Towards Real-World Quantum Machine Learning

Utkarsh Singh

Thesis submitted to the University of Ottawa in partial fulfillment of the requirements for the degree of Doctorate in Philosophy in Physics

Department of Physics
Faculty of Science
University of Ottawa

uOttawa

Abstract

Quantum machine learning (QML) promises new representational and computational capabilities, yet practical deployment on near-term hardware is hampered by resource overheads, depth constraints, and fragile trainability. This thesis advances resource-aware QML by proposing architectures and kernels that retain expressivity while sharply reducing qubit counts, circuit depth, and entangling-gate budgets. The work is presented in an article-based format with three core contributions.

First, I introduce a Coherent Feed-Forward Quantum Neural Network (CFF-QNN) that preserves quantum coherence across all layers, mirrors the flexibility of classical feed-forward networks (adjustable hidden layers and nodes), and decouples qubit requirements from input feature dimension. Compared to prevailing QNN baselines, the CFF-QNN reduces both depth and CNOT count by over 50% while achieving strong performance on standard benchmarks (e.g., 91% accuracy on Wisconsin breast cancer and 85% on credit-card fraud). This contribution is accompanied by an international patent filing (WO2025050205A1).

Second, I develop a resource-efficient quantum kernel that enables high-dimensional embeddings with substantially fewer qubits and entanglers, achieving linear scaling of entangling gates in the number of qubits. Empirically, the kernel delivers competitive or superior performance to widely used classical kernels and to popular quantum feature maps (e.g., on the Parkinson's disease dataset), with noisy simulations and small-scale runs on superconducting hardware indicating suitability for near-term devices. This contribution is covered by a companion patent filing (WO2025073041A1).

Third, I propose a quantum reservoir computing (QRC) scheme that reuses a fixed quantum feature-map circuit as the reservoir while injecting temporal memory via an explicit feedback loop. The register is recycled across time, so quantum resources remain constant, no quantum parameters are trained, and only a lightweight classical readout is fitted. Experiments on chaotic time series (e.g., Mackey–Glass) show competitive predictive accuracy with strict resource efficiency, and illuminate how feedback delay and entangling structure affect memory and error.

Collectively, these results chart a path toward practical QML: coherent architectures and kernels that are trainable, scalable in input dimension, frugal in quantum resources, and viable on noisy intermediate-scale devices—while providing design guidelines for future, larger-scale implementations.

To my parents, for their endless support, quiet sacrifices, and for teaching me the value of perseverance and curiosity that led me here.

Acknowledgements

"An approximate answer to the right question is worth a great deal more than a precise answer to the wrong question."
— *John W. Tukey* (1962)

Research is never a solitary achievement. It is a long conversation across ideas, drafts, whiteboards, and time—made possible by the patience, generosity, and curiosity of many people.

I am profoundly grateful to my supervisor, **Khabat Heshami**, for unwavering guidance, high standards, and the freedom to explore difficult ideas. When I turned toward quantum machine learning, a direction new to our group and to Khabat, he still made space for it and helped steer the work even outside his original domain. That trust and support shaped both this thesis and my growth as a researcher.

A special thanks to **Aaron** for helping with everything: idea, experimentation, writing, and countless debugging sessions.

To my group: thank you for making me feel at home—even when I was the one who rarely showed up. I will remember your surprised faces every time I was at NRC. Even when our research wasn't fully aligned, our chats were always fruitful. And **Noah**, your many questions helped me sharpen my thinking.

'To my family: thank you for believing in me. Coming from a modest background in India, most people choose a job early and settle. You still backed this longer, uncertain path—emotionally and financially—and went the extra mile, every time. I am here because of you.

To my partner, **Lily (Ritu)**—thank you for your patience and care. I'm not always easy, and long distance isn't either, but you chose to stay.

To my friends: **Archi**, **Pierre-Alexis** (whom I nicknamed "PA"—sorry, I know you hate it!), **Véronique**, **Sidhesh**, and **Mihir**—you made Ottawa less boring. **Anand** and **Dharmveer** from undergrad, thanks for the calls and for reminding me you're always there. And to my childhood friends, thank you all for your constant support.

A small smile for my nocturnal neighbour—the raccoon that tugged the cable outside my window around 1 a.m. without forgetting a single day. Relentless, if nothing else—an unintended reminder to keep going.

I also acknowledge using tools like **ChatGPT** and **Claude** to help with grammar, citation formatting, and some LaTeX figures; all ideas, analysis, and any errors are my own.

I also acknowledge that this work was conducted on the unceded, ancestral territory of the **Algonquin Anishinaabe** people. I am grateful to live and learn on this land we now call Ottawa, and I recognize the ongoing presence, knowledge, and stewardship of Indigenous people.

Finally, thank you to the **University of Ottawa** and the **National Research Council of Canada** for supporting my work, and to everyone who helped me grow along the way.

Author Contributions

This is an article-based thesis. Across all chapters, I led the conception of ideas, algorithm and circuit design, implementation, experimentation (simulation and hardware runs), data analysis, figure preparation, and manuscript writing.

Chapter 2 (CFFQNN). I originated the CFFQNN architecture, designed the circuits, derived the core properties, implemented all experiments, produced figures, and wrote the manuscript text.

Chapter 3 (A Resource-Efficient Quantum Kernel). I proposed the CP feature map, developed the kernel construction and resource analysis, implemented simulations and benchmarking, coordinated device tests, prepared figures, and wrote the manuscript.

Chapter 4 (Quantum Reservoir Computing). I developed the reservoir scheme based on a fixed feature map with explicit feedback, implemented the pipeline, designed the memory/entanglement studies, ran simulations, prepared figures, and wrote the manuscript.

Co-author roles: **Khabat Heshami** (supervisor) and **Aaron** jointly contributed across all stages: strategic guidance and project framing; discussions and experimental checks; manuscript editing; and support with patent drafting. **Christoph Simon** contributed through discussions and multiple reviews of the manuscript, providing feedback that improved the work on "Recurrent Quantum Feature Maps for Reservoir Computing" (Chapter 4). **Jean-Frédéric Laprade** contributed to the implementation of experiments on the IBM Quebec hardware for "A Resource Efficient Quantum Kernel" (Chapter 3); additional hardware experiments were performed independently by me on the IBM Torino. All co-authors of included or pending publications reviewed the relevant chapters and consented to their inclusion in this thesis.

Contents

List of Figures

List of Publications

No.	Title & Authors	Status	Notes / Patent Info
1	**Coherent Feed-Forward Quantum Neural Network** U. Singh, A. Z. Goldberg and K. Heshami	Published (2024)	Patented (WO2025050205A1)
2	**A Resource-Efficient Quantum Kernel** U. Singh, J. F. Laprade, A. Z. Goldberg and K. Heshami	Under Review	Patented (WO2025073041A1)
3	**Recurrent Quantum Feature Maps for Reservoir Computing** U. Singh, A. Z. Goldberg and K. Heshami	Preprint (arXiv: 2604.03469)	Patent Filed (2025)
4	**Seeking Metal-Organic Frameworks for hydrogen storage using classical and quantum active learning** MP. Laurenco, R. Shukla, D. Gaur, U. Singh, D. Salahub, M. Naseri and S. Gusarov	Published (2025)	Research Collaboration (University of Calgary)
5	**Local and Multi-Scale Strategies to Mitigate Exponential Concentration in Quantum Kernels** C Zendejas-Morales, D Saikia and U. Singh	Preprint (arXiv: 2602.16097 (2026))	

1

Introduction

1.1 Motivation and Vision

Quantum computing harnesses quantum-mechanical phenomena to perform computations that are believed to be intractable for classical computers. Since Feynman's 1982 proposal that quantum systems could exponentially outperform classical simulations of the quantum aspect of nature [1], researchers have pursued quantum algorithms with significant speedups. Landmark examples include Shor's algorithm for integer factorization [2], which runs in polynomial time on a quantum computer versus sub-exponential time classically, and Grover's search algorithm, which achieves quadratic speedup for unstructured search [3]. These and other algorithms promise transformative applications in cryptogra-

phy, database search, and linear algebra. In 2019, Google's 53-qubit *Sycamore* processor demonstrated *quantum supremacy* by performing a random circuit sampling task in 200 seconds—an estimated 10,000-year task for the most powerful classical supercomputer [4]. In 2022, Xanadu performed a similar experiment, when their *Borealis* photonic processor demonstrated *quantum computational advantage* by performing Gaussian boson sampling in 36 μs—an estimated 9,000-year task for the best available classical algorithms and supercomputers [5]. More recently, Google reported a verifiable quantum advantage in the estimation of second-order Out-of-Time-Order Correlators (OTOCs), which are quantities used to characterize information scrambling and quantum chaos in many-body systems. In this work, the quantum processor was able to estimate these correlators with a computational cost believed to be infeasible for classical methods, providing a physics-motivated demonstration of quantum advantage beyond sampling-based tasks. [6]

While this milestone confirmed that quantum devices can surpass classical limits and inaugurated the era of *Noisy Intermediate-Scale Quantum* (NISQ) technology [7], subsequent advances in classical simulation techniques have challenged the extent of this advantage, and alternative demonstrations of quantum advantage have since been reported, such as photonic Gaussian boson sampling with Xanadu's *Borealis* processor [5].

Although fully fault-tolerant quantum computers remain under development, NISQ devices with tens or hundreds of qubits are now accessible. However, their computational power is constrained by decoherence, noise, and limited gate fidelities, which restrict algorithmic depth. The immediate challenge is thus to extract useful computations from these devices before large-scale, error-corrected quantum computing becomes feasible.

In parallel, classical machine learning (ML) has undergone its own revolution. Over the past decade, deep learning has enabled artificial neural networks trained on massive datasets to rival or surpass human performance in tasks such as image recognition, speech processing, and decision making [8, 9]. These breakthroughs were powered by advances in parallel hardware such as GPUs and TPUs [10], together with the availability of large an-

notated datasets. Yet this success comes at a cost: ever-larger models require exponentially increasing computational and energy resources [11]. This scaling bottleneck motivates the search for complementary computational paradigms.

This raises a central question: *can quantum computing enhance machine learning, and vice versa?* The emerging field of Quantum Machine Learning (QML) explores this interplay in both directions. On the one hand, quantum systems may accelerate or improve learning by leveraging quantum parallelism, interference, and entanglement [12, 13]. On the other hand, machine learning has proven useful for quantum technologies themselves, assisting in quantum error correction, device calibration, and control [14, 15]. With the growing availability of programmable quantum hardware, QML represents a promising frontier for developing practical, resource-efficient learning models that complement or outperform their classical counterparts in certain contexts.

Recent developments have extended QML into domains of tangible scientific and industrial relevance, including quantum chemistry, drug discovery, finance, and materials science [16–18]. For example, quantum simulations combined with QML techniques have been applied to molecular Hamiltonians and drug candidate design, showing early promise in areas such as KRAS inhibitor discovery and molecular property prediction [19, 20]. At the same time, major cloud platforms—IBM Quantum, Amazon Braket, and Microsoft Azure Quantum—now provide public access to quantum hardware and hybrid algorithm support [21–23], democratizing experimentation on real devices. Advances in parameterized quantum circuit (PQC) algorithms and error mitigation strategies further support this transition from theory to practice [24, 25]. Taken together, the convergence of real-world use cases, accessible infrastructure, and rapidly evolving algorithms makes this a pivotal moment for QML research. Importantly, however, most QML approaches remain to be validated at scale, and any claims of advantage must be grounded in task-specific performance comparisons under realistic resource constraints.

Early optimism in QML was driven by theoretical proposals showing exponential speedups

under idealized assumptions. For instance, Rebentrost et al. introduced a quantum support vector machine with logarithmic runtime scaling via quantum linear algebra subroutines [26], while Lloyd et al. demonstrated quantum principal component analysis for feature extraction from quantum-encoded data [27]. These algorithms build upon the Harrow–Hassidim–Lloyd (HHL) method for solving linear systems in polylogarithmic time under sparsity and well-conditioning assumptions [28]. While these results established that quantum algorithms could in principle outperform classical methods, they often rely on quantum random access memory or input data encoded as quantum states, which are not yet practical on near-term hardware.

With the advent of NISQ processors, the focus has shifted toward hybrid and noise-tolerant algorithms. Variational quantum algorithms, which combine classical optimization with parameterized quantum circuits, have become central to this effort [29, 30]. Examples include variational quantum classifiers and quantum neural networks (QNNs), which employ trainable gate parameters in analogy with classical neural networks [31, 32]; quantum kernel methods, which embed data into high-dimensional Hilbert spaces for similarity-based learning [33]; and quantum generative adversarial networks (QGANs), which extend adversarial learning to quantum data [34]. Reinforcement learning has also been explored in quantum-enhanced forms [14]. Despite promising results, challenges such as barren plateaus—regions of vanishing gradients that hinder QNN training—remain significant [35]. Proposed remedies include shallow circuit designs and architectural regularizations, though systematic benchmarks against classical methods are still required.

In response, new architectures are being developed to better align with hardware capabilities. For example, our work presented in chapter 2 introduced a *coherent feed-forward QNN* (CFFQNN) that mirrors classical feed-forward networks while eliminating unnecessary entanglement layers and adding summation nodes analogous to perceptrons, achieving higher accuracy with fewer gates. Building on this direction, our second work 3 proposed a *resource-efficient quantum feature map* that reduces qubit counts and entangling oper-

ations for high-dimensional data. Such contributions exemplify the push toward scalable, NISQ-friendly QML methods.

Another promising paradigm is reservoir computing (RC), originally developed for efficient temporal processing using recurrent networks [36, 37]. Frameworks such as echo state networks and liquid state machines rely on training only a simple output layer, leaving the high-dimensional reservoir dynamics fixed [38]. This idea has been extended into the quantum domain: Fujii and Nakajima (2017) introduced *Quantum Reservoir Computing* (QRC), where quantum dynamics serve as the reservoir and only the classical readout is trained [39]. Remarkably, small reservoirs of just 5–7 qubits have shown memory capacities comparable to classical recurrent networks with hundreds of nodes. Enhancements such as time-multiplexing [40] and memory-capacity tuning [41] further improve performance, and experimental efforts are now exploring QRC implementations on superconducting qubits and continuous-variable platforms.

Despite these advancements, the field of quantum machine learning still encounters major challenges that slow down its adoption in real-world applications. All variational quantum algorithms, including quantum neural networks, usually suffer from barren plateaus, where gradients vanish exponentially with system size, rendering optimization intractable [35, 42]. Quantum kernel methods also face the issue of exponential concentration, where the off-diagonal kernel values concentrate around their mean in high-dimensional Hilbert spaces. This limits their ability to discriminate between two data points [43]. On top of the algorithmic issues, current (NISQ era) hardware poses additional obstacles for these algorithms. All near-term algorithms must contend with device noise, decoherence, and resource-intensive data encoding, which worsen trainability and scalability issues. These challenges highlight the importance of developing architectures and algorithms that are resource-efficient, resistant to noise, and able to maintain expressive power without excessive overhead.

While there are many significant issues, the convergence of quantum computing and

machine learning continues to create a rich and dynamic field of research. While a clear and scalable quantum advantage has not yet been demonstrated in practice, advances in algorithms, architectures, and hardware continue to close the gap. This thesis contributes to this effort by developing resource-efficient and NISQ-compatible QML methods, with a particular focus on quantum reservoir computing and quantum neural networks, to bring quantum machine learning closer to real-world applications.

1.2 Classical Machine Learning Achievements and Limitations

Machine Learning (ML) is a subfield of artificial intelligence concerned with designing algorithms that can learn patterns from data and make predictions or decisions without being explicitly programmed. Unlike traditional rule-based programming, where explicit instructions are provided, ML models infer the underlying relationship between inputs and outputs directly from observed examples. Formally, given a dataset

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N},$$

where each $\mathbf{x}_i \in \mathbb{R}^d$ is a feature vector containing d real-valued attributes, and y_i is the corresponding target value. Depending on the task, $y_i \in \mathbb{R}$ (for regression) or $y_i \in \{-1, 1\}$ (for binary classification).

The goal of machine learning is to construct a function

$$f : \mathbb{R}^d \to \mathbb{R} \quad \text{or} \quad f : \mathbb{R}^d \to \{-1, 1\},$$

that accurately predicts outputs for previously unseen inputs. Hence, the objective is not merely to fit the training data, but to generalize, meaning that the learned function

captures the underlying structure of the data rather than memorizing specific examples.

While more general settings exist—such as multiclass classification, structured outputs, or probabilistic predictions—these two cases are presented here for simplicity, as they are most directly relevant to the models and methods discussed in this thesis.

ML paradigms are typically divided into three main categories (Fig. 1.1):

- **Supervised Learning:** The model is trained on labeled data, learning a mapping from inputs $\mathbf{x}$ to outputs y. This includes regression (continuous outputs) and classification (discrete outputs).
- **Unsupervised Learning:** The algorithm extracts hidden structure from unlabeled data, e.g., clustering or dimensionality reduction.
- **Reinforcement Learning:** The algorithm (agent) learns an optimal policy of actions by interacting with an environment to maximize a reward signal.

Supervised learning is the most common setting and is especially relevant for this thesis, as both classical and quantum machine learning models are often trained on labeled datasets. Unsupervised and reinforcement learning are also active research areas, with quantum extensions being explored in contexts such as quantum clustering [44] and quantum-enhanced reinforcement agents.[45]

The following subsections introduce key supervised learning methods, including regression, classification, kernel methods, and neural networks. These concepts form the foundation for the quantum machine learning models developed in later chapters.

1.2.1 Regression and Classification

Two of the most fundamental tasks in supervised machine learning are *regression* and *classification*. The distinction lies in the nature of the target variable y.

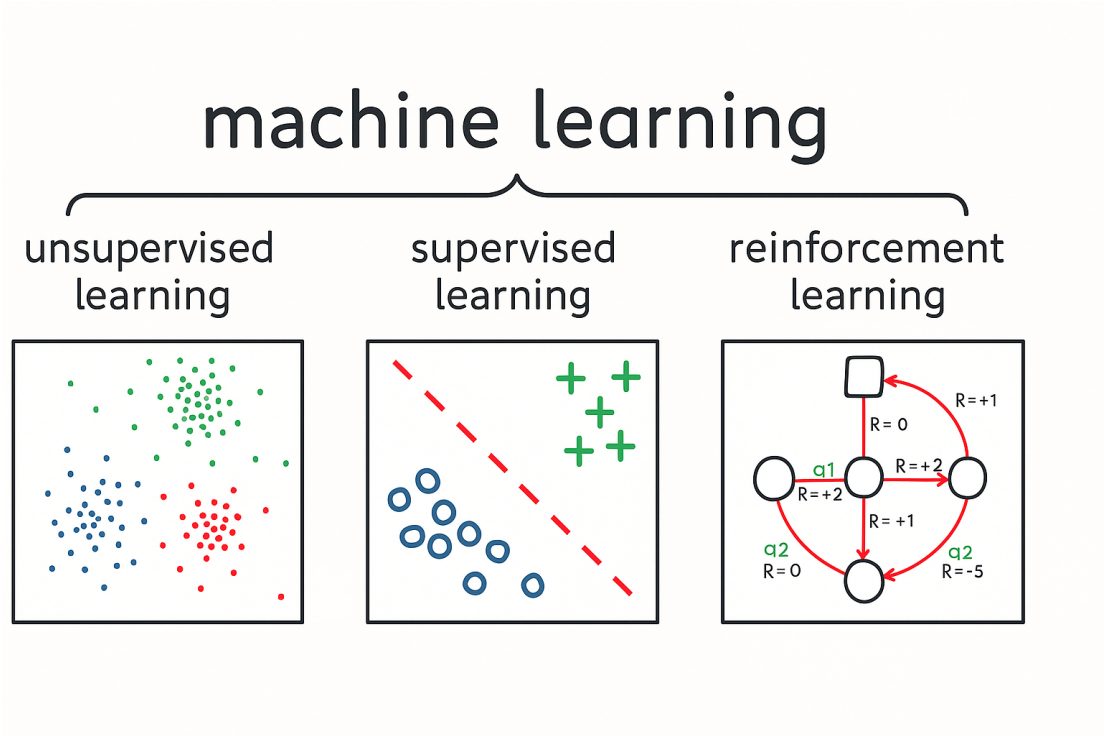


Figure 1.1: Schematic overview of the three main types of machine learning. Unsupervised learning (left): in this approach, unlabeled data points are grouped according to patterns or similarities present in the data. Supervised learning (center): in supervised learning, labeled input-output examples are used to learn a predictive mapping from features to targets. Reinforcement learning (right): here, an agent interacts with an environment, takes actions, and updates its behaviour based on rewards or penalties received over time. This figure is conceptual; the symbols and arrows illustrate information flow rather than numerical axes or measured physical quantities.

1.2.1.1 Linear Regression

In regression, the objective is to predict a continuous-valued output $y \in \mathbb{R}$ from an input vector $\mathbf{x} \in \mathbb{R}^d$. The simplest and widely used model is **linear regression**, which assumes a linear relationship between inputs and outputs:

$$\hat{y} = f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b, \tag{1.1}$$

where $\mathbf{w} \in \mathbb{R}^d$ are the weights and $b \in \mathbb{R}$ is the bias term.

This model defines a hyperplane in the d-dimensional feature space. The learning task consists of choosing $\mathbf{w}$ and b such that this hyperplane best approximates the observed

data.

The model parameters are obtained by minimizing the *mean squared error (MSE)* loss:

$$\mathcal{L}(\mathbf{w}, b) = \frac{1}{N} \sum_{i=1}^{N} (y_i - (\mathbf{w}^\top \mathbf{x}_i + b))^2. \tag{1.2}$$

Where, N denotes the number of training samples, $\mathbf{x}_i \in \mathbb{R}^d$ is the input feature vector for the i-th sample, $y_i \in \mathbb{R}$ is the corresponding target value, $\mathbf{w} \in \mathbb{R}^d$ is the weight vector, and $b \in \mathbb{R}$ is the bias term.

To write the problem more compactly, we can collect all input vectors into a single matrix

$$X = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix} \in \mathbb{R}^{N \times d},$$

which is called the design matrix. In simple terms, this matrix just stacks all data points into one table, where each row corresponds to one sample and each column corresponds to one feature. Similarly, the target values are collected into a vector $\mathbf{y} \in \mathbb{R}^N$.

Using this notation, the optimal weights can be written as

$$\mathbf{w}^* = (X^\top X)^{-1} X^\top \mathbf{y}, \tag{1.3}$$

provided $X^\top X$ is invertible. This solution corresponds to the best linear fit to the data in the least-squares sense.

For large datasets, computing the closed-form solution can become computationally expensive, as it requires forming and inverting the matrix $X^\top X$. Instead, the parameters are typically learned iteratively using gradient descent: [46]

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \tag{1.4}$$

where η is a small positive number called the learning rate, which controls the step size of the update.

1.2.1.2 Logistic Regression (Classification)

In classification, the goal is to assign an input $\mathbf{x}$ to one of K discrete classes. For example, a model might decide whether an email is spam or not, or which type of object appears in an image.

For binary classification ($K = 2$), where $y \in \{0, 1\}$ or $\{-1, 1\}$, a common model is **logistic regression**. Instead of directly predicting a class label, logistic regression predicts the probability that the input belongs to a given class:

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b), \tag{1.5}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function. This function maps any real number to a value between 0 and 1, allowing the output to be interpreted as a probability.

To obtain a final class label, the predicted probability is compared to a threshold:

$$\hat{y} = \begin{cases} 1 & \text{if } P(y = 1 \mid \mathbf{x}) > \tau, \\ 0 & \text{otherwise.} \end{cases}$$

The threshold τ is often chosen as 0.5, but it can be adjusted depending on the problem.

The model parameters are typically learned by minimizing the *binary cross-entropy* loss function:

$$\mathcal{L}(\mathbf{w}, b) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i) \right]. \tag{1.6}$$

It is the most commonly used loss function for this setting.

For problems with more than two classes ($K > 2$), logistic regression generalizes to **softmax regression**:

$$P(y = k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^\top \mathbf{x}+b_k}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}+b_j}}, \tag{1.7}$$

This expression assigns a probability to each class, such that all probabilities sum to one.

with the cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{I}(y_i = k) \log P(y_i = k|\mathbf{x}_i). \tag{1.8}$$

where $\mathbb{I}(y_i = k)$ is the indicator function, which equals 1 if the true label of the i-th sample is k, and 0 otherwise. In other words, the model is rewarded for assigning high probability to the correct class and penalized otherwise.

1.2.1.3 Connection to Artificial Neural Networks

Both regression and classification form the foundation of more complex models such as neural networks. For example, a neural network with no hidden layers and a linear activation is equivalent to linear regression, while networks with sigmoid or softmax outputs generalize logistic regression to more expressive models.

1.2.2 Support Vector Machines and Kernel Methods

Support Vector Machines (SVMs) are powerful supervised learning algorithms widely used for both classification and regression tasks. They are particularly effective in high-

dimensional spaces and form the mathematical foundation for kernel-based quantum machine learning approaches.

1.2.2.1 Geometric intuition

Consider a binary classification problem where each data point $\mathbf{x}_i \in \mathbb{R}^d$ belongs to one of two classes labelled $y_i \in \{-1, +1\}$. The goal is to find a boundary that separates the two classes.

An SVM seeks a linear decision boundary of the form

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b = 0, \tag{1.9}$$

which defines a hyperplane in the feature space. Points for which $f(\mathbf{x}) > 0$ are assigned to one class, and those with $f(\mathbf{x}) < 0$ to the other.

Unlike simple linear classifiers, SVMs do not just find any separating hyperplane. Instead, they choose the one that maximizes the margin, defined as the distance between the hyperplane and the closest data points. These closest points are called *support vectors*.

Maximizing the margin improves robustness and generalization, as the classifier becomes less sensitive to small perturbations in the data.

1.2.2.2 Hard-Margin SVM

If the data are perfectly linearly separable, the optimal hyperplane can be found by solving:

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{1.10}$$

$$\text{subject to:} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad \forall i. \tag{1.11}$$

Here, $\|\mathbf{w}\|$ controls the margin: minimizing $\|\mathbf{w}\|^2$ corresponds to maximizing the distance between classes. This is known as the **hard-margin SVM**.

1.2.2.3 Soft-Margin SVM

In practice, data are rarely perfectly separable. To allow for classification errors, slack variables $\xi_i \geq 0$ are introduced, which measure how much the i-th data point violates the margin constraint.

$$\min_{\mathbf{w},b,\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \tag{1.12}$$

$$\text{subject to:} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0. \tag{1.13}$$

Here, $C > 0$ controls the trade-off between maximizing the margin and minimizing classification errors. A large C penalizes errors more strongly, while a smaller C allows a wider margin at the cost of more misclassifications.

1.2.2.4 Dual Formulation

Instead of solving the optimization problem directly in terms of the model parameters $(\mathbf{w}, b)$, it can be reformulated using Lagrange multipliers $\alpha_i \geq 0$. This leads to an equivalent representation known as the dual formulation, where the optimization is expressed entirely in terms of the training data:

$$\max_{\alpha} \quad \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \tag{1.14}$$

$$\text{subject to:} \quad \sum_{i=1}^{N}\alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C. \tag{1.15}$$

Here, each α_i determines the importance of the corresponding training point. Only a subset of these coefficients will be non-zero, and the associated data points are called support vectors, because they define the decision boundary. The parameter $C > 0$ controls the trade-off between maximizing the margin and allowing classification errors.

The key advantage of this formulation is that the data appear only through inner products $\mathbf{x}_i^\top \mathbf{x}_j$. This is particularly useful, as it allows the model to be extended to nonlinear settings by replacing the inner product with a kernel function.

Once the optimal α_i are found, the final decision function becomes:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^\top \mathbf{x} + b\right). \tag{1.16}$$

1.2.2.5 Kernel Trick

Many datasets are not linearly separable in their original feature space. A common strategy is to map the data into a higher-dimensional space where separation becomes possible:

$$\phi : \mathbb{R}^d \to \mathcal{H}, \tag{1.17}$$

where $\mathcal{H}$ is a (possibly high-dimensional) feature space.

Instead of computing $\phi(\mathbf{x})$ explicitly, which may be computationally expensive, the SVM only requires inner products of the form

$$\langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle.$$

The kernel trick replaces these inner products with a kernel function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle, \tag{1.18}$$

This allows the model to operate in a high-dimensional space without explicitly computing the mapping ϕ.

Common kernels include:

- **Linear kernel:** $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \mathbf{x}'$
- **Polynomial kernel:** $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}' + c)^p$
- **Gaussian (RBF) kernel:** $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$

The decision function becomes:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b\right). \tag{1.19}$$

These classical kernels not only motivated the construction of quantum kernels, where we use a quantum circuit as a feature map ϕ, but they also serve as natural baselines for comparison. In Chapter 3, we evaluated our proposed quantum feature map in the kernel setup by benchmarking against all three of the classical kernels introduced above.

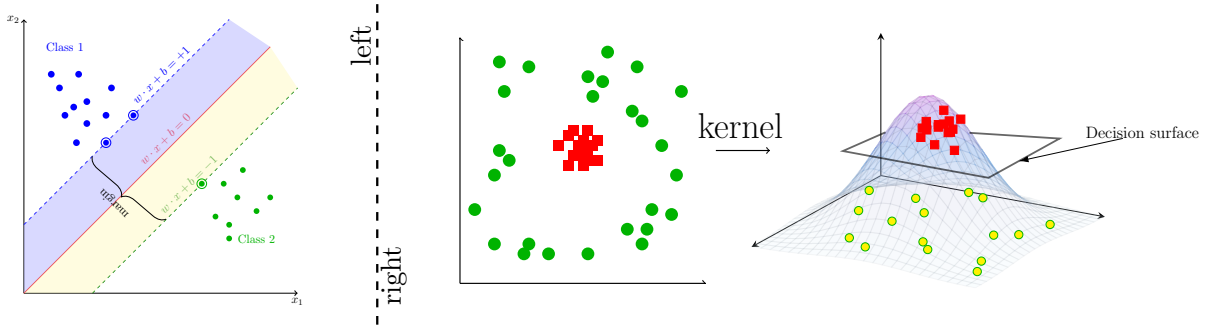


Figure 1.2: Geometric illustration of Support Vector Machines (left: linear separation, right: Classification using kernel trick).

1.2.2.6 Support Vector Regression (SVR)

SVMs can be extended to regression, known as **support vector regression**. The objective is to find a function $f(\mathbf{x})$ that approximates a continuous target variable.

In SVR, the objective is to find a function $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$ that deviates from the true targets y_i by at most ϵ for all training points, while being as flat as possible. This leads to:

$$\min_{\mathbf{w},b,\xi,\xi^*} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{1.20}$$

$$\text{subject to:} \begin{cases} y_i - \mathbf{w}^\top \mathbf{x}_i - b \leq \epsilon + \xi_i \\ \mathbf{w}^\top \mathbf{x}_i + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{1.21}$$

Here, ϵ defines a margin of tolerance within which errors are ignored, and ξ_i, ξ_i^* are slack variables that allow deviations beyond this margin.

Unlike standard regression methods that penalize all errors, SVR introduces an ϵ-insensitive region, meaning that small errors within this region do not contribute to the loss. This makes the model more robust to noise and small fluctuations in the data.

Kernel extension: As in classification, SVR can be extended using kernel functions. This allows nonlinear regression by implicitly mapping inputs into a higher-dimensional feature space. The resulting model can capture complex relationships while maintaining the advantages of the SVM framework.

1.2.2.7 Connection to Quantum Kernels

Kernel methods provide a natural bridge to quantum machine learning. Quantum feature maps $\phi_q(\mathbf{x})$ embed classical data into the Hilbert space of a quantum system. Such as, into the angles of a parameterized quantum circuit:

$$|\phi_q(\mathbf{x})\rangle = U(\mathbf{x})|0\rangle. \tag{1.22}$$

The corresponding kernel is defined as the overlap between quantum states:

$$k_q(\mathbf{x}, \mathbf{x}') = |\langle \phi_q(\mathbf{x}) | \phi_q(\mathbf{x}') \rangle|^2, \tag{1.23}$$

which can be computed on a quantum computer.

This can be interpreted as a similarity measure computed in the Hilbert space of a quantum system. Importantly, this feature space grows exponentially with the number of qubits, enabling representations that may be difficult to reproduce classically.

This concept forms the basis of quantum kernel methods and is directly connected to our later work on a *resource-efficient quantum kernel* presented in Chapter 3 [47].

1.2.3 Neural Networks

Artificial Neural Networks (ANNs) are flexible models designed to learn complex mappings from inputs to outputs by composing affine transformations with nonlinear activations [8, 48]. They underpin many state-of-the-art systems in vision, language, and control. This subsection introduces the perceptron, multilayer feed-forward networks, backpropagation, optimization, regularization, and a brief view of convolutional networks. These ingredients are directly relevant to the quantum analogues discussed later and in Chapter 2.

1.2.3.1 Perceptron and Linear Separability

The perceptron (see figure 1.3), originally proposed by Rosenblatt [49], is one of the earliest models of a binary linear classifier. It maps an input vector $\mathbf{x} \in \mathbb{R}^d$ to a discrete label $y \in \{-1, +1\}$ through the decision rule

$$\hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x} + b), \tag{1.24}$$

where $\mathbf{w} \in \mathbb{R}^d$ is the weight vector and $b \in \mathbb{R}$ is the bias term. Geometrically, the perceptron defines a separating hyperplane in $\mathbb{R}^d$ with normal vector $\mathbf{w}$, assigning labels according to which side of the hyperplane an input lies.

For linearly separable data, the perceptron learning algorithm iteratively adjusts the parameters whenever a misclassification occurs. Given a training example $(\mathbf{x}_i, y_i)$, if the prediction violates the margin condition $y_i(\mathbf{w}^\top \mathbf{x}_i + b) \leq 0$, the update rule is applied:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\, y_i \mathbf{x}_i, \tag{1.25}$$

$$b \leftarrow b + \eta\, y_i, \tag{1.26}$$

where $\eta > 0$ is the learning rate. This procedure continues until all training points are correctly classified or a predefined stopping criterion is reached. The celebrated Perceptron Convergence Theorem guarantees that if the data are strictly linearly separable, the algorithm converges in a finite number of updates [50]. Although limited in its expressive power, the perceptron established the foundation for modern neural network architectures [8].

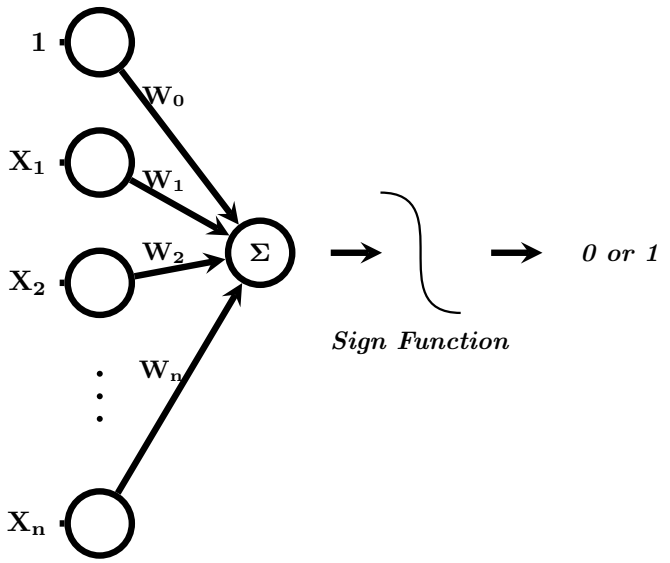


Figure 1.3: A perceptron, inspired by neural networks in the brain. Inputs X_i are combined with weights W_i including a bias W_o that are processed nonlinearly to produce a binary output

1.2.3.2 Multilayer Perceptrons (MLPs)

To model more complex relationships, neural networks introduce hidden layers and nonlinear activation functions. A single hidden-layer network (multilayer perceptron, MLP) takes the form:

$$f_\Theta(\mathbf{x}) = \mathbf{v}^\top \sigma(W\mathbf{x} + \mathbf{b}) + c, \tag{1.27}$$

where $W \in \mathbb{R}^{m\times d}$ is the weight matrix of the hidden layer, $\mathbf{b} \in \mathbb{R}^m$ is the bias vector, $\mathbf{v} \in \mathbb{R}^m$ contains the output weights, $c \in \mathbb{R}$ is the output bias. The set of all parameters is denoted by $\Theta = \{W, \mathbf{b}, \mathbf{v}, c\}$, and $\sigma(\cdot)$ (e.g., ReLU, tanh, sigmoid) is a nonlinear activation function applied element-wise.

By stacking multiple layers (figure 1.4), the network becomes:

$$\mathbf{h}^{(1)} = \sigma^{(1)}(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}), \tag{1.28}$$

$$\mathbf{h}^{(\ell)} = \sigma^{(\ell)}(W^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}), \quad \ell = 2, \ldots, L, \tag{1.29}$$

$$\hat{y} = g(W^{(L+1)}\mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}), \tag{1.30}$$

where g (identity for regression, softmax for classification) is the output nonlinearity.

In this case, each layer can be viewed as transforming the data into a new representation. Early layers extract simple features, while deeper layers capture more abstract patterns. This hierarchical representation is a key reason for the success of deep learning.

Neural networks are highly expressive: even relatively simple architectures can approximate a wide class of functions. This property is formalized by the universal approximation theorem [51], which states that a feed-forward neural network with a single hidden layer and a suitable nonlinear activation function (e.g., sigmoid or tanh) can approximate any

continuous function on a compact subset of $\mathbb{R}^d$ to arbitrary accuracy, given sufficiently many hidden units.

In practical terms, this means that neural networks are not limited by their functional form, but rather by factors such as the number of parameters, the availability of data, and the efficiency of training algorithms. However, while a shallow network can in principle approximate any function, doing so may require an impractically large number of neurons. This motivates the use of deep architectures, which can represent complex functions more efficiently through hierarchical feature representations.

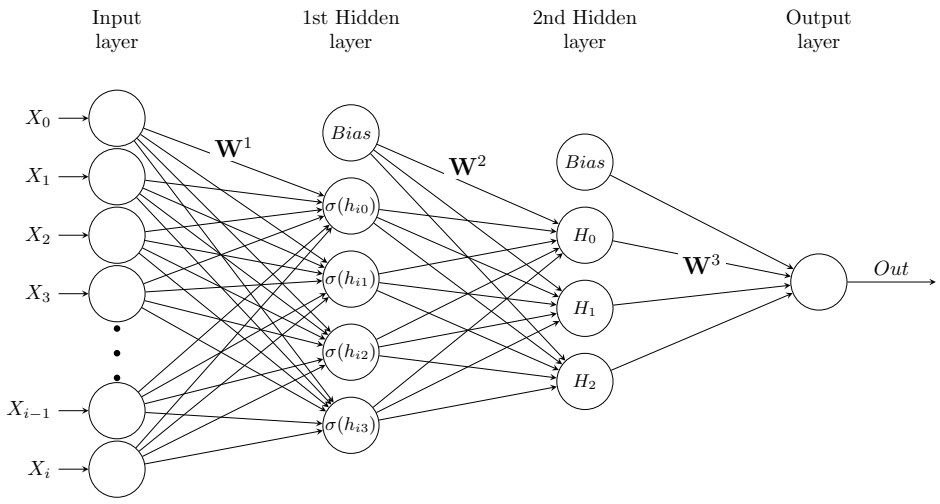


Figure 1.4: Architecture of a feed-forward MLP with two hidden layers. Here $\mathbf{W}$ represents the weight parameters, $\mathbf{X}$ are data points, σ is a non-linear activation function, and $h_{ij} = W_{ij}X_i$. This structure directly motivates the coherent feed-forward quantum architecture in Chapter 2.

Loss functions: Neural networks are trained by minimizing a loss function, which quantifies the difference between the predicted output $\hat{y}$ and the true target y. The choice of loss function depends on the task.

For regression with targets $y \in \mathbb{R}$, the mean squared error (MSE) is the most widely used loss function, given by

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{1.31}$$

For multiclass classification with labels $y_i \in \{1, \dots, K\}$, the softmax function converts outputs into probabilities

$$p_\Theta(y = k \mid \mathbf{x}) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}, \qquad \mathbf{z} = W^{(L+1)}\mathbf{h}^{(L)} + \mathbf{b}^{(L+1)}, \tag{1.32}$$

and the cross-entropy loss is used:

$$\mathcal{L}_{\mathrm{CE}}(\Theta) = -\frac{1}{N}\sum_{i=1}^{N} \log p_\Theta(y_i \mid \mathbf{x}_i). \tag{1.33}$$

The loss function defines the objective of learning: it measures how well the model performs on the training data. Training a neural network consists of adjusting its parameters to minimize this loss. This optimization problem is typically solved using gradient-based methods, described later in this chapter.

Activation Functions: A neural network becomes expressive only when nonlinear transformations are introduced between layers. If every layer performed only a linear transformation, then even a deep network would be equivalent to a single linear map. Activation functions provide this essential nonlinearity, allowing neural networks to model complex input-output relationships.

Common activation functions include:

$$\mathrm{sigmoid}(z) = \frac{1}{1+e^{-z}}, \quad \tanh(z) = \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}, \quad \mathrm{ReLU}(z) = \max(0, z). \tag{1.34}$$

The sigmoid function maps real numbers to the interval $(0, 1)$ and is therefore useful when outputs are interpreted probabilistically. The hyperbolic tangent maps inputs to the interval $(-1, 1)$ and is often preferred when zero-centered activations are desirable. The Rectified Linear Unit (ReLU) outputs zero for negative inputs and grows linearly for positive inputs. ReLU and its variants are widely used in deep learning because they are

simple to compute and often reduce training difficulties associated with vanishing gradients.

In summary, activation functions determine how information is transformed between layers and play a central role in the expressive power and trainability of neural networks. This point is also important for quantum machine learning: while nonlinear activation functions are natural in classical neural networks, implementing analogous nonlinear operations coherently in quantum circuits is nontrivial and often requires alternative strategies.

1.2.3.3 Backpropagation and Gradient-Based Optimization

Training a neural network requires computing how changes in parameters affect the loss. This is achieved using **backpropagation**, which applies the chain rule of calculus to efficiently compute gradients layer by layer [52].

For a single sample $(\mathbf{x}, y)$ and a layer ℓ with pre-activations $\mathbf{a}^{(\ell)} = W^{(\ell)}\mathbf{h}^{(\ell-1)} + \mathbf{b}^{(\ell)}$ and activations $\mathbf{h}^{(\ell)} = \sigma^{(\ell)}(\mathbf{a}^{(\ell)})$, an error signal is propagated backward:

$$\boldsymbol{\delta}^{(\ell)} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(\ell)}} = (W^{(\ell+1)\top}\boldsymbol{\delta}^{(\ell+1)}) \odot \sigma'^{(\ell)}(\mathbf{a}^{(\ell)}), \tag{1.35}$$

where $\odot$ denotes element-wise multiplication. The error signal at the output layer is given by

$$\boldsymbol{\delta}^{(L+1)} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}} \quad \text{(e.g., for softmax+CE, } \boldsymbol{\delta}^{(L+1)} = \mathbf{p} - \mathbf{y}_{\text{onehot}}). \tag{1.36}$$

Backpropagation can be understood as passing information about the prediction error from the output layer back through the network. Each layer adjusts its parameters based on how much it contributed to the final error.

Gradients can be estimated as

$$\frac{\partial \mathcal{L}}{\partial W^{(\ell)}} = \boldsymbol{\delta}^{(\ell)}\, \mathbf{h}^{(\ell-1)\top}, \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(\ell)}} = \boldsymbol{\delta}^{(\ell)}. \tag{1.37}$$

The gradients are then used to update parameters via gradient descent:

$$\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}, \tag{1.38}$$

where $\eta > 0$ is the learning rate, which determines the step size of each update, and $\nabla_\Theta \mathcal{L}$ is the gradient of the loss function with respect to the parameters.

Optimization algorithms: The parameters of a neural network are updated iteratively using gradient-based optimization. The simplest method is **stochastic gradient descent (SGD)**, which updates Θ according to equation 1.38.

In practice, gradients are computed on small subsets of the data, called minibatches, rather than the full dataset. This reduces computational cost and introduces useful stochasticity that can help optimization.

More advanced methods improve convergence by adapting the update step. One widely used example is the **Adam** optimizer, which keeps track of moving averages of past gradients and their squares:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{1.39}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{1.40}$$

where $g_t = \nabla_\Theta \mathcal{L}_t$ is the gradient of the loss function at iteration t, computed on the current minibatch. The quantity m_t is an exponentially weighted average of past gradients (the first moment), while v_t is the second moment estimate, an exponentially weighted average of the squared gradients. The constants β_1 and β_2 are decay rates, typically chosen close to 1, that determine how strongly past gradients influence the current averages.

The parameter update is then given by

$$\Theta_t = \Theta_{t-1} - \eta \frac{m_t/(1-\beta_1^t)}{\sqrt{v_t/(1-\beta_2^t)} + \epsilon}, \tag{1.41}$$

where the factors $1-\beta_1^t$ and $1-\beta_2^t$ provide bias correction in the early stages of training, and ϵ is a small positive constant added for numerical stability to avoid division by zero.

Compared with standard SGD, Adam adapts the effective learning rate for each parameter individually. This often leads to faster and more stable training, especially in large neural networks or when gradients vary significantly across parameters.

1.2.3.4 Initialization, Normalization, and Regularization

Initialization: The choice of initial parameter values plays an important role in training neural networks. If weights are initialized too large, signals can grow exponentially across layers (exploding activations); if too small, they can shrink toward zero (vanishing activations). Both effects make training difficult.

To mitigate this, modern initialization schemes choose the scale of the weights based on the structure of each layer. In particular, Xavier (or Glorot) initialization is commonly used for activation functions such as tanh or sigmoid [53], while He initialization is preferred for ReLU-type activations [54]. These methods scale the initial weights according to the fan-in and fan-out of a layer, which refer to the number of input and output connections, respectively. This ensures that the variance of signals remains approximately constant as they propagate through the network, improving training stability.

Normalization. During training, the distribution of activations within a network can change as parameters are updated. This effect can slow down optimization. **Batch Normalization** addresses this by normalizing intermediate activations within each minibatch.

Given a vector of pre-activation values $\mathbf{a}$ for a minibatch $\mathcal{B}$, the normalized activations are computed as:

$$\hat{\mathbf{a}} = \frac{\mathbf{a} - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}, \qquad \mathbf{h} = \gamma \odot \hat{\mathbf{a}} + \beta, \tag{1.42}$$

where $\mu_{\mathcal{B}}$ and $\sigma_{\mathcal{B}}^2$ are the mean and variance of $\mathbf{a}$ computed over the minibatch, ϵ is a small constant added for numerical stability, and γ, β are learnable parameters that allow the network to rescale and shift the normalized activations. The symbol $\odot$ denotes element-wise multiplication.

Regularization: Highly expressive neural networks are prone to overfitting, meaning they fit the training data well but perform poorly on unseen data. Regularization techniques are used to improve generalization.

A common approach is **weight decay** (or L_2 regularization), which adds a penalty term

$$\lambda \sum_{\ell} \|W^{(\ell)}\|_F^2 \tag{1.43}$$

to the loss function. Here, $\lambda > 0$ controls the strength of the penalty, and $\|\cdot\|_F$ denotes the Frobenius norm, which measures the overall magnitude of the weight matrix.

Another widely used technique is **dropout**, where activations are randomly set to zero during training:

$$h \leftarrow \xi \odot h, \qquad \xi \sim \text{Bernoulli}(p),$$

where $\odot$ denotes element-wise multiplication, ξ is a random binary mask and p is the probability of keeping a unit active. This prevents the network from relying too strongly on specific neurons and encourages more robust representations.

Additional strategies include **early stopping**, where training is halted when validation performance stops improving, and **data augmentation**, where the training dataset is

artificially expanded using transformations such as rotations or noise.

1.2.3.5 From Classical NNs to Quantum Analogues

Many architectural ideas translate to quantum models: layers (unitary blocks), parameters (gate angles), and readouts (measurements). However, implementing nonlinear activations coherently is nontrivial in quantum circuits. Practical QNNs often obtain effective nonlinearity via measurement, data re-uploading, or controlled unitaries.

Chapter 2 introduces the *Coherent Feed-Forward Quantum Neural Network (CFFQNN)*, which mirrors classical feed-forward structure while preserving coherence by replacing explicit nonlinearities with coherent summation nodes and hardware-efficient blocks, achieving competitive accuracy with fewer entangling gates. This connection motivates careful consideration of classical design principles (layering, parameter sharing, regularization) when proposing quantum counterparts.

1.2.4 Recurrent Models and Reservoir Computing

Sequential data, such as speech, weather records, or financial time-series, require models capable of capturing temporal dependencies. Recurrent Neural Networks (RNNs) address this by introducing a hidden state that evolves over time. At each timestep t, the network updates its internal state according to

$$\mathbf{h}_t = \phi(W_{in}\mathbf{u}_t + W_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h), \tag{1.44}$$

$$\mathbf{y}_t = g(W_{out}\mathbf{h}_t + \mathbf{b}_y), \tag{1.45}$$

where $\mathbf{u}_t$ is the input at time t, $\mathbf{h}_t$ is the hidden state, ϕ is a nonlinear activation, g is an output activation function and $\mathbf{y}_t$ is the output.

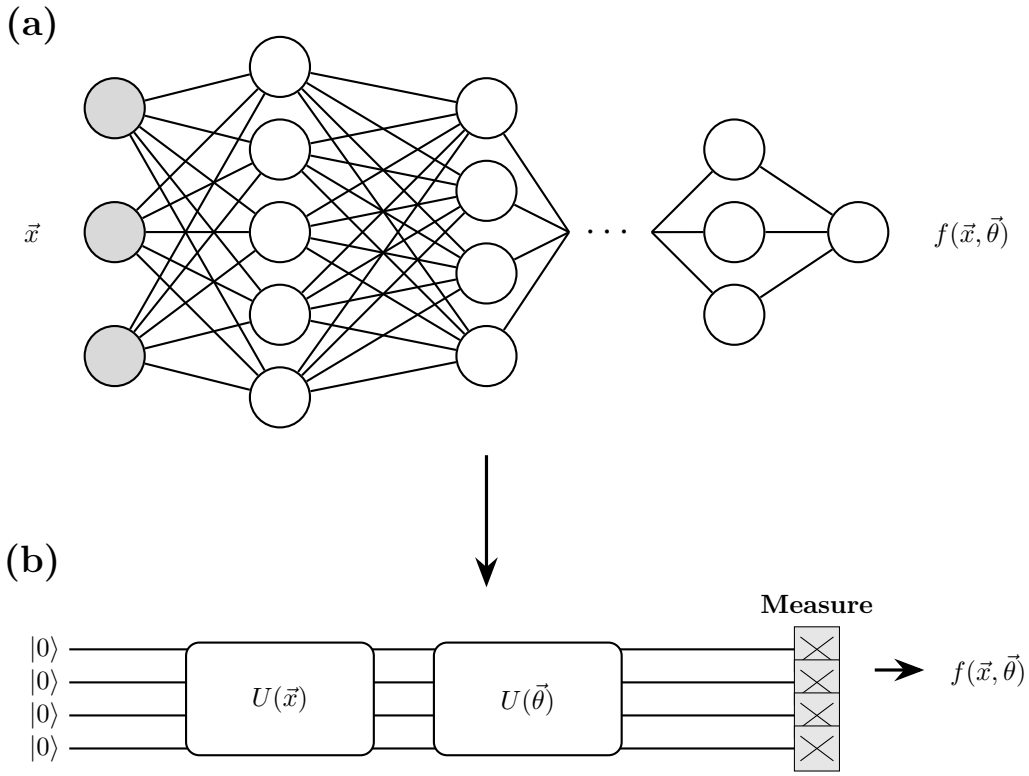


Figure 1.5: Comparison between a classical multilayer perceptron (MLP) and a quantum neural network (QNN). **(a)** In a classical MLP, an input vector $\vec{x}$ is propagated through successive layers with trainable parameters $\vec{\theta}$, producing an output $f(\vec{x};\vec{\theta})$. **(b)** In a QNN, the input $\vec{x}$ is encoded into a quantum state via $U(\vec{x})$, followed by a variational circuit $U(\vec{\theta})$ with trainable parameters. The output $f(\vec{x};\vec{\theta})$ is obtained from quantum measurements.

The key feature of recurrent neural networks is the presence of the term $W_{hh}\mathbf{h}_{t-1}$, which feeds the previous hidden state back into the network. This introduces a recursive structure in time, allowing the same transformation to be applied repeatedly across different timesteps. In particular, the matrix W_{hh} is shared across all timesteps, meaning that the same set of parameters is reused regardless of the length of the input sequence (please see the figure 1.6). This parameter sharing enables RNNs to process sequences of arbitrary length without increasing the number of trainable parameters.

This same repeated structure creates difficulties during training with Backpropagation Through Time (BPTT), where gradients are propagated backward across multiple timesteps.

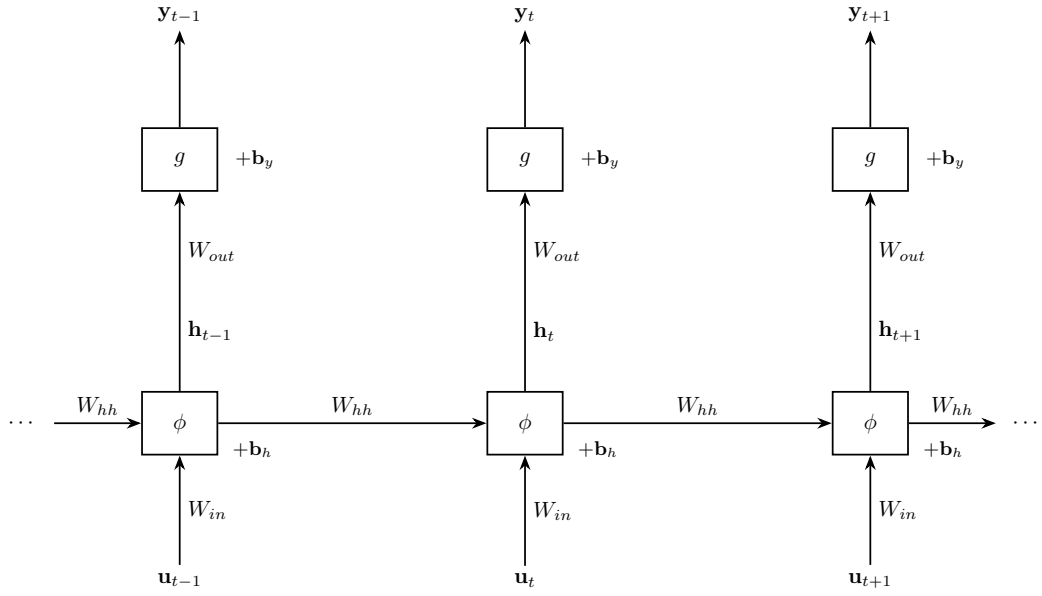


Figure 1.6: Unrolled representation of a Recurrent Neural Network (RNN). At each timestep t, the input u_t is combined with the previous hidden state h_{t-1} to produce a new hidden state h_t, which in turn generates the output y_t. The matrices W_{in}, W_{hh}, and W_{out} denote input, recurrent, and output weights, respectively, and are shared across all timesteps. This parameter sharing allows the network to process sequences of arbitrary length while maintaining a fixed number of parameters.

From the recurrence

$$\mathbf{h}_t = f(W_{hh}, \mathbf{h}_{t-1}), \tag{1.46}$$

we see that the hidden state is obtained by repeatedly applying the same transformation. As a result, during backpropagation, the gradient involves repeated multiplication by W_{hh}.

After k timesteps, this leads to terms of the form W_{hh}^k, which control how gradients behave over time. If the magnitude of W_{hh} is less than one, the gradients shrink exponentially and become too small for numerical computation, leading to the vanishing gradient problem. If it is greater than one, the gradients grow exponentially, causing instability and the exploding gradient problem [55].

To overcome this limitation, gated architectures such as Long Short-Term Memory (LSTM) [56] and Gated Recurrent Units (GRUs) [57] were introduced. These models control information flow via gating mechanisms, mitigating gradient instabilities and improv-

ing performance on sequential tasks. Nevertheless, their training remains resource-intensive and sensitive to hyperparameters.

Reservoir Computing: As an alternative, Reservoir Computing (RC) was proposed [38, 58]. The key idea is to use a large, randomly connected recurrent network (reservoir) with fixed internal weights. Inputs are nonlinearly projected into a high-dimensional dynamical space, and only the linear output layer (the readout) is trained, usually via ridge regression. This dramatically simplifies training compared to full RNN optimization.

Formally, the reservoir dynamics are given by [58]

$$\mathbf{h}_t = \phi(W_{in}\mathbf{u}_t + W_{hh}\mathbf{h}_{t-1}), \tag{1.47}$$

which is structurally identical to an RNN. However, in contrast to standard RNNs, the matrices W_{in} and W_{hh} are randomly initialized and kept fixed during training. Only the output layer

$$\mathbf{y}_t = W_{out}\mathbf{h}_t \tag{1.48}$$

is trained, typically using linear or ridge regression. Please note that the symbols here carry the same meanings as in the RNN section.

The effectiveness of RC relies on dynamical properties such as the Echo State Property (ESP) and its Memory Capacity (MC) [59]. The Echo State Property (ESP) ensures that the influence of initial conditions vanishes over time and that the state $\mathbf{h}_t$ is determined primarily by the input sequence. And memory capacity quantifies how well the reservoir can retain information about past inputs. A good reservoir must balance these two competing effects: it must have sufficient memory to retain past information, while also being responsive to new inputs. A detailed discussion of these properties, along with their mathematical formulations, is presented in Chapter 4, where we also extend the framework to Quantum Reservoir Computing (QRC).

Motivation for Quantum Extensions: Quantum systems evolve in exponentially large Hilbert spaces with natural recurrent dynamics and entanglement. When used as reservoirs, quantum systems can therefore provide rich, high-dimensional representations of temporal data. In Chapter 4, we build on this idea and introduce Recurrent Quantum Feature Maps for reservoir computing, where a quantum feature map circuit with an external feedback loop acts as the reservoir.

1.2.5 Limitations of Classical Machine Learning

Despite the remarkable success of classical ML across domains such as computer vision, natural language processing, and scientific discovery, it faces intrinsic limitations that motivate the exploration of quantum-enhanced approaches:

- **Scalability and computational cost:** Training state-of-the-art deep learning models requires enormous computational resources and energy consumption. For example, training large language models with hundreds of billions of parameters can cost millions of dollars and generate substantial carbon footprints [11]. This raises concerns about sustainability and accessibility.

  While quantum computing is sometimes proposed as a way to address such challenges, it is important to note that current quantum devices are limited in size and reliability. As a result, quantum approaches do not yet provide a general solution to scalability issues, but rather offer a potential alternative framework for certain structured problems.

- **No-Free-Lunch Theorem (NFL):** The NFL theorem [60] implies that, averaged over all possible data distributions, no learning algorithm performs better than another. This result is independent of whether the algorithm is classical or quantum. Consequently, quantum machine learning does not bypass this limitation; any advantage must arise from exploiting specific structures in the data or problem domain.

- **High-dimensional and structured data:** While kernel methods and deep networks are powerful, they often struggle with very high-dimensional or highly structured data (e.g., quantum many-body wavefunctions or molecular descriptors). Without careful feature engineering, regularization, or inductive biases, classical models risk overfitting or inefficiency.

- **Data requirements and generalization:** Deep learning models typically require large amounts of labelled data to generalize effectively. However, in many important applications—such as medical diagnostics, quantum chemistry, and materials science—data can be scarce, expensive to obtain, or noisy. [61, 62] This limits the applicability of standard data-intensive approaches and motivates the development of methods that can learn efficiently from limited datasets.

Toward Quantum Machine Learning. These limitations have inspired the field of Quantum Machine Learning (QML), which explores whether quantum computation can provide new representational or computational advantages. Early theoretical works demonstrated this potential: Rebentrost et al. proposed a quantum support vector machine (QSVM) with logarithmic runtime under specific assumptions [26], Lloyd et al. introduced quantum principal component analysis (QPCA) for feature extraction from quantum states [27], and Harrow, Hassidim, and Lloyd developed the celebrated HHL algorithm for solving linear systems with exponential speedup under sparsity and conditioning constraints [28].

Although many of these algorithms remain impractical on today's noisy intermediate-scale quantum (NISQ) devices due to requirements such as quantum random access memory (qRAM), they established the theoretical foundation for QML. More recently, attention has shifted to variational and hybrid quantum-classical approaches, which can be implemented on near-term hardware. These approaches will be the focus of the subsequent sections, and provide the context for the contributions of this thesis.

1.3 Quantum Computing Foundations

This section introduces the essential concepts from quantum mechanics, quantum computation, and their relevance to quantum machine learning (QML). The goal is to provide a self-contained overview that equips readers from diverse backgrounds with the necessary foundations. Mathematical rigor is emphasized, and illustrative figures are referenced throughout to complement the formalism.

1.3.1 Mathematical Foundations

Quantum mechanics is formulated on the structure of complex Hilbert spaces, with physical states, observables, and dynamics expressed in terms of linear algebra.

1.3.1.1 Hilbert Spaces and Dirac Notation

A quantum system is described by a state vector $|\psi\rangle \in \mathcal{H}$, where $\mathcal{H}$ is a complex Hilbert space with inner product $\langle\phi|\psi\rangle$. Physical states are normalized such that $\langle\psi|\psi\rangle = 1$. Global phases are physically irrelevant, so states live in the projective Hilbert space $\mathbb{P}(\mathcal{H})$.

The Dirac notation is widely used: kets $|\psi\rangle$ represent column vectors, bras $\langle\psi|$ are their Hermitian conjugates, inner products $\langle\phi|\psi\rangle$ yield scalars, and outer products $|\psi\rangle\langle\phi|$ define linear operators.

1.3.1.2 Basis States and Tensor Products

For a qubit, the computational basis $\{|0\rangle, |1\rangle\}$ spans $\mathbb{C}^2$. A general qubit state is

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \qquad \alpha, \beta \in \mathbb{C}, \quad |\alpha|^2 + |\beta|^2 = 1. \tag{1.49}$$

Any single-qubit pure state can be represented on the Bloch sphere:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,. \tag{1.50}$$

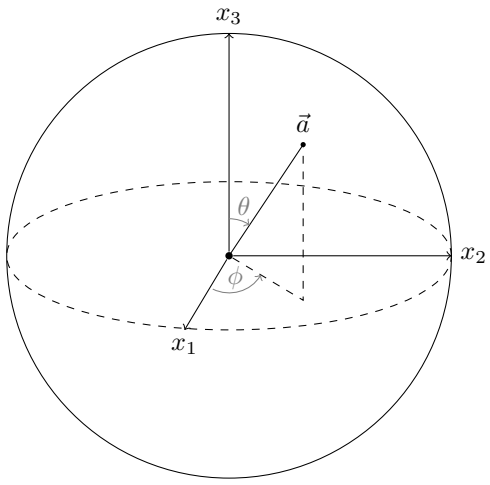


Figure 1.7: The Bloch sphere representation of a qubit.[63] Polar angle θ and azimuthal angle ϕ parameterize any pure state.

For composite systems, the Hilbert space is the tensor product $\mathcal{H}_A \otimes \mathcal{H}_B$. Entangled states, such as the Bell state (shown in figure 1.8)

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \tag{1.51}$$

cannot be factorized into product states.

1.3.1.3 Operators and Measurements

Operators acting on $\mathcal{H}$ include:

- Hermitian operators ($A = A^\dagger$): observables with real eigenvalues.
- Unitary operators ($U^\dagger U = I$): reversible dynamics.
- Projectors ($P^2 = P$): describe measurement outcomes.

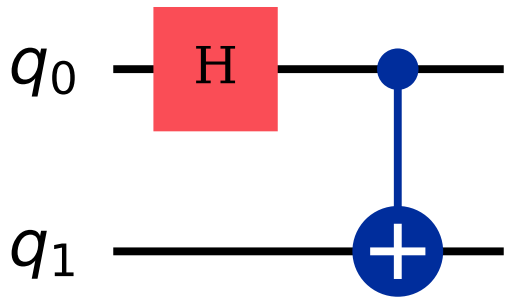


Figure 1.8: Circuit for preparing the Bell state $|\Phi^+\rangle$: apply a Hadamard to the first qubit followed by a CNOT.

The spectral theorem ensures any Hermitian A has a decomposition $A = \sum_i \lambda_i |a_i\rangle \langle a_i|$. Measurements of A yield eigenvalue λ_i with probability $|\langle a_i|\psi\rangle|^2$, collapsing the state to $|a_i\rangle$.

1.3.1.4 Density Matrices and Mixed States

Mixed states are described by density operators:

$$\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|, \qquad \rho = \rho^\dagger, \;\; \rho \geq 0, \;\; \operatorname{Tr} \rho = 1. \tag{1.52}$$

A state is pure if $\operatorname{Tr}(\rho^2) = 1$; otherwise, it is mixed. Mixed states capture statistical ensembles and decoherence.

The reduced state of subsystem A is obtained by the partial trace:

$$\rho_A = \operatorname{Tr}_B(\rho_{AB}). \tag{1.53}$$

1.3.1.5 Metrics and Entanglement

Distance between states is quantified by:

$$F(\rho,\sigma) = \left(\operatorname{Tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}\right)^2 \qquad \text{(Fidelity)}, \tag{1.54}$$

$$D_{\mathrm{tr}}(\rho,\sigma) = \frac{1}{2}\operatorname{Tr}|\rho-\sigma| \qquad \text{(Trace distance)}. \tag{1.55}$$

The Schmidt decomposition expresses bipartite states as

$$|\psi\rangle_{AB} = \sum_{i=1}^{r} \lambda_i |i\rangle_A \otimes |i\rangle_B, \qquad \sum_i \lambda_i^2 = 1, \tag{1.56}$$

with entanglement entropy

$$S(\rho_A) = -\operatorname{Tr}(\rho_A \log \rho_A). \tag{1.57}$$

1.3.2 Postulates of Quantum Mechanics

The foundations of quantum theory can be summarized in six postulates [64–66]:

1. **State space:** A system is described by a normalized state $|\psi\rangle \in \mathcal{H}$.
2. **Observables:** Physical quantities correspond to Hermitian operators.
3. **Measurement:** Outcomes are eigenvalues of observables, with probabilities given by the Born rule.
4. **Collapse:** Post-measurement, the state collapses to the measured eigenstate.
5. **Time evolution:** Closed systems evolve by the Schrödinger equation $i\hbar\partial_t |\psi(t)\rangle = H|\psi(t)\rangle$.
6. **Composite systems:** The Hilbert space of joint systems is the tensor product of subsystems.

1.3.3 Quantum Computing Primitives

Quantum computing is the application of quantum mechanics to information processing. The fundamental unit is the qubit, a two-level system, which, unlike a classical bit, can exist in a superposition of two states.

Quantum Gates: In quantum computing, computation is performed by applying quantum gates, which are unitary transformations acting on one or more qubits. These gates modify the state of the system in a reversible way. Common single-qubit gates include the Pauli gates X, Y, Z, which perform rotations on the qubit state, and the Hadamard gate H, which creates superposition. A key two-qubit gate is the controlled-NOT (CNOT), which introduces entanglement between qubits. Another important gate is the T gate, a phase rotation, which is essential for achieving universality.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{1.58}$$

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad \text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \qquad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \tag{1.59}$$

The set $\{H, T, \text{CNOT}\}$ is universal: any unitary can be approximated by sequences of these gates.

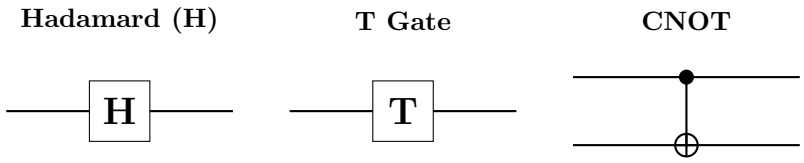


Figure 1.9: Examples of basic single- and two-qubit gates. Universal sets allow the construction of arbitrary quantum circuits.

Quantum Circuits: Quantum algorithms are typically implemented using quantum circuits, which are sequences of quantum gates applied to qubits, followed by measurements at the end to obtain classical outputs. A simple example is the preparation of a Bell state, shown in Fig. 1.8, where a Hadamard gate is applied to the first qubit, followed by a controlled-NOT (CNOT) gate to create entanglement between the qubits. The complexity of a quantum circuit is often characterized by its *width*, which is the number of qubits used, and its *depth*, which is the number of sequential layers of gates applied. These quantities are important, as they determine the resources required to run the algorithm on real quantum hardware.

1.3.4 Quantum Algorithms

Several algorithms demonstrate quantum advantages:

- **Shor's algorithm:** Shor's algorithm factors an integer N in polynomial time by reducing the problem to order-finding, solvable efficiently with the quantum Fourier transform (QFT) [2]. While classical factoring requires sub-exponential time, Shor's algorithm runs in $O((\log N)^3)$ time on a quantum computer.

- **Grover's algorithm:** Grover's algorithm provides a quadratic speedup for unstructured search problems. Given a search space of N items, it finds a marked item in $O(\sqrt{N})$ steps, compared to $O(N)$ classically [3].

- **Quantum Fourier Transform (QFT):** Computes the discrete Fourier transform over $\mathbb{Z}_N$ with a circuit depth that scales polynomially in $\log N$, specifically $\mathcal{O}((\log N)^2)$, compared to the classical Fast Fourier Transform which requires $\mathcal{O}(N \log N)$ operations. Here, $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ denotes integers modulo N.

- **HHL algorithm:** The Harrow–Hassidim–Lloyd (HHL) algorithm solves systems of linear equations $A\mathbf{x} = \mathbf{b}$ in time $O(\text{poly}(\log N))$ under certain input assumptions, an

exponential speedup over classical algorithms [28].

1.3.5 The NISQ Era and Hybrid Methods

Present devices are noisy intermediate-scale quantum (NISQ) processors with 50–1000 qubits, limited by decoherence and imperfect gates [7]. To exploit them, hybrid algorithms have been developed.

Variational Quantum Algorithms: Variational quantum algorithms combine quantum circuits with classical optimization. A parameterized quantum circuit (PQC) is used to prepare a quantum state that depends on a set of adjustable parameters θ. The goal is to find the values of these parameters that minimize a given cost function:

$$\theta^* = \arg\min_{\theta} \langle \psi(\theta) | \, C \, | \psi(\theta) \rangle \,, \tag{1.60}$$

where C represents the quantity we want to minimize, such as energy or error.

In practice, this is done using a hybrid loop: the quantum computer prepares the state and measures the cost, while a classical optimizer updates the parameters θ based on the results. This process is repeated until convergence. (Check figure 1.10)

Two important examples are:

- **Variational Quantum Eigensolver (VQE)**: estimates the lowest energy (ground state) of a quantum system [67].
- **Quantum Approximate Optimization Algorithm (QAOA)**: is designed to solve optimization problems, such as finding the best solution among many possible configurations [68].

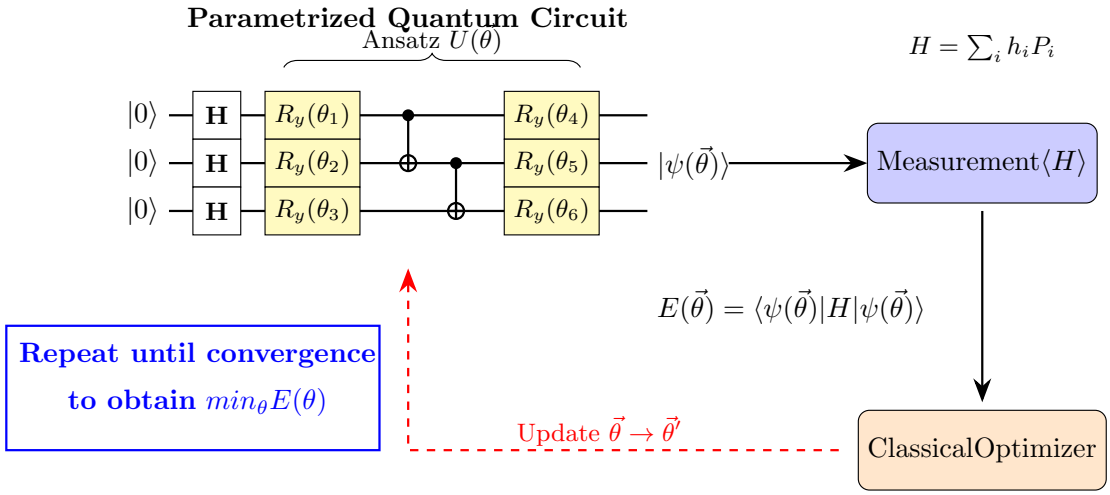


Figure 1.10: Hybrid variational loop used in the Variational Quantum Eigensolver (VQE). A parameterized quantum circuit (ansatz) $U(\boldsymbol{\theta})$ prepares the state $|\psi(\boldsymbol{\theta})\rangle$ from an initial state $|0\rangle^{\otimes n}$. The expectation value of the Hamiltonian $H = \sum_i h_i P_i$ is estimated on quantum hardware via measurements, yielding the objective function $E(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$. This value is passed to a classical optimizer, which updates the parameters $\boldsymbol{\theta}$ iteratively. The loop is repeated until convergence, typically corresponding to the minimization of $E(\boldsymbol{\theta})$, providing an approximation to the ground state energy.

1.3.6 Summary and Relevance to QML

This section established the mathematical, physical, and computational foundations of quantum mechanics and quantum computing. Concepts such as superposition, entanglement, unitary evolution, and variational hybrid algorithms directly underpin quantum machine learning. In QML, parameterized quantum circuits play the role of trainable models, with measurement defining prediction and loss evaluation. Understanding these fundamentals is thus essential before moving to the study of QML architectures and algorithms.

1.4 Quantum Machine Learning

Quantum Machine Learning (QML) is an emerging field at the intersection of quantum computing and machine learning (ML). Its goal is to exploit quantum-mechanical resources

such as superposition, entanglement, and interference to enhance learning tasks. While the field is still in its infancy, QML provides novel perspectives and potential advantages in representational power, kernel methods, and optimization landscapes [12–14].

QML methodologies can be broadly categorized as:

- **Quantum-enhanced classical ML:** classical models are accelerated by quantum subroutines (e.g., quantum linear algebra, quantum kernel estimation).
- **Quantum-inspired algorithms:** classical algorithms inspired by quantum principles (e.g., tensor network learning).
- **Fully quantum models:** parameterized quantum circuits that act as trainable neural networks or generative models.

In this thesis, the primary emphasis will be on Quantum-enhanced classical ML and fully quantum models, particularly those based on parameterized quantum circuits. Brief introductions to the other categories are provided to place this work within the broader QML landscape.

1.4.1 Data Encoding and Quantum Feature Maps

To process classical data $\mathbf{x} \in \mathbb{R}^d$, it must be embedded into a quantum state $|\phi(\mathbf{x})\rangle$. This is a crucial step, as the choice of encoding determines the effective feature space.

Encoding strategies: Common encoding strategies include:

- **Basis encoding:** Maps classical bitstrings directly to computational basis states. For a d-dimensional binary input $\mathbf{x} \in \{0, 1\}^d$, this requires d qubits, with each feature encoded into one qubit.

- **Amplitude encoding:** Normalize $\mathbf{x}$ and embed into amplitudes of a quantum state:

$$\mathbf{x} \mapsto |\phi(\mathbf{x})\rangle = \sum_{i=1}^{d} \frac{x_i}{\|\mathbf{x}\|} |i\rangle . \tag{1.61}$$

  This requires only $\mathcal{O}(\log d)$ qubits, but preparing such states can be computationally expensive.

- **Angle encoding:** Encodes each feature as a rotation angle applied to a qubit, e.g.,

$$\mathbf{x} \mapsto \bigotimes_{i=1}^{d} R_Y(x_i) |0\rangle . \tag{1.62}$$

  This method typically requires $\mathcal{O}(d)$ qubits and is efficient to implement on current quantum hardware.

While these encoding strategies provide direct ways to map classical data into quantum states, they often do not explicitly capture correlations between different features. To address this limitation, more structured approaches known as quantum feature maps have been developed. These maps extend simple encodings by incorporating entangling operations, allowing the encoding of higher-order correlations among input features in a high-dimensional Hilbert space.

A widely used example of such a feature map is the ZZ feature map, which augments angle encoding with entangling interactions between qubits. We briefly describe ZZFeatureMap below:

ZZ Feature Map

The ZZ feature map is one of the most widely used quantum feature maps in quantum machine learning, particularly in quantum kernel methods [33, 69]. It encodes classical data into a quantum state using both single-qubit rotations and entangling operations, thereby mapping data into a high-dimensional Hilbert space.

Given an input vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, the ZZ feature map prepares a quantum state of the form

$$|\phi(\mathbf{x})\rangle = U_{\mathrm{ZZ}}(\mathbf{x}) \, |0\rangle^{\otimes n}, \tag{1.63}$$

where the unitary $U_{\mathrm{ZZ}}(\mathbf{x})$ is defined as

$$U_{\mathrm{ZZ}}(\mathbf{x}) = \exp\left(i \sum_{j=1}^{n} x_j Z_j + i \sum_{j<k} \phi(x_j, x_k) Z_j Z_k \right). \tag{1.64}$$

Here, Z_j denotes the Pauli-Z operator acting on qubit j, and $\phi(x_j, x_k)$ is a nonlinear function of the input features, commonly chosen as

$$\phi(x_j, x_k) = (\pi - x_j)(\pi - x_k). \tag{1.65}$$

This structure enables the encoding of both linear and pairwise nonlinear relationships between features. The resulting quantum kernel between two inputs $\mathbf{x}$ and $\mathbf{x}'$ is given by the equation 1.23.

While the ZZ feature map can generate expressive representations, it typically requires n qubits for n features and a number of two-qubit gates that scales quadratically with the input dimension. This leads to practical limitations on near-term quantum devices due to noise and circuit depth, motivating the development of more resource-efficient feature maps.

The encoded quantum states can be used to define similarity measures between data points, forming the basis of quantum kernel methods, which will be discussed in section 1.4.5

1.4.2 Parameterized Quantum Circuits (PQCs)

Most QML models rely on parameterized quantum circuits (PQCs), also called variational circuits. A general PQC is:

$$U(\theta, \mathbf{x}) = U_{\text{enc}}(\mathbf{x})\, U_{\text{ansatz}}(\theta), \tag{1.66}$$

where U_{enc} encodes data and U_{ansatz} is a trainable circuit with parameters θ.

Given input $\mathbf{x}$, the PQC prepares:

$$|\psi(\theta, \mathbf{x})\rangle = U(\theta, \mathbf{x})\, |0\rangle^{\otimes n}. \tag{1.67}$$

Outputs are defined by expectation values of observables:

$$f_\theta(\mathbf{x}) = \langle \psi(\theta, \mathbf{x})|\, \hat{O}\, |\psi(\theta, \mathbf{x})\rangle. \tag{1.68}$$

Training involves minimizing a cost function:

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(\mathbf{x}_i), y_i), \tag{1.69}$$

with gradients estimated using the *parameter-shift rule* [32], which enables gradient evaluation on quantum hardware by expressing derivatives as finite differences of expectation values at shifted parameter values:

$$\frac{\partial}{\partial \theta_j} f_\theta(\mathbf{x}) = \frac{1}{2} \left[f_{\theta + \frac{\pi}{2} e_j}(\mathbf{x}) - f_{\theta - \frac{\pi}{2} e_j}(\mathbf{x}) \right]. \tag{1.70}$$

1.4.3 Quantum Neural Network Models

Over the past decade, several quantum neural network (QNN) architectures have been proposed, inspired by both classical deep learning and quantum circuit design. Among these, variational quantum classifiers (VQCs) have become the most widely studied approach, but several other models extend the neural network paradigm into the quantum domain.

Variational Quantum Classifiers (VQCs) use parameterized quantum circuits (PQCs) as decision functions for classification tasks [32, 70]. In this approach, input data are encoded using a quantum feature map, transformed by a variational ansatz with tunable parameters, and finally measured to produce class predictions. The parameters are trained via a classical optimizer. Please see 1.5(b).

Other Quantum Neural Networks (QNNs) generalize classical neural network principles using layered unitary operations and measurement schemes [31]. Notable examples include:

- **Coherent Feed-Forward QNNs:** mimic the structure of classical feed-forward networks, where information flows layer by layer, but the computation remains fully quantum and coherent throughout. A detailed discussion is provided in the chapter 2.
- **Quantum Convolutional Neural Networks (QCNNs)** are inspired by classical CNNs and use repeated local operations (filters) and pooling steps to extract hierarchical features from data [71].
- **Quantum Perceptrons:** are quantum versions of the classical perceptron, where quantum gates are used to implement simple decision-making units [72].

Expressibility vs. trainability: Highly expressive circuits approximate Haar-random states, but risk *barren plateaus* [30, 42], where gradient variance vanishes exponentially:

$$\mathrm{Var}[\nabla_\theta C] \sim \mathcal{O}\left(\frac{1}{2^n}\right). \tag{1.71}$$

Strategies to mitigate this include shallow circuits, problem-inspired ansätze, and local cost functions.

1.4.4 Quantum Generative Models

Generative QML models learn probability distributions:

- **Quantum GANs (QGANs):** a quantum generator trained against a discriminator [34].
- **Born Machines:** directly model probability amplitudes of quantum states [73, 74].
- **Quantum Boltzmann Machines:** inspired by energy-based models [75].

1.4.5 Quantum Kernel Methods

Kernel methods naturally integrate with QML. Schuld and Killoran [69] demonstrated that quantum computers can implement such feature maps natively: encoding a classical data point $\mathbf{x}$ into a quantum state $|\phi(\mathbf{x})\rangle$ implicitly defines a feature map into a Hilbert space. The corresponding quantum kernel is given by the inner product between quantum states:

Using this kernel, learning models such as support vector machines can be expressed as

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}, \mathbf{x}_i) = \sum_{i=1}^{N} \alpha_i \langle \phi(\mathbf{x}), \phi(\mathbf{x}_i) \rangle. \tag{1.72}$$

where $\{\mathbf{x}_i\}_{i=1}^{N}$ are training samples and α_i are learned coefficients.

A key advantage of quantum kernels lies in the ability of quantum systems to efficiently prepare and manipulate states in exponentially large Hilbert spaces. By leveraging quantum feature maps that are difficult to simulate classically, it may be possible to construct kernels that are classically intractable, thereby offering potential advantages for certain learning tasks [33].

However, the practical realization of such advantages depends critically on the choice of feature map, circuit structure, and noise resilience. In particular, issues such as exponential concentration can degrade performance, making the design of expressive yet resource-efficient quantum feature maps an active area of research. Check chapter 3 for more details.

1.4.6 Quantum Reservoir and Recurrent Models

Quantum Reservoir Computing (QRC) [39] uses fixed quantum systems as reservoirs:

$$\mathbf{h}(t) = f\left(U\mathbf{h}(t-1), \mathbf{x}(t)\right), \tag{1.73}$$

with only the classical readout layer trained. Variants include:

- **Time-multiplexed QRC:** In this case, the input time series is encoded using a sliding window over time, where each input to the reservoir consists of a sequence of consecutive time steps, e.g., $(t-n, t-n+1, \ldots, t)$. This window is then shifted forward at each iteration, allowing the reservoir to process temporal information by incorporating multiple past inputs into a single state. This approach increases the effective memory of the system without increasing the complexity.

- **Feedback-based QRC:** Here the output generated at time step $t-1$ is fed back into the system and influences the state at time step t. In most implementations, this feedback is classical: the measured output is processed and re-encoded into the

quantum system. This recurrence allows the reservoir to incorporate past information and model temporal dependencies.

Quantum analogues of recurrent networks also exist, though less explored [76].

1.4.7 Challenges and Practical Considerations

Despite rapid development, QML faces significant challenges:

Trainability (barren plateaus in QNNs): Gradients in many parameterized quantum circuits vanish exponentially with system size or depth, making stochastic optimization effectively impossible; global cost functions and highly expressive/ entangling ansätze are especially prone to this "barren plateau" issue. Noise can induce the same effect even at modest depths, compounding trainability issues on NISQ devices. More generally, the geometry of the loss landscape (via Fisher information/effective dimension) ties expressivity to gradient concentration, creating sharp trade-offs between richness and learnability.

Quantum kernels and concentration: In quantum kernels (especially fidelity quantum kernels), as the number of qubits (n) increases, the computed kernel values $\kappa(x, x')$ between different data inputs tend to concentrate around a data-independent value with exponentially vanishing variance. This phenomenon is driven by highly expressive quantum feature maps that map data points to almost orthogonal states in the vast Hilbert space. Global measurements, strong entanglement under local readout, and realistic device noise further accelerate the effect. Consequently, distinguishing these minute kernel differences demands an exponentially growing number of circuit evaluations, leading to trivial machine learning models that cannot generalize effectively to new data. This fundamental limitation poses a major challenge to achieving a practical quantum advantage in kernel-based QML at scale [43].

Thanasilp *et al.* [43] showed that projected (local) kernels can mitigate concentration by reducing the effective dimension of the feature space and preserving larger kernel variance, although they also exhibit exponential concentration for sufficiently expressive or highly entangled feature maps. Complementary to this, local and multi-scale kernel constructions mitigate concentration by avoiding reliance on a single global overlap and instead aggregating subsystem similarities across different scales [77]. Carefully designed feature maps can also slow down this effect (see Chapter 3).

Hardware noise and circuit depth: Current NISQ processors have short coherence times, imperfect (especially two-qubit) gates, readout errors, crosstalk, and sparse connectivity. As circuits deepen and require additional routing, these errors accumulate and rapidly degrade state fidelity and the expectation values of measured observables [7]. For QML, this yields suppressed or unstable gradients in variational training and corrupted overlap estimates in kernel methods (often pushing kernels toward ill-conditioning and concentration). So shot requirements and bias–variance trade-offs can blow up before useful learning signals emerge [43]. Although error mitigation can reduce bias on small instances, recent results show that the samples required to recover noiseless expectation values grow super-polynomially in the worst case, even at shallow depths. This implies improved accuracy cannot be achieved generically without incurring exponential resources elsewhere [78, 79].

Data encoding: In most practical QML pipelines the inputs are classical, so one must first embed them into quantum states—an operation whose cost can dominate the workflow and error budget. Preparing arbitrary amplitude-encoded states generally requires a number of gates that grows exponentially with qubit count, while

widely used angle-encoding feature maps (often paired with entangling layers) introduce nontrivial two-qubit overhead that scales at least quadratically in the number of features/qubits, with qubit count growing linearly in features [33, 80]. Beyond resource cost, the embedding sets the model's inductive bias. this means if the data distribution is misaligned with the chosen feature map, kernels can exponentially concentrate toward a constant, and the model becomes information-starved even with polynomial shots [43, 81]. "Data re-uploading" (iterative injection of features through the circuit) increases expressivity but typically at the price of greater depth and parameter count [35, 82].

Finding the right problem to solve: To date, there is no widely accepted real-world deployment in which a QML pipeline achieves superior end-to-end performance or a better cost–accuracy trade-off than strong classical baselines. Large, controlled studies generally find classical wins when compute and hyperparameters are matched [83]. The gap stems from many factors including expensive and fragile NISQ resources, fundamental limits on error-mitigation scalability, trainability pathologies in variational models, and exponential concentration that collapses signal in many quantum kernels [7, 43, 78, 84]. Even embeddings motivated by conjectured hardness (e.g., IQP-style feature maps) encounter noise and variance issues in practice, while tuned classical methods remain highly competitive on standard datasets [33, 83].

1.4.8 Summary

QML integrates quantum resources into ML workflows through data encoding, variational PQCs, kernel estimation, neural network analogues, generative models, and reservoir com-

puting. While no *practical* revolution has yet been realized, QML has established foundational frameworks that may, in the long term, lead to advantages in areas such as chemistry, material discovery, and optimization. This thesis builds upon these principles to develop resource-efficient QML algorithms closer to real-world deployment.

1.5 Limits of Proving General Quantum Advantage in Machine Learning

While several quantum algorithms promise significant speedups over classical counterparts in certain machine learning (ML) tasks [26, 81, 85], it is fundamentally impossible to prove a general quantum advantage for all ML tasks, independent of the dataset or learning setting. This limitation arises not from a shortcoming of quantum computing, but from the intrinsic nature of machine learning itself.

The No-Free-Lunch Theorem. The most direct formal barrier is the No-Free-Lunch (NFL) theorem [8, 60], which states that when averaged over all possible data-generating functions, all learning algorithms perform equally. That is, without any assumptions about the structure of the data distribution, no algorithm—quantum or classical—can outperform another on average. Thus, any claim of advantage must be conditional on specific properties of the data.

Learnability is Undecidable. Even determining whether a function class is learnable in the PAC (Probably Approximately Correct) framework can be undecidable in general [86]. This implies that we cannot always determine whether learning is possible without access to the underlying data distribution or structural priors.

Dependence on Structural Assumptions. Most provable quantum speedups in ML rely on specific assumptions: access to quantum memory (qRAM) [26], well-conditioned and sparse matrices [85], or hardness assumptions from classical complexity theory [81, 87]. In their absence, quantum models are subject to the same theoretical limitations as classical models.

Uncomputability of Data Complexity. Relatedly, Kolmogorov complexity—a formal measure of the information content or structure of data—is known to be uncomputable [88]. This further prohibits any universal assessment of the learnability or compressibility of datasets, reinforcing the impossibility of general performance guarantees.

Therefore, both classical and quantum ML can only claim task-specific advantages under explicitly stated assumptions. Proving a universal quantum advantage in ML, independent of data and model assumptions, is not possible within current theoretical frameworks.

2

Coherent Feed-Forward Quantum Neural Network

Quantum neural networks (QNNs) represent one of the most actively explored paradigms in quantum machine learning (QML), but most existing approaches rely on variational circuits or quantum feature maps with multiple layers of entanglement. While these constructions have provided useful testbeds on noisy intermediate-scale quantum (NISQ) devices, they deviate significantly from the structure of classical feed-forward neural networks (FFNNs). More importantly, their circuit depth and qubit requirements scale poorly with the number of features, limiting applicability to realistic machine learning tasks.

In this work, we introduce the *Coherent Feed-Forward Quantum Neural Network (CFF-QNN)*, a bona fide quantum neural network architecture that retains coherence across all

layers without intermediate measurements. Our model mirrors the versatility of a classical FFNN by supporting adjustable hidden layers and nodes, while significantly reducing resource overhead. Specifically, the CFF-QNN achieves over a 50% reduction in circuit depth and CNOT count compared to state-of-the-art QNN models, while maintaining a qubit requirement that is independent of the data's feature dimension.

We benchmark our approach on practical datasets, including the Wisconsin breast cancer diagnostic dataset and the credit card fraud detection dataset. The CFF-QNN achieves accuracies of 91% and 85% respectively, outperforming conventional QNN methods by 5–10% while requiring fewer quantum resources. These results highlight the potential of incorporating structural ideas from classical neural networks into quantum computing, bridging a critical gap between theoretical proposals and resource-efficient, real-world QML applications.

Notably, this research has also led to an international patent filing [89], underscoring the originality of the CFF-QNN architecture and its potential for real-world deployment in quantum machine learning pipelines.

This paper represents the first core contribution of my doctoral research, establishing a coherent and scalable quantum neural network model. The subsequent chapters build on this foundation by addressing complementary directions: resource-efficient quantum kernels and quantum reservoir computing, both of which extend the theme of making QML more practical for deployment on near-term and future quantum devices.

Published Article

The following is the published version of this work:

- **Reference:** Utkarsh Singh, Aaron Z. Goldberg, and Khabat Heshami. "Coherent Feed-Forward Quantum Neural Network." *Quantum Machine Intelligence*, 6, 47 (2024). DOI: 10.1007/s42484-024-00222-8

Quantum Machine Intelligence (2024) 6:89
https://doi.org/10.1007/s42484-024-00222-8

RESEARCH ARTICLE

Coherent feed-forward quantum neural network

Utkarsh Singh[1,2] · **Aaron Z. Goldberg**[1,2] · **Khabat Heshami**[1,2,3]

Received: 26 January 2024 / Accepted: 6 November 2024 / Published online: 3 December 2024

Abstract

Quantum machine learning, focusing on quantum neural networks (QNNs), remains a vastly uncharted field of study. Current QNN models primarily employ variational circuits on an ansatz or a quantum feature map, often requiring multiple entanglement layers. This methodology not only increases the computational cost of the circuit beyond what is practical on near-term quantum devices but also misleadingly labels these models as neural networks, given their divergence from the structure of a typical feed-forward neural network (FFNN). Moreover, the circuit depth and qubit needs of these models scale poorly with the number of data features, resulting in an efficiency challenge for real-world machine learning tasks. We introduce a *bona fide* QNN model, which seamlessly aligns with the versatility of a traditional FFNN in terms of its adaptable intermediate layers and nodes, absent from intermediate measurements such that our entire model is coherent. This model stands out with its reduced circuit depth and number of requisite CNOT gates, achieving a more than 50% reduction in both compared to prevailing QNN models. Furthermore, the qubit count in our model remains unaffected by the data's feature quantity. We test our proposed model on various benchmarking datasets such as the breast cancer diagnostic (Wisconsin) and credit card fraud detection datasets. Our model achieved an accuracy of 91% on the breast cancer dataset and 85% on the credit card fraud detection dataset, outperforming existing QNN methods by 5–10% while requiring approximately 50% fewer quantum resources. These results showcase the advantageous efficacy of our approach, paving the way for the application of quantum neural networks to relevant real-world machine learning problems.

Keywords Quantum machine learning · Quantum neural networks · Quantum computation · Coherent feed forward

1 Introduction

Over the past decade, quantum machine learning (QML) (Schuld and Petruccione 2021) has emerged as a dynamic field with promising potential for advancing machine learning techniques using quantum computing, especially because quantum machines may efficiently explore large-dimensional spaces for processing large amounts of data (Nielsen and Chuang 2011; Biamonte et al. 2017; Schuld et al. 2014; Zoufal et al. 2021; Havlicek et al. 2019; Garcia et al. 2022). While initial research focused primarily on adapting standard machine learning algorithms to quantum computing, such as quantum neural networks (QNNs) (Diep 2020; Chalumuri et al. 2021; Chen and Niu 2020; Tacchino et al. 2021; Wang et al. 2019; Li et al. 2020), quantum generative adversarial networks (QGANs) (Situ et al. 2018; Zoufal et al. 2019), and quantum support vector machines (Havlicek et al. 2019; Wu et al. 2021), progress has been hindered by the lack of a clear path to scalability and practical applications of these methods. Instead, researchers have focused on developing algorithms suitable for the current generation of noisy intermediate-scale quantum (NISQ) devices (Wan et al. 2017; Beer et al. 2020; Bondarenko and Feldmann 2020; Cong et al. 2019; Li et al. 2020; Sharma et al. 2022; Zhou et al. 2023; Bharti et al. 2022), resulting in a new wave of algorithms known as NISQ-era QML algorithms (Preskill 2018).

These recent QML algorithms are predicated upon low-depth parameterized quantum circuits (Sim et al. (2019); Zhang et al. 2023), which take a hybrid approach that combines the strengths of quantum processors with classical

Khabat Heshami
khabat.heshami@nrc-cnrc.gc.ca

[1] National Research Council of Canada, 100 Sussex Drive, Ottawa, Ontario K1N 5A2, Canada

[2] Department of Physics, University of Ottawa, 25 Templeton Street, Ottawa, Ontario K1N 6N5, Canada

[3] Institute for Quantum Science and Technology, Department of Physics and Astronomy, University of Calgary, 25 Templeton Street, Calgary, Alberta T2N 1N4, Canada

processors. This hybridization allows for the development of novel algorithms that have shown some advantages in specific use cases (Abbas et al. 2021; Biamonte et al. 2017; Jiang et al. 2021; Yamasaki et al. 2023), although a useful quantum advantage remains to be seen.

Most QML models currently available employ complex encoding techniques, known as quantum feature maps, and use parameterized quantum circuits as models (Garcia et al. 2022; Havlicek et al. 2019) in place of the intermediate layers of a neural network. Sometimes, these circuits are concatenated such that the measurement of one circuit provides nonlinearity when inputting data into the subsequent circuit (Tacchino et al. 2020). In the hybrid approach, post-processing steps are similar to those used in classical machine learning (ML), updating the parametrized quantum circuits using techniques such as gradient descent, and are performed on a classical computer. Nonetheless, these feature mapping techniques often stumble when faced with real-world datasets, as the requisite number of qubits and the circuit depth escalate with the number of features in the data.

In this work, we propose a novel circuit-based approach that incorporates entanglement layers for the connections between nodes in adjacent layers, resembling a classical feed-forward neural network (FFNN). This approach offers the advantage of adaptable intermediate layers, allowing us to adjust them according to the characteristics of the data, which is particularly significant for classification tasks. Further, because all of the hidden layers can be incorporated without intermediate measurements, our approach is fully coherent throughout the evolution of the circuit and can take advantage of quantum coherence properties throughout, which is responsible for quantum advantages in related settings (Gyurik et al. 2023). Finally, by developing a data encoding scheme inspired by classical neural networks that writes multiple features onto a small number of qubits, our overall use of quantum resources is amenable to quantum computers available today.

To evaluate the effectiveness of our model, we conduct numerical experiments using the credit card fraud detection and Wisconsin breast cancer diagnostic datasets, employing the Qiskit package for simulations of quantum circuits, and compare the results to state-of-the-art QNN models. The results of these experiments are presented and show that our approach achieves significant improvements in both accuracy and computational efficiency over traditional QNN methods.

Our results highlight the potential of integrating classical neural network concepts into quantum computing frameworks, opening avenues for more sophisticated, resource-efficient quantum models in the future. As we continue to explore and refine our model that we dub the coherent feed-forward quantum neural network (CFFQNN), we anticipate its adaptability to a broader range of applications, further bridging the gap between quantum computing and real-world machine learning challenges.

1.1 Artificial neural network

The functioning of the brain inspires the idea of an artificial neural network (ANN). The brain receives and processes information via a network of neurons, where each neuron receives inputs from a number of neurons, processes them, and produces an output that is then input to subsequent neurons. In an ANN, perceptrons or nodes are used to mimic biological neurons: each is linked to others with variable weights, and the structure of the connections between nodes and their weights can then be used to process data; one artificial neuron is shown in Fig. 1.

Perceptrons were introduced by Rosenblatt (1957) as binary classifiers that form the foundational concept behind artificial neural networks and deep learning. A perceptron takes multiple input values X_i and produces a single binary-outcome output y. Each input is associated with a weight W_i, which signifies the importance of that input. The perceptron computes a weighted sum of its inputs with an overall bias b via

$$z = \sum_i W_i X_i + b \tag{1}$$

and passes this sum through an activation function, typically a step function, to produce its output y:

$$y = \sigma(z). \tag{2}$$

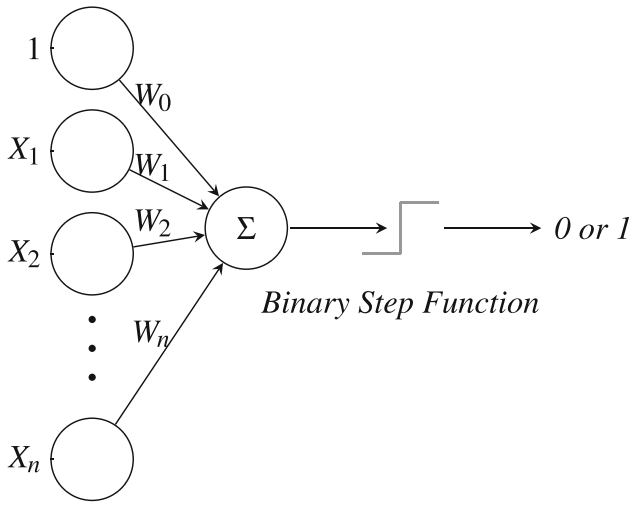


Fig. 1 A perceptron, inspired by neural networks in the brain. Inputs $\{X_i\}$ are combined with weights $\{W_i\}$ including a bias W_0 that are processed nonlinearly to produce a binary output

Here, to make it a differentiable non-linear activation function, σ is typically taken to be a logistic sigmoid or rectified linear unit function. If the weighted sum exceeds a certain threshold, the perceptron outputs a 1 (or "active"); otherwise, it outputs a 0 (or "inactive").

The perceptron's strength lies in its simplicity, adept at modelling linearly separable data, but it falters with non-linear data. This spurred the evolution of multi-layer perceptrons (MLPs) or neural networks capable of handling complex, non-linear patterns. ANNs optimize weights in each layer using methods like backpropagation and gradient descent, allowing them to approximate any function with high accuracy (Cybenko 1989; Hornik 1991). FFNNs, a key type of ANN, allow unidirectional information flow and have shown impressive performance in tasks like classification and regression. Any exemplary FFNN will be shown below in Fig. 5a.

1.2 Quantum neural network

Quantum neural networks are among the most popular algorithms in QML. Even though the field is not fully developed, there is a rapidly growing set of people exploring potential quantum advantages (Sagingalieva et al. 2023; Bondarenko and Feldmann 2020; Jiang et al. 2021; Farhi and Neven 2018).

The momentum behind recent advancements in this domain can largely be attributed to the variational techniques prevalent in numerous hybrid algorithms (Garcia et al. 2022). In general, one starts by encoding the classical data $\mathbf{X} \in \mathbb{R}^N$ onto the quantum state using a feature map $U(\mathbf{X})$ that can be implemented by a quantum circuit (Asfaw et al. 2020). Next, a variational circuit $U_{var}(\boldsymbol{\theta})$ is applied with intelligently chosen single-qubit rotations $R(\theta_i)$ and entanglement layers on an input state $U(\mathbf{X})|0\rangle^{\otimes n}$, where the θ_i parameters are the trainable weights and n is the number of qubits employed and usually scales linearly with N. Finally, the expectation value of some observable (e.g., $\hat{Z}^{\otimes n}$) is measured for the classical post-processing to predict the class $\tilde{y}$ of the input data. Optimizing the weight parameters present in the variational circuit is done using some classical optimizers, eventually minimizing the cost function $C(y(\mathbf{X}, \boldsymbol{\theta}), \tilde{y})$ in consideration (Garcia et al. 2022). A schematic diagram of a typical QNN is shown in Fig. 2.

While this approach of QNNs demonstrates promising outcomes in certain applications, it faces substantial challenges in scalability and gate complexity, particularly on NISQ devices (Cerezo et al. 2022; Beer et al. 2020). The number of qubits required in QNNs tends to increase linearly with the number of data features, quickly exceeding the limited qubit capacity of current quantum hardware and thus restricting their applicability to smaller datasets, instead of making use of the exponential scaling of Hilbert space dimension with number of qubits. Additionally, the number of (CNOT) gates, crucial for creating entanglements in quantum circuits, scales nearly quadratically with the number of features. This scaling is problematic on NISQ devices due to increased error rates and circuit depth, leading to higher likelihoods of decoherence and computational inefficiency. The combination of these scalability issues with the gate complexity challenges significantly hinders the practical implementation of QNNs on existing quantum platforms, making the handling of complex, feature-rich datasets a formidable task and posing a significant bottleneck in fully leveraging quantum computing for advanced machine learning applications.

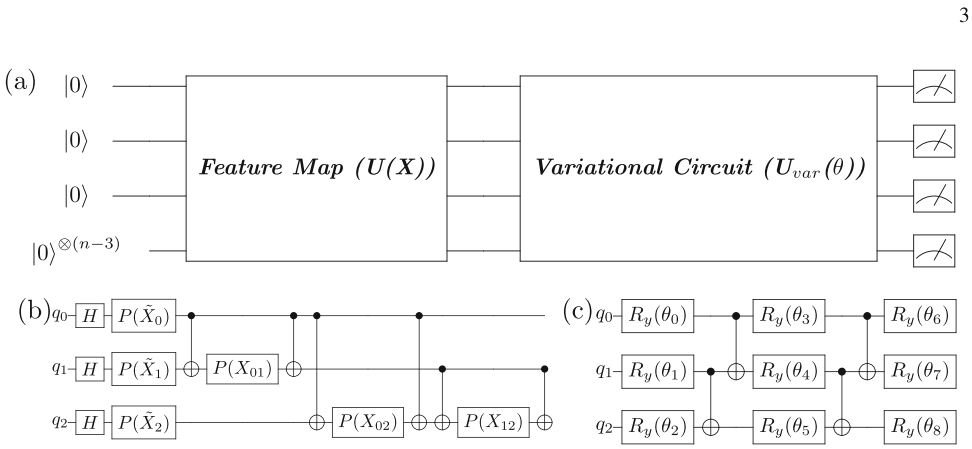


Fig. 2 **a** Architecture of a QNN acting on n qubits; the data $\mathbf{X}$ are loaded with a feature map $U(\mathbf{X})$, and the data are processed using a parametrized circuit $U_{var}(\theta)$. Subsequent measurement allows the parameters θ to be optimized and updated. These steps may be repeated. **b** A 3-qubit feature map circuit administering the commonly used ZZFeatureMap. Here, H represents the Hadamard gate, P represents the phase gate, $\tilde{X}_i = 2X_i$, and $X_{ij} = 2(\pi - X_i)(\pi - X_j)$. **c** A 3-qubit variational circuit with weight parameters $\{\theta_j\}$ explicit

2 Results

2.1 The model: coherent feed-forward quantum neural network

Here, we introduce our CFFQNN model that uses a quantum-classical hybrid approach to process data. The source code for all of our work can be found on GitHub as detailed below. The initial encoding layer is similar to the first hidden layer of a conventional ANN, as illustrated in Fig. 3, with the classical data loaded onto quantum states. Subsequent layers consist of a network single-qubit and controlled-rotation (entangling) gates, all of which are parametrized by their rotation angles, exemplified in Fig. 4. Ultimately, the qubits undergo measurement. Notably, the parameterized controlled rotations are adaptable to cater to specific network demands, and the measurement process can be tailored based on both the data and the desired structural outcome. This circuitry inherently mirrors the architecture of an ANN, as can be seen by comparing Fig. 5 a versus b.

We elect to perform all rotations about the y-axis for reasons that will become clear shortly. These are mathematically described by the single-qubit rotation gate expressed in the single-qubit computational basis $\{|0\rangle, |1\rangle\}$ as

$$R_y(\theta) = \exp(-\mathrm{i}\theta\sigma_y/2) = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

using the Pauli matrix σ_y. Sequential rotations about the same axis commute and act additively as $R_y(\theta_1)R_y(\theta_2) = R_y(\theta_1+\theta_2)$.

In the first layer, the data points with weights are encoded as the rotation angle of the R_y gate acting on some initial state, where the latter is taken to be some general state $R_y(b)|0\rangle$. All of the data points are successively encoded onto the same qubit, which is schematized in Fig. 3. Since all of the rotations are about the same axis, such an encoding can be performed with a single single-qubit gate parametrized by $\theta = z = \sum_{i=1}^{N} X_i W_i + b$:

$$R_y(X_N W_N)\cdots R_y(X_2 W_2)R_y(X_1 W_1)[R_y(b)|0\rangle] = R_y(z)|0\rangle. \tag{3}$$

This is the first efficiency resulting from all of the rotations being about the same axis, which reduces the number of data

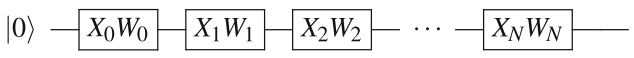


Fig. 3 The depiction of data encoding stage. Rotation gates with angles X_0W_0 act on a single qubit in analogy with inputs acting on a single neuron. An extra rotation gate with $X_0 = 1$ and $W_0 = b$ is added for flexibility to bias the initial qubit

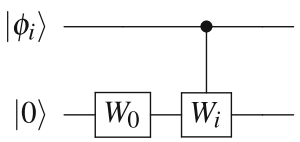


Fig. 4 An illustration of a single intermediate node in CFFQNN and how the weight values are applied from one node to the next using controlled rotations by angle W_i, with a possible single-qubit rotation by bias angle W_0

encoding gates, and also is responsible for better mimicking an ANN by directly encoding the variable z without giving a preference to the *ordering* among nodes within a given layer.

This data encoding procedure can be repeated on multiple distinct qubits to allow for more non-linear processing of the data in the quantum circuit. Therefore, a CFFQNN with n qubits in the first layer can be described as

$$|\Psi\rangle = [R(z)|0\rangle]^{\otimes n}. \tag{4}$$

Such single gates are exemplified in the "1st hidden layer" segment of Fig. 5b.

Since n is, in principle, independent from N, the size of the circuit need not to depend on the number of data features, unlike the case for a traditional QNN.

In the subsequent layers corresponding to hidden layers of an ANN, we initialize new qubits by applying a parametrized rotation gate as a biasing term $R_{y;\,j}(\theta_{0j})$ acting on the jth qubit, and then we apply parametrized controlled-rotation gates that are controlled by the qubits in the first layer. For example, if we want to connect the first node in the first layer to the first node in the second layer, we apply a rotation on the $(n+1)$th qubit controlled by the state of the *1*st qubit:

$$CR_y^{1\mapsto n+1}(\theta) = |0\rangle_1\langle 0| \otimes \mathbb{I}_{n+1} + |1\rangle_1\langle 1| \otimes R_{y;\,n+1}(\theta). \tag{5}$$

The other nodes are similarly connected by controlled-rotation gates $CR_y^{i\mapsto j}(\theta_{ij})$, each parametrized by independent weights θ_{ij}. Even though the rotations are controlled by different control qubits in different states, all of the rotations on a new qubit commute, such that the aggregate effect on a node in a hidden layer is independent from any ordering among the nodes in the previous layer.

Figure 4 displays a single intermediate node in the CFFQNN's hidden layer. It illustrates how the node applies weight values (denoted as W) to these nodes depending on the value of the last layer nodes. After this operation, the state $|\phi_i\rangle \otimes |0\rangle$ changes to a new state $|\psi_{ij}\rangle = CR_y^{i\mapsto j}(\theta_{ij})|\phi_i\rangle \otimes |0\rangle$ that may continue to be acted upon by other control layers.

While a classical ANN uses a perceptron to decide whether or not a certain node should forward its output to the next one, the CFFQNN retains the branches of the wavefunction associated with the control qubit being in each of states $|0\rangle$ and $|1\rangle$ without collapsing the state via measurement. For

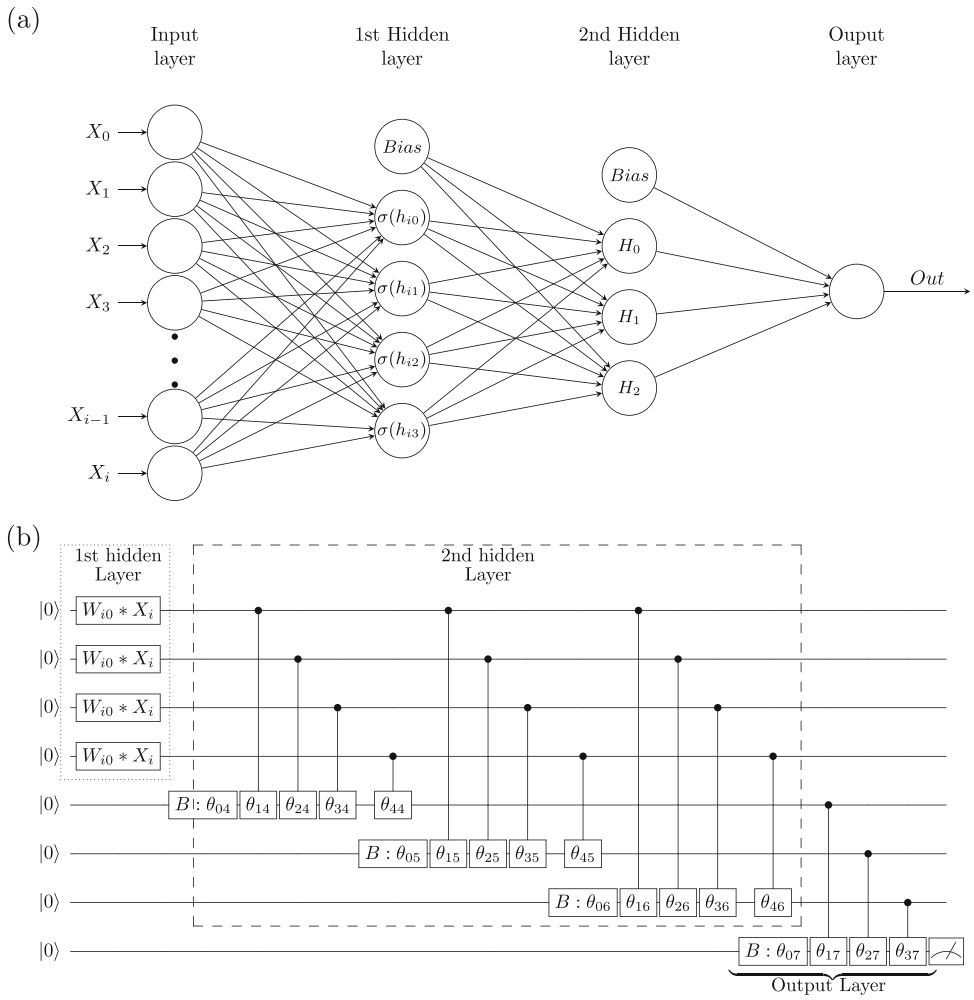


Fig. 5 **a** Architecture of an artificial neural network with two hidden layers. Here, W represents the weight parameters, $\mathbf{X}$ are data points, σ is a non-linear activation function, and $h_{ij} = W_{ij} X_i$. **b** Architecture of a CFFQNN with two hidden layers where $\mathbf{X}$ are data points, and $\mathbf{W}$ and $\boldsymbol{\theta}$ represent the weight parameters. The number of modes in a layer of the ANN corresponds to the number of qubits in a layer of the CFFQNN. In contrast to earlier QNN models such as those in Fig. 2c, the parameters of the CFFQNN change the controlled operations such that the CFFQNN circuits are not solely parametrized by their single-qubit gates; this is what allows the CFFQNN to resemble an ANN

example, the branch of the state where all of the control qubits are in state $|0\rangle$ only applies the bias rotation to the target qubits, while the branch where all of the control qubits are in state $|1\rangle$ applies the gate $R_{y;\,j}(\sum_{0=1}^{n} \theta_{ij})$ to qubit j. Overall, the action takes the form

$$U = \prod_{ij} CR_y^{i\mapsto j}(\theta_{ij}) = \sum_{\mathbf{X}} |\mathbf{X}\rangle\langle\mathbf{X}| \otimes \bigotimes_j R_{y;\,j}\left(\sum_{i=0}^{n} X_i \theta_{ij}\right). \tag{6}$$

Here, $\mathbf{X} = X_1, \cdots, X_n$ is a bit string with elements $X_i \in \{0, 1\}$, the sum runs over all such strings, and we have included $X_0 = 1$ to represent the bias term; the tensor product over j implies that the rotation on the jth qubit associated with the branch of the wavefunction where the qubits are in state $|\mathbf{X}\rangle$ is by an angle $\sum_{i=0}^{n} X_i \theta_{ij}$, corresponding to the standard factor in ANNs. This structure is schematized in the "2nd hidden layer" segment of Fig. 5b, where the connections between control and target qubits are explicit and their weights are given accordingly. All output values of the perceptron are essentially kept coherently, and the system is ready to have the process repeated in a subsequent layer.

Put another way, we have replaced the non-linear activation function σ in a standard perceptron by the controlled operations CR_y. Unlike σ, the output of CR_y is not deterministic; it is probabilistic. Nevertheless, if we measure one of the control qubits, we know we will find state $|0\rangle$ with probability $p(0) = \cos^2 \frac{z}{2}$ times and state $|1\rangle$ with the rest of the probability $p(1) = \sin^2 \frac{z}{2}$, depending on the value of $z = \sum_{i=1}^{n} X_i W_i + b$. We can turn such probabilities into binary outputs by simply choosing the larger of the two, which are split at the value $\alpha = \pi/2$:

$$\tilde{y} = \begin{cases} 1, & p(1) > p(0) \iff z > \alpha \\ 0, & p(1) < p(0) \iff z < \alpha \end{cases}. \tag{7}$$

In this sense, we have created a coherent QNN that retains all of the properties of an ANN while allowing for data to be processed without each perceptron being restricted to a binary output.

After repeating the process for multiple layers, the number of controlled (entangling) gates is given by the number of connections between the layers. This is at most a quadratic function of the number of nodes per layer and a linear function in the depth of the neural network, which are expected to grow with the number of features in the data but do not follow a fixed relationship. One can thus create a few-qubit quantum neural network with $n \ll N$ and verify empirically its success for a given machine learning tasks.

Finally, after all of the intermediate (hidden) layers, we perform a measurement and calculate the expectation value of some operator, which we further use to decide the class of the data point as in a classical NN. In our work, we often measure the value of the final (Nth) qubit $\langle Z_N \rangle$ as depicted in the "output layer" segment of Fig. 5b. The measurement result is fed into a classical non-linear activation function σ to produce the binary outcome

$$\tilde{y} = \begin{cases} 1, & \sigma(\langle Z_N \rangle) = 1 \\ 0, & \sigma(\langle Z_N \rangle) = 0 \end{cases}. \tag{8}$$

More general outcomes can be considered by either dividing the ranges of expectation values $\langle Z_N \rangle$ into more than two segments or by measuring more final qubits to yield more possible final outcomes. The results of such outcomes can be used to update the encoding weights W_i and intermediate weights θ_{ij} throughout the network in an iterative fashion.

This method can create a complete FFNN like a standard classical one without doing intermediate measurements. An overall schematic diagram of a CFFQNN with two hidden layers with four and three qubits respectively is shown in Fig. 5b.

2.1.1 Different variations of the model

In our work, we deploy two distinct versions of the CFFQNN model: the standard CFFQNN and a variant that we dub FixedCFFQNN. While FixedCFFQNN retains the architectural design of the CFFQNN, it diverges in one key aspect: the weights in its initial layer remain untrained. This reduction in the number of parameters to be trained significantly speeds the training process while still outperforming previous QNNs.

2.1.2 Hyperparameters of the CFFQNN model

The architecture of CFFQNN closely mirrors that of an ANN, sharing many of the same hyperparameters. This allows for customization in terms of the number of layers and nodes within each layer. Additionally, the measurement scheme can be tailored to fit specific data needs; even the parameter α in Eq. (8) is a hyperparameter. For instance, throughout our research, we employed a single measurement strategy for CFFQNN, measuring only the final qubit. For the FixedCFFQNN, we adopted a partial measurement approach, targeting all qubits except the qubits in the initial layer.

2.2 Numerical results

We evaluate the CFFQNN's performance against the prevailing QNN model across various datasets. For the conventional QNN model, we employ the ZZFeatureMap (contributors 2023) to encode classical data into the quantum circuit and the RealAmplitudes (contributors 2023) circuit as the variational circuit with the trainable weights. We use the COBYLA (Pedregosa et al. 2011) optimizer to optimize the weights (θs). Figure 2 illustrates a 3-qubit ZZFeatureMap circuit alongside a RealAmplitudes circuit with two repetition layers. Additionally, to provide a comprehensive performance perspective, we compared the efficacy of all quantum models against the classical MLPClassifier-an FFNN.

To predict the input data's output class, we predominantly employed the Statevector simulator from Qiskit (contributors 2023; Asfaw et al. 2020), designed to emulate the ideal quantum states of a quantum system without any external noise or decoherence. This simulator offers an exact depiction of the quantum state, facilitating precise calculations and forecasts. It proves especially valuable for theoretical investigations and grasping the optimal performance of quantum algorithms.

2.3 Data and metrics

We evaluated the efficacy of our quantum machine learning model using the credit card fraud detection and breast cancer diagnostic datasets. These datasets serve as standard benchmarks for testing and comparing different machine learning models. Details about these models can be found in Section 4.

2.4 Results on credit card dataset

For this study, we used the PCA to reduce the dimension of credit card dataset to seven features. For both CFFQNN and FixedCFFQNN, we utilized a network structure with three nodes in the initial layer, two in the subsequent layer, and a single node in the concluding layer.

Figure 6a presents a performance comparison between CFFQNN, FixedCFFQNN, QNN, and the classical MLPClassifier. The bar chart clearly demonstrates the superior performance of both CFFQNN variants over the conventional QNN model. Despite augmenting the QNN with 35

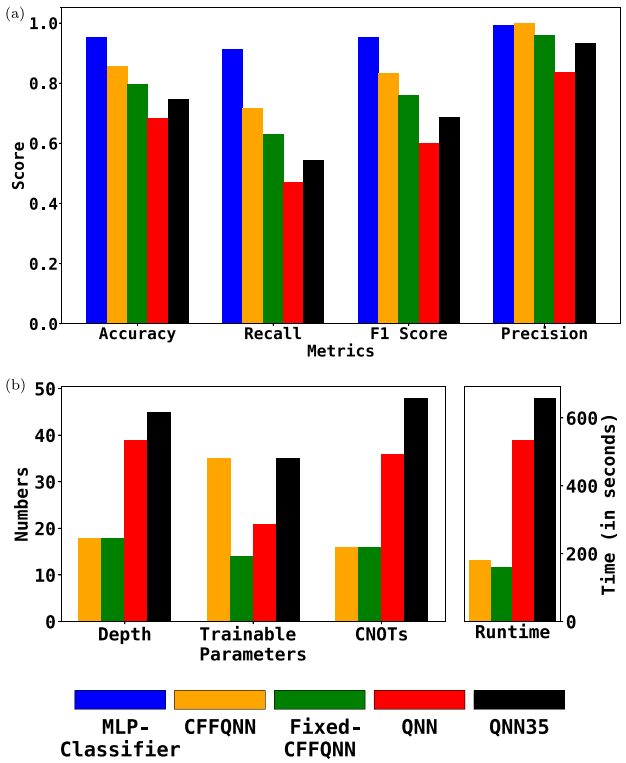


Fig. 6 **a** Performance comparison of various models on the credit card fraud detection dataset, highlighting metrics such as accuracy, recall, F1 score, and precision. Higher scores are better, showcasing the advantage of the CFFQNN over QNNs. **b** Comparative analysis of resource utilization for QNN and CFFQNN models on the same dataset. The depth, number of trainable parameters, and CNOTs are indicated by the vertical axis on the left, while the runtime is indicated by the vertical axis on the right. Note that the bars from left to right represent the MLP-Clasifier, CFFQNN, FixedCFFQNN, QNN, and QNN35, respectively. The resource utilization bar for the MLPClassifier is omitted in **b** since it is not a quantum model. Lower resource utilization is better, again demonstrating the advantages of the CFFQNN

parameters for a balanced comparison, there was no notable enhancement in its results.

Figure 6b outlines the quantum resources utilized by each quantum model. Evidently, the CFFQNN models require a reduced circuit depth, fewer CNOT gates, and a shorter simulator runtime compared to the QNN model. We specify the number of CNOT gates and circuit depth as resources, instead of the overall circuit complexity, because the former are fixed by the framework of the QNN alone while the latter depends on the actual values that the weight parameters inside the network take, which are varied during training. Nonetheless, we expect a close correlation between CNOT gates, circuit depth, and overall circuit complexity due to the directional nature of our QNN, where later qubits only interact with earlier qubits in one layer. Additionally, the FixedCFFQNN model exhibits a reduced requirement on the number of parameters.

2.5 Results on breast cancer dataset

For the breast cancer dataset, we narrowed down the feature space to 7 features and executed 100 iterations of COBYLA to enhance the quantum models. We adopted a network structure similar to that used for the credit card dataset in our proposed quantum models.

In Fig. 7a, it is evident that both CFFQNN and FixedCFFQNN outperform the existing QNN models. The QNN's recall, F1, and precision scores are all zero, indicating a failure of the model to effectively learn from the data, resulting in the misclassification of all test data into a single category.

Figure 7b provides a comparative analysis of the resources consumed by each of the four quantum models. While the resource utilization is largely consistent across all models, it is noteworthy that the QNN models demand a significantly higher number of CNOT gates compared to the CFFQNN variants.

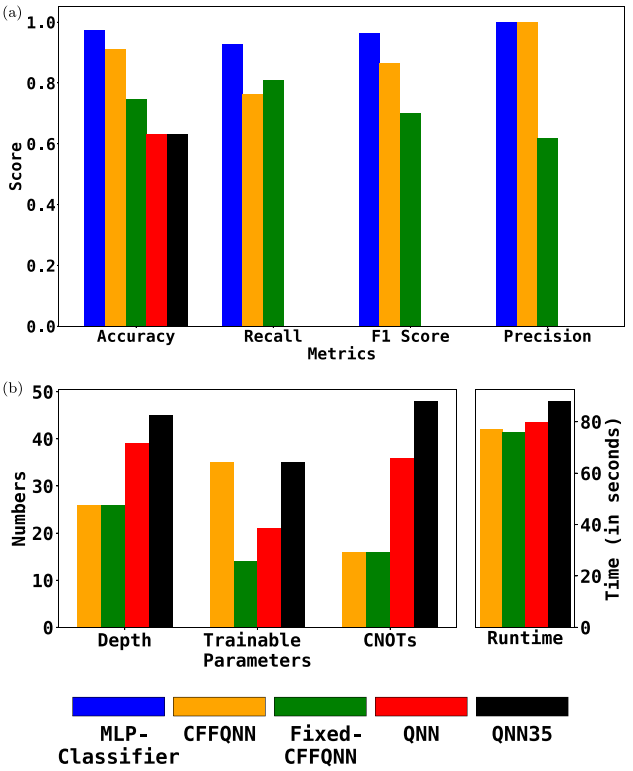


Fig. 7 **a** Performance comparison of various models on the breast cancer diagnostic dataset, highlighting metrics such as accuracy, recall, F1 score, and precision. Higher scores are better, showcasing the advantage of the CFFQNN over QNNs. **b** Comparative analysis of resource utilization for QNN and CFFQNN models on the same dataset. The depth, number of trainable parameters, and CNOTs are indicated by the vertical axis on the left, while the runtime is indicated by the vertical axis on the right. Note that the bars from left to right represent the MLP-Clasifier, CFFQNN, FixedCFFQNN, QNN, and QNN35, respectively. The resource utilization bar for the MLPClassifier is omitted in **b** since it is not a quantum model. Lower resource utilization is better, again demonstrating the advantages of the CFFQNN

2.6 Results on other datasets

To verify that the CFFQNN did not merely succeed by chance, we performed further analyses that are described in Appendix 1. For a further classification problem related to diagnosing heart disease, the CFFQNN outperforms existing QNNs by over 10 percentage points in accuracy while using half or fewer CNOT gates. For a regression task related to student performance, the mean square error of our CFFQNN is slightly better than that of existing QNNs, while our fixed CFFQNN beats both by a factor of almost 20. These display that our advantages can be broadly applicable to many tasks that involve neural networks.

3 Discussion and conclusion

Our approach and results demonstrate a step forward in quantum machine learning through the introduction of the CFFQNN. Our CFFQNN is the direct quantum upgrade of a classical ANN, with all possible outputs of each classical perceptron upgraded to a branch of the quantum state that exists simultaneously and interacts with all other branches. The advantages of the CFFQNN over previous QNNs stem from its unique integration of the $\sum_i X_i W_i + b$ term, inspired by classical neural networks, into the initial layer within a higher-dimensional Hilbert space. This approach, combined with the strategic incorporation of quantum entanglements in subsequent layers, marks a significant advancement over traditional QNN models.

The CFFQNN inherits the flexibility and empirical nature of classical ANNs. Depending on the size of the data and its particular features, different numbers of hidden layers and nodes therein are optimal for training without overfitting (see Appendix 2 for an analysis of the non-linear impact of changing numbers of layers on model performance). Just as there is no hard-and-fast rule for specifying the optimal size of a classical ANN, our model should be tuned for the task at hand and has been successful for all of the tasks described here.

A pivotal advantage of our model is that its qubit count and number of CNOT gates in its circuit are independent from the number of features in the dataset. This independence is particularly advantageous in dealing with complex, realistic datasets with varying feature sizes, ensuring scalability and flexibility. Furthermore, the model's design allows for direct adjustments in intermediate layers, a feature that enables it to effectively support deep network architectures, a limitation in many existing QNN models. These low resource counts make the CFFQNN suitable for quantum hardwares that exist around the world, and we look forward to an experimental demonstration of the CFFQNN in the near term.

Through numerical experimentation, we have demonstrated the superior performance of CFFQNN in classification tasks, including a notable variant: the FixedCFFQNN. In the latter approach, we did not train any parameters in the first layer, yet it still outperformed existing QNN models. This untrained variant underscores the inherent efficiency and robustness of the CFFQNN architecture. Achieving high accuracy while requiring minimal quantum resources, both the standard and Fixed CFFQNN variants are significant steps toward the practical use of quantum machine learning. Compared to QNN models utilizing the ZZFeature Map, the CFFQNN exhibits not only enhanced performance but also a more efficient utilization of quantum resources. These characteristics of our model will persuade realization on various quantum computing hardware as initial steps toward a scalable implementation for the practical application of quantum machine learning.

4 Methods

We create an instance of the CFFQNN to perform data classification on two standard datasets. The CFFQNN model is simulated using Qiskit, while the datasets employed are accessible through Kaggle. We here detail parameters and techniques used in creating, training, and evaluating the model.

First, we select the maximum number of qubits to be used in our network as seven: this is small enough to be simulable on a standard personal computer yet large enough to exhibit genuinely quantum features such as a large Hilbert space spanned by 128 elements. The major question is whether this is a sufficient number of qubits to perform nontrivial machine learning tasks. Standard QNNs require one qubit per feature in the dataset, so we limit our investigation to data with 7 features. In comparison, the CFFQNN can handle more features with the same total number of qubits, so we note that the number of features is limited by the QNNs against which we seek to compare the CFFQNN, not by the CFFQNN itself.

Next, we take real-world datasets and reduce their feature spaces to make the data size suitable for simulation, noting that many-qubit quantum systems are inherently challenging to simulate and that therein lies potential quantum advantages. The Credit Card Fraud detection dataset has 30 features with which one seeks to evaluate whether a given transaction was fraudulent or not, a binary classification problem, while the Breast Cancer Diagnostic dataset's 30 features are used to evaluate whether a given patient does or does not have breast cancer, another binary classification. Since some of the features may be highly correlated with each other or may contribute less to the overall variance in the data distribution, a linear transformation of the coordinates in the

feature space can elicit the principal components, which are the new coordinate axes that account for most of the independent information contained in the features and allow one to neglect axes where the data change less. Such principal component analysis (PCA) is standard in data processing and here reduces both 30-feature datasets to seven principal features each. These details are summarized in Table 1.

In the case of the Credit Card dataset, we also addressed the issue of class imbalance. To ensure unbiased training and evaluation, we eliminated the excess class instances, balancing the dataset. This step enabled our model to learn from both the minority and majority classes more effectively, thereby enhancing its ability to detect fraudulent transactions accurately. By employing these preprocessing techniques on the selected datasets, we aimed to create a robust and reliable framework for evaluating the effectiveness of our quantum machine learning model.

The standard QNN is programmed as follows. Every qubit is initialized in the superposition state $(|0\rangle + |1\rangle)/\sqrt{2}$ by means of a Hadamard transformation, and then the data features are encoded using a phase gate

$$P(\bar{X}_i) = |0\rangle\langle 0| + \mathrm{e}^{\mathrm{i}\bar{X}_i}|1\rangle\langle 1| \tag{9}$$

acting on the ith qubit; this is the ZFeatureMap with $\bar{X}_i = 2X_i$ and is the first stage of the ZZFeatureMap.

The ZZFeatureMap then continues to sequentially entangle the $i$the and jth qubits and again upload the same data onto the quantum state, using the sequence of gates:

$$G_{ij} = \mathrm{CNOT}^{i\mapsto j}[\mathbb{I}_i \otimes P_j(X_{ij})]\mathrm{CNOT}^{i\mapsto j} \tag{10}$$

which uses the gate $\mathrm{CNOT}^{i\mapsto j} = (|0\rangle\langle 0| \otimes \mathbb{I} + |1\rangle\langle 1| \otimes \sigma_x)$ and the non-linear function of the parameters $X_{ij} = 2(\pi - X_i)(\pi - X_j)$. All of the qubits are pairwise entangled using a sequence of G_{ij} operators for various i and j. However, the relationship between the number of CNOT gates and the number of qubits is not fixed in a strict mathematical sense, but rather it depends on the specific architecture of the ZZFeatureMAp quantum circuit and the requirements of the algorithm being implemented. Here, we used the circuit with *full* entanglement option, which requires $\frac{N(N-1)}{2}$ CNOT gates to fully entangle all pairs of qubits for single repetition of the circuit.

After all of the features are uploaded, the next step of the QNN is a parametrized quantum circuit. A single repetition of this consists of at least $2N$ single-qubit rotation gates $R_y(\theta_i)$, N acting on each qubit, separated by fixed entangling gates. Each qubit experiences one parametrized R_y gate, then a sequence of entangling gates $\prod_{i=1}^{i-1}\mathrm{CNOT}^{N-1\mapsto N}\cdots\mathrm{CNOT}^{2\mapsto 3}\mathrm{CNOT}^{1\mapsto 2}$ is applied, and then the process is repeated in alternating fashion and ends with parametrized single-qubit rotation gates for a total of $2(N-1)$ controlled operations in the parametrized circuit. All the qubits are then measured in the computational basis, and the measurement result is processed in the same way as for the CFFQNN detailed below.

In comparison, the data may be encoded into any number of qubits for the CFFQNN, with more qubits being required for subsequent manipulations that correspond to hidden layers of ANNs. Just like in classical machine learning, there is no *a priori* method for determining how many layers and how many nodes in each layer will be required for the success of training the network for a particular dataset. We choose to encode our datasets' seven features into three qubits, corresponding to the first hidden layer, process them with a second hidden layer comprised of two qubits, and then funnel the quantum information into a final qubit such that the total number of qubits is only six.

The three qubits have the same data redundantly uploaded into them using no entangling operations: the operator $R_y(\sum_{i=1}^{N} W_i X_i + b)$ is applied to each of the first three qubits as in the main text. To process the data and forward it to the second hidden layer, a controlled operation is required between each pair of qubits from the first and second layers, such that 12 operations of the form $CR_y^{i\mapsto j}(\theta_{ij})$ with unique parameterized weights θ_{ij} are applied. A single-qubit rotation corresponding to a biasing term is also applied to each qubit in the second layer. Finally, three controlled operations $CR_y^{j\mapsto N}(\theta_{jN})$ are performed between the qubits in the second hidden layer and the final qubit along with a biasing term on the final qubit, for a total of 16 controlled operations. The final qubit is measured in the computational basis.

For both setups, the single output parameter $\langle Z_N\rangle$ is fed into a classical non-linear function and used to classify the input data. At least 70% of the available data points are used, and the models are scored on how well they correctly predict the classification of those data. The parameterized circuits

Table 1 Properties of the datasets used to evaluate the CFFQNN and compare it to existing neural networks

Datasets	Features	Features used	Training size	Testing size	Labels
Credit card fraud detection (balanced)	30	7	688	296	2
Breast cancer diagnostic (Wisconsin)	30	7	455	114	2

Table 2 Metrics used to evaluate and compare the performances of different neural networks on the same datasets

Metrics	Equations
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
Accuracy	$\frac{TP+TN}{TP+FP+FN+TN}$
F1 score	$\frac{2\times Precision\times Recall}{Precision+Recall}$

are then updated with new parameters obtained from the COBYLA (Pedregosa et al. 2011) optimizer, and this process is repeated iteratively until the parameters converge. Those parameters are then used to test the models' performances on the test data points that they have not seen before. The overall models are then evaluated using the numbers of true positive (TP), false positive (FP), true negative (TN), and false negative (FN) results. These can be combined into a number of metrics as in Table 2. All of these metrics range from zero to one, with larger numbers implying superior models.

All of the source code for creating and comparing these models is detailed in a GitHub repository and will be available upon request.

Appendix 1. Additional results

This section provides additional performance results comparing our proposed quantum neural network (QNN) model with an existing QNN model that uses a ZZFeatureMap and a RealAmplitude circuit.

Heart disease dataset

The Heart Disease dataset, sourced from Kaggle (LAPP 2024), contains 918 instances with 13 features used to predict the presence of heart disease. To enhance the model's performance and due to limited resources for running quantum simulations, principal component analysis (PCA) was applied to reduce the number of features to 7, capturing the most significant variance in the data.

The existing QNN model achieved an accuracy of 65.76%. In contrast, our proposed CFFQNN model demonstrated superior performance with an accuracy of 78.80% when using a layer structure of [2,3,1]. Even with a simpler layer structure of [1,1,1], our model achieved an accuracy of 76.09%.

In terms of circuit complexity, the existing QNN model has a circuit depth of 39 and utilizes 36 CNOT gates. Our proposed model, with a layer structure of [2,3,1], achieved a reduced circuit depth of 26 with 18 CNOT gates. Furthermore, with a layer structure of [1,1,1], our model significantly reduced the circuit depth to 14 and required only 4 CNOT gates.

These results highlight the enhanced efficiency and effectiveness of our CFFQNN model, demonstrating both higher accuracy and reduced circuit complexity compared to the existing model.

Student performance dataset (linear regression)

We further evaluated our proposed quantum neural network models using the student performance dataset from Kaggle (Narayan 2024), focusing on regression tasks. The mean square error (MSE) values for different models are as follows: our proposed CFFQNN achieved an MSE of 0.1869, the fixed CFFQNN achieved a significantly lower MSE of 0.0102, and the existing QNN model resulted in an MSE of 0.1982. These results demonstrate the superior performance of the fixed CFFQNN model in regression tasks, indicating its ability to generalize better compared to both the CFFQNN and the existing QNN models.

In terms of computational resources, the depth, number of parameters, and number of CNOT gates for each model are as follows: the CFFQNN has a depth of 20, 21 parameters, and 12 CNOT gates; the fixed CFFQNN has a similar depth of 20 but with only 11 parameters and 12 CNOT gates; and the existing QNN model has a greater depth of 31, with 15 parameters and 24 CNOT gates. The resource efficiency of the fixed CFFQNN, with fewer parameters and a reduced number of CNOT gates while maintaining a lower circuit depth, highlights its computational advantage over the existing QNN model.

Appendix 2. Impact of model depth

Resource limitations

In our study, we utilized a dataset with only 7 features due to resource constraints. To conduct a thorough analysis of the CFFQNN model's depth, it is necessary to use larger datasets, which require computational resources that we currently do not have access to. Training deeper models on such datasets demands significant computational power and memory. Moreover, even if we were able to perform such an analysis, it is not guaranteed that increasing the model depth would result in improved performance. This is a phenomenon observed in classical neural network structures as well.

Non-trivial relationship between depth and performance

The relationship between model depth and performance is complex and not necessarily linear. Increasing the number of layers in a neural network does not always result in improved performance. For instance, He et al. (2016) introduced the

concept of residual networks (ResNets), demonstrating that simply adding more layers can lead to training degradation due to the vanishing gradient problem. Similarly, Li et al. (2016) highlighted that deeper models can suffer from overfitting, and their performance may plateau or even degrade beyond a certain depth.

In our specific model, we observed similar trends. For example, our proposed QNN model with a layer structure of [2,3,1] achieved the highest accuracy of 0.7880 on the Heart disease dataset. Interestingly, even a simpler structure of [1,1,1] achieved a notable accuracy of 0.7609. However, increasing the complexity to structures like [3,1,1] and [3,2,1] resulted in lower accuracies of 0.5978 and 0.6359, respectively. The detailed performance of various layer structures is summarized in Table 3.

These results highlight the non-linear relationship between depth and performance, illustrating that increased model depth does not necessarily equate to better performance. Additionally, in our specific model, performance is influenced by initial points and the type of measurement used. These factors further complicate the relationship between depth and performance.

Practical approaches to determining model depth

In the absence of a fixed methodology to determine the optimal number of layers, it is common practice in neural network design to rely on heuristic and empirical approaches. Practitioners often start with their experience and intuition when designing a model, iterating through different configurations and systematically testing various hyperparameters to find an optimal setup. Goodfellow et al. (2016) suggest that the architecture of deep learning models often involves iterative experimentation and validation to find a suitable balance between model complexity and performance. This iterative process involves starting with a baseline model and gradually increasing complexity while monitoring validation performance. These methodologies are valid for our CFFQNN model as well, where initial experimentation and validation play a crucial role in finding the optimal depth and configuration.

Table 3 Results for heart disease dataset for different layers

Layers	Accuracy
[1, 1, 1]	0.760
[1, 2, 1]	0.722
[1, 3, 1]	0.679
[1, 4, 1]	0.766
[2, 1, 1]	0.750
[2, 2, 1]	0.760
[2, 3, 1]	**0.788**
[3, 1, 1]	0.597
[3, 2, 1]	0.635
[3, 3, 1]	0.750

Acknowledgements The authors acknowledge that the NRC headquarters is located on the traditional unceded territory of the Algonquin Anishinaabe and Mohawk people. The authors would like to acknowledge the use of IBM Quantum services for this work and in particular the Qiskit package (contributors 2023). The authors thank Farid Ghobadi and Barry Sanders for insightful discussions. AZG acknowledges support from NSERC's postdoctoral fellowship. KH acknowledges support from NSERC's Discovery Grant program.

Author Contribution US conceived the model and developed numerical tests for performance analysis and comparison. All authors reviewed the results and developed the overall description and conclusions of the proposed model. All authors contributed to the writing and editing of the manuscript.

Funding Open access funding provided by National Research Council Canada library.

Data Availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of Interest The authors declare no competing interests.

References

Abbas A, Sutter D, Zoufal C, Lucchi A, Figalli A, Woerner S (2021) The power of quantum neural networks Nat. Comput Sci 1:403. https://doi.org/10.1038/s43588-021-00084-1

Asfaw A, Bello L, Ben-Haim Y, Bravyi S, Capelluto L, Vazquez AC, Ceroni J, Chen R, Frisch A, Gambetta J, Garion S, Gil L, Gonzalez SDLP, Harkins F, Imamichi T, McKay D, Mezzacapo A, Minev Z, Movassagh R, Nannicini G, Nation P, Phan A, Pistoia M, Rattew A, Schaefer J, Shabani J, Smolin J, Temme K, Tod M, Wood S (2020) Learn quantum computation using Qiskit. https://qiskit.org/textbook/preface.html

Beer K, Bondarenko D, Farrelly T, Osborne TJ, Salzmann R, Scheiermann D, Wolf R (2020) Training deep quantum neural networks Nat. Commun 11:1. https://doi.org/10.1038/s41467-020-14454-2

Bharti K, Cervera-Lierta A, Kyaw TH, Haug T, Alperin-Lea S, Anand A, Degroote M, Heimonen H, Kottmann JS, Menke T, Mok W-K, Sim S, Kwek L-C, Aspuru-Guzik A (2022) Noisy intermediate-scale quantum algorithms. Rev Mod Phys 94:015004. https://doi.org/10.1103/RevModPhys.94.015004

Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S (2017) Quantum machine learning. Nature 549:195–202. https://doi.org/10.1038/nature23474

Bondarenko D, Feldmann P (2020) Quantum autoencoders to denoise quantum data Phys. Rev Lett 124:130502. https://doi.org/10.1103/PhysRevLett.124.130502

Cerezo M, Verdon G, Huang H-Y, Cincio L, Coles PJ (2022) Challenges and opportunities in quantum machine learning. Nat Comput Sci 2:567. https://doi.org/10.1038/s43588-022-00311-3

Chalumuri A, Kune R, Manoj BS (2021) A hybrid classical-quantum approach for multi-class classification. Quantum Inf Process 20:1. https://doi.org/10.1007/s11128-021-03029-9

Chen B-Q, Niu X-F (2020) Quantum neural network with improved quantum learning algorithm. Int J Theor Phys 59:1978. 10.1007/s10773-020-04470-9

Cong I, Choi S, Lukin MD (2019) Quantum convolutional neural networks Nat. Phys 15:1273. https://doi.org/10.1038/s41567-019-0648-8

contributors Q (2023) Qiskit: an open-source framework for quantum computing. https://doi.org/10.5281/zenodo.2573505

Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Systems 2:303. https://doi.org/10.1007/BF02551274

Diep DN (2020) Some quantum neural networks. Int J Theor Phys 59:1179. https://doi.org/10.1007/s10773-020-04397-1

Farhi E, Neven H (2018) Classification with quantum neural networks on near term processors. Quantum Phys

García DP, Cruz-Benito J, García-Peñalvo FJ (2022) Systematic literature review: quantum machine learning and its applications. https://doi.org/10.48550/arXiv.2201.04093

Goodfellow I, Bengio Y, Courville A (2016) Deep learning (publisher MIT Press). http://www.deeplearningbook.org

Gyurik C, Molteni R, Dunjko V (2023) Limitations of measure-first protocols in quantum machine learning. arXiv:2311.12618 [quant-ph]

Havlíček V, Córcoles AD, Temme K, Harrow AW, Kandala A, Chow JM, Gambetta JM (2019) Supervised learning with quantum-enhanced feature spaces. Nature 567:209. https://doi.org/10.1038/s41586-019-0980-2

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: IEEE Conference on computer vision and pattern recognition (CVPR) (IEEE) pp 27–30. https://doi.org/10.1109/CVPR.2016.90

Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4:251. https://doi.org/10.1016/0893-6080(91)90009-T

Jiang W, Xiong J, Shi Y (2021) A co-design framework of neural networks and quantum circuits towards quantum advantage. Nat Commun 12:1. https://doi.org/10.1038/s41467-020-20729-5

LAPP D (2024) Heart Disease Dataset. Accessed 11 Jul 2024. https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

Li L, Jamieson K, DeSalvo G, Rostamizadeh A, Talwalkar A (2016) Hyperband: a novel bandit-based approach to hyperparameter optimization. https://doi.org/10.48550/arXiv.1603.06560

Li Y, Zhou R-G, Xu R, Luo J, Hu W (2020) A quantum deep convolutional neural network for image recognition. Quantum Sci Technol 5:044003. https://doi.org/10.1088/2058-9565/ab9f93

Narayan N (2024) Student performance (multiple linear regression). Accessed 11 Jul 2024. https://www.kaggle.com/datasets/nikhil7280/student-performance-multiple-linear-regression/data

Nielsen MA, Chuang IL (2011) Quantum computation and quantum information: 10th Anniversary Edn 10th ed. (Cambridge University Press USA)

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in python. J Mach Learn Res 12:2825

Preskill J (2018) Quantum Computing in the NISQ era and beyond Quantum 2:79. arXiv:1801.00862v3. https://doi.org/10.22331/q-2018-08-06-79

Rosenblatt F (1957) The perceptron, a perceiving and recognizing automaton Project Para, Report: Cornell Aeronautical Laboratory (Cornell Aeronautical Laboratory). https://books.google.ca/books?id=P_XGPgAACAAJ

Sagingalieva A, Kordzanganeh M, Kenbayev N, Kosichkina D, Tomashuk T, Melnikov A (2023) Hybrid quantum neural network for drug response prediction. Cancers 15. https://doi.org/10.3390/cancers15102705

Schuld M, Petruccione F (2021) Machine learning with quantum computers. (Springer International Publishing. Cham, Switzerland)

Schuld M, Sinayskiy I, Petruccione F (2014) An introduction to quantum machine learning. Contemp Phys 56:172. https://doi.org/10.1080/00107514.2014.964942

Sharma K, Cerezo M, CincioP L, Coles J (2022) Trainability of dissipative perceptron-based quantum neural networks Phys. Rev Lett 128:180505. https://doi.org/10.1103/PhysRevLett.128.180505

Sim S, Johnson PD, Aspuru-Guzik A (2019) Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms Adv. Quantum Technol 2:1900070. https://doi.org/10.1002/qute.201900070

Situ H, He Z, Wang Y, Li L, Zheng S (2018) Quantum generative adversarial network for generating discrete distribution. https://doi.org/10.1016/j.ins.2020.05.127. arXiv:1807.01235

Tacchino F, Mangini S, Kl P, Barkoutsos C, Macchiavello D, Gerace I, Tavernelli D. Bajoni (2021) Variational learning for quantum artificial neural networks. IEEE Trans Quantum Eng 2:1. https://doi.org/10.1109/TQE.2021.3062494

Tacchino F, Barkoutsos P, Macchiavello C, Tavernelli I, Gerace D, Bajoni D (2020) Quantum implementation of an artificial feed-forward neural network. Quantum Sci Technol 5:044010. https://doi.org/10.1088/2058-9565/abb8e4

Wan KH, Dahlsten O, Kristjánsson H, Gardner R, Kim MS (2017) Quantum generalisation of feedforward neural networks npj Quantum Inf 3:1. https://doi.org/10.1038/s41534-017-0032-4

Wang J, Chen Y, Chakraborty R, Yu SX (2019) Orthogonal convolutional neural networks. https://doi.org/10.48550/arXiv.1911.12207

Wu SL, Sun S, Guan W, Zhou C, Chan J, Cheng CL, Pham T, Qian Y, Wang AZ, Zhang R, Livny M, Glick J, KlBarkoutsos P, Woerner S, Tavernelli I, Carminati F, Di Meglio A, Li ACY, Lykken J, Spentzouris P, Chen SY-C, Yoo S, Wei T-C (2021) Application of quantum machine learning using the quantum kernel algorithm on high energy physics analysis at the LHC. https://doi.org/10.1103/PhysRevResearch.3.033221. arXiv:2104.05059

Yamasaki H, Isogai N, Murao M (2023) Advantage of quantum machine learning from general computational advantages. https://doi.org/10.48550/arXiv.2312.03057

Zhang S-X, Allcock J, Wan Z-Q, Liu S, Sun J, Yu H, Yang X-H, Qiu J, Ye Z, Chen Y-Q, Lee C-K, Zheng Y-C, K.Jian S, Yao H, Hsieh C-Y, Zhang S (2023) TensorCircuit: a quantum software framework for the NISQ era. Quantum 7:912. arXiv:2205.10091v2. https://doi.org/10.22331/q-2023-02-02-912

Zhou M-G, Liu Z-P, Yin H-L, Li C-L, Xu T-K, Chen Z-B (2023) Quantum neural network for quantum neural computing. Research 6. https://doi.org/10.34133/research.0134

Zoufal C, Lucchi A, Woerner S (2019) Quantum generative adversarial networks for learning and loading random distributions. npj Quantum Inf 5:1. https://doi.org/10.1038/s41534-019-0223-2

Zoufal C, Lucchi A, Woerner S (2021) Variational quantum Boltzmann machines. Quantum Machine Intelligence 3. https://doi.org/10.1007/s42484-020-00033-7

3

A Resource-Efficient Quantum Kernel

The second contribution of my doctoral research centers on quantum kernel methods, which have emerged as a promising approach within quantum machine learning by enabling non-linear classification and regression through data embeddings in high-dimensional Hilbert spaces. A key ingredient in these methods is the quantum feature map, which encodes classical data into quantum states so that inner products in Hilbert space can be evaluated efficiently on quantum hardware. Among the most widely used constructions is the ZZFeatureMap, which provides a straightforward foundation for kernel-based learning. However, despite their promise, such conventional feature maps suffer from critical drawbacks: they require a number of qubits that grows linearly with the number of input features and

a number of entangling gates that scales quadratically. On current noisy intermediate-scale quantum (NISQ) devices, where both qubit counts and gate fidelities are limited, these scaling properties quickly exceed feasible hardware limits, severely constraining the practical applicability of quantum kernel methods in real-world machine learning tasks.

To address this, we introduce the *CPMap*, a resource-efficient quantum feature map that addresses these scalability challenges. The CPMap reduces the qubit requirement by at least a factor of two and decreases the number of entangling gates to linear in the number of qubits, enabling implementation within the noise and connectivity constraints of available quantum processors. By embedding this feature map into a quantum kernel framework, we demonstrate improved performance on benchmark classification tasks relative to the widely used ZZFeatureMap, while maintaining parity or outperforming strong classical baselines.

Our results, obtained from both noisy simulations and small-scale experiments on IBM superconducting quantum hardware, highlight that the CPMap kernel not only improves classification accuracy but also reduces resource consumption by more than half. This demonstrates that carefully designed feature maps can make quantum kernel methods viable on near-term devices, offering one of the most promising avenues for realizing early quantum advantage in machine learning.

This research also resulted in a patent filing [90], underscoring the novelty of the CPMap approach and its potential value for practical quantum machine learning applications. Within the broader thesis, this chapter sets the stage for the third paper on quantum reservoir computing, where resource-efficient ideas are extended to temporal and dynamical learning tasks.

Preprint Reference

The following is the preprint version of this work:

Reference: Utkarsh Singh, Jean-Frédéric Laprade, Aaron Z. Goldberg, and Khabat Heshami. "CPMap: A Resource-Efficient Quantum Kernel." arXiv:2507.03689 (2025). https://arxiv.org/abs/2507.03689

A Resource Efficient Quantum Kernel

Utkarsh Singh,[1,2] Jean-Frédéric Laprade,[3] Aaron Z. Goldberg,[1] and Khabat Heshami[1,2,4]

[1]*National Research Council of Canada, 100 Sussex Drive, Ottawa, Ontario K1N 5A2, Canada*
[2]*Department of Physics, University of Ottawa, 25 Templeton Street, Ottawa, Ontario, K1N 6N5 Canada*
[3]*Institut quantique, Université de Sherbrooke Sherbrooke, QC J1K 2R1, Canada*
[4]*Institute for Quantum Science and Technology, Department of Physics and Astronomy, University of Calgary, Alberta T2N 1N4, Canada*

Quantum processors may enhance machine learning by mapping high-dimensional data onto quantum systems for processing. Conventional feature maps for encoding data onto a quantum circuit are impractical, as the number of entangling gates scales quadratically with the dimension of the dataset. We introduce a quantum feature map designed to handle high-dimensional data with a significantly reduced number of qubits and entangling operations. Our approach preserves essential data characteristics while promoting computational efficiency, evidenced by extensive experiments on benchmark datasets that demonstrate a marked improvement in both accuracy and resource utilization when using our feature map as a kernel for characterization, as compared to state-of-the-art quantum feature maps. Our noisy simulation results highlight our map's ability to function within the constraints of noisy intermediate-scale quantum devices. Through numerical simulations and small-scale implementation on a superconducting circuit quantum computing platform, we demonstrate that our scheme performs on par or better than a set of classical algorithms for classification. Our approach empirically delays the onset of exponential concentration relative to existing feature maps under the diagnostics considered here, which can improve the practical operating regime of fidelity kernels on near-term devices. Our results indicate that resource-efficient feature maps can broaden the range of kernel-based QML experiments that are feasible on near-term platforms, and motivate further validation on larger datasets and additional hardware.

I. INTRODUCTION

High-dimensional data is prevalent in modern machine learning tasks, including image and speech recognition, natural language processing, and medical diagnostics. While classical machine learning techniques can handle these high-dimensional problems, they often require substantial computational resources, particularly as the dimensionality of the data increases [1–5]. Quantum computing has emerged as a promising avenue to address these challenges. Quantum kernel methods, for instance, have shown potential for accelerating data analysis by efficiently learning relationships in high-dimensional spaces encoded into quantum states [6–11]. To harness large-dimensional Hilbert spaces for processing data, quantum computers need to efficiently encode classical data onto quantum states. These quantum feature maps are required for quantum support vector classification [12–16] and quantum neural networks [12, 13, 16–18], including data reuploading [19], and are germane to essentially all quantum machine learning paradigms. Yet, the most popular feature maps require numbers of qubits and controlled-not (CNOT) gates linear and quadratic in the number of data features, respectively, severely shortening their practical application. We introduce a feature map whose qubit requirement is reduced by a factor of at least two and whose CNOT-gate requirement is reduced to linear in the number of qubits, and show these improvements to be sufficient for practical application on current devices.

Quantum feature maps [14] translate classical data into a quantum form that can be manipulated using quantum circuits, enabling the application of quantum algorithms [13]. The qubit and entangling gate requirements for conventional feature maps pose a significant limitation for practical applications, particularly in era of noisy intermediate-scale quantum (NISQ) devices, where the number of qubits is limited and gate errors are prohibitively large [21]. The development of efficient quantum feature maps that can effectively encode high-dimensional classical data onto quantum states while minimizing the number of qubits remains an active area of research, with several proposed embedding schemes under exploration [6, 11, 22].

Here, after introducing our quantum feature map with the reduced resource requirements, we examine its performance using a kernel method in classification of various datasets. We compare our results with several classical methods and the commonly used Pauli feature map known as the ZZ feature map and consistently show competing performance.We emphasize that ZZFeatureMap is used here as a widely available reference implementation of a Pauli-style entangling feature map, rather than as the only relevant baseline. [6, 26, 40–49] Many alternative embeddings exist (in-

cluding data re-uploading and other hardware-efficient constructions), and to avoid over-interpreting the comparison as "CPMap vs. ZZ", we additionally benchmark CPMap against a suite of fixed (non-trained) data re-uploading feature maps in Appendix E. We present a detailed background of quantum feature maps and their challenges in Section I A. In Section II A we introduce our approach. We lay out our experimental methodology and results in Section II B, including small-scale tests on IBM's ibm_quebec and IBM_torino machines. Our results stimulate considerations of quantum machine learning as an early practical application of application-starved NISQ devices. In our work, we utilize this feature map within a predefined quantum kernel function known as a the fidelity quantum kernel for the support vector classifier (SVC) algorithm in Qiskit [16, 23]; as a result, we will refer to both the feature map and the kernel as CPMap.

A. Feature Maps

Quantum feature maps (QFMs) play a pivotal role in quantum machine learning, enabling the encoding of classical data into the quantum state space. This encoding process transforms classical vectors into the amplitudes of quantum states, thus facilitating quantum processing. In this section, we delve into the mathematical foundations of QFMs and explore their applications and limitations.

1. *Mathematical Definition*

A quantum feature map is defined as a function $\Phi : \mathbb{R}^n \to \mathscr{H}$, where $\mathscr{H}$ is the Hilbert space of a quantum system. Given a classical vector $\mathbf{x} \in \mathbb{R}^n$, the map Φ transforms $\mathbf{x}$ into a quantum state:

$$\Phi(\mathbf{x}) = U_\Phi(\mathbf{x})|0\rangle = \sum_{i=0}^{2^n-1} f_i(\mathbf{x})|i\rangle, \tag{1}$$

where $f_i(\mathbf{x})$ are QFM-dependent functions of the classical data and $\{|i\rangle\}$ represents the orthonormal set of computational basis states. The most common class of feature maps comprises the Pauli feature maps.

2. *Pauli Feature Maps*

The Pauli feature map for encoding classical data into the state space of a quantum system is formalized as

$$U_{PFM}(\mathbf{x}) = \exp\left(i \sum_{S\subseteq T} \phi_S(\mathbf{x}) \prod_{i\in S} P_i\right) H^{\otimes n}. \tag{2}$$

Here, S indexes subsets of qubits, T encompasses all such subsets, and each P_i represents one of the Pauli matrices $\{I, X, Y, Z\}$ acting on the qubit labeled by i. The factor $H^{\otimes n}$ applies a Hadamard gate H on each of the n qubits to transform computational basis states into superpositions thereof. The function ϕ_S is defined as the data-mapping function, with $\phi_S(\mathbf{x}) = x_i$ when S is a singleton and $\phi_S(\mathbf{x}) = \prod_{j\in S}(\pi - x_j)$ otherwise, thus capturing both individual feature impacts and the impacts of higher-order interactions between features within the quantum state [6, 24]; in this sense the scalar functions $\phi(\mathbf{x})$ are components of the mapping function Φ.

Among the commonly used Pauli feature maps is the *Pauli-Z Feature Map* (also referred to as the ZFeatureMap), where each subset S is a singleton and each P_i corresponds to a Pauli-Z operation. This map encodes classical data by applying phase shifts relative to each data feature x_i, such that each qubit in the circuit experiences a phase rotation as dictated by the respective feature value. The corresponding transformation is expressed as

$$U_Z(\mathbf{x}) = \bigotimes_i e^{-ix_i Z} H, \tag{3}$$

where the tensor product runs over all qubits. The result is a quantum state separable between all qubits where for each qubit a data feature is encoded in the relative phase between computational basis states [6, 16].

A second widely adopted feature map is the *Pauli-ZZ Feature Map* (ZZFeatureMap), which extends the Pauli-Z map by incorporating interactions between qubits. This is achieved through alternating single-qubit Z-rotations and two-qubit ZZ-entangling gates. The latter take the form

$$U_{ZZ}(\mathbf{x}) = \exp\left(i \sum_{\{i,j\}\in T} \phi_{\{i,j\}}(\mathbf{x}) Z_i Z_j\right) H^{\otimes n} \tag{4}$$

and enable the map to encode both individual contributions and pairwise interactions between features into an entangled quantum state. Owing to its capacity to enrich the expressivity of the quantum feature space and

to the difficulty of simulating it classically, the ZZFeatureMap is one of the most commonly used data encoding methods in quantum classification tasks [6, 24–27].

3. *Challenges and Limitations*

All of the aforementioned quantum feature maps require the number of qubits to grow linearly with the dimensionality of the data, requiring one qubit per datum. This requirement becomes prohibitive with high-dimensional data, particularly with current NISQ devices [21]. The ZZFeatureMap tends to perform better than the ZFeatureMap so, for this study, we will only consider the ZZFeatureMap. In the case of ZZFeatureMap, the circuit depth grows linearly and the number of CNOT gates in the circuit grows quadratically with the number of features in the data; this will later be seen in Fig. 2. Appendix B explains why we choose the ZZFeatureMap over the ZFeatureMap for our comparisons, as the latter can seldom be used on its own when nonlinear functions of the input data are necessary. Our feature map has better resource costs than the ZZFeatureMap while still performing well, which is crucial for comparison because one can always find feature maps that are less resource intensive but with inferior performance.

II. RESULTS

A. CPMap

Our proposed *CPMap* is inspired by the structure of quantum convolutional neural networks (QCNNs) [17], which themselves borrow ideas from classical convolutional neural networks. Similar to their classical counterparts, QCNNs combine layers that extract salient features from data with pooling operations that progressively reduce dimensionality. This hierarchical reduction allows limited quantum resources to be focused on increasingly abstract representations of the input.

The CPMap follows a similar guiding principle but adapts it to the data encoding setting. Instead of discarding qubits after a pooling-like operation, the CPMap recycles them to encode additional features. In effect, pooling is replaced by partitioning: qubits freed by earlier feature aggregation are immediately reused for loading new data. Since CPMap is a feature mapping technique used for kernel construction rather than a trainable neural network, parameters are fixed rather than optimized, and measurements occur only at the end. Data are coherently uploaded and processed layer by layer, enabling more features to be embedded on a restricted number of qubits.

The CPMap alternates two types of fixed two-qubit unitaries. The first set, denoted C, plays a role analogous to convolutional filters, while the second set, denoted P, performs feature focusing similar to pooling. Together, these unitaries reduce the number of active qubits by approximately half at each layer: $n \mapsto n/2$. Iterating this process allows n qubits to encode roughly $2n$ features. More precisely, the maximum number of features that can be encoded on n qubits follows the meta-Fibonacci sequence with parameter $s = 0$ [28, 29]:

$$\begin{aligned} a(N) =& a(N - a(N-1)) + a(N - 1 - a(N-2)), \\ & \text{with} \quad a(1) = a(2) = 1. \end{aligned} \tag{5}$$

This construction yields a substantial increase in feature capacity on any given hardware.

The structure of CPMap is illustrated in Fig. 1, which highlights the replacement of QCNN's measurement-and-feedforward stage with the unitary P. More general focusing strategies can be adopted. For instance, if each pooling step reduces n qubits to n/m instead of $n/2$, then $n(m-1)/m$ qubits remain available for subsequent iterations, allowing even more features to be embedded on the same hardware. These flexible design choices make CPMap well suited for encoding large classical datasets into limited quantum registers.

To instantiate C and P, we restrict to two-qubit unitaries of the form

$$U = (A_1 \otimes A_2)\, N(\alpha, \beta, \gamma)\, (A_3 \otimes A_4), \tag{6}$$

where $A_i \in \mathrm{SU}(2)$ and

$$N(\alpha, \beta, \gamma) = \exp[i(\alpha X \otimes X + \beta Y \otimes Y + \gamma Z \otimes Z)]\,. \tag{7}$$

In general this requires 15 parameters, but in our implementation, we choose $A_i = \mathbb{I}$, leaving only the three free parameters (α, β, γ). A circuit decomposition of $N(\alpha, \beta, \gamma)$ into three CNOT gates is shown in Fig. 3 [30], and the circuit for the unitaries C and P is shown in Fig.1.

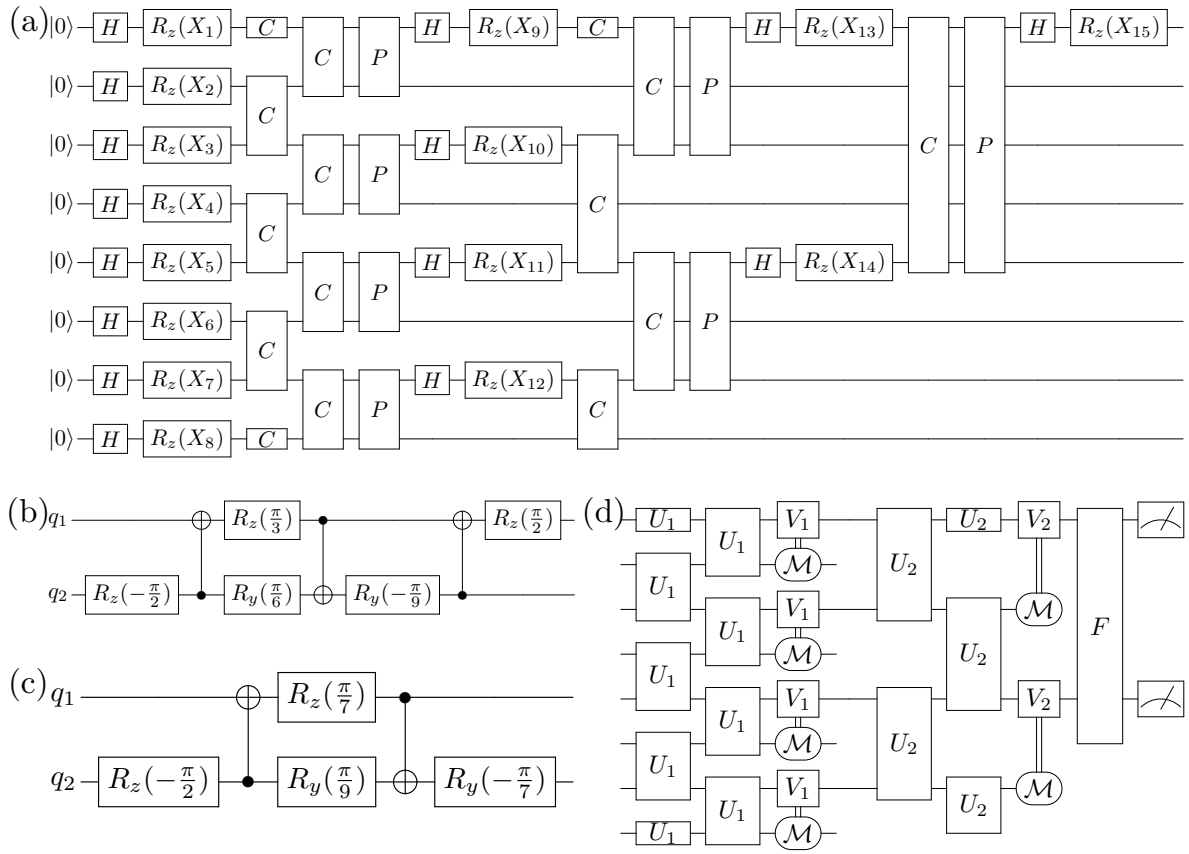


FIG. 1. (a) A CPMap with 8 qubits encoding 15 features. The qubits are then output to be used coherently in any application (including measurement, repetition of the kernel, and input to a quantum neural network). H is the Hadamard gate and $R_z(X_i)$ is a single-qubit rotation around the z axis by angle X_i. (b) Diagram of the C unitary. (c) Diagram of the P unitary. (d) An illustration of a quantum CNN with 8 qubits. The two-qubit unitaries U_1 and U_2 perform convolutions, then the pooling operations are done by measurements $\mathscr{M}$ on one qubit followed by feedforward unitaries V_1 and V_2 on the other qubit; all such operations involve parameters that must be trained. While (d) uses pooling operations, (a) adopts a similar architectural flow but replaces them with the unitary P, allowing more data features to be encoded before the subsequent layer.

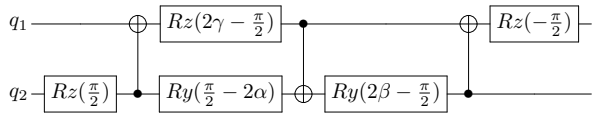


FIG. 3. A circuit for implementing the three-parameter two-qubit gate $N(\alpha,\beta,\gamma)$, requiring three CNOT gates.

Let n be the number of physical qubits and $x \in \mathbb{R}^F$ the feature vector. At each layer ℓ, a subset S_ℓ of active qubits is used to encode a contiguous batch of features $\mathbf{x}_\ell$ via $U_Z(\mathbf{x}_\ell)$.

The subset S_0 is chosen to be all of the qubits and each subsequent subset S_ℓ is chosen to be half of the qubits from the previous subset (e.g. taking every second qubit such that the data encoding takes the hierarchical structure depicted in Fig. 1(a)).

For a pair set $E \subseteq \{\{i,j\} \subset \{1,\ldots,n\} : i \neq j\}$ with disjoint pairs, define

$$C[E] := \bigotimes_{\{i,j\}\in E} C_{i,j}, \qquad P[E] := \bigotimes_{\{i,j\}\in E} P_{i,j}, \quad (8)$$

where $C_{i,j}$ and $P_{i,j}$ are the fixed two-qubit unitaries used in the circuit (their internal decomposition is not important for the formal description). Even and odd pairings of S_ℓ define sets of two-qubit gates C and P (chosen as nearest-neighbours in Fig. 1(a) for convenience on

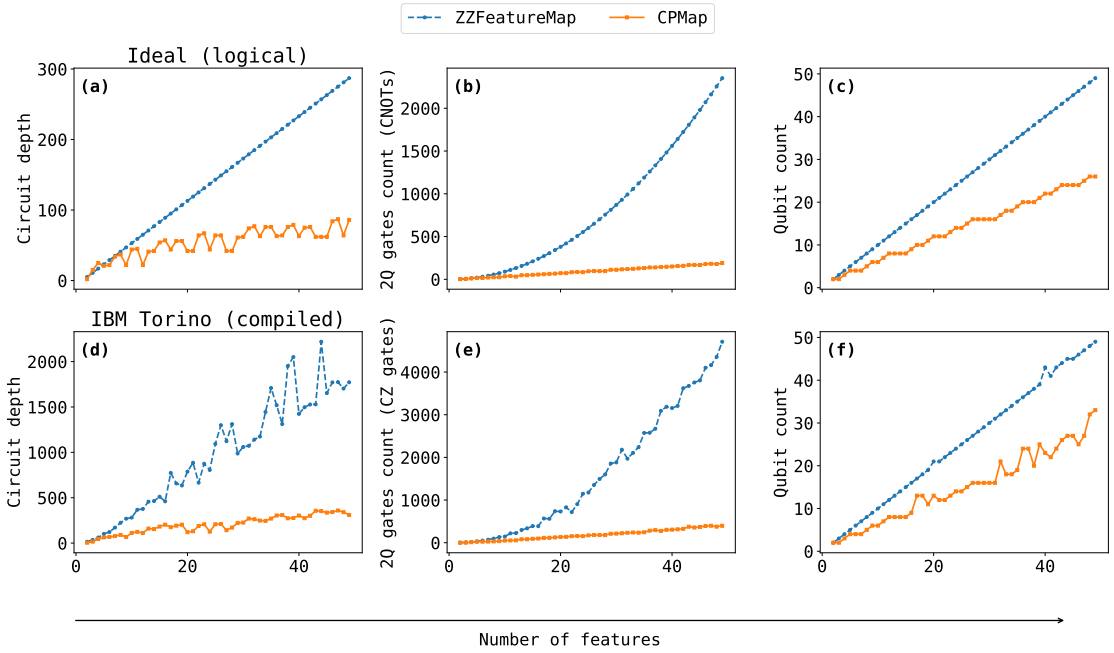


FIG. 2. **Resource scaling of CPMap vs. ZZFeatureMap (ideal vs. hardware-compiled).** The top row reports ideal/logical circuit resources (no transpilation) as a function of the number of features: (a) circuit depth, (b) two-qubit gates count (CNOT) operations, and (c) qubit count required by the feature map. The bottom row reports the same quantities after transpiling the circuits to the IBM Torino backend (native gate set and coupling constraints), where the native two-qubit operation is CZ; thus subplot (e) reports CZ counts for the compiled circuits. Across both settings, CPMap exhibits substantially reduced two-qubit resources and depth growth compared to ZZFeatureMap, and the qualitative scaling advantage is preserved after device-aware compilation.

quantum devices). The ℓth layer then acts as

$$\mathscr{L}_\ell(x) = P[E_\ell^{(1)}]\,C[E_\ell^{(1)}]\,C[E_\ell^{(0)}]\,D_{S_\ell}(x_{B_\ell})\,H_{S_\ell}. \tag{9}$$

Applying L layers produces the feature map

$$U_{\mathrm{CP}}(x) = \mathscr{L}_L(x)\cdots\mathscr{L}_1(x), \qquad |\psi_{\mathrm{CP}}(x)\rangle = U_{\mathrm{CP}}(x)\,|0\rangle^{\otimes n}. \tag{10}$$

We use this feature map in the fidelity quantum kernel setting, which is given by

$$K_{\mathrm{CP}}(x,x') = \left|\langle\psi_{\mathrm{CP}}(x)|\psi_{\mathrm{CP}}(x')\rangle\right|^2. \tag{11}$$

For example, in an 8-qubit circuit we can encode $F = 15$ features. The active sets are reduced as $|S_1| = 8, |S_2| = 4, |S_3| = 2, |S_4| = 1$. Each layer applies the even/odd C sweeps, and selective P, and passes the retained wires forward. This process yields the compact embedding structure shown in Fig. 1.

CNOT count for CPMap: The CPMap is constructed using the C and the P unitaries in multiple layers (ℓ), where C has 3 CNOT gates and P has 2 CNOT gates inside their circuit implementation. At each layer $\ell = 0, 1, 2, \ldots$, the active width is given by $|S_\ell| = n/2^\ell$. And inside each of these layers, we apply C on all active pairs and P on half as many pairs, so the per–layer CNOT cost is

$$\mathrm{CNOT}_\ell \;=\; 3\cdot\frac{n}{2^\ell} \;+\; 2\cdot\frac{n}{2^{\ell+1}} \;=\; \frac{4n}{2^\ell}. \tag{12}$$

Summing L full layers gives

$$\mathrm{CNOT}_{\le L} \;=\; \sum_{\ell=0}^{L-1}\frac{4n}{2^\ell} \;=\; 8n\big(1-2^{-L}\big). \tag{13}$$

In terms of the number of features encoded after L layers,

$$F_L \;=\; \sum_{\ell=0}^{L-1}\frac{n}{2^\ell} \;=\; 2n\big(1-2^{-L}\big), \tag{14}$$

so the CNOT count can be written as

$$\boxed{\mathrm{CNOT}_{\le L} \;=\; 4F_L} \tag{15}$$

and, in the limit $L \to \infty$ (i.e., $F \to 2n$), $\mathrm{CNOT}_{\mathrm{total}} \to 8n = 4F$.

1. Resource Complexity

Encoding a data point with F features using the ZZFeatureMap typically requires $n = F$ qubits and approximately $\mathcal{O}(F^2)$ two-qubit entangling operations, leading to rapidly increasing depth as F grows. In contrast, CPMap encodes F features using substantially fewer qubits, with $n \approx F/2$ (cf. Eq. (15)), and exhibits an approximately linear growth in entangling cost, i.e., $\mathcal{O}(F)$ two-qubit operations in the ideal (pre-compilation) circuit. Figure 2 reports this comparison in two settings: (top row) the logical circuits prior to transpilation, where the two-qubit count is reported in terms of CX (CNOT) gates, and (bottom row) the same circuits after transpilation to the IBM Torino backend, where the native two-qubit operation is CZ and the compiled two-qubit cost is therefore reported as CZ count. Importantly, the qualitative resource advantage of CPMap (lower depth growth, fewer two-qubit gates, and reduced qubit requirements) is preserved under device-aware compilation, making it particularly attractive for near-term (NISQ) implementations.

For completeness, we provide a hardware-feasibility analysis based on backend-timing circuit durations and device coherence times (T_1, T_2) in Appendix H (see Fig. 20). This shows that the compiled gate schedule remains well within coherence across the tested feature dimensions.

B. Numerical Results

The performance of the CPMap was benchmarked against several established quantum and classical kernels to ascertain its capabilities. We use several datasets characterized in Table I to check its performance as an SVC for classifying data into discrete categories.

Owing to the skewed nature of the majority of our test data, we opted for the Matthews Correlation Coefficient (MCC) as our primary evaluation criterion. For an in-depth explanation, kindly refer to Methods and Appendix A. MCC serves as a reliable metric, offering a comprehensive assessment of binary classification outcomes by considering both true and false positives as well as negatives. A score of +1 in MCC denotes perfect prediction accuracy, 0 suggests no better than a random prediction, and -1 signifies a complete mismatch between the predicted and actual outcomes. It overcomes metrics such as accuracy that can give misleadingly positive results when classifiers are tested on skewed data.

In our research, we extensively utilized Qiskit's Statevector simulator, a tool designed to simulate the ideal quantum states of a quantum system without any external noise or decoherence. This simulator provides a precise representation of the quantum state, allowing for accurate computations and predictions. It is particularly beneficial for theoretical explorations and understanding the ideal behaviour of quantum algorithms. Alongside the Statevector simulator, we also employed Qiskit's noisy simulators. These are designed to mimic real IBM quantum devices by leveraging system snapshots. Such snapshots capture vital data about the quantum setup, including the coupling map, foundational gates, and qubit attributes (T1, T2, error rates, and more), proving instrumental for transpiler testing and conducting system simulations with noise. These noisy simulators present a more grounded view of quantum operations in tangible settings. Their use enabled us to assess the durability of our algorithms in authentic environments and refine them to better withstand quantum disruptions and other unforeseen challenges. We then proceeded to run small trials on ibm_quebec and ibm_torino.

1. Results with Seven Features After PCA

In the initial phase of our study, we employed principal component analysis (PCA) to reduce the feature set size to seven across all datasets. Specifically, we limited the sample size of the Stellar Classification Dataset to 2,000 and excluded the QSO class to expedite the simulation process. For support vector classification, we leveraged the capabilities of both the SciKit Learn [31] and QisKit [23] libraries.

Our findings, depicted in Fig. 4, reveal that the CPMap consistently outshines the ZZFeatureMap quantum kernel across a diverse range of datasets. Remarkably, there were instances where the CPMap not only matched but exceeded the performance of standard classical kernels [linear, polynomial (poly), radial basis function (RBF), sigmoid], highlighting its promising applications in the realm of quantum machine learning. The accompanying plots, which display MCC scores, further substantiate the superior efficacy of the CPMap. While some classical kernels ex-

Dataset	Number of Features	Number of Instances	Classes
Ionosphere	34	351	2
Breast cancer Diagnostic (Wisconsin)	30	569	2
Credit card Fraud detection (Balanced)	30	984	2
Parkinson's disease (PD)	22	195	2
Stellar Classification Dataset - SDSS17	17	100000*	3*
Heart Disease	11	918	2
Titanic	10	891	2
Agaricus Lepiota Mushrooms	21	8145	2

TABLE I. Properties of various datasets used in this study. *The Stellar Classification Dataset properly has three classes, but we used only two of them, Galaxy and Star, combining for a total of 81039 data points.

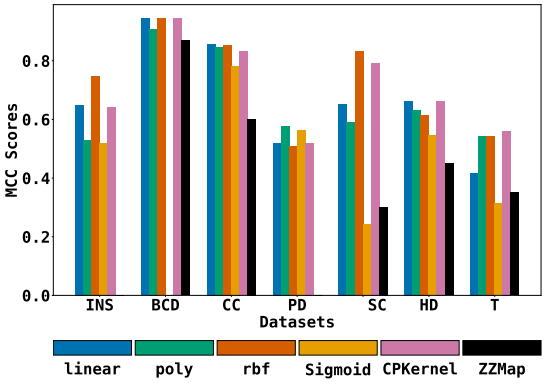


FIG. 4. MCC-score-based analysis of different kernels on different datasets scaled to each have 7 features, comparing the effectiveness of each kernel in handling specific types of data. The CPMap (second from right for each dataset) outperforms the ZZFeatureMap (rightmost) on all datasets in this 7-feature benchmark, and is competitive with strong classical kernels (linear, poly, RBF, sigmoid), sometimes exceeding them depending on the dataset. For some datasets, certain kernels completely failed, so there is no bar visible for the ZZFeatureMap for the INS and PD datasets and for the sigmoid kernel for the BCD dataset. Dataset acronyms: Ionosphere (INS), Breast Cancer Diagnostic (BCD), Credit Card Fraud (CC), Parkinson's Disease (PD), Stellar Classification (SC), Heart Disease (HD), and Titanic Survival (T).

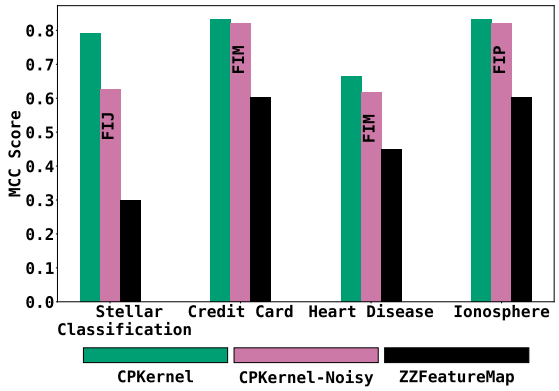


FIG. 5. MCC score for the noisy simulation of CPMap on different datasets. Here we used *'fake_ibmq_jakarta' (FIJ)*, *'fake_ibmq_manila' (FIM)* and *'fake_ibmq_perth' (FIP)* backends.

hibited strong performance on specific datasets, they faltered on others. In contrast, the CPMap consistently delivered robust results, often rivaling or surpassing the best-performing classical kernels.

For some datasets, we repeated the CPMap up to two times but uploaded the same features for each repetition. This increases the total number of CNOT gates by a factor of two but maintains the total number of qubits and the significant reduction in the number of CNOT gates; see Appendix C 1 for details. An additional multi-class experiment on the Stellar Classification dataset (balanced $N = 1500$) is reported in Appendix F, and an expanded comparison against fixed data re-uploading feature-map variants (DR1–DR6) is provided in Appendix E.

2. *Results for the Noisy Simulation with Seven Features*

In addition to standard simulations, we conducted noisy simulations on three specific datasets: Stellar Classification, Balanced Credit Card, and Heart Disease.

As demonstrated in Figure 5, CPMap remains competitive under these noise models and, in this benchmark, achieves higher MCC than ZZFeatureMap even when ZZFeatureMap is evaluated in the ideal (noiseless) setting.

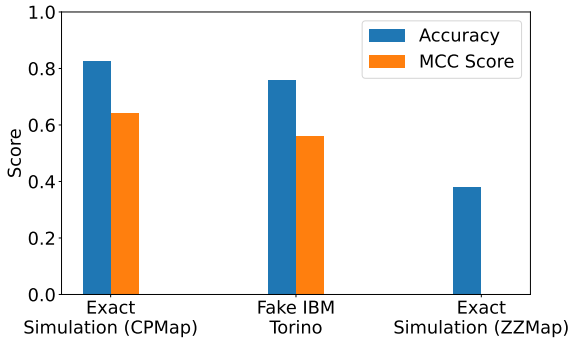


FIG. 6. Performance comparison of the CPMap under ideal and noisy conditions. The CPMap maintains reasonable classification performance despite noise, achieving an MCC of 0.56 and accuracy of 0.76 in the noisy case, compared to 0.64 and 0.83, respectively, in the ideal case. While the ZZMap achieves 0 MCC. These results highlight the noise resilience of the CPMap encoding strategy.

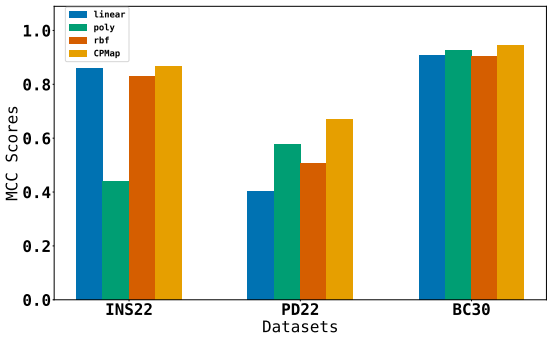


FIG. 7. This bar chart illustrates the MCC scores achieved by four kernel functions—from left to right: linear, polynomial (poly), radial basis function (RBF), and the CPMap—when applied to three different datasets.

3. *Results for the Noisy Simulation with sixteen Features*

To check the effect of realistic noise when dealing with a higher-dimensional problem, we ran a similar noise test using the Parkinson's disease dataset using the Qiskit fake back-end model of `ibm_torino`. This time, we used the dataset with 16 features (i.e., 9 qubits for CPMap). The results are shown in figure 6.

The CPmap under ideal conditions achieved an accuracy of 0.83 and MCC of 0.64, while under noisy conditions, the performance degraded, but it still achieved an accuracy of 0.76 and MCC of 0.56, highlighting its resilience to the realistic noisy conditions of current devices. The ZZFeatureMap failed to classify the dataset, achieving an MCC of 0.

4. *A case for quantum usefulness*

Next, we turn to datasets with so many features that it is prohibitive to simulate the ZZFeatureMap on a classical computer due to the exorbitant qubit and gate-count requirements. We perform PCA on the Ionosphere dataset to reduce it to 22 features, in order to analyze it with the exact same CPMap as for the Parkinson's dataset that has exactly 22 features.

The bar chart presented in Fig. 7 illustrates the performance of various kernels, including linear, poly, RBF, and CPMap, across the three distinct datasets: Ionosphere (INS22), Parkinson's Disease (PD22), and Breast Cancer (BC30). Notably, the CPMap outperforms its counterparts across all datasets, achieving the highest MCC scores, indicative of its superior predictive capabilities. On the PD22 dataset, CPMap's score is a significant leap from poly's score and is even greater compared to the linear and RBF kernels. This trend persists with the INS22 and BC30 datasets, where CPMap consistently maintains the lead. Similar results are presented in Appendix D with even more features per dataset and results for different numbers of repetitions of the CPMap in Appendix C 2.

The consistent outperformance of CPMap in our experiments indicates an empirical advantage for this embedding under the studied settings, particularly when feature counts are larger and circuit resources are constrained. The substantial margin by which CPMap leads for datasets with large numbers of features hints at its unique ability to capture complex patterns that classical kernels might not discern as effectively, thereby bolstering the hypothesis that quantum machine learning could offer computational benefits over traditional algorithms.

C. Statistical Significance Testing

In addition to demonstrating quantum usefulness, we further validated the performance of our custom CPMap kernel against traditional RBF, poly, and linear kernels using Parkinson's disease dataset. We employed statistical significance testing to confirm the superiority of CPMap in terms of accuracy and the MCC.

We performed paired t-tests to evaluate the significance of performance differences between the CPMap kernel and the traditional kernels (RBF, Poly, and Linear) in terms of accuracy and MCC scores across 80 independent runs. The results are summarized in Table II.

TABLE II. Paired t-Test Results Comparing CPMap with Other Kernels

Comparison	Metric	t-Statistic	p-Value
RBF vs. CPMap	Accuracy	8.45	1.15×10^{-12}
RBF vs. CPMap	MCC	8.54	7.83×10^{-13}
POLY vs. CPMap	Accuracy	11.38	2.57×10^{-18}
POLY vs. CPMap	MCC	11.32	3.28×10^{-18}
Linear vs. CPMap	Accuracy	11.60	9.78×10^{-19}
Linear vs. CPMap	MCC	11.26	4.26×10^{-18}

The t-statistics and p-values indicate that the performance differences between the CPMap kernel and each of the traditional kernels are statistically significant. The t-statistics exceed typical critical values, and the p-values are significantly below standard significance levels (e.g., 0.05, 0.01), thereby rejecting the null hypothesis of no difference in performance.

Performance Metrics Summary: To provide a detailed view of the CPMap kernel's performance, we calculated the mean and standard deviation of accuracy and MCC scores across the 80 runs for each kernel. These metrics are detailed in Table III.

TABLE III. Mean and Standard Deviation of Accuracy and MCC for Different Kernels

Kernel	Accuracy Mean	Accuracy Std	MCC Mean	MCC Std
Linear	0.8615	0.0442	0.6102	0.1333
Poly	0.8660	0.0424	0.6135	0.1354
CPMap	0.9234	0.0388	0.7898	0.1118
RBF	0.8785	0.0477	0.6532	0.1438

Figures 8 and 9 illustrate the distribution of accuracy and MCC scores across the different kernels. These visualizations highlight the consistent superior performance of the CPMap kernel.

This statistical analysis and visualization confirm the significant advantage of the CPMap kernel over traditional kernels in the context of Parkinson's disease data, reinforcing the potential for quantum machine learning to achieve empirical improvement in predictive tasks.

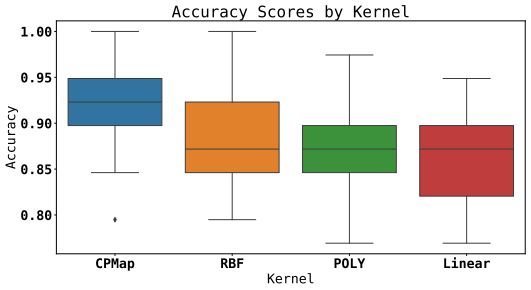


FIG. 8. Box Plot of Accuracy Scores for Different Kernels

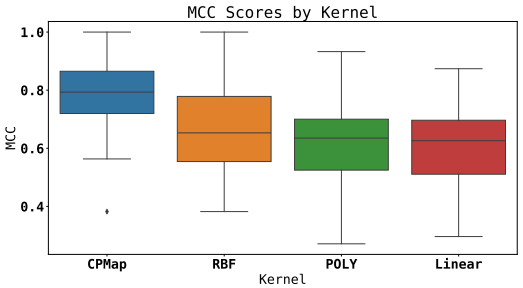


FIG. 9. Box Plot of MCC Scores for Different Kernels

D. Test on IBM's Hardware

To investigate the feasibility and performance of our CPMap quantum kernel on real superconducting quantum processors, we conducted experiments on two IBM devices: the 127-qubit ibm_quebec (Eagle-class) and the 133-qubit ibm_torino (Heron-class). All experiments targeted a binary classification task using the Parkinson's disease dataset, which originally contains 195 samples and 22 numerical features per instance. The qubit configurations, data preprocessing, and mitigation strategies can be found in the Methods section.

1. *Test with full dataset on ibm_quebec*

We evaluated the CPMap kernel on IBM's 127-qubit ibm_quebec device using the full 22-feature Parkinson dataset encoded on 12 qubits. To assess kernel performance, we ran tests across three transpiler optimization levels and applied error mitigation techniques including dynamical decoupling (DD), zero noise extrapolation (ZNE), and probabilistic error cancellation (PEC). In all cases, the average entropy of the output was approximately same as the number of qubits,

which implies the outputs yielded no meaningful information. Specifically, entropy values ranged from 11.58 to 11.66 across all runs—very close to the maximum possible for a 12-qubit system. This was attributed both to CPMap's requirement for all-to-all connectivity—which introduces numerous SWAP operations and deepens the circuit—and to the high noise levels inherent to the ibm_quebec device.

We also performed a small test on the newer 133-qubit ibm_torino (Heron-class) device using a subset of the Parkinson dataset. In contrast to ibm_quebec, the output entropy on ibm_torino remained within a reasonable range, with values ranging from 5.59 to 9.79—well below the 12-qubit maximum—indicating that the circuits retained meaningful structure. This suggests that the full dataset could feasibly be processed on this hardware. However, due to limited device access, we conducted the experiment with 7 PCA-reduced features mapped to 4 qubits. The classification results are reported below.

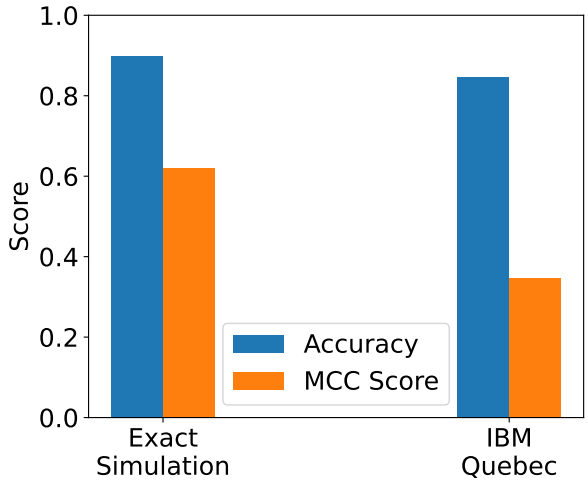


FIG. 10. Performance comparison of the CPMap kernel for Parkinson's disease classification. Accuracy and MCC Score are shown for both exact simulation and IBM's ibm_quebec (4-qubit encoding).

2. *Test with PCA-reduced dataset on ibm_quebec*

We tested the CPMap kernel on IBM's ibm_quebec device using 4 qubits (7 input features) and no error mitigation for Parkinson's disease classification. Experiments were performed on qubit chains 39-40-41-42 or 39-40-41-53. The calibration data for these qubits indicate the following average values: 3.39×10^{-3} s for T1, 2.37×10^{-3} s for T2, and 3.82×10^{-3} for the two-qubit gate error rate. In this trial, we obtained an MCC score of 0.34 and an accuracy comparable to simulation (see Fig 10). While the result is still affected by hardware noise, it marks the first instance where this kernel produced a meaningful result on real hardware.

As mentioned above, one limiting factor is the device's restricted qubit connectivity, which requires the transpiler to insert multiple SWAP gates during compilation, increasing the number of CNOT operations and overall circuit depth. This opens the path to explore optimal compilation of our kernel in architectures with limited connectivity and realization in platforms such as trapped ions where qubit connectivity is not constrained.

3. *Test with PCA reduced dataset on ibm_torino*

In this case, we deployed our CPMap kernel circuit on IBM's ibm_torino, a 133-qubit superconducting quantum processor belonging to the Heron family. Heron-class devices feature improvements in qubit coherence and native two-qubit gate implementations. At the time of execution, the backend-reported median coherence times were approximately $T_1 \approx 1.83 \times 10^{-4}$ s and $T_2 \approx 1.41 \times 10^{-4}$ s.

Due to limited hardware access, we performed this experiment using a PCA-reduced representation with $d = 7$ features, encoded on 4 qubits using CPMap. The overlap circuits required for the fidelity kernel were executed on hardware using IBM Runtime's `Sampler` primitive with otherwise default runtime settings. We used 1024 shots per circuit and did not apply any error mitigation or correction techniques for this result. We did not manually select a qubit layout; instead, we allowed the IBM Runtime compilation workflow to choose the physical qubits and initial layout automatically. All circuits were transpiled with optimization level 3.

The quantum kernel matrix was computed in a single run across the full dataset and training and test matrices were extracted without recomputation. We allocated 80 samples for training and 20 for testing.

The quantum kernel matrix was computed in a single run (i.e., without recomputing entries across training and test evaluation), and the training and test kernel submatrices were extracted from this precomputed kernel. We allocated 80 samples for training and 20

samples for testing.

Despite inevitable hardware noise, the quantum kernel executed on ibm_torino demonstrated strong performance. It achieved an MCC of 0.68 and an accuracy of 0.85 (see Fig. 11). Among the three kernels evaluated, the simulated CPMap kernel achieved the highest performance (MCC of 0.90, accuracy of 0.95), followed by the classically optimized RBF kernel (MCC of 0.78, accuracy of 0.90), and finally, the CPMap executed on real hardware. Notably, the ZZFeatureMap kernel, simulated under *ideal noiseless* conditions, yielded an MCC of 0.00, failing to capture any useful structure in this task. This further underscores the practical relevance of our CPMap.

The observed performance gap between the statevector simulation and the real-device execution can be attributed to decoherence time, gate and measurement errors in the compiled overlap circuits. Nevertheless, the successful deployment of CPMap on ibm_torino confirms its feasibility on today's NISQ devices and underscores the promise of quantum kernels in practical machine learning tasks.

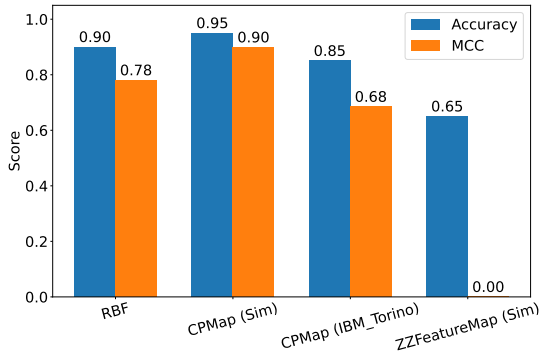


FIG. 11. Comparison of CPMap based kernel on the curated Parkinson dataset with PCA-reduced input ($d = 7$), executed on `ibm_torino` (Sampler, 1024 shots, optimization level 3; no error mitigation). The simulated CPMap-based Kernel achieves the highest performance, followed by RBF and quantum hardware (ibm_torino).

E. Parameter sensitivity and kernel geometry

CPMap contains a small set of fixed design parameters that determine how pairs of qubits interact inside the feature map. These parameters are not trained, so it is important to check whether the method is stable and whether parameter variations induce meaningful (rather than arbitrary) changes in the resulting kernel. To do this, we performed a random parameter search on a 400-sample subset of the Breast Cancer dataset: we randomly sampled CPMap angle settings (varying both the C and P-block parameters), constructed the corresponding kernel matrices, and evaluated the same SVM pipeline using the mean performance over 50 stratified train/test splits. For each sampled setting, we also computed kernel–label alignment, a standard scalar summary of how well the kernel similarity matrix matches the class structure (higher alignment means that samples from the same class tend to be more similar under the kernel than samples from different classes). As shown in Fig. 12, parameter settings with higher alignment consistently yield higher classification performance (correlation ≈ 0.57). This indicates that the CPMap parameters control the geometry of the induced kernel in a systematic way, rather than acting as arbitrary constants, and it provides a practical diagnostic for selecting reasonable parameter regimes. Another point to note is that the performance varies substantially across sampled settings (a wide spread in MCC), demonstrating that the CP parameters are not innocuous constants; rather, they can materially influence the quality of the kernel and downstream accuracy. Taken together, these results support two practical conclusions: (i) CPMap is not overly brittle—many parameterizations remain competitive—but (ii) the choice of angles does matter, and alignment provides a simple diagnostic for identifying promising regimes without introducing trainable parameters. Additional experimental details and kernel diagnostics are provided in Supplementary Section G.

F. Issue of Exponential Concentration:

Exponential concentration of quantum kernels, as formalized in [38], presents a fundamental limitation for fidelity-based kernels. Given a quantum feature map $U(x)$ on n qubits, the fidelity kernel is defined as

$$k(x,x') = \left|\langle 0^n|U(x)^\dagger U(x')|0^n\rangle\right|^2 = p_0(x,x'), \quad (16)$$

and the variance of p_0 across random pairs (x,x') quantifies the kernel's discriminative spread. As the feature dimension increases, the overlaps concentrate around their mean, leading to

$$\mathrm{Var}[p_0] \sim e^{-c\cdot n}, \qquad c > 0, \quad (17)$$

so that kernels become progressively uninformative in high dimensions.

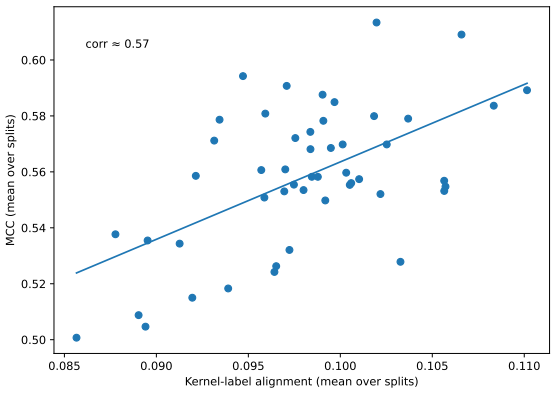


FIG. 12. Kernel geometry vs performance under random parameter search on Breast Cancer ($n = 400$, $d = 12$). Each point is one parameter setting evaluated over 50 stratified splits. The x-axis is kernel–label alignment (higher means the kernel similarities better match class structure) and the y-axis is mean MCC. A strong positive trend (corr ≈ 0.57) shows that CPMap parameters systematically shape kernel geometry in a way that predicts performance. The wide dispersion in MCC further shows that the parameters materially affect results. The solid line is a least-squares fit.

Our proposed CPMap and its implementation based on the fidelity kernel, by construction, is not exempt from the issue of exponential concentration. However, the CPMap changes the rate of the exponential concentration by quadratically lowering the CNOT counts and halving the qubit overhead. In addition to the effects of resource efficiency on slowing the exponential concentration, we observed that CPMap behaves more favorably as a function of the number of qubits, thereby providing a more realistic paradigm for information processing on NISQ devices.

To analyze the issue of exponential concentration, we tested both feature maps (ZZFeatureMap and CPMap) with the fidelity quantum kernel on the Ionosphere dataset, a binary classification dataset commonly used in kernel-based learning. The diagnostic in Fig. 13 (top) plots $\log \mathrm{Var}[p_0]$ as a function of feature dimension F, while the bottom panel shows the same diagnostic against the actual number of qubits used. At the same number of features, the ZZFeatureMap exhibits faster decay (minimum $\log \mathrm{Var}[p_0] \approx -11.5$) compared to CPMap (minimum ≈ -5.7), indicating stronger concentration due to using more qubits per feature. Even when we compared with the same number of qubits, CPMap consistently maintains higher variance in overlap values than ZZ, implying that it retains discriminative power for a longer period and has a better decay rate c by a factor of 2-3.

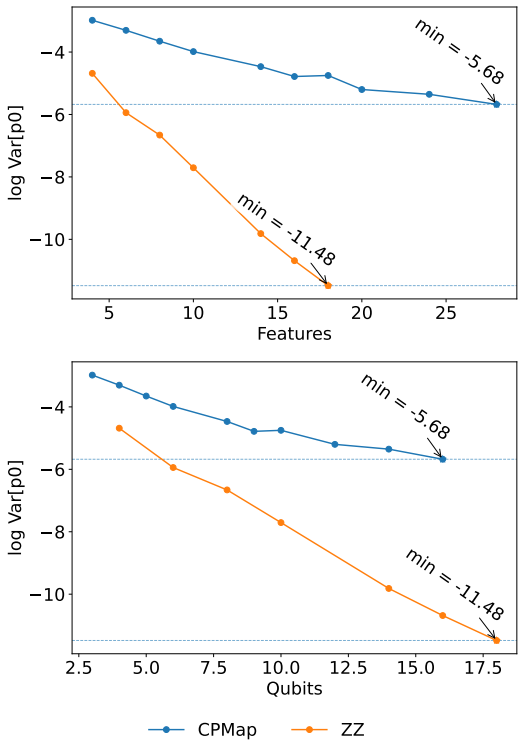


FIG. 13. The plots compare the variance of kernel overlaps (shown as $\log \mathrm{Var}[p_0]$) across two feature maps: the proposed CPMap and the conventional ZZFeatureMap. (Top) Variance as a function of feature dimension F. (Bottom) Variance as a function of effective qubit count n. In both cases, the CPMap maintains consistently higher variance and delays the onset of exponential concentration relative to the ZZFeatureMap. These results show that the CPMap mitigates concentration effects and preserves discriminative power in higher-dimensional settings.

Additionally, from a practical point of view, extremely low variance (e.g., $\mathrm{Var}[p_0] \sim 10^{-11}$ for ZZFeatureMap) means we need an extremely large number of measurement shots (on the order of 10^{11}) to have a meaningful discriminative power. This is not practical for the current devices. At the same time, CPMap has a consistently higher variance, which suggests it can obtain meaningful information with fewer shots, making it far more suitable for the NISQ era.

Although CPMap does not eliminate the fundamental limitation of exponential concentration, it *extends the useful operating regime* of fidelity kernels on near-term devices. By reducing the qubit

and CNOT overhead, CPMap enables exploration of higher-dimensional datasets before entering the severe concentration regime, offering a practical pathway for applying quantum kernels in real-world machine learning settings.

III. DISCUSSION AND CONCLUSION

Our proposed feature map paves the way for the application of quantum machine learning algorithms to more complex and higher-dimensional data, even with limited qubit resources. The CPMap's remarkable efficiency in qubit utilization, evident through its requirement of approximately $\frac{N}{2}$ qubits to represent N features, marks a substantial improvement over the ZZFeatureMap. This efficiency not only eases the computational load but also enhances the scalability of quantum models. In our comparative analysis with classical kernels such as the RBF, linear, and polynomial kernels, the CPMap sometimes demonstrated superior performance. Our findings show that, for the studied datasets and preprocessing choices, CPMap can achieve competitive (and sometimes higher) predictive performance than standard classical kernels, while using substantially fewer quantum resources than common Pauli feature maps.

The operational feasibility of the CPMap is another cornerstone of our research. Its reduced circuit depth, as opposed to the computationally intensive ZZFeatureMap, enabled us to successfully simulate complex datasets, due to the significant reduction in number of entangling gates required for our scheme. This aspect of the CPMap not only addresses the current limitations in quantum computing resources but also makes quantum machine learning models more accessible and practical for a broader range of applications.

Note on classical simulability. CPMap is intentionally shallow and locally structured to improve NISQ feasibility. The circuit structure underlying CPMap resembles QCNN-style architectures. Such architectural choices can also place a model in regimes where classical approximation methods remain effective. Recent work by Bermejo [50] argues that QCNNs can be *effectively* classically simulable in practically relevant regimes, most notably, when their action is restricted to information accessible via low-bodyness observables and when the benchmark instances are "locally-easy," so that a classical algorithm equipped with (Pauli) classical shadows can reproduce the relevant behaviour.

In contrast, our method uses CPMap as a *feature map*, for kernel method, where training and evaluation require estimating state overlaps of the form $|\langle 0|U(y)^\dagger U(x)|0\rangle|^2$ across many pairs. Whether such overlap estimation is efficiently approximable classically can depend strongly on the circuit depth, entanglement growth, and the input distribution (e.g., the tensor-network bond dimension required for accurate contraction). We therefore do not make a general simulability or hardness claim here; rather, we position CPMap as a resource-efficient embedding and treat questions of classical simulability as regime-dependent.

The implications of this work extend beyond the specific benchmarks considered here. By reducing qubit requirements and limiting the growth of two-qubit entangling operations, CPMap makes kernel-based quantum learning experiments more accessible at higher feature counts, where many standard feature maps quickly become resource-intensive. This motivates several directions for future work, particularly in applying quantum algorithms to more complex and diverse datasets with applications in health sciences to material discovery. Further exploration into its realization on existing noisy quantum computing platforms and future small-scale fault-tolerant devices, as opposed to simulations, could offer deeper insights into its practical utility and performance in real-world applications.

Rather than positioning CPMap as a universal replacement for Pauli feature maps, we view it as a resource-efficient alternative that can be advantageous when qubit count and entangling-gate budgets are the dominant constraints. Our numerical benchmarks and small-scale hardware demonstrations indicate that CPMap can achieve competitive performance while using fewer qubits and fewer entangling gates (CNOTs) than standard Pauli-style constructions at comparable feature counts. These results suggest CPMap as a useful candidate for near-term kernel workflows and for systematic benchmarking against strong classical baselines. Clarifying the regimes where CPMap provides consistent benefits—across larger datasets, wider classes of quantum embeddings, and more realistic noise models—remains an important next step.

IV. METHODS

In this study, we introduce an instance of the CPMap, specifically designed for data classification using the kernel method across various standard datasets. The CPMap simulation is conducted utilizing two dis-

tinct environments: Qiskit's Statevector simulator and its Noisy counterpart, providing a comprehensive analysis of its performance under varied conditions. The datasets employed in this investigation are sourced from standard online repositories, namely Kaggle and the UCI dataset library, ensuring a diverse and robust set of data for evaluation. To facilitate the support vector classification process, we employ the built-in SVC function from Scikit-learn, a widely recognized tool in machine learning. This section details the parameters, methodologies, and evaluation techniques utilized in the development and assessment of the CPMap, aiming to demonstrate its effectiveness and versatility in data classification tasks.

To address the limitations of conventional metrics on imbalanced datasets (see Appendix A), the MCC was employed. MCC is a balanced metric that takes into account both over-predictions and under-predictions across classes. It is defined as

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}, \tag{18}$$

where TP, TN, FP, and FN are the numbers of true positives, true negatives, false positives, and false negatives, respectively. The MCC returns a value in $(-1, 1)$, with 1 representing perfect prediction, -1 indicating total disagreement between prediction and observation, and 0 suggesting no better than random prediction. MCC is particularly valuable for imbalanced datasets as it considers all four components of the confusion matrix and is less susceptible to the bias of a large class [32].

To facilitate a comprehensive comparison, we utilized seven distinct features for each dataset when evaluating the CPMap against the ZZFeatureMap. This choice is strategically made, considering that the ZZFeatureMap demands one qubit per feature and, notably, the number of CNOT gates required for the ZZFeatureMap increases quadratically with the number of features. Such rapid growth renders the simulation of larger datasets challenging. Opting for seven features strikes a balance, allowing for a meaningful comparison while keeping the computational requirements within manageable limits. In contrast to the ZZFeatureMap, the CPMap demonstrates remarkable efficiency in qubit utilization, requiring a mere four qubits to represent these seven features—a notable advancement over the ZZFeatureMap kernel. Moreover, we have benchmarked the CPMap's performance against several classical kernels, namely the RBF, linear, and polynomial kernels, to establish a baseline comparison.

For the noisy simulation, we targeted three distinct datasets: Stellar Classification, Balanced Credit Card, and Heart Disease. To accommodate computational limitations and manage extended runtimes, we applied PCA to reduce the dimensionality of the Heart Disease and Balanced Credit Card datasets, ultimately selecting the seven most informative features for these simulations. These noisy simulations were conducted using Qiskit's advanced noisy simulators, designed to mimic the behaviour of actual quantum computers, thereby providing a more realistic assessment of our models' performance in quantum computing environments. For the Balanced Credit Card and Heart Disease datasets, we utilized the **fake_ibmq_manila** backend, a simulator that emulates the noise characteristics of the IBMQ Manila quantum device. To avoid cherry picking, for the Stellar Classification dataset, we employed the **fake_ibmq_jakarta** backend, replicating the conditions of the IBMQ Jakarta, and for the Ionosphere dataset, we used **fake_ibm_perth**. These choices allowed us to assess the robustness of our models against realistic quantum noise and error rates. The results from these simulations, highlighting the impact of quantum noise on model accuracy and reliability, are detailed in Fig. 5, providing critical insights into the potential real-world performance of quantum machine learning algorithms.

In our experiments, we also processed the datasets in their entirety using the CPMap without resorting to Principal Component Analysis (PCA) for dimensionality reduction. This approach provides a more authentic test of the CPMap's capability to handle high-dimensional data. Specifically, in the case of the Breast Cancer dataset, which requires sixteen qubits to represent thirty features, we observed that the CPMap necessitates significantly less circuit depth. This efficiency advantage enabled us to successfully simulate the model on a personal computer without encountering computational bottlenecks. It is crucial to note that running the Breast Cancer dataset with all 30 features using the ZZFeature Map was not feasible due to its computational expensiveness and the near impossibility of simulating 30 qubits with such high depth on personal computers. Notably, the quantum Kernel based on CPMap exhibited superior performance compared to the classical kernels in analyzing the Breast Cancer, Heart Disease datasets, thereby demonstrating its potential for practical applications in quantum machine learning and indicating a promising direction for further research in this area.

We conducted experiments on IBM's superconduct-

ing quantum processors. Specifically, we tested our circuits on two real devices: ibm_quebec (127-qubit Eagle-class) and ibm_torino (133-qubit Heron-class). These evaluations were performed on the Parkinson's disease dataset. On ibm_quebec, we used all 22 features mapped to 12 qubits, as well as a PCA-reduced 7-feature input encoded on 4 qubits. For the 12 qubit case, error mitigation techniques including dynamical decoupling, zero noise extrapolation, and probabilistic error cancellation were applied in combination with varying transpilation levels to analyze hardware-induced deviations. On ibm_torino, we performed an end-to-end kernel classification task using a curated subset of 100 samples selected via classical SVM prefiltering, ensuring a mix of easy and ambiguous cases. We did not use any error mitigation technique for this task. The quantum kernel matrix was constructed in a single batch run, and training/testing splits were extracted without recomputation. These experiments allowed us to benchmark our kernel's performance under real-device noise and confirm its competitiveness with classically optimized models, demonstrating the CPMap's potential viability on current NISQ hardware.

V. ACKNOWLEDGMENTS

The authors would like to acknowledge the use of IBM Quantum services for this work and in particular the Qiskit package [23], as well as fruitful discussions with Anaelle Hertz and Barry Sanders. We thank Marco Armenta for assisting with the execution of our experiments on the IBM Quebec quantum device. AZG and KH acknowledge that the NRC headquarters is located on the traditional unceded territory of the Algonquin Anishinaabe and Mohawk people. K.H. acknowledges funding from the NSERC Discovery Grant and Alliance programs.

VI. DATA AVAILABILITY

Data and code related to this research can be found at this private GitHub repository upon reasonable request.

[1] L. Chen, in *Encyclopedia of Database Systems* (Springer, Boston, MA, Boston, MA, USA, 2009) pp. 545–546.
[2] V. Berisha, C. Krantsevich, P. R. Hahn, S. Hahn, G. Dasarathy, P. Turaga, and J. Liss, npj Digital Med. **4**, 1 (2021)
.
[3] R. E. Bellman, *Adaptive Control Processes: A Guided Tour* (Princeton University Press, Princeton, NJ, 1961)
.
[4] C. M. Bishop, *Pattern Recognition and Machine Learning* (Springer, New York, NY, 2006)
.
[5] M. Verleysen and D. François, in *Computational Intelligence and Bioinspired Systems* (Springer, 2005) pp. 758–770
.
[6] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, Nature **567**, 209 (2019).
[7] E. Peters, J. Caldeira, A. Ho, S. Leichenauer, M. Mohseni, H. Neven, P. Spentzouris, D. Strain, and G. N. Perdue, npj Quantum Inf. **7**, 1 (2021).
[8] T. Kusumoto, K. Mitarai, K. Fujii, M. Kitagawa, and M. Negoro, npj Quantum Inf. **7**, 1 (2021).
[9] S. L. Wu, S. Sun, W. Guan, C. Zhou, J. Chan, C. L. Cheng, T. Pham, Y. Qian, A. Z. Wang, R. Zhang, M. Livny, J. Glick, P. Kl. Barkoutsos, S. Woerner, I. Tavernelli, F. Carminati, A. Di Meglio, A. C. Y. Li, J. Lykken, P. Spentzouris, S. Y.-C. Chen, S. Yoo, and T.-C. Wei, Phys. Rev. Res. **3**, 033221 (2021).
[10] D. Alaminos, M. B. Salas, and M. A. Fernández-Gámez, Comput. Econ. **59**, 803 (2022).
[11] S. Lloyd, M. Mohseni, and P. Rebentrost, Nat. Phys. **10**, 631 (2014).
[12] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017).
[13] M. Schuld, I. Sinayskiy, and F. Petruccione, Contemp. Phys. **56**, 172 (2015).
[14] M. Schuld and N. Killoran, Phys. Rev. Lett. **122**, 040504 (2019).
[15] M. Schuld, arXiv 10.48550/arXiv.2101.11020 (2021), 2101.11020.
[16] A. Asfaw, L. Bello, Y. Ben-Haim, S. Bravyi, L. Capelluto, A. C. Vazquez, J. Ceroni, R. Chen, A. Frisch, J. Gambetta, S. Garion, L. Gil, S. D. L. P. Gonzalez, F. Harkins, T. Imamichi, D. McKay, A. Mezzacapo, Z. Minev, R. Movassagh, G. Nannicini, P. Nation, A. Phan, M. Pistoia, A. Rattew, J. Schaefer, J. Shabani, J. Smolin, K. Temme, M. Tod, and S. Wood, *Learn Quantum Computation Using Qiskit* (2020).
[17] I. Cong, S. Choi, and M. D. Lukin, Nat. Phys. **15**, 1273 (2019).
[18] C. Zoufal, A. Lucchi, and S. Woerner, npj Quantum Inf. **5**, 1 (2019).

[19] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, Quantum **4**, 226 (2020).

[20] B. Roy, All about data encoding for quantum machine learning.

[21] J. Preskill, Quantum **2**, 79 (2018), 1801.00862v3.

[22] T. Hubregtsen, J. Pichlmeier, P. Stecher, and K. Bertels, Quantum Mach. Intell. **3**, 9 (2021).

[23] Qiskit contributors, Qiskit: An open-source framework for quantum computing (2023).

[24] X. Vasques, H. Paik, and L. Cif, Sci. Rep. **13**, 1 (2023).

[25] H.-J. Kim, G.-J. Song, K.-B. Jang, and H.-J. Seo, in *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)* (IEEE) pp. 01–03.

[26] A. Abbas, D. Sutter, C. Zoufal, A. Lucchi, A. Figalli, and S. Woerner, Nat. Comput. Sci. **1**, 403 (2021).

[27] D. T. Mukhamedieva, E3S Web Conf. **494**, 04026 (2024).

[28] View of Meta-Fibonacci Sequences, Binary Trees and Extremal Compact Codes (2024), [Online; accessed 22. Jan. 2024].

[29] 1,2,2,3, 4, 4, 4, 5, 6, 6, 7, 8, 8, 8, 8, 9, 10, 10, 11, 12, 12, 12, 13, 14, 14, 15, 16, 16, 16, 16, 16, 17 - OEIS (2024), [Online; accessed 22. Jan. 2024].

[30] F. Vatan and C. Williams, Phys. Rev. A **69**, 032315 (2004).

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, Journal of Machine Learning Research **12**, 2825 (2011).

[32] D. Chicco, BioData Mining **10**, 1 (2017).

[33] T. Fawcett, Pattern Recognit. Lett. **27**, 861 (2006).

[34] D. Chicco and G. Jurman, BMC Genomics **21**, 1 (2020).

[35] N. Japkowicz and S. Stephen, Intell. Data Anal. **6**, 429 (2002).

[36] L. A. Jeni, J. F. Cohn, and F. De La Torre, in *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction* (IEEE, 2013) pp. 245–251.

[37] M. Sokolova and G. Lapalme, Information Processing & Management **45**, 427 (2009)
.

[38] S. Thanasilp, S. Wang, M. Cerezo, and Z. Holmes, Nature Communications **15**, 5200 (2024)
.

[39] R. K. Jha, N. Kasabov, S. Bhattacharyya, D. Coyle, and G. Prasad, Sci. Rep. **16**, 8142 (2026)
.

[40] M. Y. El Hafidi, A. Toufah, and M. A. Kadim, Online resource, 2025
.

[41] E. Akpinar, Proc. IEEE HPEC 1–6 (2024)
.

[42] D. Alvarez-Estevez, IEEE Trans. Quantum Eng. **6**, 1–15 (2025)
.

[43] C. Boy, E. Altamura, D. Manawadu, I. Tavernelli, S. Mensa, and D. J. Wales, Mach. Learn.: Sci. Technol. **6**, 045076 (2025)
.

[44] S. Chandrasekhar, S. R. Pokhrel, and N. Singh, arXiv:2510.06563 (2025)
.

[45] V. Novák, I. Zelinka, L. Přibylová, L. Martínek, and V. Benčurik, arXiv:2506.01708 (2025)
.

[46] E. Choi, J. Sul, J. E. Kim, S. Hong, B. Izquierdo Gonzalez, P. Cembellin, and Y. Wang, Manuf. Lett. **41**, 1415–1422 (2024)
.

[47] S. Kwon, J. Huh, S. J. Kwon, S.-H. Choi, and O. Kwon, Symmetry **17**, 186 (2025)
.

[48] N. Innan, M. A.-Z. Khan, and M. Bennai, Int. J. Quantum Inf. **22**, 2350044 (2024)
.

[49] N. Singh and S. R. Pokhrel, IEEE Trans. Artif. Intell. 1–13 (2025)
.

[50] P. Bermejo, P. Braccia, M. S. Rudolph, Z. Holmes, L. Cincio, and M. Cerezo, PRX Quantum (2026)
.

Appendix A: On the Suitability of Regular Machine Learning Metrics

In the realm of machine learning, the evaluation of model performance is paramount. Common metrics such as accuracy, precision, recall, and F1 score are frequently employed to gauge the efficacy of models. However, these metrics, while widely accepted, are not universally applicable across all scenarios.

Imbalanced datasets, where one class significantly outnumbers the other, present a unique challenge for machine learning models and the evaluation metrics used to assess their performance. This is particularly true for binary classification tasks, where the minority class is often of greater interest than the majority class.

1. Limitations of Conventional Metrics

Traditional performance metrics, such as accuracy, can be misleading in the context of imbalanced datasets [33]. Consider a dataset with 95% samples of class A and only 5% samples of class B. A naive classifier predicting all samples as class A will still achieve a superficially high accuracy of 95%. This demonstrates that accuracy alone is not sufficient to evaluate model performance on imbalanced datasets [34, 35].

Similarly, other metrics such as recall, precision, and the F1 score can sometimes provide a skewed perspective when classes are imbalanced. There are situations where models can achieve high values for these metrics by being biased towards the majority class, rendering them less effective as measures of model performance [36, 37]. This is why we employ the Matthews Correlation Coefficient (MCC), as described in the Methods section.

2. Numerical Comparison:

The bar plot (Fig. 14) comparing the accuracy and MCC scores across the three datasets Ionosphere, Parkinson's Disease, and Stellar Classification illustrates why the MCC score can be a more informative metric than accuracy, particularly in specific contexts such as imbalanced datasets or when true negatives are significant.

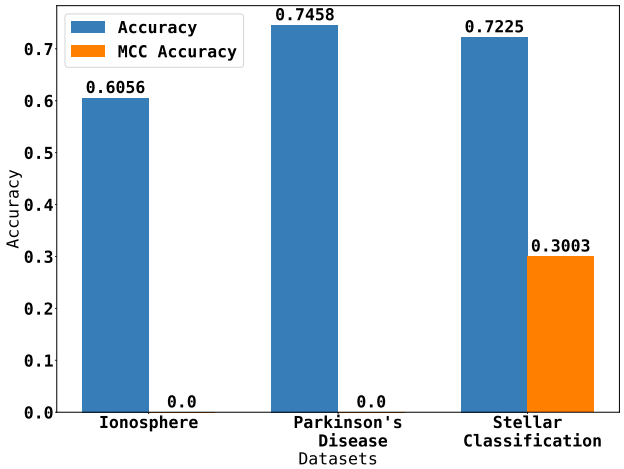


FIG. 14. Evaluating accuracy (left bar, blue) vs. MCC (right bar, orange) across Ionosphere, Parkinson's Disease, and Stellar Classification datasets. Note that the MCC bar vanishes for the left two datasets.

Firstly, in the Stellar Classification dataset, while the accuracy is relatively high (0.7225), the MCC score is notably lower (0.3003). This disparity suggests that although the model has a high rate of correct predictions

(accuracy), its ability to balance true and false positives and negatives is less proficient, as captured by the MCC score. MCC takes into account all four categories of the confusion matrix (true positives, false positives, true negatives, and false negatives), offering a more balanced measure of the quality of binary classifications, especially in cases where class distribution is uneven.

Secondly, for the other two datasets, the MCC scores are zero despite the accuracies being 0.6056 and 0.7458, respectively. This indicates a situation where the model might be making correct predictions by chance or through bias towards the majority class, a common issue in imbalanced datasets. In such scenarios, accuracy alone can be misleading, as it does not distinguish between the types of errors made by the model. MCC, on the other hand, by being sensitive to the balance among all four confusion matrix categories, gives a more faithful representation of the model's performance.

3. Conclusion

When assessing machine learning models on datasets with imbalances, exclusive dependence on conventional metrics can result in skewed insights. The MCC presents a more encompassing and impartial assessment criterion, aptly fitting for tasks with class disparities. It is pivotal to select a performance measure that resonates with the distinct demands and traits of the problem at hand. Particularly in quantum machine learning, given the distinct nature of data and the nuances of the models, a mere reliance on metrics established for tasks other than the one at hand may fall short in truly capturing the model's efficacy.

Appendix B: ZFeatureMap vs. ZZFeatureMap

The ZFeatureMap is a simple feature map that involves rotations around the z axis of the Bloch sphere. On the other hand, the ZZFeatureMap is made from two-qubit gates that involve both rotations around the z axis and controlled-X (CNOT) gates. This makes the ZZFeatureMap more expressive and capable of capturing more complex relationships in the data.

The primary reason for preferring the ZZFeatureMap over the ZFeatureMap in quantum machine learning is the former's ability to generate entanglement between qubits. Entanglement is a uniquely quantum phenomenon and is believed to be one of the reasons quantum algorithms can outperform classical ones [6, 16]. By using the ZZFeatureMap, quantum machine learning algorithms can leverage this entanglement to potentially achieve better performance.

While both the ZFeatureMap and ZZFeatureMap have their applications, the latter is generally preferred in quantum machine learning due to its ability to capture more complex data relationships and leverage quantum entanglement.

Appendix C: Repeating the kernel to reupload the same data

1. Seven features

In Table IV, we present the efficacy of two quantum machine learning kernels, CPMap and ZZFeatureMap, across a variety of datasets, evaluated using the Matthews Correlation Coefficient (MCC). Each dataset was reduced to seven principal components, except for the Titanic dataset, where PCA was not applied, as indicated by an asterisk. The MCC scores serve as a quantitative measure of model performance, with a score of zero indicating no learning.

Regarding the number of repetitions (NOR), CPMap consistently outperformed ZZFeatureMap, even when only a single repetition was used. Specifically, the MCC scores for ZZFeatureMap dropped to zero for some datasets, suggesting no learning occurred, a situation not observed with CPMap. For ZZFeatureMap, the optimum performance was achieved with a single repetition, while additional repetitions led to a decrease in MCC scores,

contrary to expectations. In contrast, CPMap's performance was enhanced with repetitions, as demonstrated by higher MCC scores across several datasets. This pattern also underscores the variable impact of the number of repetitions on different algorithms and the importance of tailored approaches in model training.

Dataset	NOF Used	NOR for CPMap	NOR for ZZFeatureMap	MCC Score CPMap	MCC Score ZZFeatureMap	MCC Score CPMap with 1 Reps
Ionosphere	7	1	1	0.643	0.0	0.643
Breast cancer Diagnostic	7	2	1	0.944	0.869	0.926
Credit card (Balanced)	7	1	1	0.831	0.602	0.831
Parkinson's disease (PD)	7	2	1	0.520	0.0	0.451
Stellar Classification	7	2	1	0.791	0.300	0.778
Heart Disease	7	2	1	0.664	0.449	0.621
Titanic	7*	1	1	0.561	0.351	0.561

TABLE IV. Data and model characteristics for datasets with seven features. NOF: Number of features; NOR: Number of repetitions; *without PCA.

2. All features

In Table V, we present the efficacy of the CPMap in learning tasks across six datasets, emphasizing the model's robustness without substantial feature reduction, each with varying requirements for quantum resources, especially the number of repetitions of the CPMap circuits. The datasets under consideration included Ionosphere, Breast Cancer Diagnostic, Credit Card (Balanced), Parkinson's Disease (PD), Stellar Classification, and Agaricus Lepiota Mushrooms. The number of features utilized ranged from 16 to 30, with an asterisk indicating datasets where Principal Component Analysis (PCA) was not employed. The quantum resource requirement, measured in qubits, was predominantly 12, except for Stellar Classification, which required 9. The assessment metric, MCC, was reported for multiple repetitions of the CPMap and a singular repetition to investigate the impact of repetition on performance. Notably, the MCC scores exhibited a broad spectrum, with Parkinson's Disease recording the lowest at 0.672, while Agaricus Lepiota Mushrooms achieved an impressive high of 0.996. This variance underscores the model's performance sensitivity to the inherent characteristics of the dataset.

Dataset	NOF Used	NOR for CPMap	Qubits Required	MCC Score	MCC Score with 1 repetition	Accuracy
Ionosphere	22	2	12	0.866	0.832	0.930
Breast cancer Diagnostic	30*	2	16	0.943	0.908	0.974
Credit card (Balanced)	22	5	12	0.950	0.942	0.934
Parkinson's disease (PD)	22*	1	12	0.672	0.672	0.881
Stellar Classification	16	3	9	0.956	0.946	0.982
Agaricus Lepiota Mushrooms	21*	6	12	0.996	0.910	0.998

TABLE V. Data and model characteristics for datasets with all features. NOF: Number of features; NOR: Number of repetitions; *without PCA.

Appendix D: Results for higher number of features

This section presents findings using exclusively the CPMap and a greater number of features, including the full set. In this comparison, the ZZFeatureMap was not utilized, as simulation attempts on our computers consistently resulted in memory errors. This suggests that employing ZZFeatureMap is computationally demanding.

Figure 15 presents a comparison of MCC scores for several datasets, with all the features, encompassing Parkinson's Disease with 22 features (PD22), Breast Cancer with 30 features (BC30), Agaricus Lepiota Mushroom with 21 features (ALM21), and Stellar Classification with 17 features (SC17). For the Stellar Classification dataset specifically, the analysis was focused on two classes, totaling 81,039 entries. The bar graph demonstrates that CPMap consistently achieves high MCC scores across these datasets, showcasing its capacity to effectively manage and interpret datasets with a substantial number of features. These results indicate that CPMap has the potential to surpass traditional kernel functions, affirming its suitability for complex, large-scale data analysis.

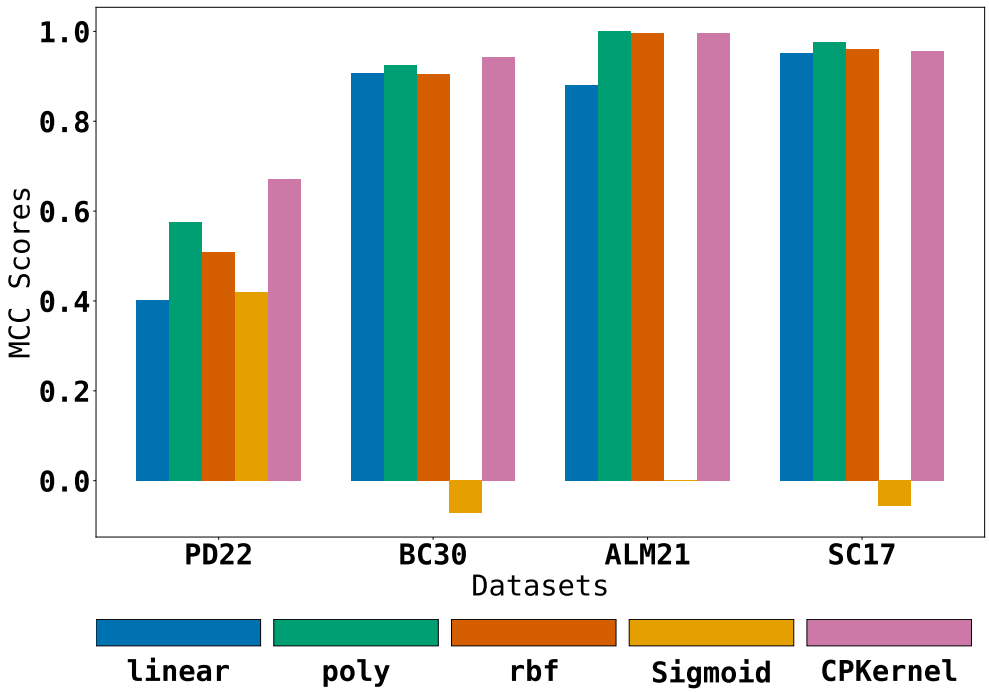


FIG. 15. Comparative Analysis of MCC Scores for various datasets with all the Features, including Parkinson's Disease with 22 features (PD22), Breast Cancer with 30 features (BC30), Agaricus Lepiota Mushroom with 21 features (ALM21) and Stellar Classification with 17 features (SC17).

Appendix E: Comparison with Data Re-Uploading Feature Maps

In the main text, we benchmark CPMap against the widely used ZZFeatureMap [6]. To further strengthen the positioning of CPMap within the broader landscape of resource-efficient quantum embeddings, we additionally benchmark CPMap against six fixed (non-trained) circuit variants inspired by the data re-uploading framework of Pérez-Salinas *et al.* [19]. Data re-uploading was originally introduced as a *trainable* model in which classical inputs are injected at multiple layers and trainable parameters are optimized. Here, to remain within the *fixed-embedding* kernel setting studied throughout this paper, we use the same circuit templates but *freeze* the additional rotation offsets: the offsets are sampled once at initialization and then held fixed for all runs. This yields a deterministic kernel matrix for a given dataset and initialization, enabling a controlled comparison of encoding strategies without introducing additional trainable degrees of freedom.

Base circuits and DR variants: We consider two base encoding blocks (Fig. 16) designed to embed a $d = 7$ feature vector under different resource constraints. **Circuit 1** uses a one-qubit-per-feature strategy across 7

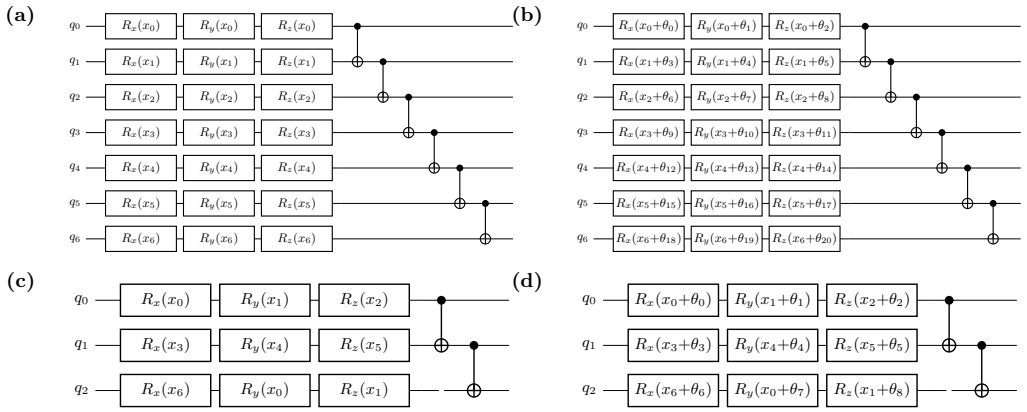


FIG. 16. Base circuit blocks used to construct the frozen data re-uploading feature maps DR1–DR6. **(a)** Circuit 1 (7 qubits): one-qubit-per-feature encoding with successive single-qubit rotations per feature and a linear entangling chain. **(b)** Circuit 1 with frozen random offsets θ added to rotation angles (random-feature style kernel). **(c)** Circuit 2 (3 qubits): feature packing that distributes the $d = 7$ features across qubit–rotation pairs via modular indexing, followed by linear entanglement. **(d)** Circuit 2 with frozen random offsets. DR1–DR6 are obtained by repeating these base blocks three or six times as described in the text.

qubits: each qubit q_i encodes feature x_i via a sequence of single-qubit rotations (e.g., R_x, R_y, R_z), followed by a linear entangling chain to introduce inter-qubit correlations. **Circuit 2** implements a feature-packing strategy, encoding all 7 features into only 3 qubits by distributing feature indices across qubit-rotation pairs using modular indexing, again followed by a linear entangling pattern. For each base circuit we consider two parameterizations: a *parameter-free* version and a *frozen-offset* version in which rotation angles are augmented by fixed random offsets θ sampled once and then held constant.

From these blocks we define six re-uploading feature maps DR1–DR6 by repeating the base blocks multiple times: DR1/DR2 repeat Circuit 1 three and six times, respectively; DR3/DR4 repeat the frozen-offset variant of Circuit 1 three and six times; DR5/DR6 repeat Circuit 2 three times (with and without frozen offsets). The depth-matched variants DR2 and DR4 were included specifically to match CPMap's circuit depth (34 in the $d = 7$ setting), so that performance differences can be interpreted as differences in encoding structure rather than simply additional circuit depth. Circuit resources for all feature maps are summarized in Table VI. In all cases, the quantum kernel is given by 11, and is used as a precomputed kernel in a support vector classifier; all learning takes place in the classical SVC on top of the fixed quantum feature map.

TABLE VI. Quantum circuit resources required by CPMap and the six frozen data re-uploading feature maps (DR1–DR6) for a $d = 7$ input. Circuit depth and CX gate count are reported for the full feature map after all repetitions. DR2 and DR4 are depth-matched to CPMap (depth 34).

Feature Map	**Qubits**	**Depth**	**CX Count**	**Repetitions**
CPMap	7	34	21	1
DR1	7	19	18	3
DR2	7	34	36	6
DR3	7	19	18	3
DR4	7	34	36	6
DR5	3	15	6	3
DR6	3	15	6	3

Classification results

We evaluate CPMap and the six DR variants on five benchmark datasets using the Matthews correlation coefficient (MCC), which is robust to class imbalance and summarizes all entries of the confusion matrix. Figure 17 reports the resulting MCC values.

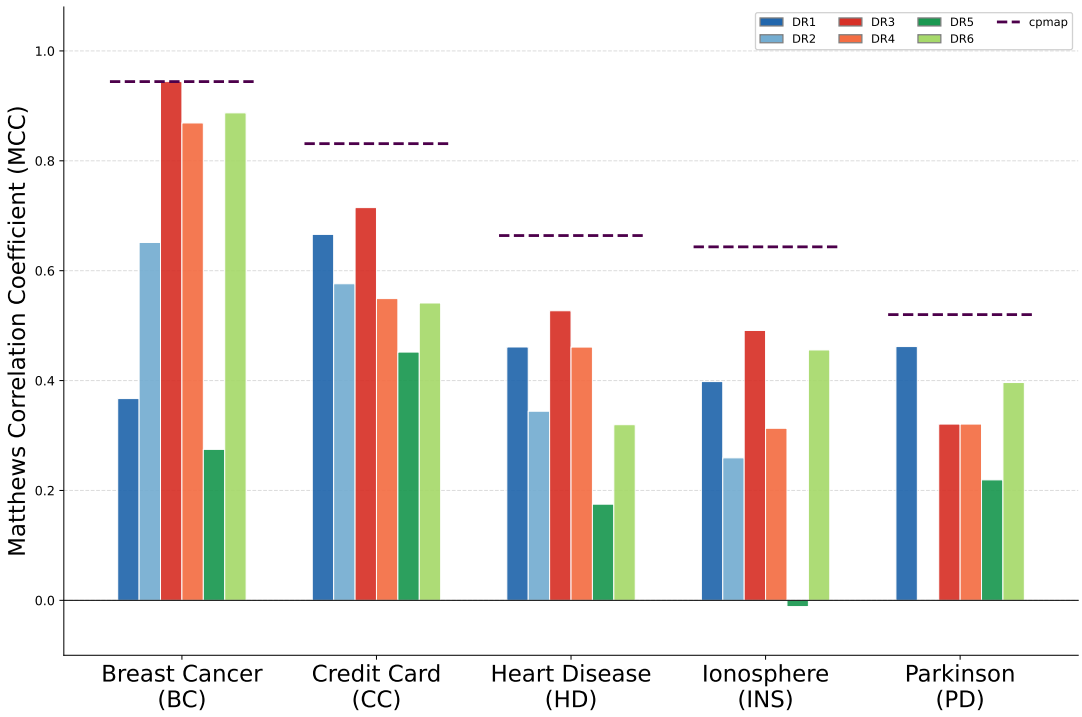


FIG. 17. MCC achieved by CPMap (dashed line) and the frozen data re-uploading feature maps DR1–DR6 across five benchmark datasets. Each configuration is evaluated on the same fixed train-test split. Higher MCC indicates stronger classification performance; values near zero correspond to near-random predictions.

Across the evaluated datasets, CPMap matches or exceeds the performance of all six frozen re-uploading variants. On **Breast Cancer**, the strongest DR variant reaches MCC $= 0.944$, matching CPMap; this is the only dataset where a generic re-uploading construction fully recovers CPMap's performance. On **Credit Card**, **Heart Disease**, **Ionosphere**, and **Parkinson's Disease**, CPMap exceeds the best DR variant by 0.165, 0.137, 0.245, and 0.124 MCC points, respectively. Notably, the depth-matched variants DR2 and DR4 (depth 34) do not consistently improve over their shallower counterparts DR1 and DR3, and remain below CPMap on all datasets except Breast Cancer. This observation is important: it indicates that the performance gap is not explained merely by additional depth in a re-uploading architecture, but reflects the benefit of CPMap's structured encoding strategy under resource constraints.

Takeaway: These additional baselines strengthen the main conclusion of the paper: CPMap provides a competitive, resource-efficient kernel embedding that remains robust when compared not only to standard Pauli-style feature maps, but also to deeper re-uploading-style embeddings in a controlled (fixed, non-trained) setting.

Appendix F: Multi-class evaluation on the Stellar Classification dataset

To assess whether CPMap remains effective beyond binary classification, we additionally evaluate it on a genuine *three-class* task using the Stellar Classification dataset. We construct a balanced subset by sampling 500 examples from each class (total $N = 1500$) and follow the same preprocessing protocol used elsewhere in the paper: features are standardized and then reduced to $d = 12$ components using PCA fit on the training and test splits. We then compute the CPMap kernel matrices using the same fixed CPMap hyperparameters as in the main experiments and train a multi-class SVM with a precomputed kernel using a one-vs-rest strategy:

`SVC(kernel="precomputed", decision_function_shape="ovr", C=C_SVM).`

On this 3-class evaluation, CPMap achieves the following performance:

$$\text{MCC} = 0.650, \qquad \text{Balanced Accuracy} = 0.767, \qquad F1_{\text{macro}} = 0.768, \qquad F1_{\text{weighted}} = 0.768.$$

These results provide evidence that CPMap can be applied to multi-class settings without modifying the feature map design or its hyperparameters, complementing the binary benchmarks reported in the main text.

Appendix G: CPMap hyperparameters and robustness

CPMap hyperparameters and default setting

CPMap is specified by a small set of fixed design angles that parameterize the two-qubit interaction blocks used in the feature map. These angles are not trained and therefore function as hyperparameters. We denote the full CPMap hyperparameter vector by

$$\Theta_{\text{CP}} = (\alpha_C, \beta_C, \gamma_C, \alpha_P, \beta_P, \gamma_P), \tag{G1}$$

where $(\alpha_C, \beta_C, \gamma_C)$ parameterize the C-block and $(\alpha_P, \beta_P, \gamma_P)$ parameterize the P-block (see main text for the circuit definition).

Unless stated otherwise, we use the default setting

$$(\alpha_C, \beta_C, \gamma_C, \alpha_P, \beta_P, \gamma_P) = \left(-\frac{\pi}{3}, \frac{\pi}{6}, -\frac{\pi}{9}, \frac{\pi}{7}, \frac{\pi}{9}, -\frac{\pi}{7}\right), \tag{G2}$$

which is a fixed, non-trained configuration chosen prior to benchmarking. The purpose of this section is to verify that CPMap does not rely on a finely tuned hyperparameter choice and that parameter variations induce systematic changes in the induced kernel geometry.

All hyperparameter analyses use the Breast Cancer dataset restricted to a fixed $n = 100$ subset (same subset across all experiments) and the same kernel SVM evaluation pipeline. For each hyperparameter setting we evaluate performance over 50 stratified train/test splits and report the mean Matthews correlation coefficient (MCC) with 95% confidence intervals (CI). Unless stated otherwise, all other aspects of the pipeline (preprocessing, feature dimension, and circuit depth) are held fixed to isolate the effect of the CPMap angles.

a. Why we set local KAK factors to the identity.

In the Cartan/KAK form $U = (u_1 \otimes u_2)\exp\left[-\frac{i}{2}(c_x X\otimes X + c_y Y\otimes Y + c_z Z\otimes Z)\right](v_1 \otimes v_2)$, the local unitaries (u_1, u_2, v_1, v_2) do not affect the nonlocal invariants (c_x, c_y, c_z) but they add additional single-qubit layers. In CPMap we set these local factors to the identity to obtain a canonical minimal representative of each interaction family (isotropic/XY/ZZ) and to keep the feature map as resource-efficient as possible. This choice reduces depth and calibration overhead and keeps the embedding in the "fixed, non-trained" regime considered in this work. The sensitivity studies in Supplementary A-B show that CPMap remains robust and that its remaining angles systematically modulate kernel geometry and performance.

Structured parameter sweeps: sensitivity of performance to block parameters

To test robustness in a controlled manner, we perform structured one-parameter sweeps over a representative sweep angle θ within each of three interaction families (isotropic, XY-type, and ZZ-type). We examine sensitivity separately for the two CPMap blocks:

- **Varying the C-block:** we sweep the designated C-block angle while holding the remaining angles (including all P-block angles) fixed.

- **Varying the P-block:** we sweep the designated P-block angle while holding the remaining angles (including all C-block angles) fixed.

Interaction-family terminology (isotropic, XY-type, ZZ-type). In our structured sweep we group two-qubit blocks into three *interaction families*, which describe the effective Pauli-coupling content of the entangling operation. Concretely, any two-qubit unitary can be expressed (up to local unitaries) via a Cartan/KAK form

$$U \equiv (u_1 \otimes u_2) \exp\left[-\frac{i}{2}(c_x X\otimes X + c_y Y\otimes Y + c_z Z\otimes Z)\right](v_1 \otimes v_2), \tag{G3}$$

where X, Y, Z are Pauli operators and (u_1, u_2, v_1, v_2) are single-qubit unitaries. The triple (c_x, c_y, c_z) determines the nonlocal "interaction type" of the gate.

- **Isotropic (balanced mixing).** A balanced nonlocal coupling with comparable strengths along the three Pauli axes, i.e., $c_x \approx c_y \approx c_z$.

- **XY-type.** Exchange-like coupling dominated by $X\otimes X$ and $Y\otimes Y$, i.e., $c_x \approx c_y \neq 0$ and $c_z \approx 0$.

- **ZZ-type.** Ising-like coupling dominated by $Z\otimes Z$, i.e., $c_z \neq 0$ with $c_x \approx c_y \approx 0$.

In the sweep experiments, "isotropic/XY/ZZ" indicates which Pauli-coupling content we vary within the two-qubit block, while keeping the remainder of the feature map fixed. This terminology is used only to concisely describe the dominant nonlocal generator of the chosen two-qubit sub-block and does not imply that the physical device natively implements these Hamiltonians.

Figure 18 reports MCC (mean $\pm$ 95% CI) as a function of θ for each interaction family. Across both blocks and all families, performance varies smoothly and remains within a narrow band over a broad range of θ. This indicates that CPMap does not depend on a single finely tuned parameter value and that moderate deviations from the default configuration do not cause performance collapse.

Paired comparison: improvements are consistent across splits

To verify that improvements are not driven by a small number of favorable splits, we perform a paired split-wise comparison of alignment between a selected strong CPMap configuration (*best*) and the default configuration (*baseline*) using the same 50 splits. Figure 19 plots split-wise alignment values (best vs baseline). Most points lie above the diagonal, indicating that alignment improvements are consistent across splits rather than arising from outliers. This supports the conclusion that CPMap hyperparameters control kernel geometry in a reproducible way.

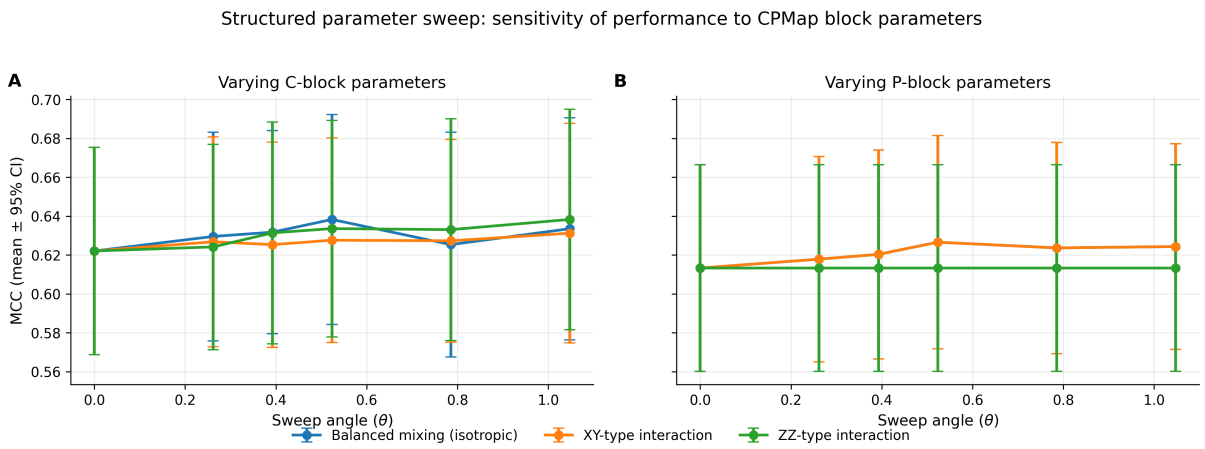


FIG. 18. **Structured parameter sweep: sensitivity of performance to CPMap block parameters (Breast Cancer, $n = 100$).** MCC (mean $\pm$ 95% CI over 50 stratified splits) as a function of sweep angle θ for three interaction families. (A) Varying the C-block while keeping the P-block fixed. (B) Varying the P-block while keeping the C-block fixed. The relatively flat response across a wide range of θ indicates robustness to hyperparameter variation.

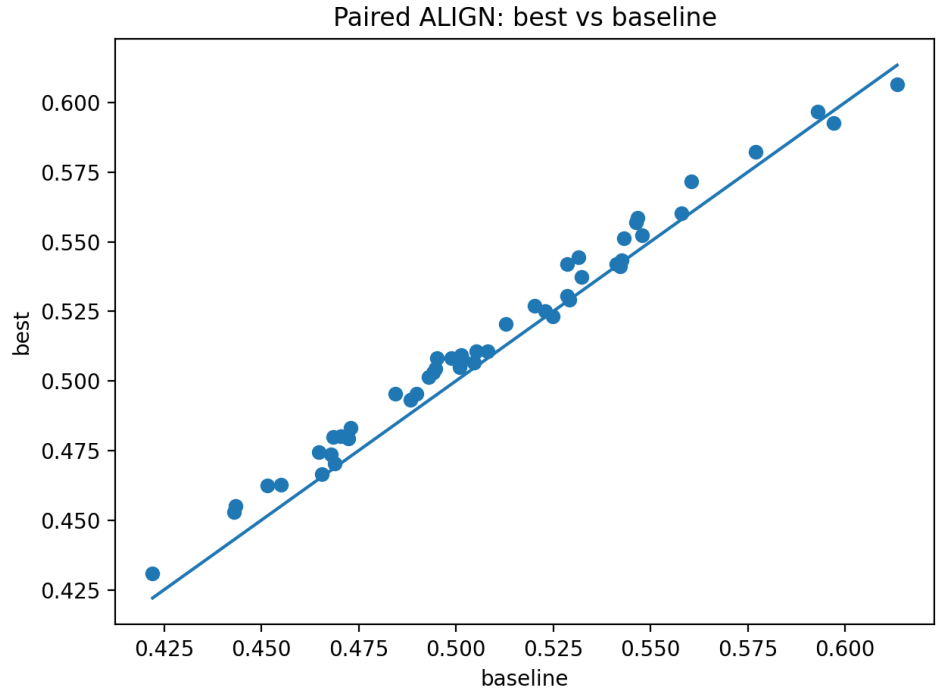


FIG. 19. **Paired alignment comparison: best vs baseline (Breast Cancer, $n = 100$).** Each point corresponds to one of the 50 stratified splits, showing the centered kernel–label alignment of a selected strong hyperparameter configuration (y-axis) versus the baseline configuration (x-axis). Points above the diagonal indicate splits where the selected configuration yields higher alignment. The consistent upward shift indicates robust improvement across splits.

Summary. The structured sweeps (Fig. S1) demonstrate that CPMap is not brittle with respect to its fixed design angles. The random screening (Fig. S2) shows that these angles systematically modulate kernel geometry and performance, and the paired analysis (Fig. S3) confirms that the effect is consistent across splits. Together these results address the concern that CPMap hyperparameters are ad-hoc by establishing robustness and providing an interpretable diagnostic (alignment) for selecting reasonable parameter regimes.

Appendix H: Coherence feasibility and runtime interpretation (hardware)

Gate-schedule duration vs. coherence. To assess whether the executed circuits are feasible within the coherence window of current superconducting hardware, we use the backend timing model to estimate the *gate-schedule duration* per shot for each compiled circuit. Concretely, after transpiling each overlap circuit to the target backend, we extract the estimated duration using Qiskit's timing estimator (reported in our logs as `duration_median_s`) and summarize it by the median across the 100 circuits in a batch. Figure 20 (left) reports the resulting median per-shot circuit execution time (in μs) as a function of feature dimension. We compare this time scale to the device coherence times using the ratio

$$\eta = \frac{t_{\text{gate}}}{\min(T_1, T_2)}, \tag{H1}$$

where t_{gate} denotes the median gate-schedule duration per shot and T_1, T_2 are the median relaxation and dephasing times from the backend properties. Figure 20 (right) shows that η remains well below unity across all tested dimensions (dashed line at $\eta = 1$), indicating that the compiled gate schedule fits comfortably within the coherence window. Therefore, the degraded performance observed at larger feature dimensions is unlikely to be explained by circuit duration exceeding coherence, and is more consistent with error accumulation from two-qubit gates, readout error, crosstalk, and other gate-level noise mechanisms.

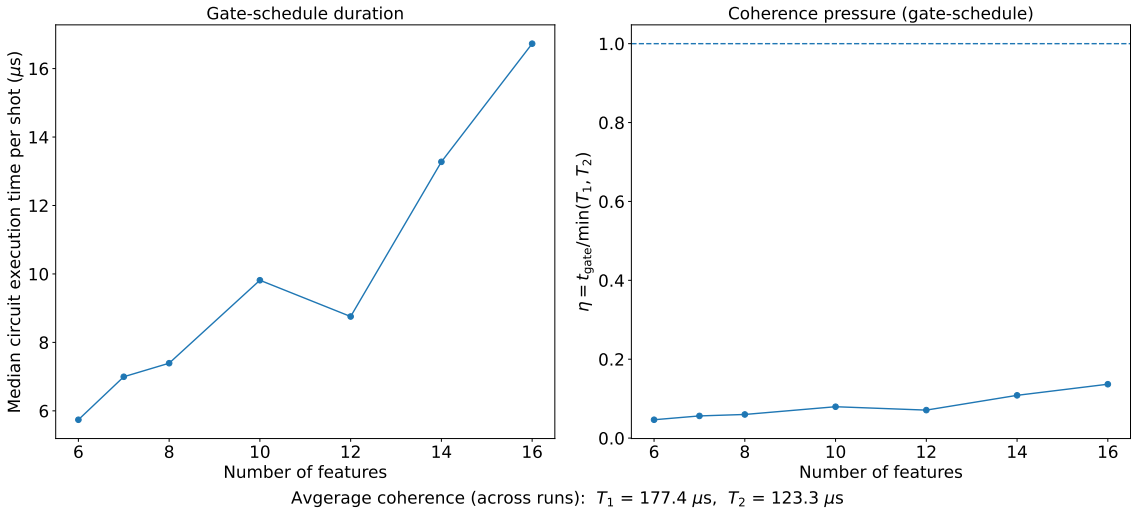


FIG. 20. **Gate-schedule duration and coherence pressure for hardware kernel circuits. (Left)** Median estimated gate-schedule duration per shot (in μs) of the transpiled circuits as a function of the number of encoded features. Each point summarizes the median across the batch of circuits executed for that feature dimension. **(Right)** Corresponding coherence-pressure ratio $\eta = t_{\text{gate}}/\min(T_1, T_2)$, where t_{gate} is the median gate-schedule duration per shot and T_1, T_2 are the backend relaxation and dephasing times (median across runs; values reported below the panels). The dashed line at $\eta = 1$ marks the boundary where the scheduled circuit duration matches the coherence window. Across all tested feature dimensions, $\eta \ll 1$, indicating that the compiled schedules remain well within the coherence time budget; performance degradation at larger feature dimensions is therefore unlikely to be explained by circuit duration exceeding coherence.

4

Quantum Reservoir Computing

Chapter Introduction

The final contribution of my doctoral research focuses on reservoir computing, a powerful machine learning paradigm originally developed in the classical setting for efficiently processing temporal and dynamical data. Classical reservoir computing methods, such as echo state networks, are valued for their simplicity, strong performance on time-series tasks, and minimal training overhead. However, their expressivity and memory are ultimately limited by classical dynamics, motivating the exploration of quantum systems as reservoirs.

In this work, we propose a quantum reservoir computing (QRC) scheme that repurposes a quantum feature map as the reservoir nonlinearity and induces temporal memory via an

explicit feedback loop.

This paper represents the third core research contribution of my thesis. While the first two contributions focused on developing resource-efficient static models (coherent quantum neural networks and quantum kernels), this work extends those ideas to temporal and dynamical learning, an essential capability for real-world applications such as forecasting, control, and signal processing. Together, these contributions form a unified trajectory toward practical quantum machine learning, where models are not only theoretically sound but also compatible with the constraints of noisy intermediate-scale quantum devices.

Preprint Reference

The following preprint will be made available on the arXiv:

Reference: Utkarsh Singh, Aaron Z. Goldberg, Christoph Simon and Khabat Heshami. "Recurrent Quantum Feature Maps for Reservoir Computing" arXiv: 2604.03469 (2026).

Recurrent Quantum Feature Maps for Reservoir Computing

Utkarsh Singh
Department of Physics, University of Ottawa, 25 Templeton Street, Ottawa, Ontario, K1N 6N5 Canada and National Research Council of Canada, 100 Sussex Drive, Ottawa, Ontario K1N 5A2, Canada

Aaron Z. Goldberg
National Research Council of Canada, 100 Sussex Drive, Ottawa, Ontario K1N 5A2, Canada

Christoph Simon
Institute for Quantum Science and Technology, Department of Physics and Astronomy, University of Calgary, Alberta T2N 1N4, Canada and Hotchkiss Brain Institute, University of Calgary, Alberta T2N 1N4, Canada

Khabat Heshami
National Research Council of Canada, 100 Sussex Drive, Ottawa, Ontario K1N 5A2, Canada Department of Physics, University of Ottawa, 25 Templeton Street, Ottawa, Ontario, K1N 6N5 Canada and Institute for Quantum Science and Technology, Department of Physics and Astronomy, University of Calgary, Alberta T2N 1N4, Canada

Reservoir computing promises a fast method for handling large amounts of temporal data. This hinges on constructing a good reservoir–a dynamical system capable of transforming inputs into a high-dimensional representation while remembering properties of earlier data. In this work, we introduce a reservoir based on recurrent quantum feature maps where a fixed quantum circuit is reused to encode both current inputs and a classical feedback signal derived from previous outputs. We evaluate the model on the Mackey-Glass time-series prediction task using our recently introduced CP feature map, and find that it achieves lower mean squared error than standard classical baselines, including echo state networks and multilayer perceptrons, while maintaining compact circuit depth and qubit requirements. We further analyze memory capacity and show that the model effectively retains temporal information, consistent with its forecasting accuracy. Finally, we study the impact of realistic noise and find that performance is robust to several noise channels but remains sensitive to two-qubit gate errors, identifying a key limitation for near-term implementations.

I. INTRODUCTION

Modern machine learning increasingly demands models that can process temporal data efficiently, particularly in settings where computational resources, latency, or energy consumption are constrained. While deep learning architectures have achieved remarkable success in domains such as vision, language, and control, their reliance on large-scale optimization and heavily parameterized models often limits their deployment in real-time and resource-limited environments [1–3]. Reservoir Computing (RC) offers a fundamentally different approach: instead of training complex recurrent dynamics, RC employs a fixed nonlinear dynamical system—the reservoir—to transform input signals into a high-dimensional representation, while only a simple linear readout is trained [4, 5]. This separation of dynamics and learning enables efficient training and has proven effective in tasks such as chaotic time-series prediction, control, and signal processing [3, 6].

Extending this paradigm into the quantum domain has led to the development of Quantum Reservoir Computing (QRC), where quantum systems serve as high-dimensional dynamical reservoirs [7]. By encoding classical inputs into quantum states and evolving them under fixed unitary dynamics, QRC leverages the exponentially large Hilbert space and intrinsic quantum correlations to enhance representational capacity. Early work demonstrated that even disordered quantum systems can match conventional recurrent neural networks on nonlinear temporal tasks [7], motivating a wide range of implementations across physical platforms, including nuclear spin ensembles [8], continuous-variable systems [9], superconducting qubits [10, 11], and large-scale neutral-atom processors [12]. More recent studies have further explored engineered dissipation and analog quantum dynamics to improve memory and scalability [13, 14]. These developments position QRC as a promising framework for temporal learning on near-term quantum hardware.

Even with these advances, QRC still struggles with memory retention after measurement, as quantum observations collapse the state and break the temporal links needed for sequence processing [15, 16]. Workarounds like reinitialization [17], mid-circuit resets, feedback-based schemes [18, 19], or weak measurements [20] help, but often increase complexity. Recent studies instead embrace dissipation and noise as useful features—amplitude damping and loss have been shown to enhance memory and task performance [13, 21]. Other strategies restrict memory artificially to balance efficiency and relevance [22]. Together, these developments aim to preserve QRC's expressivity while making it practical for real-time, hardware-compatible implementations.

In parallel, quantum feature maps and kernel methods [23, 24]—originally developed for static learning tasks—have been shown to possess high expressibility and universal approximation capabilities. These circuits encode classical data into quantum states via fixed, parameterized unitaries and have proven effective in kernel-based quantum classifiers. However, their potential as dynamical systems remains un-

derexplored. Given their ability to map data into structured, high-dimensional quantum feature spaces, an open question is whether such circuits can be repurposed as reservoirs for temporal information processing—especially if coupled with mechanisms that introduce memory and recurrence.

In this work, we address this question by proposing a feedback-driven quantum reservoir architecture based on a reusable quantum feature map circuit. The core idea is to repurpose the fixed quantum circuit by encoding both current inputs and a feedback signal—derived from previous outputs—into separate parts of the circuit. The circuit is divided into two halves: the first encodes a sliding window of input data $\{\mathbf{x}_t, \ldots, \mathbf{x}_{t+\tau}\}$, while the second half encodes a classically computed feedback term from the prior output, scaled by a tunable strength α. This design introduces temporal recurrence into the system without requiring any mid-circuit measurements or resets, thereby preserving the fading-memory property and ensuring linear runtime $\mathcal{O}(L)$. By integrating structured feedback into a fixed quantum circuit, our method bridges the gap between quantum kernel methods and reservoir computing, yielding a compact and expressive temporal learning model.

We evaluate our quantum reservoir architecture on the Mackey-Glass chaotic time-series prediction task. Across a range of delay parameters τ, our model consistently performs on par or better than classical reservoir computers, multilayer perceptrons (MLPs), and linear regression baselines in terms of mean squared error (MSE). We systematically explore how feedback strength α, entanglement, and circuit parameters affect the model's memory capacity and predictive performance. Dynamical stability is confirmed via the echo state property (ESP), and fading memory behavior is validated through standard memory capacity tests.

In summary, this work introduces a versatile quantum reservoir model that simultaneously achieves recurrence, expressivity, and interpretability. By unifying kernel-based quantum learning with dynamic feedback architectures, we take a step toward general-purpose, compact quantum models for real-time, temporal processing. The rest of this paper is organized as follows. In Sec. II, we present a detailed background on reservoir computing, including the echo state property and memory capacity. Sec. III introduces the proposed feedback-driven quantum reservoir architecture, along with the datasets and feature maps used in this work. Sec. IV presents the experimental results, including performance evaluation, noise analysis, and the role of feedback and entanglement in the reservoir dynamics.

II. RESERVOIR COMPUTING

Reservoir computing leverages a fixed, randomly initialized dynamical system—referred to as the *reservoir*—to nonlinearly embed input sequences into a high-dimensional state space, where temporal dependencies can be extracted via simple linear readout mechanisms [4, 25, 26]. A conceptual illustration of this framework is shown in Fig. 2.

The mathematical formulation of an echo state network

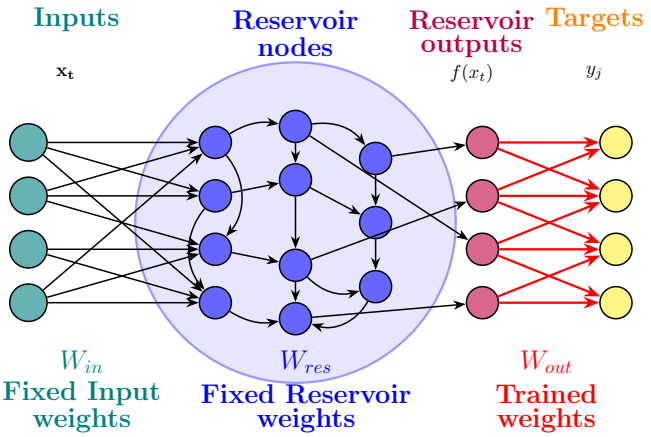


FIG. 1: Schematic representation of the classical reservoir computing framework. The input vector $\mathbf{x}_t$ is projected into a high-dimensional dynamical space by a recurrent network of fixed, randomly connected internal nodes (the reservoir), characterized by weights $\mathbf{W}_{\text{in}}$ and $\mathbf{W}_{\text{res}}$. The resulting reservoir states are then linearly mapped to the target outputs $\mathbf{y}_j$ via a trainable readout layer with weights $\mathbf{W}_{\text{out}}$. Only the output layer is optimized during training, while the reservoir dynamics remain untrained, enabling efficient learning of complex temporal patterns.

(ESN), the most widely studied reservoir computing architecture, can be expressed as follows. Let the input at time step t be denoted by $\mathbf{x}_t \in \mathbb{R}^K$, where K is the input dimension. The internal reservoir state vector $\mathbf{h}_t \in \mathbb{R}^N$ (representing the activations of the N reservoir nodes) evolves according to [25]:

$$\mathbf{h}_{t+1} = f\left(\mathbf{W}_{\text{res}}\mathbf{h}_t + \mathbf{W}_{\text{in}}\mathbf{x}_{t+1} + \mathbf{W}_{\text{fb}}\mathbf{y}_t\right), \tag{1}$$

where $\mathbf{W}_{\text{res}} \in \mathbb{R}^{N\times N}$ defines the recurrent connections within the reservoir, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N\times K}$ encodes the fixed input coupling, and $\mathbf{W}_{\text{fb}} \in \mathbb{R}^{N\times L}$ governs optional feedback from the output $\mathbf{y}_t \in \mathbb{R}^L$. The function $f(\cdot)$ is typically a nonlinear activation such as the hyperbolic tangent [1].

The final output is computed by a linear readout function that maps the current reservoir state (and optionally the input) to the prediction:

$$\mathbf{y}_{t+1} = \mathbf{W}_{\text{out}}\begin{bmatrix}\mathbf{h}_{t+1}\\ \mathbf{x}_{t+1}\end{bmatrix}, \tag{2}$$

where $\mathbf{W}_{\text{out}} \in \mathbb{R}^{L\times(N+K)}$ is learned via standard regression techniques. Crucially, only $\mathbf{W}_{\text{out}}$ is optimized during training, making the approach computationally efficient and well-suited for time-series tasks.

A. Echo State Property and Fading Memory

For a reservoir to be useful, its dynamics must be both stable and input-driven. Two closely related concepts capture this requirement: the echo state property (ESP) and the fading

memory property. The ESP states that, for a given input sequence, the reservoir state should eventually be uniquely determined by the input history rather than by the initial condition [25, 27]. In other words, two trajectories driven by the same input should converge after a transient, even if they start from different initial states. The fading memory property complements this by requiring that the influence of past inputs decays with time, so that recent inputs affect the current state more strongly than distant ones [28–30]. Together, these properties ensure that the reservoir acts as a stable causal filter with finite effective memory.

In this work, we use the ESP and fading memory in their operational sense: the reservoir should forget its initialization, remain driven by the input stream and feedback signal, and retain only a finite but useful memory of the past.

B. Memory Capacity and Computational Power

The memory capacity of a reservoir computing system quantifies its ability to reconstruct past inputs from current reservoir states. Jaeger introduced the linear memory capacity as a measure of how well an ESN can linearly reconstruct delayed versions of its input [31]. For a scalar input $u(t)$, the linear memory capacity is defined as:

$$MC = \sum_{k=1}^{\infty} MC_k \tag{3}$$

where MC_k is the capacity to reconstruct the input delayed by k time steps:

$$MC_k = \frac{(\mathrm{cov}(x_{t-k}, \hat{x}_k(t)))^2}{\mathrm{var}(x_{t-k})\,\mathrm{var}(\hat{x}_k(t))} \tag{4}$$

and $\hat{x}_k(t)$ is the best linear reconstruction of x_{t-k} from the reservoir state $\mathbf{h}_t$ [31].

III. QUANTUM RESERVOIR ARCHITECTURE

We propose a quantum reservoir based on a static quantum feature map circuit, commonly used in kernel-based quantum machine learning. This reservoir consists of a two-part quantum circuit: the **left half** applies a parameterized feature map $U(\mathbf{x}_t)$, which encodes a sliding window of time-series data $\mathbf{x}_t = [x_{t-\tau}, \ldots, x_t]$, while the **right half** applies the inverse circuit $U^\dagger(\mathbf{z}_{t-1})$, encoding a feedback vector derived from the output of the previous time step.

The feedback signal is constructed from single-qubit expectation values obtained from the circuit output distribution. Let $p_t(s)$ denote the probability of observing bitstring $s \in \{0,1\}^{n_q}$ at time step t, where n_q is the number of qubits. The i-th component of the feedback vector is defined as

$$z_{t,i} = \sum_{s\in\{0,1\}^{n_q}} p_t(s)\,(-1)^{s_i}, \tag{5}$$

where s_i denotes the i-th bit of s. This is equivalent to the expectation value $z_{t,i} = \langle Z_i \rangle_t$ of the Pauli-Z operator on qubit i. Collecting all qubit expectations gives the feedback vector

$$\mathbf{z}_t = \big(z_{t,1}, z_{t,2}, \ldots, z_{t,n_q}\big)^\top \in \mathbb{R}^{n_q}. \tag{6}$$

The feedback is then scaled by a feedback strength parameter $\alpha \in [0,1]$:

$$\tilde{\mathbf{z}}_t = \alpha \mathbf{z}_t. \tag{7}$$

The resulting reservoir can be viewed as a discrete-time, input-driven quantum dynamical system. At each time step, the circuit applies a composite unitary

$$U_t = U^\dagger\big(\Pi(\tilde{\mathbf{z}}_{t-1})\big)\, U(\mathbf{x}_t), \tag{8}$$

where $\Pi(\cdot)$ denotes a padding operation that matches the dimensionality of the feature map parameters. The quantum state is initialized as

$$\rho_0 = |0\rangle\langle 0|^{\otimes n_q}, \tag{9}$$

and evolves as

$$\rho_t = U_t \rho_0 U_t^\dagger. \tag{10}$$

Measurement in the computational basis produces a probability distribution

$$p_t(s) = \langle s|\rho_t|s\rangle, \quad s \in \{0,1\}^{n_q}. \tag{11}$$

Two different representations are then extracted from this distribution. The feedback signal is given by the expectation values in Eq. 5, while the regression features are constructed by selecting a subset of the measurement probabilities:

$$\mathbf{r}_t = \big(p_t(s_1), \ldots, p_t(s_{\lfloor \lambda 2^{n_q} \rfloor})\big). \tag{12}$$

Here, $\lambda \in (0,1]$ determines the proportion of the quantum output used for training. Specifically, if the full output vector has dimension m, only the first $\lfloor \lambda m \rfloor$ components are used as input to the regression model. This allows us to adjust the dimensionality of the learning model without modifying the quantum circuit depth or qubit count.

The final prediction is obtained via a linear readout

$$\hat{y}_t = \mathbf{w}_{\mathrm{out}}^\top \mathbf{r}_t + b, \tag{13}$$

where the target is defined as $y_t = x_{t+h}$.

For CPMap, the number of parameters required by the second half of the circuit (feedback section) exceeds the number of qubits. In this case, the feedback vector $\tilde{\mathbf{z}}_t$ is padded with zeros via $\Pi(\cdot)$ to match the required input dimension. In contrast, for the ZZFeatureMap, no padding is required, as the number of parameters matches the number of qubits.

We also consider an alternative feedback mechanism, referred to as *full-state feedback*. In this setting, instead of computing single-qubit expectation values, the feedback signal is

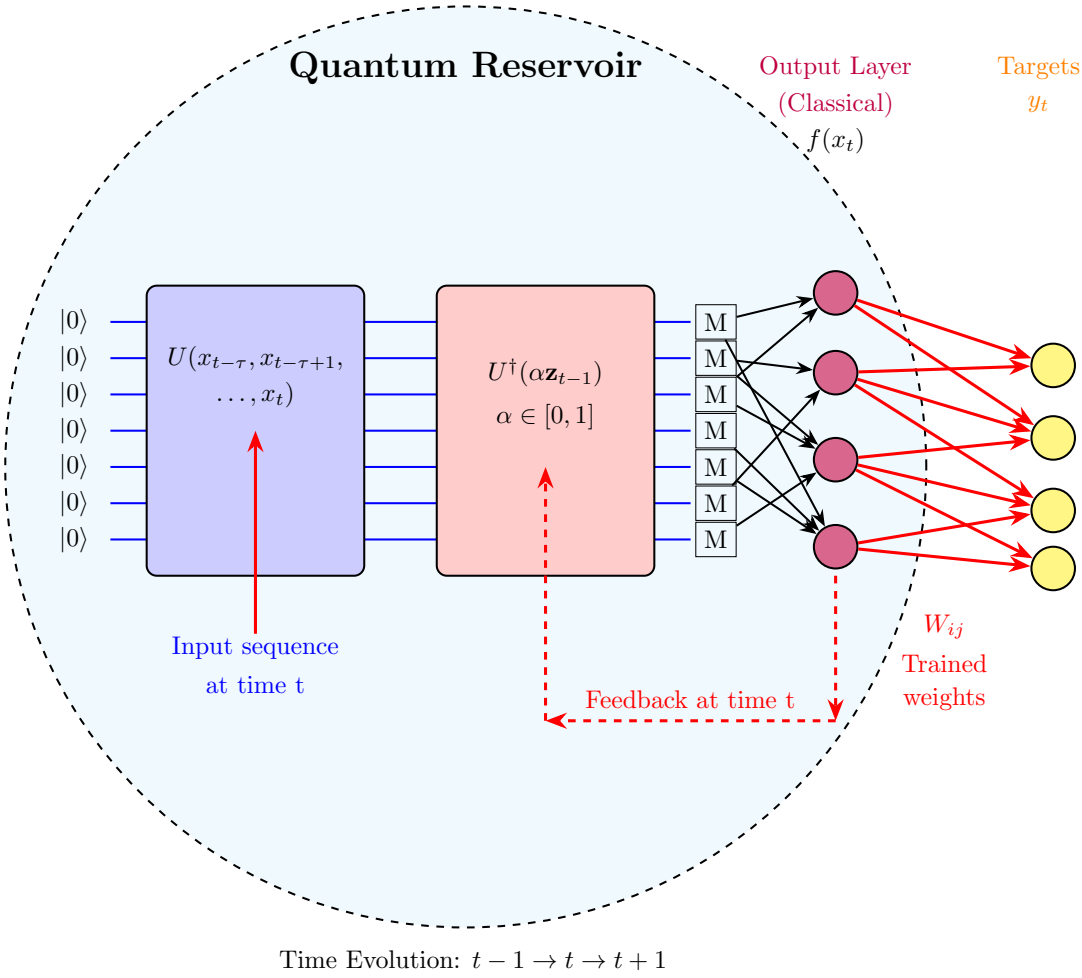


FIG. 2: Schematic of proposed feedback-driven quantum reservoir computing model. At each timestep t, a windowed input sequence $[x_{t-\tau}, \ldots, x_t]$ is encoded into the left half of a quantum feature map circuit $U(\cdot)$, while the right half applies the inverted circuit $U^\dagger(\cdot)$ using a feedback vector derived from the previous reservoir output. The feedback is modulated by a scaling parameter $\alpha \in [0,1]$, with $\alpha = 1$ corresponding to full feedback and $\alpha = 0$ to input-only evolution. The quantum circuit is initialized in $|0\rangle^{\otimes n}$, and measurement outcomes are collected to produce a classical output vector, which goes to the regression model to generate the prediction y_t.

constructed directly from the measurement probability distribution. Specifically, we take the first n components of the probability vector in lexicographic order, where n matches the input dimension, and scale them by the feedback strength parameter α. This vector is then used as the feedback input to the circuit.

This approach avoids explicit computation of single-qubit expectation values and reduces classical post-processing overhead. Details and results are provided in the Supplementary Material.

One must note that all the results presented in the main text of this work are obtained using the feedback mechanism described in Algorithm .

A. Benchmark Dataset: Mackey–Glass System

To evaluate the temporal forecasting performance of our quantum reservoir model, we employ the well-known Mackey–Glass chaotic time series [32]. Originally introduced to model physiological blood flow dynamics, the Mackey–Glass system is governed by the following nonlinear delay differential equation:

$$\frac{dx(t)}{dt} = b\frac{x(t-\tau)}{1+x(t-\tau)^n} - cx(t). \tag{14}$$

where b, c, τ, n are positive constants. For appropriate parameter values (e.g., $b = 0.2, c = 0.1, n = 10, \tau = 17$), this system exhibits deterministic chaos, characterized by sensitivity to initial conditions and complex temporal correlations. These properties make it a canonical benchmark for evaluating memory and prediction capacity in recurrent and reservoir comput-

Algorithm: Feedback-Driven Quantum Reservoir Computing

Input: Time-series input $\{x_t\}_{t=1}^{T}$, feature map $U(\cdot)$, window size τ, feedback strength α, output fraction λ, prediction horizon H

Output: Predicted outputs $\{\hat{y}_t\}_{t=\tau+1}^{T-H}$

Initialize feedback vector $\tilde{\mathbf{z}}_0 \leftarrow \mathbf{0} \in \mathbb{R}^{n_q}$

for $t = \tau+1$ **to** $T-H$ **do**

- Create input window: $\mathbf{x}_t = [x_{t-\tau}, \ldots, x_t]$
- Encode input with feature map: apply $U(\mathbf{x}_t)$ to left half of the circuit
- Encode feedback from previous step: apply $U^{\dagger}(\tilde{\mathbf{z}}_{t-1})$ to right half
- Execute full quantum circuit and measure output distribution $\mathbf{p}_t$
- Compute feedback vector $\mathbf{z}_t$ using Eq. 5
- Update feedback: $\tilde{\mathbf{z}}_t = \alpha \cdot \mathbf{z}_t$
- Construct regression features: $\mathbf{r}_t = (p_t(s_1), \ldots, p_t(s_{\lfloor \lambda 2^{n_q} \rfloor}))$;
- Store selected components as quantum reservoir state vector for training
- Store $(\mathbf{r}_t, y_t)$ where $y_t = x_{t+H}$

Train regression model: $\hat{y}_t = \mathbf{w}_{\text{out}}^{\top}\mathbf{r}_t + b$

return predicted sequence $\{\hat{y}_t\}_{t=\tau+1}^{T-H}$

ing models.

To simulate the system, we numerically integrate Eq. (14) using a Runge–Kutta method with a discretization step Δt, and sample points at uniform intervals to form a univariate time series $\{x_t\}$. The series is normalized to the interval $[0,1]$ or $[0,\pi]$, depending on the input encoding scheme of the quantum feature map.

B. Feature Maps

Our quantum reservoir is constructed using fixed, parameterized circuits originally designed for quantum kernel methods. Specifically, we incorporate two types of feature maps: the CPMap [33] and the standard ZZFeatureMap [34].

a. CPMap: The CPMap is a structured, resource-efficient quantum feature map inspired by the architectural layout of quantum convolutional neural networks (QCNNs). It enables the encoding of a high number of classical features onto a limited number of qubits by using unitary transformations that coherently compress information. The CPMap alternates between data-encoding rotation layers and carefully designed two-qubit entangling blocks that focus features from multiple qubits into fewer ones. As a result, it supports hierarchical feature encoding and can embed approximately $2n$ classical features using only n qubits. More importantly, CPMap significantly reduces quantum resource requirements compared to standard maps, requiring only $\sim 9F/2$ CNOT gates to encode F features—far fewer than the quadratic CNOT scaling of the ZZFeatureMap. In this study, CPMap serves as the default circuit for both the input and feedback encoding in our quantum reservoir.

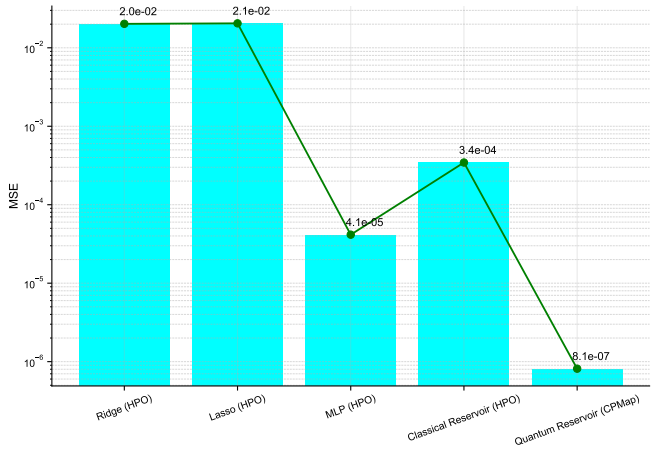


FIG. 3: Model comparison on the Mackey–Glass dataset at $\tau = 17$, window size = 20, prediction horizon = 20. Quantum reservoir achieves the lowest MSE despite no hyperparameter tuning.

b. ZZFeatureMap: For comparison, we also evaluate our architecture using the standard ZZFeatureMap [34], which encodes inputs via single-qubit R_Z rotations followed by pairwise controlled-Z entangling gates. The ZZFeatureMap is known for its simplicity, symmetry, and suitability for quantum kernel estimation. Its inclusion allows us to benchmark the performance of CPMap against an established alternative.

Each of these circuits is used in the feedback-reservoir architecture, where the left half of the circuit encodes the input sequence and the right half encodes the feedback signal via a daggered copy of the same circuit.

IV. RESULTS AND ANALYSIS

A. Performance on Mackey-Glass Dataset

We evaluate the forecasting performance of our feedback-based quantum reservoir computing (QRC) architecture on the Mackey–Glass time series with a delay parameter of $\tau = 17$, which is widely recognized in the literature for generating rich chaotic dynamics. The prediction task is configured with a window size of 20 and a prediction horizon of 20, forming a moderately long-term forecasting problem. Input sequences are normalized using a standard scaling procedure, and the dataset is partitioned into 7,500 training and 2,500 test samples. For this experiment, we focus on the CPMap-based quantum circuit, as simulating the ZZFeatureMap for this setup becomes computationally prohibitive given the circuit depth and dataset size.

Despite using a fixed quantum circuit with no internal trainable parameters and no hyperparameter optimization, the QRC model outperforms a range of classical baselines. As shown in Figure 3, it achieves the lowest mean squared error (MSE) among all tested models, including ridge regression, lasso regression, multilayer perceptrons (MLPs), and a

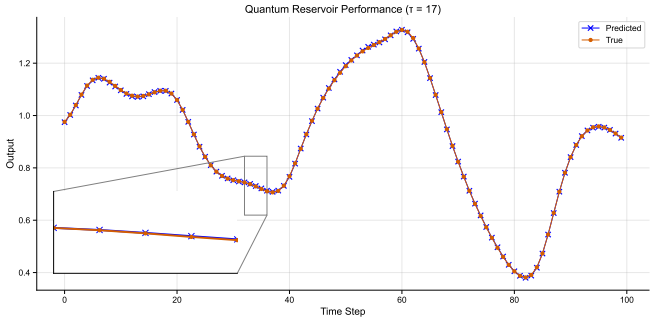


FIG. 4: Predicted vs. true signal for the quantum reservoir over 100 test time steps. The reservoir captures the chaotic dynamics with high fidelity and stability.

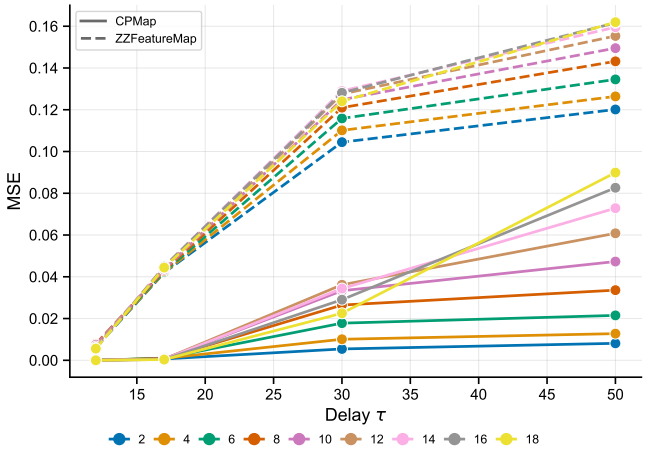


FIG. 5: Mean squared error (MSE) across Mackey–Glass delays $\tau = 15$ to 50, for various prediction horizons, and two different feature maps. The CPMap consistently outperforms the ZZFeatureMap across delays and horizons.

hyperparameter-tuned echo state network (ESN). The ESN's parameters—including spectral radius, input scaling, and regularization—were optimized via grid search, whereas the QRC model remains untouched after initialization. This result highlights the expressive power of the CPMap circuit when integrated into a feedback loop, enabling efficient temporal encoding and memory retention through purely unitary evolution.

Figure 4 illustrates the qualitative forecasting behavior of the quantum reservoir over a 100-step interval in the test set. The model captures both the fast oscillations and slower amplitude modulations of the target signal, closely tracking the chaotic dynamics without significant drift or cumulative error. These results demonstrate the reservoir's capacity to produce stable, accurate forecasts across extended time intervals.

To further evaluate the robustness of our quantum reservoir model under varying temporal dependencies, we examine performance across a range of Mackey–Glass delay parameters $\tau \in \{12, 17, 30, 50\}$ for different prediction horizons. Figure 5 shows the mean squared error (MSE) for each combination of delay and prediction horizon, separately for CPMap and ZZFeatureMap. As expected, the forecasting error generally increases with both τ and the horizon, reflecting the increasing memory demands of the task. However, the CPMap consistently yields lower MSE across most settings, demonstrating superior robustness in retaining relevant temporal information.

Notably, while both feature maps experience degradation in accuracy at high delays and long horizons, the ZZFeatureMap exhibits more pronounced performance loss, particularly for horizons greater than 10. This contrast highlights the advantage of CPMap in encoding both current input and feedback in a more expressive and noise-resilient manner. These trends remain stable across multiple experimental runs, reinforcing the generalizability of the observed advantage.

We also analyze the influence of feedback strength (α) and readout truncation (γ), showing that optimal performance arises from a balance between recurrence and the readout information. Supporting results and figures are provided in the Supplementary Material.

To further understand the performance, we examine the dynamical properties of the quantum reservoir. We find that the system exhibits a finite but effective memory capacity, retaining information over a limited temporal horizon before saturating. At the same time, the reservoir satisfies the echo state property (ESP), as trajectories initialized from different states converge under identical inputs after a short time. A detailed quantitative analysis of memory capacity and ESP verification is provided in the supplementary material.

B. Entanglement, Memory Capacity, and Prediction Error

To investigate the sensitivity of the quantum reservoir's dynamics to circuit-level hyperparameters, we performed a controlled sweep over a single circuit parameter, denoted θ_i, while keeping the other five parameters fixed. For each value of θ_i sampled uniformly from the interval $[-\pi/2, \pi/2]$, we evaluated three key quantities: the short-term memory capacity (green, right axis), the reservoir's predictive performance as measured by mean square error (MSE; red, right axis), and the average single-qubit entanglement entropy (dashed red, left axis), computed by tracing out each qubit and averaging the resulting von Neumann entropies.

As shown in Fig. 6, these three metrics exhibit nontrivial and correlated dependence on the quantum circuit configuration. Notably, regions of high memory capacity tend to coincide with improved prediction accuracy, consistent with theoretical expectations from reservoir computing. The entanglement entropy profile reveals a complementary structure: excessively high or low entanglement appears to degrade performance, suggesting an optimal intermediate regime where the circuit remains expressive yet stable. These findings highlight the sensitivity of quantum reservoir dynamics to circuit-level parameters and underscore the utility of entanglement entropy as a diagnostic for task-relevant quantum behaviour.

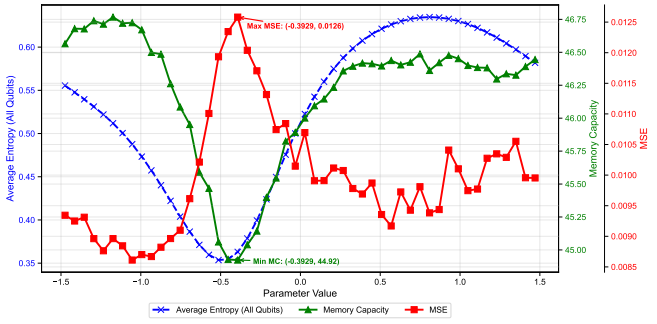


FIG. 6: Impact of quantum circuit parameter θ_i on the reservoir's entanglement structure, memory capacity, and predictive performance. A single parameter $\theta_i \in [-\pi/2, \pi/2]$ is swept while keeping all other circuit parameters fixed. The average single-qubit entanglement entropy (dashed blue) is computed across all qubits, the memory capacity (green) is derived from the explicit STM formulation, and MSE (solid red) quantifies forecasting error. The results reveal a structured dependence of the reservoir's computational behavior on θ_i, with distinct parameter regions exhibiting enhanced memory retention, moderate entanglement, and low prediction error.

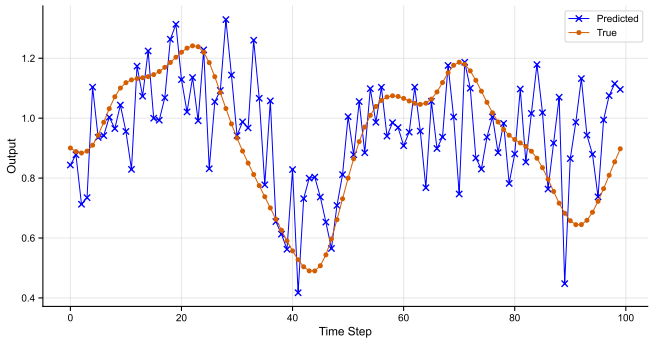


FIG. 7: Predicted versus true Mackey-Glass signal for the quantum reservoir under noisy simulation at $\tau = 17$. The noisy prediction preserves the coarse oscillatory trend but exhibits strong local fluctuations and irregular spikes, indicating degradation of fine temporal structure under realistic device noise.

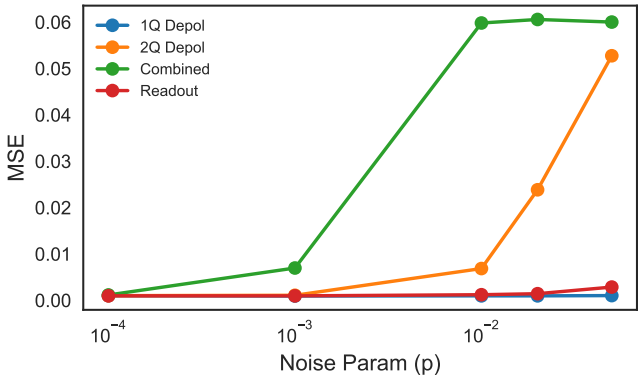


FIG. 8: Performance (MSE score) of the quantum reservoir as a function of noise strength for different noise channels. Single-qubit and readout noise have little effect, whereas two-qubit depolarizing noise leads to a sharp degradation. The combined-noise case shows the strongest deterioration.

C. Result of noisy simulation:

To assess the impact of realistic noise on the proposed quantum reservoir, we repeat the Mackey-Glass forecasting experiment using a noisy simulation based on a fake IBM_Torino backend. The dataset consists of 5000 samples with an 80/20 train-test split, window size 20, prediction horizon 20, and $\tau = 17$. As shown in Fig. 7, the QRC still captures the overall oscillatory structure of the signal. However, the predictions develop strong local fluctuations and irregular spikes, indicating that noise disrupts the fine temporal structure captured in the ideal simulation. This indicates that realistic device noise substantially disrupts the fine temporal structure learned by the reservoir, even when the coarse trend is still partially preserved. In particular, the noisy reservoir no longer matches the high-fidelity behavior observed in the ideal simulation, highlighting a clear gap between simulator performance and deployment under NISQ-like noise conditions.

To further identify which noise mechanisms are most detrimental, we performed a targeted sensitivity analysis by varying individual noise channels separately. Figure 8 summarizes the results in terms of mean squared error (MSE). We found that the reservoir is comparatively robust to single-qubit, readout, and relaxation noise over the tested range, whereas two-qubit gate noise leads to the strongest degradation in MSE value. When multiple noise sources are combined, the deterioration becomes even more severe, indicating that the instability observed in Fig. 7 is driven primarily by entangling operations rather than by all noise channels equally.

For a compact comparison, Fig. 9 shows the worst-case MSE for each noise type. Two-qubit depolarizing noise and the combined-noise setting clearly dominate, while single-qubit, readout, and relaxation noise remain comparatively small.

This behavior aligns with the role of entanglement in the reservoir. The entangling gates that build the feature space also spread errors across the system, so noise introduced at these steps does not stay local but propagates through the dynamics. As a result, the reservoir loses both memory of past inputs and the structure needed for accurate prediction. This is consistent with the trends observed in Fig. 6, where performance is tied to a balanced regime of entanglement and memory. In contrast, single-qubit and readout errors act more locally and therefore have a much weaker effect. Overall, these results point to entangling gate fidelity as the main constraint for running this model on near-term hardware.

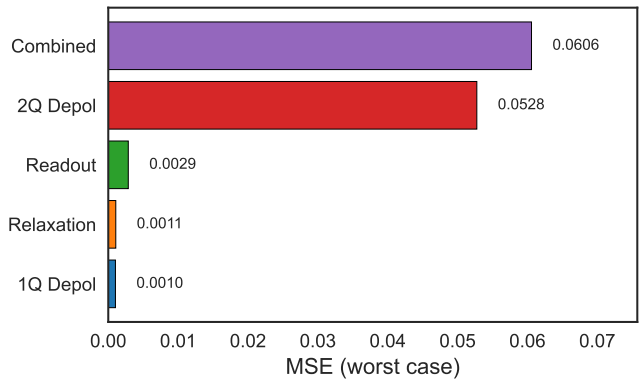


FIG. 9: Worst-case MSE for each noise type at the maximum tested noise parameter. Two-qubit and combined noise dominate the degradation.

V. DISCUSSION AND CONCLUSION

Across the Mackey-Glass forecasting experiments, the proposed model achieves strong predictive performance and compares favorably with standard classical baselines. Notably, this performance is obtained using a fixed quantum circuit with no trainable internal parameters, indicating that the combination of structured feature-map encoding and a classical feedback loop is sufficient to generate expressive and stable reservoir dynamics. The results remain consistent across different delay parameters and prediction horizons, suggesting that the architecture generalizes well to varying temporal dependencies.

The results also clarify the role of feedback in the model. The performance sweeps over α and λ show that good forecasting does not arise from circuit structure alone. Instead, it requires a balance between retaining past information through feedback and ensuring that enough of the reservoir state is accessible for prediction. Too little feedback weakens temporal memory, while excessive feedback can suppress useful input information. Likewise, truncating the readout too aggressively removes relevant dynamical features. The best performance is obtained in an intermediate regime, which is consistent with the broader reservoir computing principle that useful dynamics emerge when the system is neither too rigid nor too unstable.

This interpretation is reinforced by the dynamical analysis. The memory-capacity results show that the reservoir retains information over a finite but useful temporal window, while the echo-state tests confirm that the dynamics remain input-driven and stable. Together, these results indicate that the proposed architecture is not simply fitting the data through a large feature space, but is operating as a genuine reservoir with fading memory. The entanglement study further sharpens this picture: improved prediction is associated with parameter regions that support meaningful memory capacity and moderate entanglement, whereas very low or very high entanglement tends to be less favorable. In other words, the useful operating regime is not one of maximal quantum correlation, but one in which information spreading and dynamical stability are balanced.

At the same time, the noise study makes clear that the current performance of the model is much better in the ideal simulation than under realistic device-level noise. Although the reservoir remains relatively robust to single-qubit, readout, and relaxation noise, two-qubit errors lead to a considerable drop in performance, and combined noise quickly degrades the predictions. As a result, the model's predictive quality is closely tied to the fidelity of entangling gates. This does not invalidate the architecture, but it does place an important qualification on near-term deployment: compact circuit design alone is not sufficient unless the hardware can support the required two-qubit operations with adequate accuracy.

Taken together, these results show that recurrent quantum feature maps provide a viable and conceptually clean route to quantum reservoir computing. The proposed hybrid feedback mechanism connects quantum feature maps, reservoir dynamics, and temporal learning in a single framework, while the accompanying analyses help explain why and when the model works. The main limitation identified here is sensitivity to two-qubit noise, which points naturally to future work on shallower entangling layouts, noise-aware feature-map design, and error-mitigation strategies. It will also be important to evaluate the architecture on a broader range of real-world temporal datasets and to study whether the same balance between memory, entanglement, and stability persists beyond Mackey-Glass. More broadly, this work suggests that the most promising route for QRC may not be to maximize quantum complexity at all costs, but to engineer structured quantum dynamics that are expressive enough to be useful and simple enough to remain stable on realistic hardware.

VI. ACKNOWLEDGMENTS

The authors would like to acknowledge the use of IBM Quantum services for this work and, in particular, the Qiskit package [35, 36]. AZG and KH acknowledge that the NRC headquarters is located on the traditional unceded territory of the Algonquin Anishinaabe and Mohawk people. K.H. acknowledges funding from the NSERC Discovery Grant. C.S. and K.H. would like to acknowledge the NRC for its Applied Quantum Computing Challenge Program and NSERC for the Alliance grant QIMMIQ.

[1] H. Jaeger and H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science **304**, 78 (2004).

[2] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, An overview of reservoir computing: theory, applications and implementations, in *Proceedings of the 15th European Symposium on Artificial Neural Networks* (2007) pp. 471–482.

[3] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, Recent advances in physical reservoir computing: A review, Neural Networks **115**, 100 (2019).

[4] W. Maass, T. Natschläger, and H. Markram, Real-time computing without stable states: A new framework for neural computation based on perturbations, Neural computation **14**, 2531 (2002).

[5] M. Lukoševičius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, Computer Science Review **3**, 127 (2009).

[6] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, An experimental unification of reservoir computing methods, Neural Networks **20**, 391 (2007), echo State Networks and Liquid State Machines.

[7] K. Fujii and K. Nakajima, Harnessing disordered-ensemble quantum dynamics for machine learning, Physical Review Applied **8**, 024030 (2017).

[8] M. Negoro, K. Mitarai, K. Fujii, K. Nakajima, and M. Kitagawa, Machine learning with controllable quantum dynamics of a nuclear spin ensemble in a solid, arXiv preprint arXiv:1806.10910 (2018).

[9] J. e. a. Nokkala, Gaussian states of continuous-variable quantum systems provide universal and versatile reservoir computing, Communications Physics **4**, 53 (2021).

[10] J.-D. Chen, H. I. Nurdin, and N. Yamamoto, Temporal information processing on noisy quantum computers via reservoir computing, Physical Review Applied **14**, 024065 (2020).

[11] H. Yasuda and N. Yamamoto, Quantum reservoir computing with repeated measurements on superconducting devices, arXiv preprint arXiv:2310.06706 (2023).

[12] M. Kornjača, H.-Y. Hu, C. Zhao, J. Wurtz, P. Weinberg, M. Hamdan, A. Zhdanov, S. H. Cantu, H. Zhou, R. A. Bravo, K. Bagnall, J. I. Basham, J. Campo, A. Choukri, R. DeAngelo, P. Frederick, D. Haines, J. Hammett, N. Hsu, M.-G. Hu, F. Huber, P. N. Jepsen, N. Jia, T. Karolyshyn, M. Kwon, J. Long, J. Lopatin, A. Lukin, T. Macrì, O. Marković, L. A. Martínez-Martínez, X. Meng, E. Ostroumov, D. Paquette, J. Robinson, P. S. Rodriguez, A. Singh, N. Sinha, H. Thoreen, N. Wan, D. Waxman-Lenz, T. Wong, K.-H. Wu, P. L. S. Lopes, Y. Boger, N. Gemelke, T. Kitagawa, A. Keesling, X. Gao, A. Bylinskii, S. F. Yelin, F. Liu, and S.-T. Wang, Large-scale quantum reservoir learning with an analog quantum computer (2024), arXiv:2407.02553 [quant-ph].

[13] A. e. a. Sannia, Dissipation as a resource for quantum reservoir computing, Quantum **8**, 1291 (2024).

[14] L. C. Govia *et al.*, Quantum reservoir computing with a single nonlinear oscillator, Physical Review Research **3**, 013077 (2021).

[15] P. Mujal, J. Guerrero, D. Garcia-Beni, G. F. Calvo, and A. Alarcon, Time-series quantum reservoir computing with weak and projective measurements, npj Quantum Information **9**, 16 (2023).

[16] T. Murauer, A. Matsuura, and Y. Yamamoto, Feedback connections in quantum reservoir computing with mid-circuit measurements, arXiv preprint (2025), arXiv:2503.22380 [quant-ph].

[17] K. Kobayashi, K. Fujii, and Y. Yamamoto, Feedback-driven quantum reservoir computing for time-series analysis, PRX Quantum **5**, 040325 (2024).

[18] K. Kobayashi, K. Fujii, and N. Yamamoto, Feedback-driven quantum reservoir computing for time-series analysis, PRX Quantum **5**, 040325 (2024).

[19] L. Gonon, R. Martínez-Peña, and J.-P. Ortega, Feedback-driven recurrent quantum neural network universality (2026), arXiv:2506.16332 [quant-ph].

[20] R. Monomi, Y. Yamamoto, and K. Fujii, Feedback-enhanced quantum reservoir computing with weak measurements, arXiv preprint (2025), arXiv:2503.17939 [quant-ph].

[21] M. e. a. Domingo, Taking advantage of noise in quantum reservoir computing, Scientific Reports **13**, 11591 (2023).

[22] L. e. a. Čindrak, Enhancing the performance of quantum reservoir computing and solving the time-complexity problem by artificial memory restriction, Physical Review Research **6**, 013076 (2024).

[23] H. Goto, M. C. Tran, and K. Nakajima, Universal approximation property of quantum machine learning models in quantum-enhanced feature spaces, Physical Review Letters **127**, 090506 (2021).

[24] K. Matsumoto and M. C. Tran, Iterative quantum feature maps, arXiv preprint (2025), arXiv:2506.19461 [quant-ph].

[25] H. Jaeger, *The "echo state" approach to analysing and training recurrent neural networks-with an erratum note*, Tech. Rep. (Bonn, Germany: German National Research Center for Information Technology GMD Technical Report, 2001).

[26] M. Lukoševičius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, Computer Science Review **3**, 127 (2009).

[27] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, Re-visiting the echo state property, Neural networks **35**, 1 (2012).

[28] S. Boyd and L. O. Chua, *Fading memory and the problem of approximating nonlinear operators with Volterra series* (IEEE Transactions on circuits and systems, 1985).

[29] L. Grigoryeva and J.-P. Ortega, Echo state networks are universal, Neural Networks **108**, 495 (2018).

[30] L. Gonçalves, N. Marques, R. Marques, and R. Rego, Reservoir computing universality with stochastic inputs, IEEE Transactions on Neural Networks and Learning Systems **31**, 4749 (2020).

[31] H. Jaeger, *Short term memory in echo state networks*, Tech. Rep. (GMD-Forschungszentrum Informationstechnik, 2002).

[32] M. C. Mackey and L. Glass, Oscillation and chaos in physiological control systems, Science **197**, 287 (1977).

[33] U. Singh, J.-F. Laprade, A. Z. Goldberg, and K. Heshami, A resource efficient quantum kernel (2025), arXiv:2507.03689 [quant-ph].

[34] 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**, 209 (2019).

[35] Qiskit contributors, Qiskit: An open-source framework for quantum computing (2023).

[36] A. Asfaw, L. Bello, Y. Ben-Haim, S. Bravyi, L. Capelluto, A. C. Vazquez, J. Ceroni, R. Chen, A. Frisch, J. Gambetta, S. Garion, L. Gil, S. D. L. P. Gonzalez, F. Harkins, T. Imamichi, D. McKay, A. Mezzacapo, Z. Minev, R. Movassagh, G. Nannicini, P. Nation, A. Phan, M. Pistoia, A. Rattew, J. Schaefer, J. Shabani, J. Smolin, K. Temme, M. Tod, and S. Wood, *Learn Quantum Computation Using Qiskit* (2020).

[37] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, Information processing capacity of dynamical systems, Scientific reports **2**, 514 (2012).

A. Effect of Feedback Strength and Readout Truncation

We next examine how two core parameters—feedback strength α and readout limit λ—influence forecasting performance. As described previously, α controls the contribution of the feedback vector $\tilde{\mathbf{z}} = \alpha \cdot \mathbf{z}$, where $\mathbf{z}$ is the output of the previous reservoir iteration. Setting $\alpha = 0$ removes recurrence entirely, while $\alpha = 1$ reuses the full previous output. The parameter $\lambda \in (0, 1]$ determines the fraction of output amplitudes used in the readout vector after each quantum circuit execution. For instance, $\lambda = 0.6$ means only the first 60% of states—ranked lexicographically—are retained as features for regression.

Figure 10 summarizes the effect of both parameters on mean squared error (MSE), evaluated on the Mackey–Glass dataset with $\tau = 17$, window size = 20, and prediction horizon = 20. The model achieves its best performance at $(\alpha, \lambda) = (0.79, 1.0)$, indicating that strong—but not maximal—feedback and full readout provide optimal dynamics for this task. Lower values of α degrade memory, while overly large α may cause feedback dominance and input suppression. Similarly, reducing λ discards useful dynamical information, limiting the reservoir's expressive capacity.

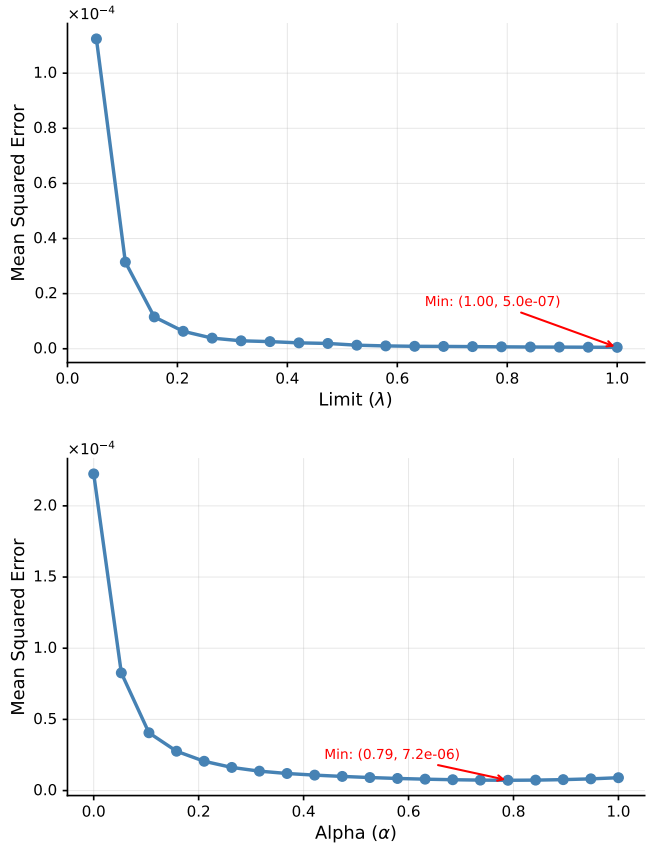
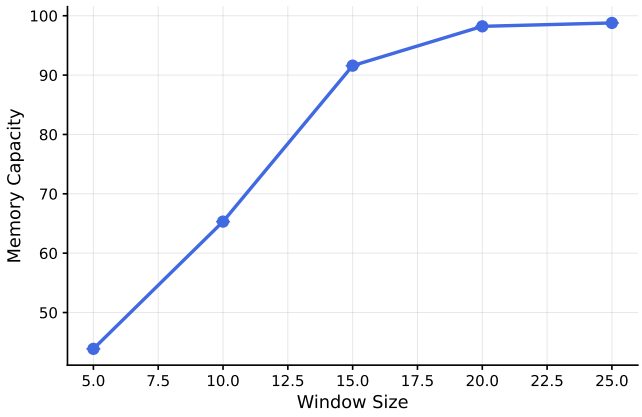


FIG. 10: Variation of test MSE with readout limit λ (top) and feedback strength α (bottom). The best performance is achieved at $(\alpha, \lambda) = (0.79, 1.0)$, demonstrating the importance of both recurrence and output dimensionality in quantum reservoir dynamics.

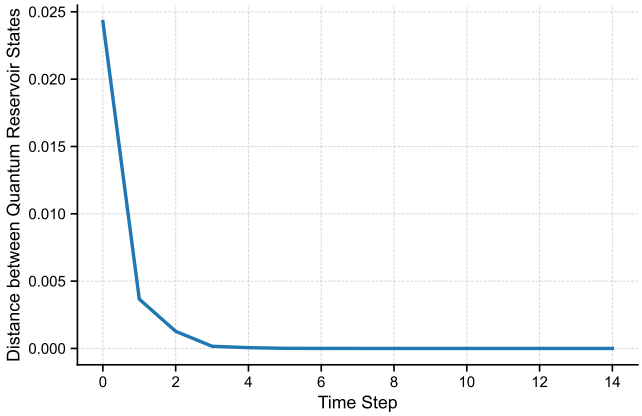
These results highlight that performance in feedback-driven quantum reservoirs arises not only from circuit design, but also from a careful balance between recurrence and observability.

B. Memory Dynamics and Echo State Property

We assess two fundamental dynamical properties of our quantum reservoir system: its linear memory capacity and its stability under the echo state property (ESP). The memory capacity quantifies how well past inputs can be reconstructed from the current reservoir state, reflecting the system's ability to retain temporal information. As shown in Figure 11, the memory capacity increases with window size and begins to saturate around a window length of 20, indicating that the reservoir effectively retains information over several time steps without being overwhelmed. Beyond this point,



(a) Memory capacity vs. input window size.



(b) State distance over time for ESP verification.

FIG. 11: Dynamical analysis of the quantum reservoir. (a) Memory capacity increases with window size and saturates around 20, indicating a limit in temporal retention. (b) Convergence of reservoir trajectories confirms that the quantum system satisfies the echo state property.

additional inputs offer diminishing returns, consistent with

capacity saturation observed in classical echo state networks [31, 37].

To evaluate stability, we measure the state distance between two initially different reservoir trajectories driven by identical input sequences. The results confirm that, after a short transient, the state distance diminishes and converges toward zero, demonstrating that the reservoir dynamics are input-driven rather than history-dependent. This convergence behavior verifies that our quantum reservoir satisfies the ESP, a critical condition for consistent temporal processing and fading memory. Together, these findings validate the use of our feedback-based quantum reservoir as a stable and memory-efficient temporal model.

C. Performance of QRC in presence of relaxation noise:

We also assessed the performance in the presence of relaxation noise. As shown in Fig. 12, the MSE changes very little over a wide range of T_1 values, which suggests relaxation alone is not the main factor limiting performance in this setting.

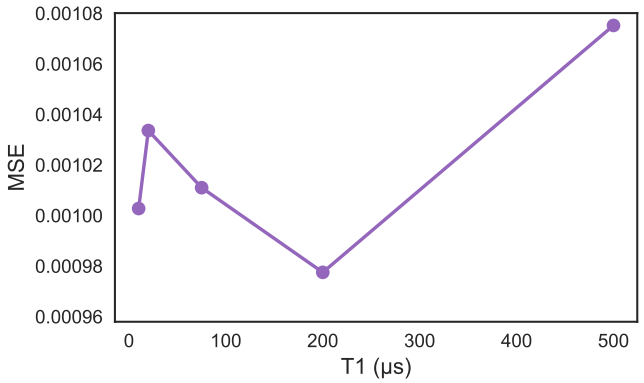


FIG. 12: MSE as a function of relaxation time T_1. The weak dependence on T_1 indicates that relaxation noise has a limited impact on performance in the explored regime.

D. Full-state feedback:

We evaluate the full-state feedback variant (CPQRC-lite) on the Mackey–Glass time-series prediction task using 5000 samples, a window size of 20, prediction horizon of 20, and delay parameter $\tau = 17$. The results are summarized in Fig. 13 and compared with standard baselines, including Ridge regression, MLP, classical reservoir computing, and the original CPQRC model. Here, CPQRC represents the QRC framework proposed in the paper with the CPMap circuit.

CPQRC-lite achieves a mean squared error of 4.7×10^{-5}, which is lower than classical reservoir computing (1.5×10^{-4}) and comparable to the MLP baseline (9.4×10^{-5}). The original CPQRC model achieves the lowest error of 1.6×10^{-6}.

Algorithm: Feedback-Driven Quantum Reservoir Computing without single-qubit expectations

Input: Time-series input $\{x_t\}_{t=1}^{T}$, feature map $U(\cdot)$, window size τ, feedback strength α, output fraction λ, prediction horizon H

Output: Predicted outputs $\{\hat{y}_t\}_{t=\tau+1}^{T-H}$

Initialize feedback vector $\tilde{\mathbf{z}}_0 \leftarrow \mathbf{0} \in \mathbb{R}^{n_q}$

for $t = \tau + 1$ **to** $T - H$ **do**

- Create input window: $\mathbf{x}_t = [x_{t-\tau}, \ldots, x_t]$
- Encode input with feature map: apply $U(\mathbf{x}_t)$ to left half of the circuit
- Encode feedback from previous step: apply $U^{\dagger}(\Pi(\tilde{\mathbf{z}}_{t-1}))$ to right half
- Execute full quantum circuit and measure output distribution $\mathbf{p}_t$
- Extract first $n = \dim(\mathbf{x}_t)$ components from $\mathbf{p}_t$: $\mathbf{z}_t = (p_t(s_1), \ldots, p_t(s_n))$;
- Update feedback: $\tilde{\mathbf{z}}_t = \alpha \cdot \mathbf{z}_t$
- Construct regression features: $\mathbf{r}_t = \left(p_t(s_1), \ldots, p_t(s_{\lfloor \lambda 2^{n_q} \rfloor})\right)$
- Store $(\mathbf{r}_t, y_t)$ where $y_t = x_{t+H}$

Train regression model: $\hat{y}_t = \mathbf{w}_{\text{out}}^{\top} \mathbf{r}_t + b$

return predicted sequence $\{\hat{y}_t\}_{t=\tau+1}^{T-H}$

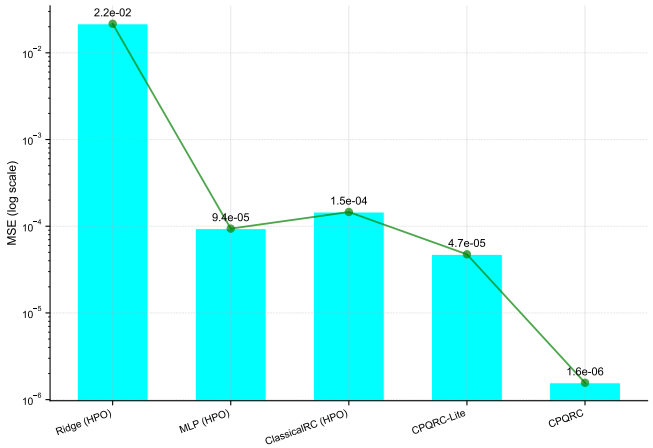


FIG. 13: Model comparison on the Mackey–Glass dataset at $\tau = 17$, window size = 20, prediction horizon = 20. CPQRC-lite achieves lower error than classical reservoir computing and performs comparably to MLP, while the original CPQRC model attains the lowest MSE.

These results show that the full-state feedback approach maintains competitive performance while simplifying the feedback construction by directly using the measurement distribution, without requiring explicit computation of expectation values.

5

Conclusion, Summary & Future Work

This thesis begins from a practical observation: despite the excitement surrounding quantum machine learning (QML), most approaches remain far from real-world deployment. They typically demand many qubits, rely on deep entangling circuits, and degrade under the realities of noisy intermediate-scale quantum (NISQ) hardware. My aim was simple to state, but demanding in practice: bring QML closer to real problems without giving up the representational advantages that motivate quantum models. The path led to three complementary contributions. First, I introduced a coherent feed-forward quantum neural network (CFFQNN) that preserves coherence across layers and mirrors the adjustable hidden structure of classical feed-forward networks, while sharply reducing circuit depth and

two-qubit gates and decoupling qubit growth from input dimension (Chapter 2). Second, I developed a resource-efficient quantum kernel (CPMap) that embeds high-dimensional data with fewer qubits and linearly scaling entanglers, performing competitively with strong classical kernels and standard quantum feature maps under noisy simulations and small hardware runs (Chapter 3). Third, I proposed a quantum reservoir computing (QRC) scheme that reuses a fixed feature-map circuit as a nonlinear reservoir and injects memory via explicit feedback, so the qubit register is recycled across time and only a lightweight classical readout is trained (Chapter 4). Together these results constitute a coherent strategy for constructing quantum models that respect device constraints yet deliver meaningful learning behaviour.

There are still many open questions at the boundary between model design and hardware reality. Beyond what I have established in the three core chapters, a natural direction is to extend these approaches to the photonic settings. During my research stay in Germany, I began a boson-sampling–inspired quantum kernel that maps inputs to linear-optical transformations and compares output distributions via overlap-style statistics. The theory is complete and early simulations are promising; the next step is experimental validation. In parallel, a colleague and I are extending the reservoir approach to the photonics domain.

The methods developed here are designed to work beyond toy problems. A concrete near-term target is the materials-discovery pipeline in which the CPMap kernel—and, where appropriate, the CFFQNN—serve as drop-in learners within active-learning workflows for property prediction such as band gap, stability, and transport. I have already conducted a collaborative study along these lines with the quantum computational chemistry group at the University of Calgary [91], and the plan is to integrate these techniques fully into their research pipeline. The goal is not merely higher test accuracy, but operation under the constraints that matter in practice: few available qubits, tight budgets on two-qubit errors, and the extra time cost of compiling circuits.

Device note: We conducted small-scale hardware tests to validate feasibility, but sustained QPU access remained limited; for ML workloads, the overheads of error correction and extensive mitigation made longer runs impractical. Accordingly, the CPMap kernel results reported here are without error correction. In exploratory trials on IBM's older Eagle-class hardware, measured outputs exhibited near-maximal entropy (effectively information-poor under our metrics). Newer Heron-class devices show significantly improved stability and fidelity in preliminary checks, and a priority for future work is to re-run the CPMap kernel and related models on Heron-class or newer backends under controlled, longer sessions.

Despite these caveats, I believe the central claim of the thesis is borne out: QML can be made practically closer to real-world application by choosing architectures and encodings that respect device constraints from the outset. The CFFQNN demonstrates that coherent, neural-like structure does not have to come with punitive depth or ballooning qubit counts. The CPMap kernel shows that high-dimensional embeddings can be implemented with tight, predictable resource footprints while remaining competitive. The feature-map-with-feedback reservoir illustrates that temporal modeling can be achieved without training large variational stacks, by trading parameter count for dynamical recurrence. These are not isolated tricks; they are facets of a single design stance that prefers shallow, topology aware entanglement, decouples qubits from feature dimension wherever possible, and leans on classical optimization when quantum training is fragile.

Looking forward, the future goals are clear and actionable. The boson-sampling kernel begun in Germany will move from simulation to experiment, providing a photonic testbed for constant-depth quantum embeddings. The circuit-based reservoir line will explore dynamic circuits and mid-circuit measurements/reset to achieve longer effective memory without deepening the coherent path, and will also look to photonic implementations. On hardware, I plan multi-session runs on IBM's newer Heron-class backends to re-evaluate the CPMap kernel and related models under stable conditions. Throughout, I will keep the

emphasis on reproducible pipelines, shared code and seeds, and comparisons that measure not only whether a model works, but at what resource cost and under what noise.

To close, I return to the original motivation. QML has been rich in promise but short on deployable systems. This thesis does not claim to have solved that whole problem; rather, it offers a concrete step: coherent architectures and resource-efficient kernels and reservoirs that survive contact with real devices and hold their ground against strong classical baselines. The two patents associated with the first two contributions signal that these designs matter outside the page. More importantly, the principles behind them—coherence where it helps, shallow entanglement where it counts, and disciplined use of quantum resources—provide a practical foundation.

References

1. Feynman, R. P. Simulating physics with computers. International Journal of Theoretical Physics **21,** 467–488 (1982).
2. Shor, P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing **26** (1997).
3. Grover, L. K. Quantum Mechanics Helps in Searching for a Needle in a Haystack. Physical Review Letters **79,** 325–328 (1997).
4. Arute, F., Arya, K., Babbush, R. & et al. Quantum supremacy using a programmable superconducting processor. Nature **574,** 505–510 (2019).
5. Madsen, L. S. et al. Quantum computational advantage with a programmable photonic processor. Nature **606,** 75–81 (2022).
6. Google Quantum AI and Collaborators. Observation of constructive interference at the edge of quantum ergodicity. Nature **646,** 825–830 (2025).
7. Preskill, J. Quantum Computing in the NISQ era and beyond. Quantum **2,** 79 (2018).
8. Goodfellow, I., Bengio, Y. & Courville, A. Deep Learning (MIT Press, 2016).
9. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature **521,** 436–444 (2015).
10. Bottou, L., Curtis, F. E. & Nocedal, J. Optimization Methods for Large-Scale Machine Learning. SIAM Review **60,** 223–311 (2018).
11. Strubell, E., Ganesh, A. & McCallum, A. Energy and Policy Considerations for Deep Learning in NLP. arXiv preprint. arXiv: `1906.02243` (2019).
12. Biamonte, J. et al. Quantum machine learning. Nature **549,** 195–202 (2017).
13. Schuld, M., Sinayskiy, I. & Petruccione, F. An introduction to quantum machine learning. Contemporary Physics **56,** 172–185 (2015).
14. Dunjko, V. & Briegel, H. J. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. Reports on Progress in Physics **81,** 074001 (2018).
15. Carleo, G. et al. Machine learning and the physical sciences. Reviews of Modern Physics **91,** 045002 (2019).
16. Avramouli, M., Savvas, I., Vasilaki, A., Garani, G. & Xenakis, A. Quantum Machine Learning in Drug Discovery: Current State and Challenges. PCI '22 394–401 (2023).
17. Mironowicz, P., H., A. S., Mandarino, A., Yilmaz, A. E. & Ankenbrand, T. Applications of Quantum Machine Learning for Quantitative Finance. arXiv: `2405.10119 [quant-ph]` (2024).
18. Graña, I. F., Varsamopoulos, S., Ando, T., Maeshima, H. & Matsuzawa, N. N. Materials Discovery With Quantum-Enhanced Machine Learning Algorithms. arXiv: `2503.09517 [cond-mat.mtrl-sci]` (2025).
19. Tang, E. A quantum-inspired classical algorithm for recommendation systems. STOC '19 217–228 (June 2019).
20. McArdle, S., Endo, S., Aspuru-Guzik, A., Benjamin, S. & Yuan, X. Quantum Computational Chemistry. Reviews of Modern Physics **92,** 015003 (2020).

21. Nguyen, T., Sipola, T. & Hautamäki, J. Machine Learning Applications of Quantum Computing: A Review. European Conference on Cyber Warfare and Security **23,** 322–330. ISSN: 2048-8602 (June 2024).

22. IBM Quantum. IBM Quantum Platform `https://quantum-computing.ibm.com/`. `https://quantum-computing.ibm.com/`. 2023.

23. Microsoft Quantum. Azure Quantum `https://azure.microsoft.com/en-us/products/quantum`. `https://azure.microsoft.com/en-us/products/quantum`. 2024.

24. Sim, S., Johnson, P. & Aspuru-Guzik, A. Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms. Advanced Quantum Technologies **2,** 1900070 (2019).

25. Stokes, J., Izaac, J., Killoran, N. & Carleo, G. Quantum Natural Gradient. Quantum **4,** 269 (2020).

26. Rebentrost, P., Mohseni, M. & Lloyd, S. Quantum Support Vector Machine for Big Data Classification. Physical Review Letters **113,** 130503 (2014).

27. Lloyd, S., Mohseni, M. & Rebentrost, P. Quantum principal component analysis. Nature Physics **10,** 631–633 (2014).

28. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum Algorithm for Linear Systems of Equations. Physical Review Letters **103,** 150502 (2009).

29. Bharti, K. & et al. Noisy intermediate-scale quantum algorithms. Reviews of Modern Physics **94,** 015004 (2022).

30. Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L. & Coles, P. J. Challenges and opportunities in quantum machine learning. Nature Computational Science **2,** 567–576 (2022).

31. Farhi, E. & Neven, H. Classification with Quantum Neural Networks on Near Term Processors. arXiv: `1802.06002 [quant-ph]` (2018).

32. Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. Physical Review A **98,** 032309 (2018).

33. Havlíček, V. & et al. Supervised learning with quantum-enhanced feature spaces. Nature **567,** 209–212 (2019).

34. Zoufal, C., Lucchi, A. & Woerner, S. Quantum generative adversarial networks for learning and loading random distributions. npj Quantum Information **5,** 103 (2019).

35. Abbas, A. & et al. The power of quantum neural networks. Nature Computational Science **1,** 403–409 (2021).

36. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. Computer Science Review **3,** 127–149 (2009).

37. Jaeger, H. & Haas, H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304,** 78–80 (2004).

38. Maass, W., Natschläger, T. & Markram, H. Real-time computing without stable states: novel frameworks for neural computation based on perturbations. Neural Computation **14,** 2531–2560 (2002).

39. Fujii, K. & Nakajima, K. Harnessing disordered-ensemble quantum dynamics for machine learning. Physical Review Applied **8,** 024030 (2017).

40. Nakajima, K., Fujii, K., Negoro, M., Mitarai, K. & Kitagawa, M. Boosting computational power through spatial multiplexing in quantum reservoir computing. Physical Review Applied **11,** 034021 (2019).

41. Kutvonen, A., Fujii, K. & Sagawa, T. Optimizing a quantum reservoir computer for time series prediction. Scientific Reports **10,** 1–12 (2020).

42. McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren plateaus in quantum neural network training landscapes. Nature Communications **9,** 4812 (2018).

43. Thanasilp, S. et al. Exponential Concentration in Quantum Kernel Methods. Nature Communications (2024).

44. Gopalakrishnan, D. et al. qLUE: A Quantum Clustering Algorithm for Multi- Dimensional Datasets 2024. arXiv: 2407.00357 [quant-ph].

45. Alomari, A. & Kumar, S. A. P. A Survey of Quantum Reinforcement Learning Approaches: Current Status and Futu in 2025 IEEE Conference on Artificial Intelligence (CAI) (2025), 1375–1382.

46. Bishop, C. M. Pattern Recognition and Machine Learning Section 3.1.3: Sequential learning. ISBN: 978-0-387-31073-2 (Springer, New York, 2006).

47. Singh, U., Laprade, J.-F., Goldberg, A. Z. & Heshami, K. A Resource Efficient Quantum Kernel. arXiv preprint arXiv:2507.03689 (2025).

48. Bishop, C. M. Pattern Recognition and Machine Learning (Springer, 2006).

49. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65,** 386–408 (1958).

50. Minsky, M. & Papert, S. Perceptrons: An Introduction to Computational Geometry (MIT Press, Cambridge, MA, 1969).

51. Hornik, K., Stinchcombe, M. & White, H. Multilayer feedforward networks are universal approximators. Neural Networks **2,** 359–366. ISSN: 0893-6080 (1989).

52. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. Learning representations by back-propagating errors. Nature **323,** 533–536 (1986).

53. Glorot, X. & Bengio, Y. Understanding the difficulty of training deep feedforward neural networks in Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (eds Teh, Y. W. & Titterington, M.) **9** (PMLR, Chia Laguna Resort, Sardinia, Italy, 2010), 249–256.

54. He, K., Zhang, X., Ren, S. & Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNe in Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (IEEE Computer Society, USA, 2015), 1026–1034. ISBN: 9781467383912.

55. Bengio, Y., Simard, P. & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks **5,** 157–166 (1994).

56. Hochreiter, S. & Schmidhuber, J. Long short-term memory. Neural Computation **9,** 1735–1780 (1997).

57. Cho, K. et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation, 1724–1734 (2014).

58. Jaeger, H. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note (2001).

59. Yildiz, I. B., Jaeger, H. & Kiebel, S. J. Re-visiting the echo state property. Neural Networks **35,** 1–9 (2012).

60. Wolpert, D. H. & Macready, W. G. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Com **1,** 67–82 (1997).

61. Iovanac, N. C. & Savoie, B. M. Improved Chemical Prediction from Scarce Data Sets via Latent Space Enrichment. The Journal of Physical Chemistry A **123,** 4295–4302. eprint: https://doi.org/10.1021/acs.jpca.9b01398 (2019).

62. De Andrés-Galiana, E. J., Fernández-Martínez, J. L., Fernández-Brillet, L., Cernea, A. & Kloczkowski, A. Addressing Noise and Estimating Uncertainty in Biomedical Data through the Exploration of Chemical Space. International Journal of Molecular Sciences **23,** 12975 (2022).

63. Nielsen, M. A. & Chuang, I. L. Quantum Computation and Quantum Information 10th Anniversary Edition (Cambridge University Press, 2010).

64. Nielsen, M. A. & Chuang, I. L. Quantum Computation and Quantum Information 10th Anniversary Edition (Cambridge University Press, 2010).

65. Griffiths, D. J. & Schroeter, D. F. Introduction to Quantum Mechanics 3rd. ISBN: 9781107189638 (Cambridge University Press, Cambridge, UK, 2018).

66. Sakurai, J. Modern Quantum Mechanics Revised. ISBN: 978-0201539295 (Addison-Wesley, Reading, MA, 1994).

67. Peruzzo, A. et al. A variational eigenvalue solver on a photonic quantum processor. Nature Communications **5,** 4213 (2014).

68. Farhi, E., Goldstone, J. & Gutmann, S. A Quantum Approximate Optimization Algorithm. arXiv: `1411.4028 [quant-ph]` (2014).

69. Schuld, M. & Killoran, N. Quantum Machine Learning in Feature Hilbert Spaces. Physical Review Letters **122.** ISSN: 1079-7114 (Feb. 2019).

70. Schuld, M., Bocharov, A., Svore, K. M. & Wiebe, N. Circuit-centric quantum classifiers. Physical Review A **101.** ISSN: 2469-9934 (Mar. 2020).

71. Cong, I., Choi, S. & Lukin, M. D. Quantum convolutional neural networks. Nature Physics **15,** 1273–1278 (2019).

72. Tacchino, F., Macchiavello, C., Gerace, D. & Bordone, P. An artificial neuron implemented on an actual quantum processor. Quantum Science and Technology **5,** 044010 (2020).

73. Benedetti, M. et al. A generative modeling approach for benchmarking and training shallow quantum circuits. npj Quantum Information **5.** ISSN: 2056-6387 (May 2019).

74. Liu, J.-G. & Wang, L. Differentiable learning of quantum circuit Born machines. Physical Review A **98.** ISSN: 2469-9934 (Dec. 2018).

75. Amin, M. H., Andriyash, E., Rolfe, J., Kulchytskyy, B. & Melko, R. Quantum Boltzmann Machine. Physical Review X **8.** ISSN: 2160-3308 (May 2018).

76. Bausch, J. Recurrent Quantum Neural Networks in Advances in Neural Information Processing Systems (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H.) **33** (Curran Associates, Inc., 2020), 1368–1379.

77. Zendejas-Morales, C., Saikia, D. & Singh, U. Local and Multi-Scale Strategies to Mitigate Exponential Concentration 2026. arXiv: `2602.16097 [quant-ph]`.

78. Quek, Y. et al. Exponentially tighter bounds on limitations of quantum error mitigation. Nature Physics **20,** 1648–1658 (2024).

79. Wang, S., Fontana, E., Cerezo, M. & Holmes, Z. Can Error Mitigation Improve Trainability of Noisy Variational Quantum Algorithms? Quantum **8,** 1287 (2024).

80. Plesch, M. & Brukner, Č. Quantum-state preparation with universal gate decompositions. Physical Review A **83,** 032302 (2011).

81. Huang, H.-Y. et al. Power of data in quantum machine learning. Nature Communications **12,** 2631 (2021).

82. Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E. & Latorre, J. I. Data re-uploading for a universal quantum classifier. Quantum **4,** 226 (2020).

83. Schnabel, J. et al. Quantum kernel methods under scrutiny: a benchmarking study. Quantum Machine Intelligence **7,** 21 (2025).

84. McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R. & Neven, H. Barren plateaus in quantum neural network training landscapes. Nature Communications **9,** 4812 (2018).

85. Harrow, A. W., Hassidim, A. & Lloyd, S. Quantum algorithm for linear systems of equations. Physical Review Letters **103,** 150502 (2009).

86. Ben-David, S., Hrubes, P., Moran, S., Shpilka, A. & Yehudayoff, A. Learnability can be undecidable. Nature Machine Intelligence, 44–48 (2019).

87. Huang, H.-Y., Kueng, R. & Preskill, J. Information-theoretic bounds on quantum advantage in machine learning. Phys. Rev. Lett. **126,** 190505 (2021).

88. Li, M. & Vitányi, P. An Introduction to Kolmogorov Complexity and Its Applications 3rd. ISBN: 978-0-387-49820-1 (Springer, 2008).

89. Singh, U., Goldberg, A. Z. & Heshami, K. pat. WO2025050205A1 (2025).

90. Singh, U., Goldberg, A. Z. & Heshami, K. pat. WO2025073041A1 (2025).

91. Lourenço, M. P. et al. Seeking metal–organic frameworks for hydrogen storage using classical and quantum active learning. Phys. Chem. Chem. Phys. **27,** 23365–23379 (43 2025).