國立臺灣大學理學院物理學研究所

碩士論文

Department of Physics

College of Science

National Taiwan University

Master's Thesis

可訓練的核函數在量子變分線路中的效益

Benefits of Trainable Kernels in Variational Quantum Circuits

黃靜嚴

Jing-Yan Huang

指導教授: 管希聖 博士

Advisor: Hsi-Sheng Goan Ph.D.

中華民國 113 年 7 月

July, 2024

doi:10.6342/NTU202402911

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

可訓練的核函數在量子變分線路中的效益

Benefits of Trainable Kernels in Variational Quantum Circuits

本論文係___黃靜嚴___(姓名)__R09222059_（學號）在國立臺灣大學___物理_____(系/所/學位學程)完成之碩士學位論文，於民國_113_年_7__月_29_日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department / Institute of _____Physics_____ on __29__(date)__7___ (month)__2024___ (year) have examined a Master's thesis entitled above presented by___Jing-Yan Huang__ (name)_____R09222059_____ (student ID) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

管希聖	林澤	
（指導教授 Advisor）		
林俊達	劉子毅	

Acknowledgements

能夠完成這篇論文，首先我想感謝我的指導老師管希聖教授，在平時的討論中給予了寶貴的建議，同時也啟發了我許多的靈感。也很感謝教授在論文撰寫以及口試報告上提出了許多修改方向，讓這篇論文有更好的組織性。我也要感謝林澤教授、林俊達教授以及劉子毓教授，願意擔任我的口試委員，並且給予了相當多的指導與建議。

我還要感謝才念庭學長不厭其煩的解答我的疑問，也要感謝他提供之前研究時使用的程式碼給我參考，這對我來說是一塊很重要的入門磚。

再來我想感謝呂柏融、吳宗道、簡大閔、黃禹超等同儕在平時討論時給予的建議及指教，這對我研究方向和內容的發想與改進都起到相當重要的幫助。我也想特別感謝吳宗道學長將 Jax 這個 Python package 介紹給了大家，大大減少了數值模擬所需要的時間，讓我能夠將實驗推展到更大的規模。

最後，感謝所有支持與幫助我的人，謝謝你們。

doi:10.6342/NTU202402911

摘要

量子運算在機器學習領域的應用越來越受到關注。在各種量子學習模型中，變分量子電路因其易於實作的特性而顯得尤其突出。透過了解這個模型的數學特性，可以發現具有可訓練的量子核函數的模型似乎有提升表現的潛力。我針對不同的問題測試了幾種具有此特性的方法，結果證實這些方法確實提高了模型的表達能力，也突破了使用固定核函數的模型的表現限制。

關鍵字：量子機器學習、量子編碼、量子變分線路、核方法、離散資料

doi:10.6342/NTU202402911

Abstract

The application of quantum computing to machine learning problems has grown increasingly popular. Among the various quantum models, variational quantum circuits are particularly notable for their ease of implementation. Investigating the mathematical properties of such models reveals that those equipped with trainable quantum kernels may achieve enhanced performance. We tested several methods possessing this characteristic across different problems, and the results confirm that such methods not only improve model expressibility but also surpass the performance limits of models with fixed kernels.

Keywords: quantum machine learning, quantum encoding, variational quantum circuit, kernel method, discrete data

doi:10.6342/NTU202402911

Contents

doi:10.6342/NTU202402911

doi:10.6342/NTU202402911

doi:10.6342/NTU202402911

List of Figures

doi:10.6342/NTU202402911

Chapter 1 Intrduction

Since Shor's algorithm was proposed in 1994 [1], the field of quantum computing has received significantly more attention than ever before. This heightened interest is primarily due to quantum computing's potential to solve problems faster or with exponentially fewer resources than classical computers [2–4]. However, not all problems can be efficiently solved via quantum computing, and therefore the potential applications of quantum computing across various classical computing domains have been explored in recent years [5–8], with machine learning emerging as a notable area of research [9].

Classical machine learning has made substantial contributions to everyday applications, such as facial recognition, image generation, and voice-to-text transformation, among others. These applications, however, can be resource-intensive when dealing with large and complex tasks, prompting the question of whether quantum computing could offer time or resource savings. Yet, empirically testing quantum algorithms for large and complex machine learning problems remains challenging due to the current scarcity of quantum computational resources [10] and the impracticality of simulating large-scale quantum computing on classical systems due to the exponentially growing computational complexity.

While empirical testing on a large scale remains impractical, this limitation opens the

doi:10.6342/NTU202402911

door for theoretical exploration of quantum machine learning's mathematical structure. By analyzing potential enhancements within this framework, we can identify candidate approaches that may improve current quantum machine learning algorithms. These theoretical advancements can then be tested empirically on a smaller scale, providing a proof of concept that helps bridge the gap between theoretical potential and practical application.

In this thesis, we explore the mathematical properties of a popular quantum-classical hybrid machine learning algorithm known as variational quantum circuits (VQCs) [11]. We then use these properties to demonstrate the limitations of modifying the circuit structure within the algorithm and discuss methods to overcome these limitations to further enhance the complexity and expressibility of these circuits. We introduce several encoding methods that can break these limitations, including data reuploading [12], trainable embedding [13], and a simplified version of trainable embedding called reduced trainable embedding. The latter maintains a constant parameter number scaling related to the number of qubits, unlike the linear scaling observed in trainable embedding. These methods were tested in various experiments, including the Frozen Lake using the VQC version of reinforcement learning, and two classification problems involving MNIST hand-written digits and the breast cancer dataset. The experimental results show that these methods can not only increase complexity and expressibility theoretically but also enhance the performance of models empirically. Additionally, reduced trainable embedding performed only slightly worse than regular trainable embedding, even though it uses significantly fewer parameters, making it a promising candidate for quantum encoding methods in quantum machine learning.

In Chapter 2, we will discuss the motivation behind this thesis, providing an introduction to the primary quantum model architecture used throughout this thesis—variational

doi:10.6342/NTU202402911

quantum circuits. We will explore its relationship with a well-known classical algorithm —kernel methods—and discuss potential ways to enhance quantum models based on this relationship. Chapter 3 will offer a detailed introduction to methods that could potentially perform better due to the relationship mentioned in Chapter 2, as well as the baseline method for comparison. Chapters 4 and 5 will present the experiments and their results, respectively. Chapters 6 and 7 will cover the discussion and conclusion, respectively.

doi:10.6342/NTU202402911

Chapter 2 Motivation

Given that classical machine learning is a well-established field, while quantum machine learning is relatively nascent by comparison, exploring relationships between these two analogous algorithms can enhance our understanding of the quantum approach. In this chapter, we will provide an overview of variational quantum circuits and kernel methods, demonstrating how kernel methods can encompass variational quantum circuits within their solution spaces. This relationship between quantum models and kernel methods was indicated and proved in [14].

2.1 Variational Quantum Circuits

A Variational Quantum Circuit (VQC) is a hybrid algorithm that integrates classical and quantum computing for machine learning applications [11]. It was initially developed with the goal of addressing machine learning challenges through the use of Noisy Intermediate-Scale Quantum (NISQ) devices. VQCs bear a strong resemblance to classical neural networks (NNs) in that they process input data to produce output values, which are used to make predictions or solve problems. Furthermore, these output values can be trained via the circuit's trainable parameters. The fundamental architecture of a VQC, as illustrated in Figure 2.1, comprises two key quantum operators. The first operator is

doi:10.6342/NTU202402911

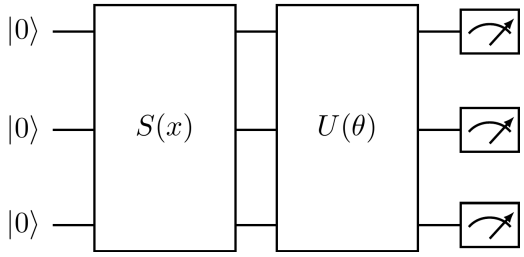


Figure 2.1: This image depicts the typical structure of a Variational Quantum Circuit (VQC), which consists of three parts: (1) the encoding operator $S(x)$, which comprises quantum gates that receive input data x as their parameters; (2) the variational block $U(\theta)$, which includes quantum gates that take trainable parameters θ as their input; and (3) the measurements, used to read out the features.

responsible for data encoding, which involves taking input data x and using it as the parameters for the encoding process. The second operator, denoted as $U(\theta)$, is the variational component that includes the trainable parameters θ. The process concludes with measurements that yield the expectation value of a Hermitian operator H:

$$\langle H \rangle_{x_1,\theta} = \left\langle 0 \left| S^\dagger(x_1) U^\dagger(\theta) H S(x_1) U(\theta) \right| 0 \right\rangle . \tag{2.1}$$

Furthermore, given that $H_\theta = U^\dagger(\theta) H U(\theta)$ is also a Hermitian operator, the expectation value can alternatively be expressed as

$$\langle H_\theta \rangle_{x_1} = \left\langle 0 \left| S^\dagger(x_1) H_\theta S(x_1) \right| 0 \right\rangle . \tag{2.2}$$

In a supervised learning problem, given a training set $\{(x_i, y_i) | i = 1, ..., m\}$, the training methodology involves iteratively adjusting θ to minimize a predefined loss function:

$$L = L((y_1, \langle H_\theta \rangle_{x_1}), ..., (y_m, \langle H_\theta \rangle_{x_m})), \tag{2.3}$$

with the gradient descent of θ being computed on a classical computer.

Usually, to increase the complexity of a circuit, we add more variational layers to our

doi:10.6342/NTU202402911

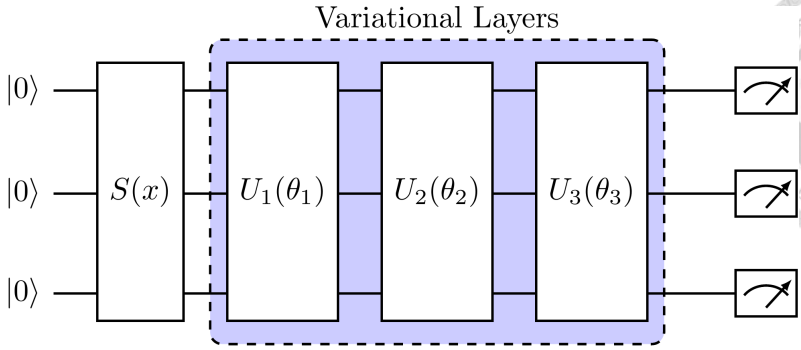


Figure 2.2: A Variational Quantum Circuit (VQC) with multiple variational layers, illustrating increased complexity and potential for more sophisticated processing capabilities.

circuit, as shown in Fig. 2.2.

2.2 Kernel Methods

In many machine learning scenarios, transforming the original input data into a higher-dimensional space, commonly referred to as feature space, proves beneficial. This transformation facilitates easier linear separations between classes of data that are not linearly separable in their original space. However, directly mapping to the feature space can be computationally expensive, especially when the feature space dimension is large. Kernel methods offer a solution by working with a kernel function—a real-valued function that computes the inner product of two input data points in the feature space—instead of performing the actual mapping [15]. This approach can achieve the same effectiveness as using the feature mapping directly for certain algorithms. A prime example is the Support Vector Machine (SVM) [16], which effectively conducts a linear separation in the feature space using only the corresponding kernel. This capability of using only the kernel, rather than performing the separation in the feature space, is underpinned by the representer theorem.

doi:10.6342/NTU202402911

2.3 Representer Theorem

A representer theorem is a theorem that asserts the solutions to certain regularization functionals in spaces of any dimension can be expressed as linear combinations of the data's representers in these functional spaces [17]. This concept was first introduced by Kimeldorf and Wahba in 1970 [18, 19] and was later generalized by Schölkopf, Herbrich, Smola, and Williamson in 2001 [20]. The following representer theorem is the generalized form:

Theorem 1 (Nonparametric Representer Theorem)**.** *Suppose we are given a nonempty set* χ*, a positive definite real-valued kernel* k *on* $\chi \times \chi$*, a training set* $(x_1, y_1), (x_2, y_2), ..., (x_m, y_m) \in \chi \times \mathbb{R}$*, a strictly monotonically increasing real-valued function* g *on* $[0, \infty)$*, an arbitrary cost function* $c : (\chi \times \mathbb{R}^2)^m \to \mathbb{R} \cup \infty$*, and a class of functions*

$$\mathcal{F} = \{f \in \mathbb{R}^{\chi} | f(.) = \sum_{i=1}^{\infty} \beta_i k(., z_i), \beta_i \in \mathbb{R}, \\ z_i \in \chi, \|f\| < \infty\}. \tag{2.4}$$

Here, $\mathbb{R}^{\chi}$ *is the space of functions mapping* χ *into* $\mathbb{R}$*, and* $\|.\|$ *is the norm in the reproducing kernel Hilbert space (RKHS)* H_k *associated with k, i.e. for any* $z_i \in \chi, \beta_i \in \mathbb{R}, i \in \mathbb{N}$*,*

$$\left\| \sum_{i=1}^{\infty} \beta_i k(., z_i) \right\|^2 = \sum_{i,j=1}^{\infty} \beta_i \beta_j k(z_i, z_j). \tag{2.5}$$

Then any $f \in \mathcal{F}$ *minimizing the regularized risk functional*

$$c((x_1, y_1, f(x_1)), ..., (x_m, y_m, f(x_m))) + g(\|f\|) \tag{2.6}$$

doi:10.6342/NTU202402911

admits a representation of the form

$$f(.) = \sum_{i=1}^{m} \alpha_i k(., x_i). \tag{2.7}$$

The representer theorem informs us that to find the optimal solution for a model belonging to the Reproducing Kernel Hilbert Space (RKHS) associated with a given kernel, it is sufficient to know the kernel values for all pairs of input data.

2.4 VQCs are Kernel Methods

To understand the relationship between Variational Quantum Circuits (VQCs) and kernel methods via the representer theorem, let us explore the mathematical foundations of VQCs. By encoding data x into a quantum circuit, as shown in Figure 2.1, this process can be regarded as a feature map from the input data set χ to a Hilbert space V:

$$\Phi^{'} : x \to S(x) \,|\, 0 \,\rangle \equiv |\, \psi_x \,\rangle \,. \tag{2.8}$$

Alternatively, by expanding the codomain, it represents a mapping from χ to $V \otimes V^*$:

$$\Phi : x \to |\psi_x\rangle \langle\psi_x| \equiv \rho(x), \tag{2.9}$$

where V^* is the dual space of V. For $\rho_1, \rho_2 \in V \otimes V^*$, we define an inner product as follows:

$$\langle \rho_1, \rho_2 \rangle = tr[\rho_1 \rho_2]. \tag{2.10}$$

Since every finite inner product space is a Hilbert space, $V \otimes V^*$, endowed with this inner product, becomes a Hilbert space. A positive definite real-valued kernel on $\chi \times \chi$ can then

doi:10.6342/NTU202402911

be naturally defined by:

$$k(x_1, x_2) = tr[\rho(x_1)\rho(x_2)] \tag{2.11}$$

with the corresponding reproducing kernel Hilbert space (RKHS):

$$\mathcal{F} = span(\{k(., x) : x \in \mathcal{X}\}). \tag{2.12}$$

One can prove the output of a VQC defined in equation 2.2 can be expressed as follows:

$$\langle H_\theta \rangle_x = tr[\rho(x)H_\theta], \tag{2.13}$$

and to apply the representer theorem to VQCs, we must prove that $tr[\rho(.)H_\theta] \in \mathcal{F}$.

We recall Theorem 3 in [14]:

Theorem 2. *Given a quantum model defined in equation 2.13, for all $\theta \in \mathbb{R}$, there exists a hermitian*

$$\widetilde{H} = \sum_i a_i \rho(x_i) \tag{2.14}$$

with $x_i \in \mathcal{X}$ and $a_i \in \mathbb{R}$, such that $\langle H_\theta \rangle_x = \langle \widetilde{H} \rangle_x$ for all $x \in \mathcal{X}$.

This immediately implies that any quantum model with the architecture shown in Figure 2.1 can be represented as

$$\langle H_\theta \rangle_x = tr[\rho(x) \sum_i a_i \rho(x_i)] = \sum_i a_i tr[\rho(x)\rho(x_i)], \tag{2.15}$$

which clearly belongs to $\mathcal{F}$ as defined in equation 2.12.

This finally leads us to Theorem 6 in [14]:

Theorem 3. *Given a quantum model defined in equation 2.13, for any θ^* minimizing the*

doi:10.6342/NTU202402911

regularized risk functional c, as defined in Theorem 1, on the training set specified therein, there exists $\{a_i \in \mathbb{R} | i = 1, ..., m\}$ such that

$$\langle H_{\theta^*} \rangle_x = \sum_{i=1}^{m} a_i tr[\rho(x)\rho(x_i)] = \sum_{i=1}^{m} a_i k(x, x_i). \tag{2.16}$$

2.5 Quantum Advantage

Theorem 3 shows that any quantum models formulated as in equation 2.13 for a supervised problem are, in fact, equivalent to a kernel method, with the kernel defined in equation 2.11. Provided that the kernel values for all pairs of training data are calculated, we can train efficiently without a quantum device by adjusting a_i in equation 2.16, instead of training on the quantum model (2.13) by tuning θ. To obtain all kernel values for every pair of input data, i.e., to compute $|\langle 0| S^\dagger(x_i)S(x_j) |0\rangle|^2$ for all i and j, we can perform the calculations on either a quantum device or a classical simulator. Typically, simulating a quantum device is resource-intensive compared to using an actual quantum device due to the exponential growth in computational complexity with the increase in the number of qubits, and indeed, not being classically computable is a necessary condition for achieving quantum advantage over classical methods. However, if the encoding gate $S(x)$ does not entangle the qubits, the kernel values can then be calculated qubit-wise, resulting in a linear increase in classical computational complexity relative to the number of qubits. Many well-known encoding methods do not introduce entanglement, such as basis encoding, angle encoding, and Quantum Random Access Codes (QRACs), which are used in the experiments in this thesis and will be introduced in Section 3.1. Some encoding methods, like amplitude encoding, even though they introduce entanglement, the quantum kernel can still be efficiently calculated by a classical device [21]. By using

doi:10.6342/NTU202402911

these encoding methods, we can train the model without involving any quantum device. It appears that the solution space of a quantum model can be covered by a classically computable method when using certain encoding methods. Can we still expect such quantum models to have any advantage over classical methods? The answer is positive, as reported by [22]. They indicated that some quantum kernel-based methods might exhibit extremely weak generalization performance due to the orthogonality of data points in the quantum space. Thus, using VQCs with a classically computable kernel can still offer potential advantages in terms of generalization ability.

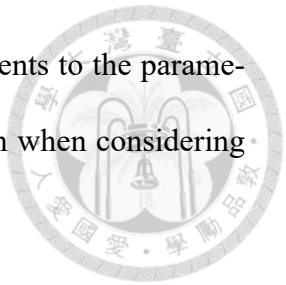
2.6 Increasing Models Complexity

Understanding the relationship between VQCs and kernel methods provides insights into how we can increase the complexity of a circuit without remaining confined to the same solution space. For instance, merely adding more parameters or quantum gates to the variational layer $U(\theta)$ in Figure 2.1 expands the solution space only to the extent described in equation 2.16. It is worth noting that $U(\theta)$ also has an inherent dimensional limitation: for q qubits, 2^2q-1 parameters are required to define a general unitary operation acting on them. To further enhance complexity, one effective approach is to modify our encoding methods.

A fundamental reason why a VQC is equivalent to a kernel method lies in the principles of quantum mechanics: any evolution of a quantum state within a closed system must be unitary, and a unitary transformation preserves norm. Assuming our quantum device represents such a closed system, this property implies that all operators are unitary. Consequently, once the encoding $S(x)$ is established and fixed, the distance between different

doi:10.6342/NTU202402911

input data points x_i and x_j remains unchanged regardless of adjustments to the parameters θ. Therefore, the kernel value $k(x_i, x_j)$ is not affected by θ even when considering $f : x \to U(\theta)S(x)|0\rangle$ as our feature map.



By altering the encoding methods, we gain the ability to shift the relative positions of input data within the Hilbert space. As a result, data belonging to different categories have the opportunity to be separated from one another in the Hilbert space. In the next chapter, we will introduce two discrete encoding methods that lead to modifiable kernels: trainable embedding and its simplified derivative, reduced trainable embedding. Additionally, we will introduce a technique known as data reuploading, which can also lead to a trainable kernel by modifying the architecture of VQCs.

doi:10.6342/NTU202402911

Chapter 3 Data Encoding Methods

In this chapter, we will introduce the encoding methods that were tested and analyzed throughout this thesis.

3.1 Quantum Random Access Codes

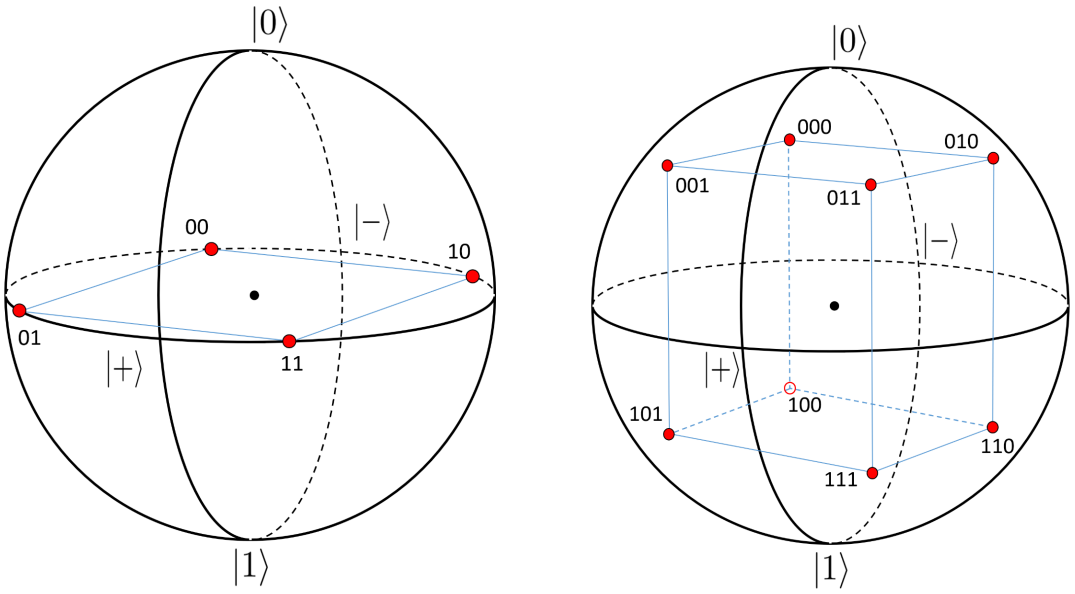
Random Access Codes (RACs) are encoding methods that compress a bit-string into a shorter one, enabling successful decoding of any one of the bits with a certain probability. Quantum Random Access Codes (QRACs), on the other hand, encode a bit-string into a smaller number of qubits, achieving compression beyond classical limits. Initially conceptualized by Stephen Wiesner in a 1983 publication [23], QRACs was later independently formulated by Ambainis et al. in [24, 25]. They proved that encoding 2 bits into 1 qubit yields, at best, a probability of 0.85 for successfully recovering any one of the bits. This type of QRAC is referred to as (2, 1, 0.85)-QRAC. Similarly, when encoding 3 bits into 1 qubit, the maximum probability of successful recovery is 0.79, denoted as (3, 1, 0.79)-QRAC. In this thesis, these encoding methods are simply referred to as (2, 1)-QRAC and (3, 1)-QRAC, respectively, and are illustrated in Figure 3.1. Only (3, 1)-QRAC will be used in this study.

The advantage of employing QRACs in quantum machine learning lies in the ability

doi:10.6342/NTU202402911

to encode input data using fewer resources (qubits) while preserving a significant level of information. Furthermore, the distance between any two quantum states encoded with QRACs is proportional to the Hamming distance of their original bit-strings [26]. This property could offer distinct benefits for quantum computational processes. QRACs have been tested on several machine learning problems and have yielded positive results [26–28].

Note that QRAC is a fixed feature map from classical bits to quantum states, and there is no trainable part within.



(a) (2,1)-QRAC (b) (3, 1)-QRAC

Figure 3.1: QRACs on the Bloch sphere. The Bloch sphere is a graphical representation of the quantum states of a qubit. A point on the sphere, characterized by the polar angle θ and the azimuthal angle φ, represents the one-qubit state $\cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle$.

3.2 Trainable Embeddings

Trainable embeddings (TE) are discrete encoding methods that assign trainable values to each unique input. This method was first applied to quantum circuits by [13], who

doi:10.6342/NTU202402911

explored encoding 3 bits into 1 qubit for comparison with a (3, 1)-QRAC. To encode n bits into 1 qubit using TE, we assign two trainable parameters to each unique n-bit string. These parameters are then used as the angles for a general 1-qubit rotation gate operating on the qubit. In this thesis, such encoding methods are referred to as $(n, 1)$-TE for encoding n bits into 1 qubit. It is important to note that a general 1-qubit rotation gate actually has three degrees of freedom. However, when this gate is applied to a fixed quantum state, the effective degrees of freedom are reduced to two. In quantum machine learning, encoding typically starts with the initial qubit state, where all qubits are in the 0 state. Thus, only 2 parameters are needed instead of 3 to specify the rotation of a 1-qubit gate due to this restriction. Since n bits can represent 2^n unique bit strings, encoding $n \times q$ bits into q qubits—by splitting the bit string into q n-bit strings—requires a total of $2^n \times 2 \times q$ trainable parameters. That is, parameters of different qubits are independent, and there are $2^n \times 2$ parameters for each qubit.

Consider the $(2, 1)$-TE as an example. Encoding a 2-bit string b_0b_1 to the i-th qubit involves mapping b_0b_1 to the 1-qubit state $Rx((\phi^i_{b_0b_1})_0)Ry((\phi^i_{b_0b_1})_1)|0\rangle$ where ϕ are trainable parameters. Therefore, to encode a bit-string $b_0b_1 \dots b_{2q-1}$ to q qubtis, it will be mapped to the quantum state:

$$S_{\text{TE}}(b_0b_1 \dots b_{2q-1})|0\rangle = \bigotimes_{i=0}^{q-1} Rx((\phi^i_{b_{2i}b_{2i+1}})_0)Ry((\phi^i_{b_{2i}b_{2i+1}})_1)|0\rangle \tag{3.1}$$

where $Rx(\phi) = e^{-i\frac{\phi}{2}X}$ and $Ry(\phi) = e^{-i\frac{\phi}{2}Y}$ are the rotation gates around the x-axis and y-axis, respectively, with X and Y being the Pauli-X and Pauli-Y matrices.

doi:10.6342/NTU202402911

3.3 Reduced Trainable Embeddings

Here, we propose a method inspired by TE, called reduced TE. Unlike the independent parameters for each qubit in TE, in reduced TE, all qubits share the same set of parameters. The number of parameters for encoding is then fixed to $2^n \times 2$ for encoding n bits into 1 qubit. In this thesis, this encoding method is referred to as TE-reduced. Sharing parameters between qubits here is very similar to QRACs, which share the same mapping from bit strings to qubit states for each qubit. However, unlike QRACs that have a fixed mapping, TE-reduced allows for arbitrary tuning of such mappings, and in addition, the mappings of QRACs are included within the codomain of TE-reduced. Therefore, models using TE-reduced as their encoding method inherently have equal or better global optima than those using QRACs, with only a fixed and small amount of extra parameters.

Taking the $(2, 1)$-TE-reduced as an example, to encode a bit-string $b_0 b_1 \dots b_{2q-1}$ to q qubits, it will be mapped to the quantum state:

$$S_{\text{TE-reduced}}(b_0 b_1 \dots b_{2q-1})|0\rangle = \bigotimes_{i=0}^{q-1} Rx((\phi_{b_{2i}b_{2i+1}})_0) Ry((\phi_{b_{2i}b_{2i+1}})_1)|0\rangle \tag{3.2}$$

3.4 Data reuploading

Data reuploading refers to a variational circuit architecture that encodes input data multiple times, interspersing these encodings with variational layers as shown in the Figure 3.2. This structure has the ability to approximate a universal model even when there is only one qubit, provided the data is reuploaded sufficiently. This concept mirrors the principle in classical neural networks where a network with just one hidden layer can approximate

doi:10.6342/NTU202402911

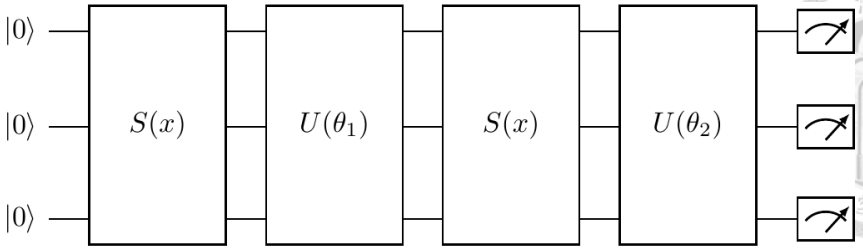


Figure 3.2: This circuit illustrates the use of a data reuploading strategy where the encoding block $S(x)$ is applied before each variational layer $U(\theta_i)$.

a universal model if it contains enough neurons [12].

If we regard $S^{'}(x, \theta_1) = S(x)U(\theta_1)S(x)$ as our feature map, then the corresponding kernel will be:

$$k(x_1, x_2) = |\langle 0|S^{\dagger}(x_1)U^{\dagger}(\theta_1)S^{\dagger}(x_1)S(x_2)U(\theta_1)S(x_2)|0\rangle|^2. \tag{3.3}$$

As long as $S(\cdot)$ and $U(\cdot)$ do not commute, the kernel value changes when adjusting θ_1. This architecture ensures that our data are mapped to non-fixed quantum states, thereby allowing the distance between each pair of data points to be adjusted.

doi:10.6342/NTU202402911

Chapter 4 Experiments

This chapter outlines three experiments: Frozen Lakes, MNIST handwritten digits, and the breast cancer dataset. The primary objective of these experiments is to compare the fixed kernel encoding methods QRACs with the trainable kernel encoding methods TE and TE-reduced, and to evaluate the impact of data reuploading on them. These encoding methods were described in Chapter 3. Additionally, we compare the performance of models with varying numbers of variational layers to illustrate the effects of increasing complexity in both the kernel and the variational components. For context, results from classical neural networks (NNs) are also provided for comparison. It is important to note that all quantum models are simulated on a classical computer, not on an actual quantum device.

4.1 Frozen Lakes

A frozen lake is represented as a 2D grid map, where the player's objective is to reach the goal in the fewest possible steps while avoiding falling into a hole. Figure 4.1 displays a 4×4 map. The actions a player can take in this game include moving one step to the left, right, up, or down. The only information available to the player is their current location and the reward received upon stepping onto a grid. The machine learning model will iteratively

doi:10.6342/NTU202402911

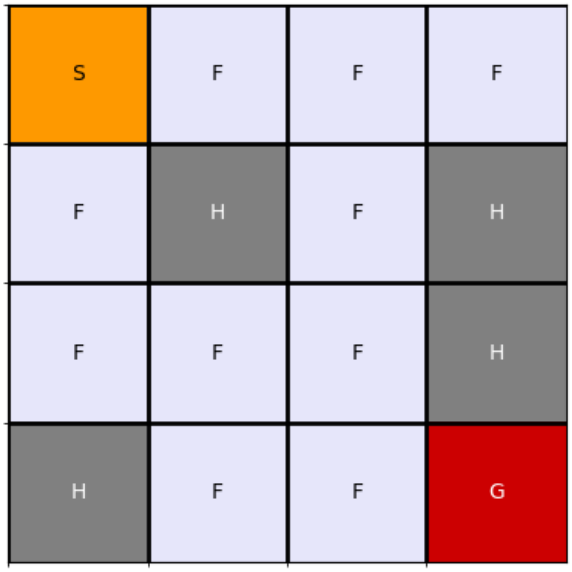


Figure 4.1: This figure illustrates a 4×4 grid map used in the Frozen Lake game. 'S' is the starting point, 'F' is the frozen lake surfaces that players can safely walk on, 'H' is the holes, and 'G' is the goal. The player' s objective is to navigate from the start point to the goal while avoiding the holes.

navigate through the map to learn the optimal path to the goal. For this experiment, we employ two algorithms: the Deep Q-Network (DQN) and the variational quantum Deep Q-Network (VQ-DQN) [29]. The VQ-DQN algorithm modifies the traditional DQN by substituting its neural networks (NNs) with VQCs, as described in Chapter 2.1.

4.1.1 Environment

For an $n \times n$ frozen lake, the total number of possible states is n^2, and in each state, the agent can take one of four actions: move left, right, up, or down. In these experiment, we test only on 4×4 maps. The rewards are assigned as follows: stepping into a hole incurs a -0.2 penalty, reaching the goal grants a $+1.0$ reward, and moving elsewhere results in a -0.01 penalty. The game concludes when the agent either falls into a hole, reaches the goal, or the number of steps—that is, the number of actions taken—reaches

doi:10.6342/NTU202402911

the limit of 2500, marking the end of an episode. The training process terminates either when the agent can consistently reach the goal via an optimal path, considered a success, or when the number of episodes reaches a predefined limit of 500, which is considered a failure.

4.1.2 Training and Performance Estimation

This experiment was conducted using 20 random seeds. For each seed, a frozen lake map was randomly generated, with the starting point and goal fixed while the locations of the holes were randomly shuffled. It was ensured that at least one accessible path to the goal remained open.

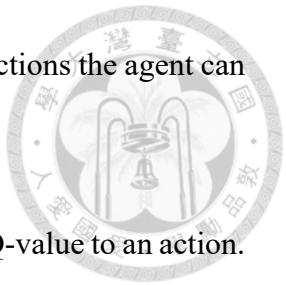
For each seed, we recorded whether the trial succeeded or failed, along with the total number of episodes and steps taken until the training process terminated. Averages for episodes and steps were calculated exclusively from the successful trials, and the overall success rate was also determined across the 20 seeds.

It is worth noting that the parameters were updated once per step, hence the overall training time is directly related to the total number of steps rather than the total number of episodes. Therefore, the total steps will serve as our main indicator of model performance.

4.1.3 Deep Q-Network

Q-learning is an algorithm that involving an agent interacting with the environment. The agent evaluates the current and future rewards that are possibly obtained when he takes a specific action, and takes the best action according to the evaluation. In practice, we define an agent doing the evaluations, and a state space S containing all possible states

doi:10.6342/NTU202402911



that the agent can be in, and an action space A including all possible actions the agent can take for each state.

Given a state s_t at time t, the agent assigns a score known as the Q-value to an action. The Q-value of an action a_t in state s_t is defined as a function that represents the sum of the immediate reward plus the discounted future rewards. The future rewards are multiplied by a discount factor γ that weighs their importance. The discount factor γ is a number between 0 and 1 (not less than zero), with values closer to 1 placing more emphasis on future rewards. This can be represented as follows:

$$Q(s_t, a_t) = R_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a) \tag{4.1}$$

Here, R_{t+1} is the reward received after taking action a_t in state s_t, and $\max_{a \in A} Q(s_{t+1}, a)$ represents the maximum Q-value for the next state s_{t+1}, across all possible actions a. This formulation allows the agent to evaluate the total expected utility of taking action a_t in state s_t, considering both the immediate and future rewards.

What a Deep Q-Network (DQN) accomplishes is the substitution of the Q-value functions, $Q(s, a)$, with a neural networks (NNs) model $Q_\Theta(s_t, a_t)$ with trainable parameters Θ. On the other hand, the Variational-Quantum Deep Q-Network (VQ-DQN) replaces the NNs used in DQN with VQCs, as discussed in Chapter 2.1. The specific VQCs utilized in this experiment are illustrated in Figure 4.2. The outputs of a VQC are connected to a dense layer consisting of four neurons. These four neurons represent the Q-values for the actions of moving left, down, right, and up, respectively, and thus they take only s_t as their input.

These two approaches achieve self-consistency by minimizing the empirical risk as-

doi:10.6342/NTU202402911

sociated with the cost function. The cost function is defined as the square of the difference between the left-hand side and the right-hand side of Equation 4.1. Specifically, given the trainable parameters θ, the cost function can be expressed as follows:

$$L(\theta, s_t, a_t, R_{t+1}, s_{t+1}) = [Q_\theta(s_t, a_t) - (R_{t+1} + \gamma \max_{a \in A} Q_\theta(s_{t+1}, a))]^2. \quad (4.2)$$

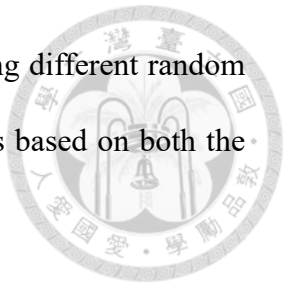
However, to solve a known problem called overestimation, we used a technique called target network [30], which utilizes two sets of parameters for a model: one set is used to calculate the left-hand side of equation 4.1 and is trained during iterations; the other set is for computing the right-hand side and is not trained during the experiment. This second set of parameters will be synchronized with the first set every 20 iterations, as specified for this experiment. By denoting the second set of parameters as θ', the cost function becomes:

$$L(\theta, s_t, a_t, R_{t+1}, s_{t+1}) = [Q_\theta(s_t, a_t) - (R_{t+1} + \gamma \max_{a \in A} Q_{\theta'}(s_{t+1}, a))]^2. \quad (4.3)$$

We also used a technique called memory replay. During the training process, the agent records four pieces of information to its memory upon taking an action: the current state s_t, the action a_t taken, the reward R_{t+1} obtained, and the next state s_{t+1} entered after the action. In our setup, a maximum of 80 sets of these data points can be stored. Once this capacity is reached, the oldest data is overwritten with new data. After each action is taken, one iteration of gradient descent on the empirical risk of the cost function 4.3 is performed on 5 sampled data points from the recorded data. The total number of steps taken by the agent during the entire training process is proportional to the training time, serving as an indicator of the model's performance. Another performance metric is the success rate of

doi:10.6342/NTU202402911

finding the optimal path to the goal across 20 experiments, each using different random seeds. Therefore, in Chapter 5, we compare the experimental results based on both the total number of steps and the success rate.



4.1.4 Data Encoding

As shown in Figure 4.2, the model outputs four values, each representing one of the Q-values of the four possible actions. Therefore, the model $Q_\theta(s_t, a_t)$ effectively comprises four related sub-models $Q^i_\theta(s_t)$, each representing the Q-value for the i-th action given the state s_t. Consequently, the model only takes states as its input. In the frozen lake problem, states or locations are represented as discrete data. For an $n \times n$ map, the number of states is n^2, and we assign labels from 0 to $n^2 - 1$ to these states. These labels are then transformed into bit strings of length $\log_2(n^2)$. The bit string is of length 4 for a 4×4 map, and we use two qubits for the quantum model, with each qubit encoding 3 bits using (3, 1)-QRAC, (3, 1)-TE, or (3, 1)-TE-reduced. The first to the third bits are encoded on the first qubit, and the second to the fourth bits are encoded on the second qubit.

4.2 MNIST Handwritten Digits

4.2.1 Environment

In this experiment, the task involves classifying handwritten digits from the MNIST dataset. Each image in the dataset is a 28×28 pixel grayscale image, with each pixel value being an integer in the range of 0 to 255. To simplify the task, the images were resized 5×5 pixels. Furthermore, pixel values were binarized: values less than 128 were

doi:10.6342/NTU202402911

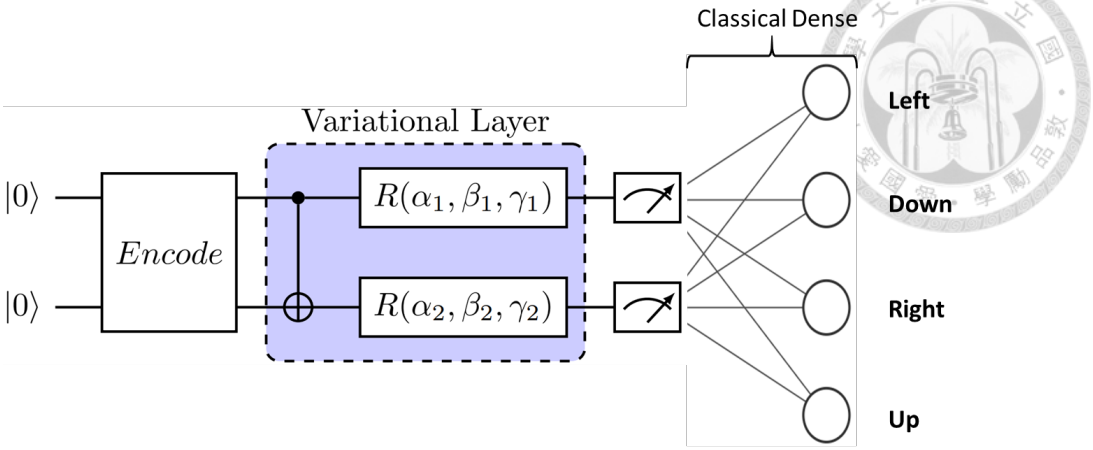


Figure 4.2: The quantum circuit used for Frozen Lake experiments. The first block is the encoding layer, which employs the methods QRAC, TE, or TE-reduced as detailed in Chapter 3. The second block is the variational layer, comprising general rotations on each qubit, with each rotation characterized by three trainable parameters. The process concludes with Pauli-Z measurements on each qubit to obtain the expectation values. Finally, a fully connected dense layer is attached to produce four output values, representing the Q-values for the four actions.

set to 0, and values greater than or equal to 128 were set to 1. Only the digits 3 and 6 were selected for classification. Figure 4.3 illustrates the process of resizing an image and transforming it into binary pixel values. Occasionally, images of the digits 3 and 6 become indistinguishable after simplification. To address this issue, the experiment was conducted in two phases: one including these indistinguishable cases and another with them removed. Initially, there were 7,141 images of the digit 3 and 6,876 images of the digit 6. After removing the indistinguishable cases, 2,523 images of digit 3 and 2,861 images of digit 6 remained.

4.2.2 Training and Performance Estimation

We used 7-fold cross-validation to compare the average performance of different models. To prevent overfitting, we allocated five folds for training, one fold for validation, and the remaining fold for testing. During the course of the 100-epoch training period, we recorded the parameters that yielded the best validation loss. Each cross-validation

doi:10.6342/NTU202402911

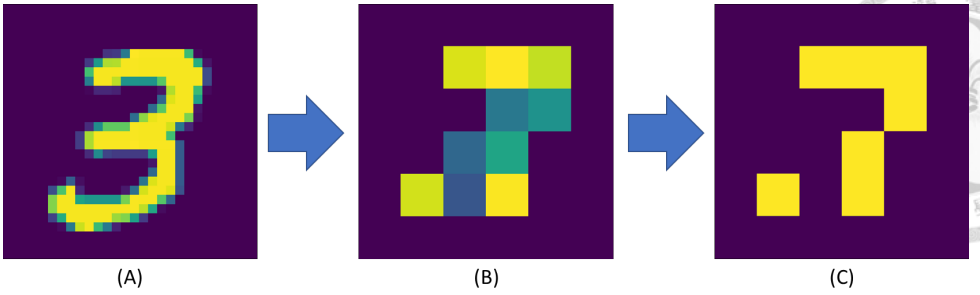


Figure 4.3: This figure illustrates the process of image simplification. The leftmost image shows an original image of the digit 3. The middle image depicts the resized version, obtained by averaging over pixels. After normalizing the pixel values, the image is further simplified into binary pixel values using a threshold of 0.5, as shown in the rightmost image.

cycle was conducted with 10 random seeds, resulting in 10 best validation losses and 10 sets of parameters. From these, we selected the set of parameters with the optimal validation loss to calculate the test accuracy, which served as the final result for one cycle of cross-validation. After repeating this process across 7 different combinations of folds, we calculated the average of the 7 test accuracies to determine the final outcome.

4.2.3 Algorithm

The quantum models used in this task are illustrated in Figure 4.4. Unlike the circuit utilized in the Frozen Lake experiments, which employed general rotation gates in conjunction with CNOT gates, this implementation makes use of Ising gates. An Ising XX gate can be expressed as:

$$XX(\theta) = e^{(-i\frac{\theta}{2}(X\otimes X))} \tag{4.4}$$

where X is the Pauli-X matrix.

Following data simplification, the task is recast as a binary classification problem: images of the digit 3 are labeled as 1, and those of the digit 6 as -1. The cost function is defined in terms of binary cross-entropy, calculated between the labels and the Pauli-Z

doi:10.6342/NTU202402911

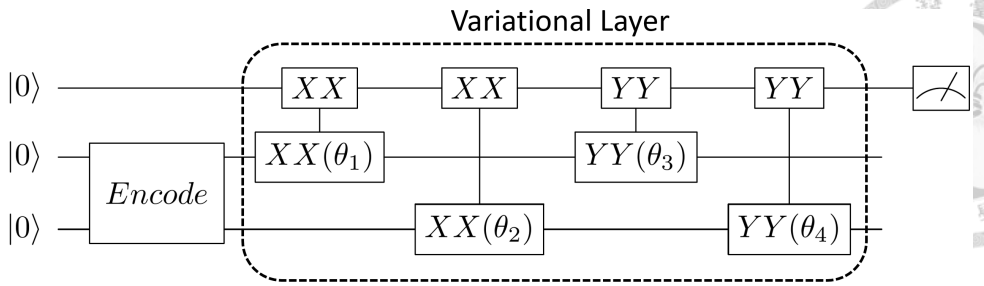


Figure 4.4: This figure illustrates the quantum circuit configuration employed for the MNIST and breast cancer datasets. Although only three qubits are depicted here, the actual implementations utilize 10 and 4 qubits for the MNIST and breast cancer datasets, respectively. In these circuits, input data are encoded onto all qubits except the first. Subsequently, Ising XX and YY gates are applied between the first qubit and all others, which are represented in the diagram as XX/YY gates. The analysis concludes with the measurement of the first qubit along the z-axis to determine its expectation value.

expectation values of the first qubit. Model parameters are updated using gradient descent with Adam as the optimizer, applied to each batch comprising 64 samples.

4.2.4 Data Encoding

After data simplification, the images are resized to 5×5 with binary pixel values. Each image can then be represented as a bit string of length 5^2. Using the (3, 1)-QRAC, (3, 1)-TE, and (3, 1)-TE-reduced encoding methods, every three bits are encoded onto the qubits sequentially. Since the length of this bit string is not divisible by three, the value 0 is encoded onto the last qubit to complete the triplet. That is, with an image size of 5×5, the bit string is of length 25, requiring 9 qubits. The first 8 qubits each encode 3 bits. The last qubit, receiving only 1 bit, has two 0 bits appended to it to form a triplet for encoding on the last qubit.

doi:10.6342/NTU202402911

4.3 Breast Cancer Dataset

4.3.1 Environment

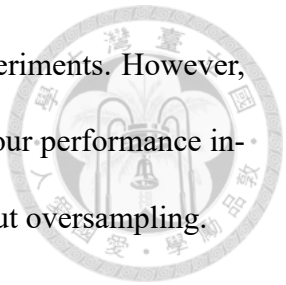
The objective of this dataset[1] is to predict whether a breast cancer patient will experience recurrence in the future, using 9 discrete features. Initially, the dataset comprised 286 instances, including 201 instances of no recurrence and 85 instances of recurrence. After removing instances with missing features, 277 instances remained, consisting of 196 no-recurrence instances and 81 recurrence instances. To reduce the number of qubits required, we selected only four features for training: Menopause, Tumor-Size, Node-Caps, and Deg-Malig, which contain 3, 12, 2, and 3 categories, respectively. Also, due to the imbalance in this dataset, the F1-score appears low for all models. To assess the maximum F1-score performance achievable, we conducted additional experiments with oversampling to balance the dataset during training.

4.3.2 Training and Performance Estimation

The training process and performance estimation in this experiment are very similar to those discussed in Section 4.2.2, with a few notable adjustments. Due to the limited number of instances in this dataset, we employed 5-fold cross-validation instead of 7-fold. Additionally, the total number of training epochs was increased to 1500, compared to 100 in the previous section. Furthermore, due to the imbalance in the dataset, both accuracies and F1-scores were recorded to provide a more comprehensive assessment of model performance. Here, we also use the validation data to select the best parameters during

[1] Dataset available at https://archive.ics.uci.edu/dataset/14/breast+cancer.

doi:10.6342/NTU202402911



training, similar to our approach in the MNIST hand-written digits experiments. However, for experiments involving oversampling, we employ the F1-score as our performance indicator instead of loss, which is the metric used in experiments without oversampling.

4.3.3 Algorithm

Since this experiment also involves binary classification, the algorithm used is the same as the one described in Section 4.2.3.

4.3.4 Data Encoding

Since we have selected only four features for training, each with 3, 12, 2, and 3 discrete categories respectively, we require 2, 4, 1, and 2 bits to represent them. This totals 9 bits. We then divide these 9 bits into three 3-bit strings, which are encoded into 3 qubits using the (3, 1)-QRAC, (3, 1)-TE, and (3, 1)-TE-reduced encoding methods.

doi:10.6342/NTU202402911

Chapter 5 Results

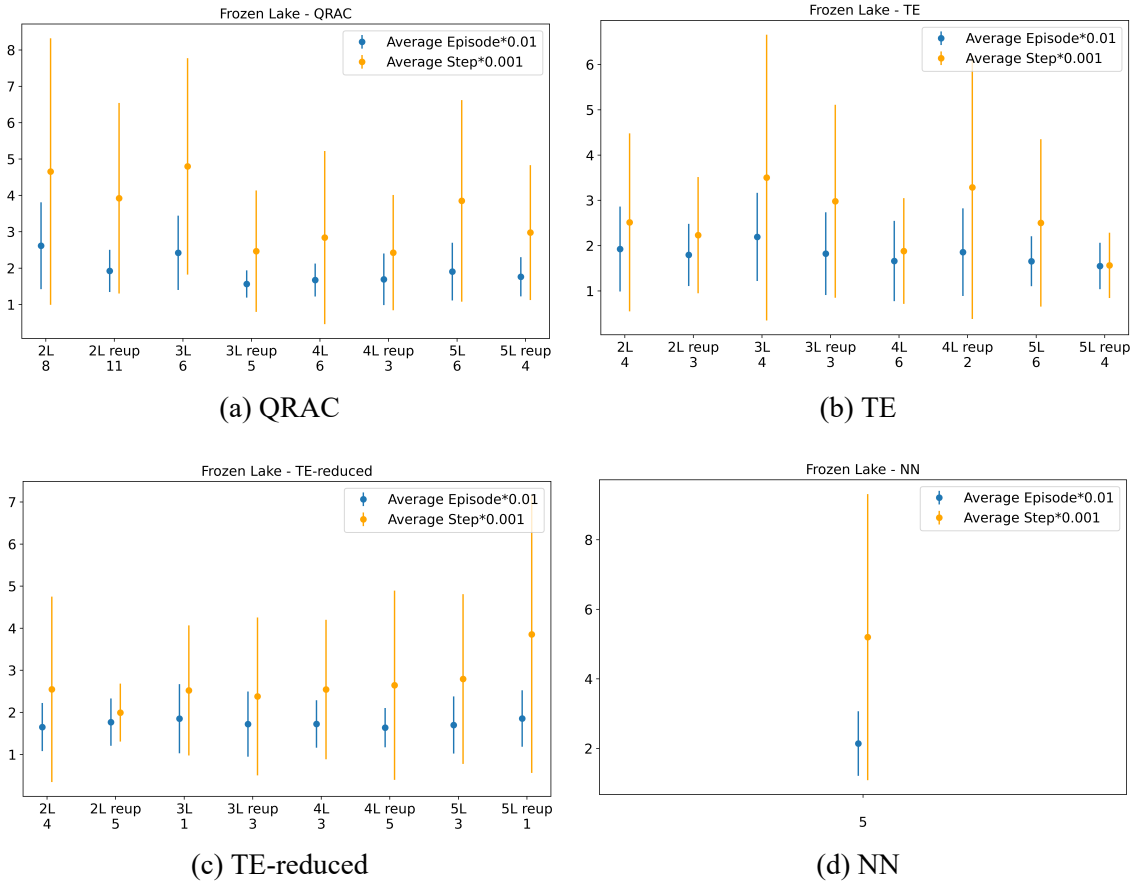
5.1 Frozen Lake

The results are presented in Figure 5.1. The yellow lines depict the average number of steps the agent required to consistently achieve an optimal path to the goal. More precisely, the agent is considered to have completed its learning when it attains an average score of 0.92 over the last 100 epochs. The number of average steps is proportional to the time taken by the agent to learn the problem, making this a primary performance indicator; lower values are preferable. Another crucial performance metric is the number of failures across 20 random seeds, indicated beneath each setting name in Figure 5.1; again, lower numbers signify better performance. It is important to note that the average number of steps is calculated only from successful random seeds, while failed attempts are excluded.

As observed, with an equal number of variational layers, the performance, in terms of training time and success rate, improves when the data reuploading method is applied to QRAC and TE. However, for TE-reduced, data reuploading appears to slightly worsen performance. Among the three encoding methods, TE and TE-reduced generally exhibit a better success rate and shorter training times. Despite being a simplified version of TE, TE-reduced still maintains comparable performance.

doi:10.6342/NTU202402911

Interestingly, enhancing the encoding part of the model can improve performance, while increasing the number of variational layers does not. This suggests that once the complexity of the variational component reaches its limit, further enhancing the complexity of the kernel, that is, the encoding part, can still boost the model's performance.



(a) QRAC (b) TE

(c) TE-reduced (d) NN

Figure 5.1: Comparative results on the frozen lake problem. In each setting name, "L" indicates the total number of variational layers used in the quantum circuits, and the presence of "reup" signifies whether the data reuploading method is employed. The numbers beneath the setting names represent the count of failures or outliers over 20 random seeds, with lower numbers indicating better performance. The yellow points represent the average number of steps taken to successfully learn how to reach the goal, while the blue points correspond to the average number of episodes required. The bars associated with these points indicate one standard deviation.

doi:10.6342/NTU202402911

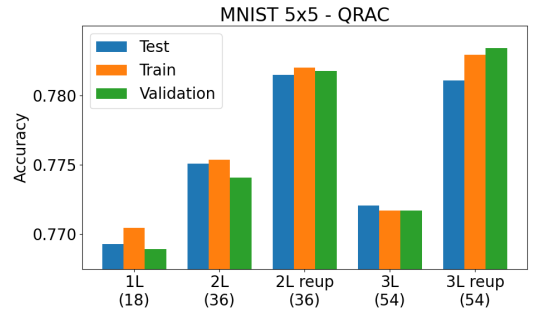
5.2 MNIST Handwritten Digits

The results are presented in Figures 5.2 and 5.3, representing experiments conducted without and with the removal of contradictory data, respectively, as mentioned in Section 4.2.1. In these figures, accuracies are displayed, with higher values indicating better performance.

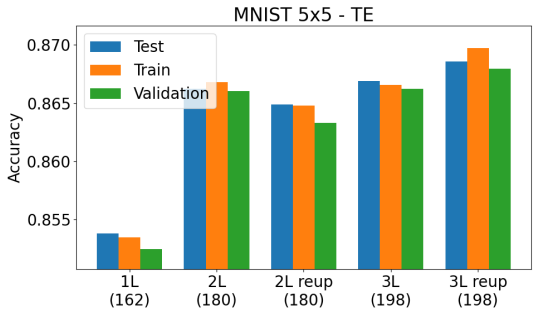
In the experiment without removing the contradictory data, models using QRAC and TE-reduced as encoding methods experience a performance boost with data reuploading. In contrast, models employing TE show no significant performance difference whether data reuploading is applied or not. This lack of improvement suggests that TE models may have reached their performance limit for this problem, as evidenced by the fact that increasing the number of variational layers to more than 2 does not yield any performance gains. Notably, the performance of quantum models using TE, TE-reduced, and classical neural networks (NNs) generally aligns, consistently outperforming those that employ QRAC.

In the experiment where contradictory data is removed, the problem becomes very straightforward, and no encoding methods derive significant benefits from data reuploading. However, it is observed that QRAC consistently exhibits the weakest performance, while the other methods display comparable levels of performance.

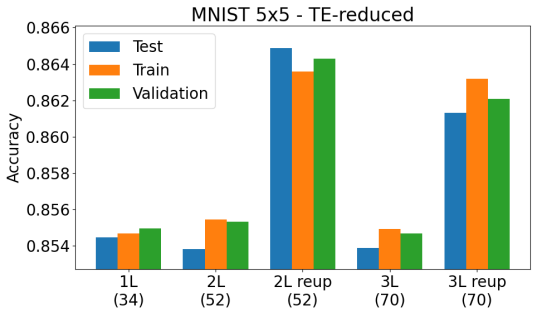
doi:10.6342/NTU202402911



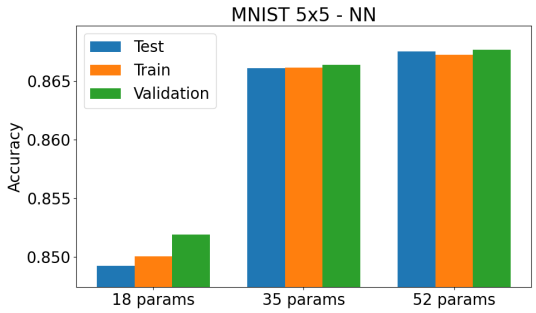
(a) QRAC



(b) TE

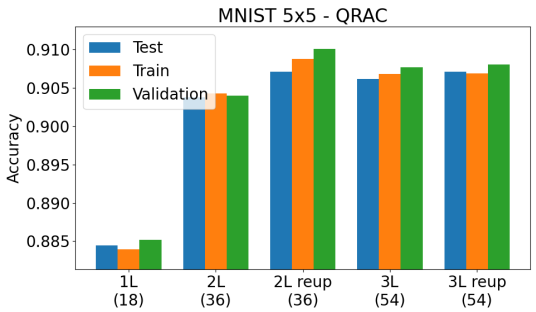


(c) TE-reduced

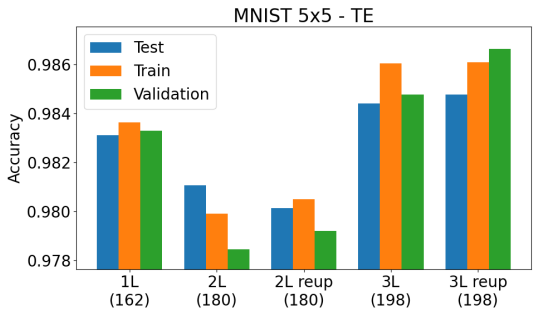


(d) NN

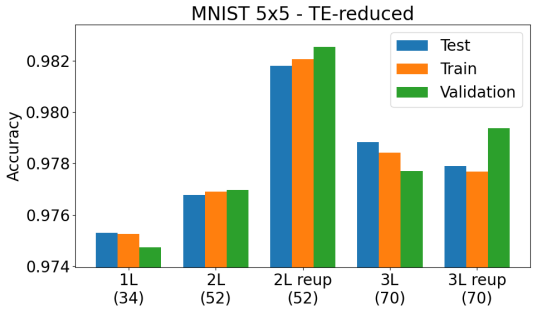
Figure 5.2: Performance of models on the MNIST hand-written digits dataset **without** removing contradictory data. The numbers in parentheses represent the total number of trainable parameters for each respective model.



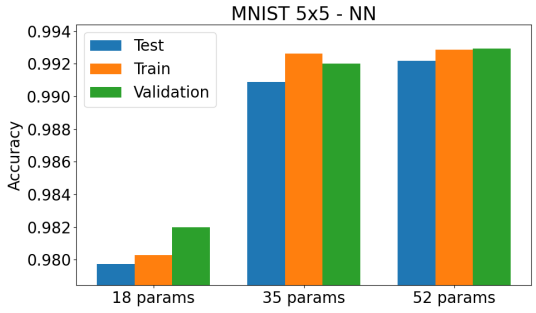
(a) QRAC



(b) TE



(c) TE-reduced



(d) NN

Figure 5.3: Performance of models on the MNIST hand-written digits dataset **with** removing contradictory data.

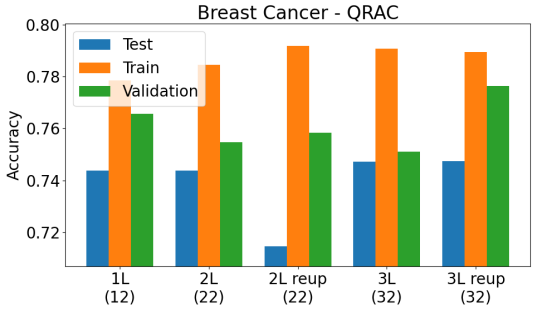
doi:10.6342/NTU202402911

5.3 Breast Cancer Dataset

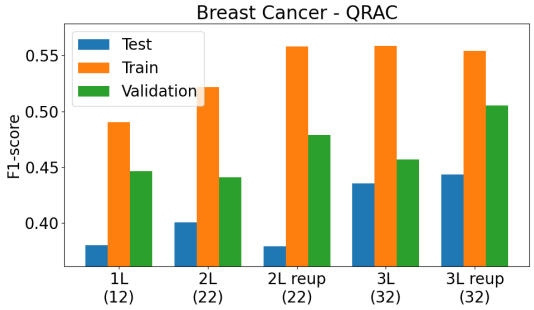
As previously noted in Section 4.3.1, this dataset is imbalanced. In addition to analyzing the original data, whose results are shown in Figure 5.4, we conducted experiments on oversampled training data. In these experiments, instances in the training set labeled '1' were duplicated to match the number of instances labeled '0'. The results of these experiments are displayed in Figure 5.5. To accurately reflect the performance on this imbalanced dataset, we present not only the accuracy but also the F1-scores for each model.

Unlike the Frozen Lake or the MNIST datasets, the impact of data reuploading on model performance in the breast cancer dataset, with or without oversampling, is not consistent. Furthermore, quantum models utilizing QRACs as their encoding method generally outperform other quantum models, despite QRACs employing fewer parameters. Additionally, models with only one variational layer often perform as well as or better than those with multiple layers. We also observe serious overfitting across all models. These suggest that simpler models are more effective for this dataset, which has a relatively small number of instances. Consequently, QRACs without data reuploading might yield better or comparable results.

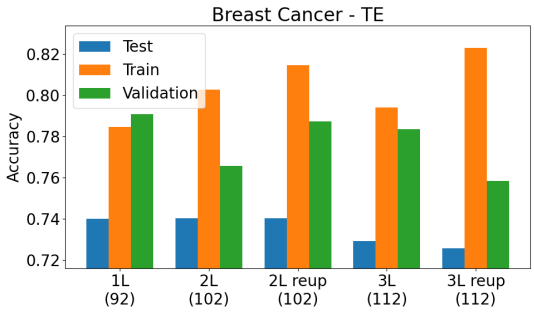
doi:10.6342/NTU202402911



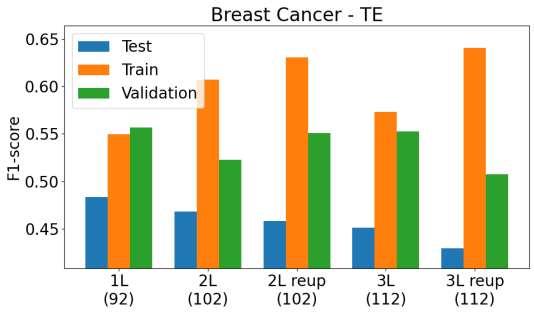
(a) QRAC - Accuracy



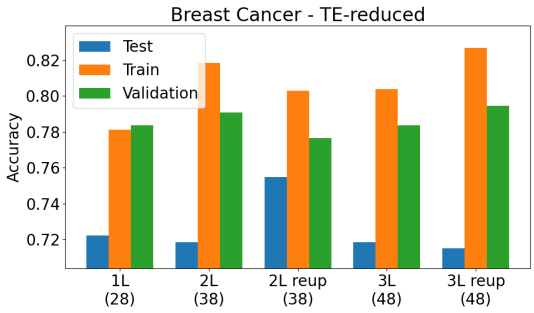
(b) QRAC - F1-score



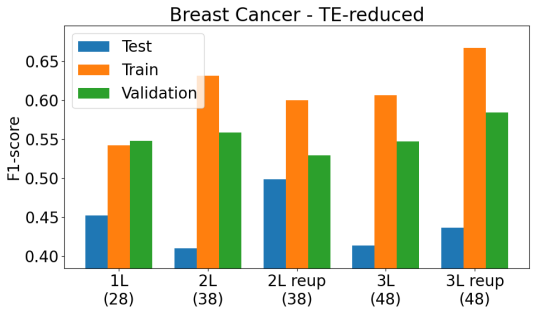
(c) TE - Accuracy



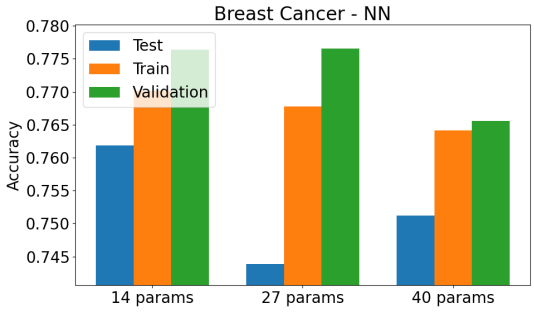
(d) TE - F1-score



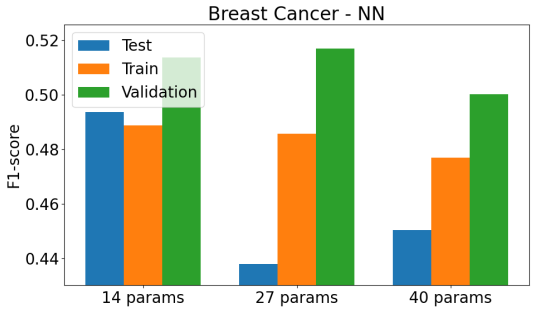
(e) TE-reduced - Accuracy



(f) TE-reduced - F1-score



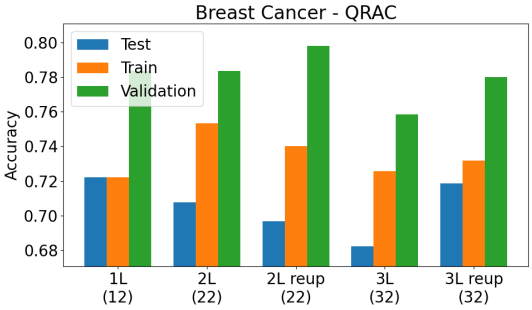
(g) NN - Accuracy



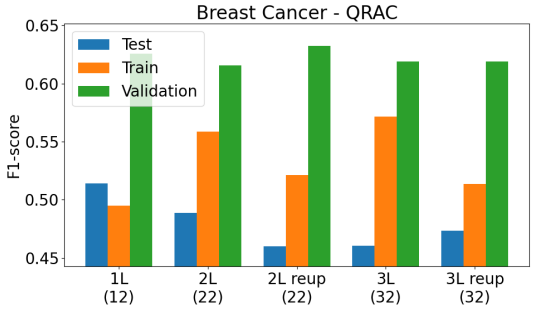
(h) NN - F1-score

Figure 5.4: Performance of models on the breast cancer dataset **without** oversampling.

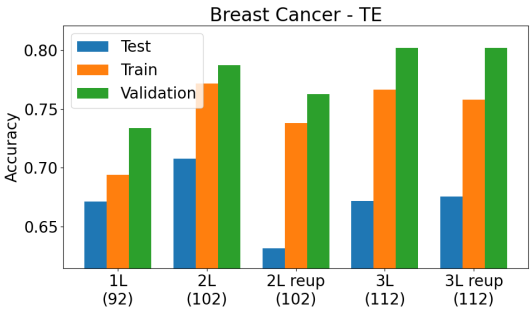
doi:10.6342/NTU202402911



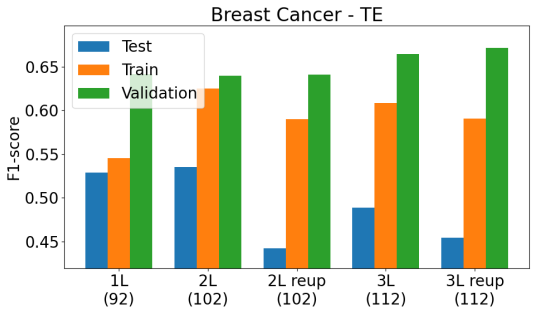
(a) QRAC - Accuracy



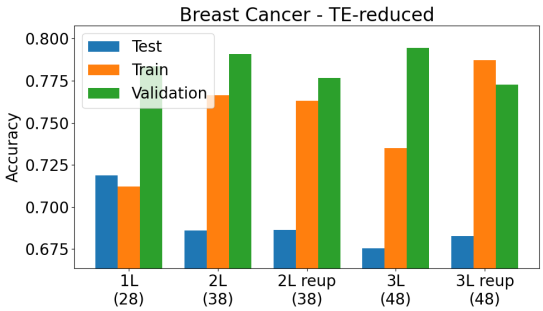
(b) QRAC - F1-score



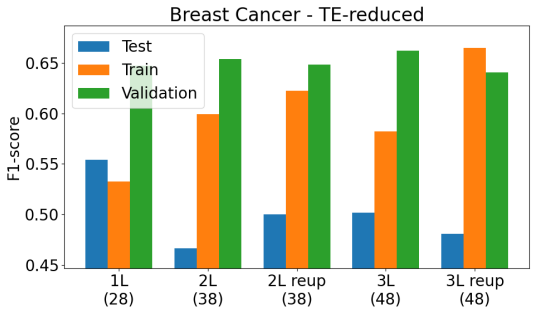
(c) TE - Accuracy



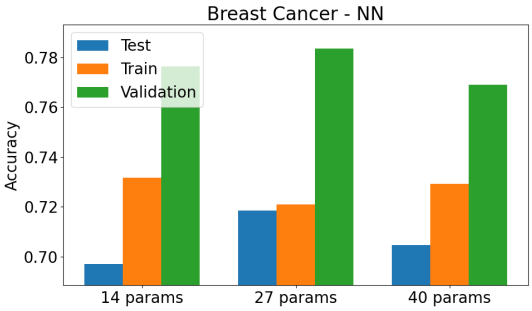
(d) TE - F1-score



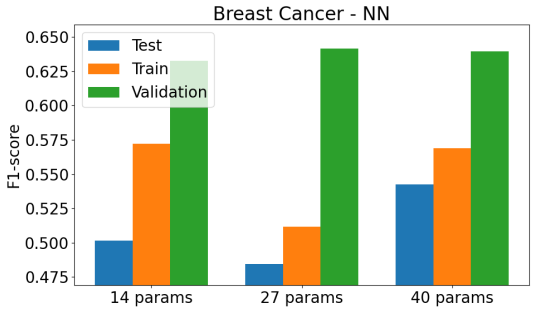
(e) TE-reduced - Accuracy



(f) TE-reduced - F1-score



(g) NN - Accuracy



(h) NN - F1-score

Figure 5.5: Performance of models on the breast cancer dataset **with** oversampling.

doi:10.6342/NTU202402911

Chapter 6 Discussion

In our experiments with the Frozen Lake problem and the MNIST hand-written digits dataset, the implementation of data reuploading appears to have been beneficial for most models. Notably, even models that did not benefit significantly from data reuploading did not experience substantial performance loss. Additionally, both TE and TE-reduced generally outperformed QRAC. In contrast, for the breast cancer dataset, neither increasing the number of layers nor implementing data reuploading consistently enhanced performance. Moreover, QRAC's performance was found to be comparable to the other encoding methods in this context.

For the first two experiments, methods that resulted in a trainable kernel generally exhibited superior performance compared to those without trainable kernels. Concurrently, increasing the number of variational layers did not enhance performance when the number was already high, corroborating the assertions in Section 2.6. These findings suggest that while increasing the complexity of $U(\theta)$ has an upper limit—either in terms of solution space or due to the inherent limitations of finite unitaries—enhancing the trainability or complexity of kernels may extend performance beyond these boundaries. However, in the breast cancer dataset experiment, neither increasing the number of variational layers nor enhancing the trainability of the encoding method improved performance, indicating a preference for simpler models and a lower requirement for expressibility in this context.

doi:10.6342/NTU202402911

Additionally, all models exhibited noticeable overfitting, even with validation data used for final model selection, likely due to the limited number of instances in this dataset. This suggests that more powerful regularization techniques may be required.

Across all three experiments, TE-reduced consistently showed slightly inferior performance compared to TE but still maintained better overall performance than QRAC. This is notable considering that TE-reduced involves far fewer parameters and matches QRAC in parameter scaling. These results imply that, except in cases where simpler models are favored, TE-reduced could be a more effective encoding method than QRAC due to its enhanced expressibility and equivalent parameter scaling.

doi:10.6342/NTU202402911

Chapter 7 Conclusion

In this thesis, we explore the properties of a quantum machine learning algorithm known as variational quantum circuits and its relationship with classical kernel methods. Based on this relationship, we demonstrate how models with trainable kernels can expand the solution space beyond the limitations of fixed kernels. We introduce several existing methods that facilitate trainable kernels and assess the performance of models with these methods through several experiments.

The results from these experiments confirm that the use of trainable kernels not only enhances the complexity and expressibility of models theoretically, but also empirically. Additionally, increasing the complexity of a trainable kernel can potentially boost the model's performance beyond the limits imposed by the variational component. Three methods—trainable embedding (TE), reduced trainable embedding (TE-reduced), and data reuploading—were evaluated, suggesting that further exploration into methods that introduce trainable kernels is warranted. However, it is important to note that these methods did not significantly enhance performance in experiments with the breast cancer dataset, which has a very limited number of instances, indicating that they may not be suitable for problems that benefit from simpler models.

Despite its marginally lower performance compared to TE, TE-reduced consistently

doi:10.6342/NTU202402911

outperformed QRAC, proving advantageous due to its fewer parameters and comparable scalability. This suggests its potential superiority in scenarios that require optimized parameter efficiency and enhanced model expressibility.

For future work, it is important to conduct larger-scale tests given the small scale of these experiments relative to typical machine learning problems. This necessity stems from current limitations in quantum computing resources, which may restrict the applicability of our findings to broader contexts beyond small-scale problems. Additionally, since the input data in all our experiments were discrete, future research should explore the applicability of these methods to problems with continuous features. Moreover, considering that all these experiments used a classical computer to simulate a quantum device without introducing noise, it is crucial to test whether the results hold on NISQ devices. Finally, as suggested in [31], and considering the relationship between VQCs and kernel methods, it would be interesting to compare quantum-kernel-based methods with VQCs. Such comparisons could help determine the performance differences between classical and quantum algorithms using the same kernels.

doi:10.6342/NTU202402911

References

[1] P. W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 1095-7111. DOI: 10.1137/s0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172.

[2] J. Preskill. “Quantum computing: pro and con”. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 454.1969 (Jan. 1998), pp. 469–486. ISSN: 1471-2946. DOI: 10.1098/rspa.1998.0171. URL: http://dx.doi.org/10.1098/rspa.1998.0171.

[3] V. E. Elfving et al. *How will quantum computers provide an industrially relevant computational advantage in quantum chemistry?* Sept. 2020. arXiv: 2009.12472 [quant-ph]. URL: https://arxiv.org/abs/2009.12472.

[4] A. J. Daley et al. “Practical quantum advantage in quantum simulation”. In: *Nature* 607 (2022), pp. 667–676. URL: https://api.semanticscholar.org/CorpusID:251132664.

[5] A. Freeman. “Materials by design and the exciting role of quantum computation/simulation”. In: *Journal of Computational and Applied Mathematics* 149.1 (2002), pp. 27–56. ISSN: 0377-0427. DOI: https://doi.org/10.1016/S0377-0427(02)00519-8. URL: https://www.sciencedirect.com/science/article/pii/S0377042702005198.

[6] A. W. Harrow, A. Hassidim, and S. Lloyd. “Quantum Algorithm for Linear Systems of Equations”. In: *Physical Review Letters* 103 (15 Oct. 2009), p. 150502. DOI: 10.1103/PhysRevLett.103.150502. URL: https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

doi:10.6342/NTU202402911

[7] L. Wang, S. K. Kowk, and W. H. Ip. “Design of an improved quantum-inspired evolutionary algorithm for a transportation problem in logistics systems”. In: *Journal of Intelligent Manufacturing* 23.6 (Dec. 2012), pp. 2227–2236. ISSN: 0956-5515. DOI: 10.1007/s10845-011-0568-7. URL: https://doi.org/10.1007/s10845-011-0568-7.

[8] D. Herman et al. *A Survey of Quantum Computing for Finance*. 2022. arXiv: 2201.02773 [quant-ph]. URL: https://arxiv.org/abs/2201.02773.

[9] D. Peral-García, J. Cruz-Benito, and F. J. García-Peñalvo. “Systematic literature review: Quantum machine learning and its applications”. In: *Computer Science Review* 51 (2024), p. 100619. ISSN: 1574-0137. DOI: https://doi.org/10.1016/j.cosrev.2024.100619. URL: https://www.sciencedirect.com/science/article/pii/S1574013724000030.

[10] J. Preskill. “Quantum Computing in the NISQ era and beyond”. In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: http://dx.doi.org/10.22331/q-2018-08-06-79.

[11] K. Mitarai et al. “Quantum circuit learning”. In: *Physical Review A* 98.3 (Sept. 2018). ISSN: 2469-9934. DOI: 10.1103/physreva.98.032309. URL: http://dx.doi.org/10.1103/PhysRevA.98.032309.

[12] A. Pérez-Salinas et al. “Data re-uploading for a universal quantum classifier”. In: *Quantum* 4 (Feb. 2020), p. 226. ISSN: 2521-327X. DOI: 10.22331/q-2020-02-06-226. URL: http://dx.doi.org/10.22331/q-2020-02-06-226.

[13] N. Thumwanit et al. *Trainable Discrete Feature Embeddings for Variational Quantum Classifier*. 2021. arXiv: 2106.09415 [quant-ph].

[14] M. Schuld. *Supervised quantum machine learning models are kernel methods*. 2021. arXiv: 2101.11020 [quant-ph].

[15] T. Hofmann, B. Schölkopf, and A. J. Smola. “Kernel methods in machine learning”. In: *The Annals of Statistics* 36.3 (2008), pp. 1171–1220. DOI: 10.1214/009053607000000677. URL: https://doi.org/10.1214/009053607000000677.

doi:10.6342/NTU202402911

[16] C. Cortes and V. Vapnik. “Support-Vector Networks”. In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 0885-6125. DOI: 10.1023/A:1022627411411. URL: https://doi.org/10.1023/A:1022627411411.

[17] G. Wahba and Y. Wang. “Representer Theorem”. In: *Wiley StatsRef: Statistics Reference Online*. John Wiley & Sons, Ltd, 2019, pp. 1–11. ISBN: 9781118445112. DOI: https://doi.org/10.1002/9781118445112.stat08200. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118445112.stat08200. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118445112.stat08200.

[18] G. S. Kimeldorf and G. Wahba. “A Correspondence Between Bayesian Estimation on Stochastic Processes and Smoothing by Splines”. In: *The Annals of Mathematical Statistics* 41.2 (1970), pp. 495–502. DOI: 10.1214/aoms/1177697089. URL: https://doi.org/10.1214/aoms/1177697089.

[19] G. Kimeldorf and G. Wahba. “Some results on Tchebycheffian spline functions”. In: *Journal of Mathematical Analysis and Applications* 33.1 (1971), pp. 82–95. ISSN: 0022-247X. DOI: https://doi.org/10.1016/0022-247X(71)90184-3. URL: https://www.sciencedirect.com/science/article/pii/0022247X71901843.

[20] B. Schölkopf, R. Herbrich, and A. J. Smola. “A Generalized Representer Theorem”. In: *Computational Learning Theory*. Ed. by D. Helmbold and B. Williamson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 416–426. ISBN: 978-3-540-44581-4.

[21] Y. Gujju, A. Matsuo, and R. Raymond. *Quantum Machine Learning on Near-Term Quantum Devices: Current State of Supervised and Unsupervised Techniques for Real-World Applications*. 2024. arXiv: 2307.00908 [quant-ph]. URL: https://arxiv.org/abs/2307.00908.

[22] S. Jerbi et al. “Quantum machine learning beyond kernel methods”. In: *Nature Communications* 14.1 (Jan. 2023). ISSN: 2041-1723. DOI: 10.1038/s41467-023-36159-y. URL: http://dx.doi.org/10.1038/s41467-023-36159-y.

doi:10.6342/NTU202402911

[23] S. Wiesner. “Conjugate coding”. In: *SIGACT News* 15.1 (Jan. 1983), pp. 78–88. ISSN: 0163-5700. DOI: 10.1145/1008908.1008920. URL: https://doi.org/10.1145/1008908.1008920.

[24] A. Ambainis et al. “Dense quantum coding and quantum finite automata”. In: *Journal of the ACM* 49.4 (July 2002), pp. 496–511. ISSN: 0004-5411. DOI: 10.1145/581771.581773. URL: https://doi.org/10.1145/581771.581773.

[25] A. Ambainis et al. *Dense Quantum Coding and a Lower Bound for 1-way Quantum Automata*. 1998. arXiv: quant-ph/9804043 [quant-ph].

[26] H. Yano et al. “Efficient Discrete Feature Encoding for Variational Quantum Classifier”. In: *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–14. ISSN: 2689-1808. DOI: 10.1109/tqe.2021.3103050. URL: http://dx.doi.org/10.1109/TQE.2021.3103050.

[27] D. Herman et al. “Expressivity of Variational Quantum Machine Learning on the Boolean Cube”. In: *IEEE Transactions on Quantum Engineering* 4 (2023), pp. 1–18. ISSN: 2689-1808. DOI: 10.1109/tqe.2023.3255206. URL: http://dx.doi.org/10.1109/TQE.2023.3255206.

[28] U. Ullah et al. “A Fully Connected Quantum Convolutional Neural Network for Classifying Ischemic Cardiopathy”. In: *IEEE Access* 10 (2022), pp. 134592–134605. DOI: 10.1109/ACCESS.2022.3232307.

[29] S. Y.-C. Chen et al. “Variational Quantum Circuits for Deep Reinforcement Learning”. In: *IEEE Access* 8 (2020), pp. 141007–141024. DOI: 10.1109/ACCESS.2020.3010470.

[30] V. Mnih et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 00280836. URL: http://dx.doi.org/10.1038/nature14236.

[31] V. Havlíček et al. “Supervised learning with quantum-enhanced feature spaces”. In: *Nature* 567.7747 (Mar. 2019), pp. 209–212. ISSN: 1476-4687. DOI: 10.1038/s41586-019-0980-2. URL: http://dx.doi.org/10.1038/s41586-019-0980-2.

doi:10.6342/NTU202402911