UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JONATHAN HENRIQUE ANDRADE DE CARVALHO

Parametrized Constant-Depth Quantum Neuron:
framework, conception, and applications

Recife

2022

JONATHAN HENRIQUE ANDRADE DE CARVALHO

Parametrized Constant-Depth Quantum Neuron:
framework, conception, and applications

Dissertation presented to the Programa de Pós-Graduação em Ciência da Computação of the Centro de Informática of the Universidade Federal de Pernambuco in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Concentration Area: Computational Intelligence

Advisor: Fernando Maciano de Paula Neto

Recife

2022

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

C331p Carvalho, Jonathan Henrique Andrade de
Parametrized constant-depth quantum neuron: framework, conception, and applications / Jonathan Henrique Andrade de Carvalho. – 2022.
64 f.: il., fig., tab.

Orientador: Fernando Maciano de Paula Neto.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências.

1. Inteligência computacional. 2. Quantum computing. 3. Quantum neuron. 4. Kernel machine. 5. Constant-depth quantum circuit. I. Paula Neto, Fernando Maciano de (orientador). II. Título

006.31 CDD (23. ed.) UFPE - CCEN 2023 – 48

Jonathan Henrique Andrade de Carvalho

“Parametrized Constant-Depth Quantum Neuron: framework, conception, and applications”

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação. Área de Concentração: Inteligência Computacional.

Aprovado em: 11 de agosto de 2022.

BANCA EXAMINADORA

_____

Prof. Dr. Paulo Salgado Gomes de Mattos Neto
Centro de Informática / UFPE

_____

Prof. Dr. Tiago Alessandro Espinola Ferreira
Departamento de Estatística e Informática / UFRPE

_____

Prof. Dr. Fernando Maciano de Paula Neto
Centro de Informática / UFPE
(Orientador)

I dedicate this dissertation to those who believe that the development of science, especially on emerging topics, can transform the individual and society in the long term.

ACKNOWLEDGEMENTS

I certainly acknowledge the success of this stage in my life to God first. With faith in the plans that God has made for me, I was able to stay focused, dedicated, and hopeful during all this time, even in difficult moments and decisions. He provided me with discernment to overcome the challenges and problems that have appeared. Thus, I glorify your name in this moment of triumph. God be praised.

I also thank the professors whom I have taken courses or interacted with throughout those academic and scientific processes. I really appreciate and respect their scientific knowledge in the corresponding expertise areas and their personal interpretations of the world and life. I am deeply thankful for each one of the ideas and thoughts that they have shared with me. A special thank is dedicated to my advisor, Fernando Maciano de Paula Neto, who guided me through the uncertainties with crucial contributions that led to successful paths. At the same time, he supported and encouraged my own decisions, which was fundamental to my continuous development as an independent researcher. To the Centro de Informática of the Universidade Federal de Pernambuco, I would like to express my acknowledgment for its environment of excellence and achievements. Such an atmosphere motivated me to preserve and even enhance its reputation. I would also like to express my acknowledgment to the Fundação de Amparo à Ciência e Tecnologia do Estado de Pernambuco for the financial support. I could not develop such research without a full-time dedication.

The development of this research also required a substantial part of the time that I would dedicate to my family and friends. I was not able to organize my time properly on several occasions. I truly recognize the importance of those moments and the consequences of abdicating them. Unfortunately, the established goals required intensive and almost tireless work on my part. I am deeply grateful for the comprehension and the support that I have received from my family and friends in all situations and under any circumstances. They provided me with lightness and serenity, which is essential to facing a hard process. I promise to dedicate myself to the fullest to attend to the expectations, making those people completely proud of me. I will stand firm in order to deserve all this in my life.

ABSTRACT

Quantum computing has been revolutionizing the development of algorithms, which includes remarkable advances in artificial neural networks. The exploration of inherently quantum phenomena holds the promise of transcending classical computing. However, only noisy intermediate-scale quantum devices are available currently. To demonstrate advantages in this quantum era, the development of quantum algorithms needs to satisfy several software requirements due to the insufficiency of quantum computing resources. In this research, we propose a kernel-based framework of quantum neurons that not only contemplates existing quantum neurons but also makes room to define countless others, including quantum neurons that comply with the present hardware restrictions. For example, we propose here a quantum neuron that is implemented by a circuit of constant depth with a linear number of elementary single-qubit gates. Existing quantum neurons are implemented by exponentially expensive circuits, even using complex multi-qubit gates. We improve the proposed quantum neuron through a parametrization that can change its activation function shape in order to fit underlying patterns that existing quantum neurons cannot fit. As an initial demonstration, we show the proposed quantum neuron producing optimal solutions for six classification problems that an existing quantum neuron can solve only two of them. After, we benchmark classical and quantum neurons in several classification problems. As a result, in the majority of the cases, the proposed quantum neuron is the best over all neurons, which solidly confirms its superiority. The parametrization offers flexibility to not only fit a wide range of problems but also to optimize the margin between classes, at least better than the classical neurons and existing quantum ones. In light of those advantages, this research paves the way to develop quantum neural networks that can demonstrate a practical quantum advantage in the current quantum era already.

Keywords: quantum computing; quantum neuron; kernel machine; constant-depth quantum circuit; parametrized activation function.

RESUMO

A computação quântica vem revolucionando o desenvolvimento de algoritmos, o que inclui notáveis avanços em redes neurais artificiais. A exploração de fenômenos inerentemente quânticos traz a promessa de transcender a computação clássica. No entanto, apenas dispositivos quânticos de escala intermediária e ruidosos estão disponíveis atualmente. Para demonstrar vantagens nesta era quântica, o desenvolvimento de algoritmos quânticos precisa satisfazer diversos requisitos de software devido à insuficiência de recursos computacionais quânticos. Nesta pesquisa, nós propomos uma estrutura de neurônios quânticos baseada em kernel que não apenas contempla neurônios quânticos existentes como também abre espaço para definir inúmeros outros, incluindo neurônios quânticos que atendam às atuais restrições de hardware. Por exemplo, nós propomos aqui um neurônio quântico que é implementado por um circuito de profundidade constante com um número linear de portas elementares de um único bit quântico. Neurônios quânticos existentes são implementados por circuitos exponencialmente custosos, mesmo usando portas complexas de múltiplos bits quânticos. Nós melhoramos o neurônio quântico proposto através de uma parametrização que consegue mudar a forma da sua função de ativação a fim de se ajustar a padrões subjacentes que neurônios quânticos existentes não conseguem se ajustar. Como uma demonstração inicial, nós mostramos o neurônio quântico proposto produzindo soluções ótimas para seis problemas de classificação que um neurônio quântico existente consegue resolver apenas dois deles. Na sequência, nós extensivamente comparamos neurônios quânticos e clássicos em diversos problemas de classificação. Como resultado, na maioria dos casos, o neurônio quântico proposto é o melhor entre todos os neurônios, o que confirma sua superioridade de forma sólida. A parametrização fornece flexibilidade para não apenas se ajustar a uma ampla variedade de problemas mas também para otimizar a margem entre classes, pelo menos melhor que os neurônios clássicos e os neurônios quânticos existentes. Devido a essas vantagens, esta pesquisa abre o caminho para desenvolver redes neurais quânticas que podem demonstrar uma vantagem quântica prática já na era quântica atual.

Palavras-chaves: computação quântica; neurônio quântico; máquina de kernel; circuito quântico de profundidade constante; função de ativação parametrizada.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

In this chapter, we first delimit the context in which this research is embedded. Then, an overview of the research and its discoveries is presented. We also present here the organization of the dissertation regarding the subsequent chapters. Basically, this research can be subdivided into two parts. The first one, which contemplates the proposed models and a first validation, is already available on arXiv (CARVALHO; PAULA NETO, 2022) and is under review in the IEEE Transactions on Neural Networks and Learning Systems, where indications of merit were received in the first revision. We also recognize a high potential for publication in the second part of this research, which extends the comparisons in order to solidly support the proposed models, but it is not available on arXiv as well as it is not submitted to a journal or a conference yet. Simultaneously with this research, we also developed works in collaboration with a research group that studies quantum walks as search algorithms for different graph structures and their applications (CARVALHO et al., 2020; CARVALHO et al., 2023; SOUZA; CARVALHO; FERREIRA, 2021; SOUZA; CARVALHO; FERREIRA, 2022). It is worth mentioning that the work developed by Carvalho et al. (2020) obtained an honorable mention at the 9th Brazilian Conference on Intelligent Systems (BRACIS 2020).

1.1 CONTEXTUALIZATION

Quantum computing is basically the act of performing computations under the quantum mechanics regime, which allows the exploration of inherently quantum phenomena, such as state superposition and particle entanglement (NIELSEN; CHUANG, 2010). This computing model is expected to achieve the so-called supremacy over classical computing, which will allow us to solve previously intractable problems in a reasonable time (PRESKILL, 2012). For the quantum community, a revolution in business and society will occur in the next years due to the achievement of that predicted quantum supremacy.

Regarding specifically the development of neural networks, more efficient models can be obtained by harnessing that quantum advantage (NGUYEN et al., 2020). Quantum neurons can implement arbitrary non-linear functions while taking advantage of super-

position and entanglement (PAULA NETO et al., 2020). The capacity of implementing arbitrary activation functions qualifies the quantum neural networks as universal approximators (MARONESE; DESTRI; PRATI, 2022). When a quantum walk is used to search the optimal synaptic weights, the optimization of classical feedforward neural networks no longer suffers from local minima and requires a number of iterations that is well-defined a priori (SOUZA; CARVALHO; FERREIRA, 2022). Implementing the operations of recurrent cells in quantum circuits will lead to recurrent neural networks that provide exponential speed-ups (CESCHINI; ROSATO; PANELLA, 2022).

However, those research efforts in quantum computing for neural networks assume the existence of fault-tolerant large-scale quantum computers. Large-scale quantum computers have enough redundant qubits to encode the information and employ error correction codes (DEVITT; MUNRO; NEMOTO, 2013). Currently, only noisy intermediate-scale quantum devices are available, so there are not enough resources to protect the system by quantum error correction (PRESKILL, 2018). The implication in the development of quantum algorithms is that a set of software requirements arises in order to successfully execute the corresponding quantum circuits on the current hardware. Aspects like state preparation, oracle expansion, connectivity, circuit rewriting, decoherence, gate infidelity, and measurement errors can compromise the computation (LEYMANN; BARZEN, 2020). Thus, the depth, width, and operations number of quantum circuits become critical factors when developing quantum algorithms. Nevertheless, the current quantum era creates an opportunity to demonstrate advantages over classical computing at this time.

Research efforts have been focusing on that new generation of quantum algorithms since quantum devices became available. Despite significant efforts at the quantum compiler level (ZULEHNER; PALER; WILLE, 2019; LAROSE, 2019), recent quantum algorithms are developed considering those present restrictions (ACASIETE et al., 2020; ACAMPORA; VITIELLO, 2021), including the development of quantum neural networks (TACCHINO et al., 2019; MANGINI et al., 2020; TACCHINO et al., 2020; GRANT et al., 2018; CONG; CHOI; LUKIN, 2019; DALLAIRE-DEMERS; KILLORAN, 2018; ZOUFAL; LUCCHI; WOERNER, 2019; KONAR et al., 2022). Particularly, this research is not interested in the so-called variational quantum models, which are considered the quantum analogs of neural networks by simply implementing a quantum circuit with an arbitrary structure that is optimized for a given learning task. Instead, this research is focused on quantum models that are inspired by the classical neuron model (HAYKIN, 2009), which better represent the quantum

counterparts of classical neural networks for sharing even more similarities.

Tacchino et al. (2019) took the first step in this direction by proposing a scheme that computes the inner product between binary-valued vectors. After, Mangini et al. (2020) proposed a particular way to work with continuous-valued vectors by reformulating the previous scheme. Both schemes have an exponential advantage in information storage since the circuit width grows in a logarithmic manner, although the circuit depth and the number of operations grow exponentially. Thus, there is room for developing reformulations that work with continuous-valued vectors in other codifications, which is directly associated with the expressive power of quantum models (SCHULD; SWEKE; MEYER, 2021). At the circuit level, a research gap exists in terms of efficiency. More efficient implementations are required to further comply with the software requirements of the current quantum hardware.

1.2 RESEARCH OVERVIEW

In this research, we first propose a framework of quantum neurons where those previous schemes (TACCHINO et al., 2019; MANGINI et al., 2020) are particular cases. Based on that framework, each quantum neuron implements a kernel machine with a non-deterministic activation function. That activation function depends only on the kernel trick that each quantum neuron applies. Thus, the quantum neurons differ from each other by their feature space mappings. It makes room for other instantiations of this generalized kernel-based framework of quantum neurons, including for actual quantum devices.

Generally, quantum kernel methods are used to estimate inner products for classical models in a hybrid setup. The advantage emerges from the fact that the inner products are estimated from classically intractable feature mappings. Those inner products can be estimated by the standard swap-test (BUHRMAN et al., 2001) or, more recently, by quantum kernel estimators (HAVLÍČEK et al., 2019; SCHULD; KILLORAN, 2019). Speedups in classical models can be also obtained by classical sampling techniques inspired by quantum models that generate the kernel matrix of inner products (DING; BAO; HUANG, 2022). However, the kernel-based framework of quantum neurons proposed here computes the inner product between an input vector and a weight vector, and then explicitly extracts such information to an ancillary qubit. Measuring the ancilla gives a standalone fully-quantum classifier based on kernel machines. If the ancilla is not measured, the inner

product can be propagated forward to other quantum neurons, which gives a quantum neural network (TACCHINO et al., 2020). Finally, this kernel-based framework of quantum neurons is expected to achieve practical quantum advantage, as is expected with quantum kernel estimators (LIU; ARUNACHALAM; TEMME, 2021).

Based on that framework, we propose here a quantum neuron of constant depth, i.e., the circuit implementation of the proposed neuron has a depth that is independent of the input size. Constant-depth quantum circuits can demonstrate quantum advantage (BRAVYI; GOSSET; KÖNIG, 2018; BRAVYI et al., 2020) and benefit from error mitigation techniques (LI; BENJAMIN, 2017; TEMME; BRAVYI; GAMBETTA, 2017; KANDALA et al., 2019). The proposed quantum neuron implements local feature mappings (STOUDENMIRE; SCHWAB, 2016) by taking advantage of qubit encoding (GRANT et al., 2018; KONAR et al., 2022). We demonstrate that this encoding strategy actually implements a tensor-product feature mapping to an exponentially larger space, which improves the separating capacity (COVER, 1965). We further improve that neuron capacity by including two parameters in its activation function. That parametrization can change the activation function shape of the proposed quantum neuron in order to fit different underlying patterns with no additional cost in the circuit implementation. Therefore, we propose a flexible quantum neuron of constant depth implemented with a linear number of elementary single-qubit gates. Existing quantum neurons (MANGINI et al., 2020; MARONESE; PRATI, 2021) are inflexible and exponentially expensive, even with complex multi-qubit gates in their circuit implementations.

Then, we proceed to a visual study that relates some interactions between the input and weight vectors in the original space with the respective neuron outputs in the feature space. Those activation function shapes reveal the problem structures that each quantum neuron can solve. By comparing the best solutions of each quantum neuron for six classification problems, we show that parametrization can really change the activation function shape in order to fit underlying patterns that an existing quantum neuron cannot fit. The proposed quantum neuron produces optimal solutions for all six cases, which demonstrates its flexibility and improved capabilities. That initial demonstration finishes with a proof-of-concept experiment on a quantum simulator to show the feasibility of those quantum neuron solutions.

The present research actually finishes by contrasting the quantum neurons against classical ones in several classification problems. Assessing whether or not quantum models

can outperform classical counterparts is critical for the consolidation of quantum computing. Demonstrations of superiority over classical computing are required to support the construction of fault-tolerant large-scale quantum computers, which involves numerous engineering challenges. One of the major roles of the quantum algorithms community in this quantum era is to demonstrate that the frontiers of classical computing can be exceeded. Here, we show the quantum neurons performing better than the classical ones in all cases of the benchmark, which constitutes solid evidence that quantum computing can go beyond. There is enough evidence here that the poor performances of classical neurons in non-linearly separable problems do not hold for quantum neurons, which in turn can produce optimal solutions for the same problems. Specifically, from forty classification problems, the classical neurons can perfectly solve only two of them against seventeen perfect solutions of the quantum neurons. In most of those cases, the quantum neurons can still provide robust separations of classes. In particular, the benchmark also generates enough evidence about the parametrization gains since the best overall neuron in the majority of the cases addressed here is the quantum neuron that we proposed. Therefore, the parametrization definitely gives flexibility to not only fit a wide range of problems but also to optimize the margin between classes.

1.3 ORGANIZATION OF THE DISSERTATION

This dissertation is organized as follows. Firstly, however, it is worth mentioning that the reader is expected to be familiar with the basic concepts of quantum computing. If that assumption is not the case, the reader should refer to the books of Nielsen and Chuang (2010) and Yanofsky and Mannucci (2008), where not only basic concepts but also advanced topics on quantum computing are covered. With that assumption stated, the dissertation proceeds by formalizing the proposed framework of quantum neurons based on kernel machines in Chapter 2, which enables a facilitated description of the existing quantum neurons. Under that generalized framework, Chapter 3 proposes the parametrized quantum neuron of constant circuit depth. The activation function shapes of the quantum neurons are presented in Chapter 4. As an initial application, Chapter 5 compares the quantum neurons in solving a few classification problems. On the other hand, Chapter 6 presents an extensive benchmark that includes classical and quantum neurons. Finally, Chapter 7 discusses the conclusions of this research.

2 QUANTUM NEURON FRAMEWORK

The classical neuron model basically consists of two steps. First, the inner product between the input vector $\vec{i}$ and the weight vector $\vec{w}$ is computed, and then that inner product $\sum_j w_j i_j$ is passed to an activation function $\varphi(\cdot)$ in order to define the neuron output y (HAYKIN, 2009). Based on that model, one can construct different classical neurons by only changing the activation function that processes the obtained inner product. Examples of common activation functions are the Heaviside function and the logistic sigmoid function, which are presented in Equation 2.1 and Equation 2.2 respectively.

$$\varphi(\vec{w} \cdot \vec{i}) = \begin{cases} 1, & \text{if } \vec{w} \cdot \vec{i} \geq 0 \\ 0, & \text{if } \vec{w} \cdot \vec{i} < 0 \end{cases} \tag{2.1}$$

$$\varphi(\vec{w} \cdot \vec{i}) = \frac{1}{1 + e^{-(\vec{w} \cdot \vec{i})}} \tag{2.2}$$

Inspired by that classical neuron model, Tacchino et al. (2019) proposed a scheme for actual quantum devices that encodes input and weight vectors, computes the inner product between them, and finally extracts the activation function output. Here, we generalize that scheme to a framework of quantum neurons based on kernel machines. Given two vectors $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ in an input space, a kernel machine $k(\cdot, \cdot)$ maps those vectors to other two vectors $\vec{w}$ and $\vec{i}$ in a feature space by a nonlinear transformation $\Phi(\cdot)$, and then computes the inner product between the transformed vectors, i.e., $k(\boldsymbol{\phi}, \boldsymbol{\theta}) = \Phi(\boldsymbol{\phi})^T \Phi(\boldsymbol{\theta}) = \vec{w} \cdot \vec{i}$ (BISHOP, 2006). The quantum neurons constructed by the framework are deeply related to that kernel trick.

The quantum neurons implement kernel methods because the given m-dimensional classical vectors $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are first mapped to N-dimensional quantum vectors $\vec{i} = \Phi(\boldsymbol{\theta})$ and $\vec{w} = \Phi(\boldsymbol{\phi})$. Those quantum vectors can be directly encoded in legitimate quantum states in the following way:

$$|\psi_i\rangle = \sum_{j=0}^{N-1} i_j \, |j\rangle \qquad \text{and} \qquad |\psi_w\rangle = \sum_{j=0}^{N-1} w_j \, |j\rangle \, .$$

Then, the inner product $\vec{w}^* \cdot \vec{i} = \langle \psi_w | \psi_i \rangle$ is computed by the quantum neurons. The final neuron output is given by the non-deterministic activation function presented in Equation 2.3.

$$\varphi(\vec{w}^* \cdot \vec{i}) = \begin{cases} 1, & \text{with probability } |\vec{w}^* \cdot \vec{i}|^2 \\ 0, & \text{with probability } 1 - |\vec{w}^* \cdot \vec{i}|^2 \end{cases} \tag{2.3}$$

In this way, the quantum activation function depends on the inner product $\vec{w}^* \cdot \vec{i}$ that in turn depends on the feature space mapping $\Phi(\cdot)$. Based on that framework, one can realize different quantum neurons, including on actual quantum computers, by only changing the mapping $\Phi(\cdot)$ that each quantum neuron implements.

Figure 1 shows the circuit implementation of the framework of quantum neurons based on kernel machines. Specifically, the quantum neurons differ from each other by the quantum operators E and D, which depend on the classical vectors $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ respectively. The operator E maps $\boldsymbol{\theta}$ to $\vec{i}$ and then encodes it in $|\psi_i\rangle$ from $|+\rangle^{\otimes n}$, where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and n is the number of encoding qubits. Thus,

$$E(\boldsymbol{\theta}) |+\rangle^{\otimes n} = |\psi_i\rangle .$$

The operator D maps $\boldsymbol{\phi}$ to $\vec{w}$ and then decodes it from $|\psi_w\rangle$ to $|+\rangle^{\otimes n}$. Thus,

$$D(\boldsymbol{\phi}) |\psi_w\rangle = |+\rangle^{\otimes n} .$$

At a higher level of abstraction, the operator U_i encodes $|\psi_i\rangle$ from a blank register $|0\rangle^{\otimes n}$:

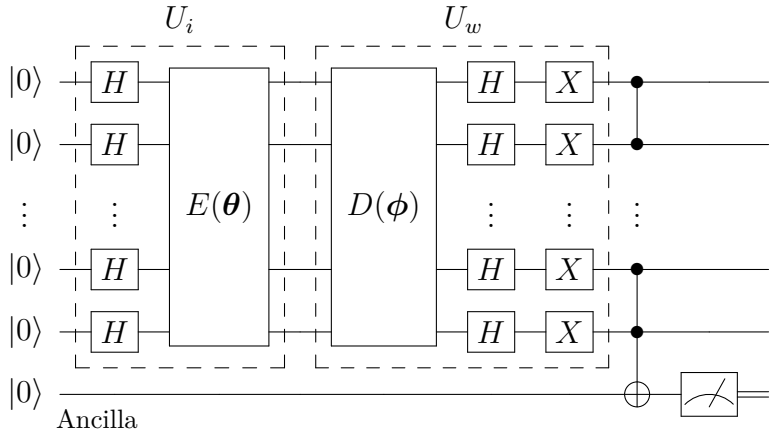
$$U_i |0\rangle^{\otimes n} = E(\boldsymbol{\theta}) H^{\otimes n} |0\rangle^{\otimes n} = E(\boldsymbol{\theta}) |+\rangle^{\otimes n} = |\psi_i\rangle .$$

On the other hand, U_w decodes $|\psi_w\rangle$ to $|1\rangle^{\otimes n}$:

$$\begin{aligned} U_w |\psi_w\rangle = X^{\otimes n} H^{\otimes n} D(\boldsymbol{\phi}) |\psi_w\rangle &= X^{\otimes n} H^{\otimes n} |+\rangle^{\otimes n} \\ &= X^{\otimes n} |0\rangle^{\otimes n} \\ &= |1\rangle^{\otimes n} . \end{aligned}$$

The operator $D(\boldsymbol{\phi})$ and then U_w are defined according to the action on $|\psi_w\rangle$, but those operators are applied in $|\psi_i\rangle$ in the circuit flow. Applying U_w to $|\psi_i\rangle$ actually computes the inner product between $\vec{i}$ and $\vec{w}$ by taking advantage of an interesting property of unitary operators. Unitary operators preserve the space geometry, which includes inner

Figure 1 – Framework Circuit to Build Quantum Neurons Based on Kernel Machines



Source: The author (2022).

First, U_i encodes $\vec{i} = \Phi(\boldsymbol{\theta})$. Then, U_w computes the inner product between $\vec{i}$ and $\vec{w} = \Phi(\boldsymbol{\phi})$. Such information is extracted to an ancillary qubit that gives a non-deterministic activation function when finally measured.

products (YANOFSKY; MANNUCCI, 2008). If A is a unitary operator, the inner product between a quantum state $|v\rangle$ and another quantum state $|v'\rangle$ is equal to the inner product between $A|v\rangle$ and $A|v'\rangle$, i.e., $\langle v'|v\rangle = \langle v'A^\dagger|Av\rangle$. Let $A = U_w$, $|v\rangle = |\psi_i\rangle$, and $|v'\rangle = |\psi_w\rangle$. We can rewrite the expression as:

$$\langle\psi_w|\psi_i\rangle = \langle\psi_w U_w^\dagger|U_w\psi_i\rangle = \langle 1^{\otimes n}|U_w\psi_i\rangle.$$

Thus, the projection of $U_w|\psi_i\rangle$ on the basis state $|1\rangle^{\otimes n}$ stores the inner product $\langle\psi_w|\psi_i\rangle$. In other words, that inner product is stored in the component of $|1\rangle^{\otimes n}$ into the superposition of $U_w|\psi_i\rangle = |\psi_{i,w}\rangle$. In this way,

$$|\psi_{i,w}\rangle = c_0|0\rangle + c_1|1\rangle + \cdots + c_{N-2}|N-2\rangle + \langle\psi_w|\psi_i\rangle|N-1\rangle.$$

To extract such information, a multi-controlled NOT gate is applied targeting an ancillary qubit. Note that only the basis state $|N-1\rangle$ in the superposition flips the ancilla to the state $|1\rangle$. Consequently, the final quantum state $|\psi_f\rangle$ is:

$$|\psi_f\rangle = c_0|0\rangle|0\rangle + c_1|1\rangle|0\rangle + \cdots + c_{N-2}|N-2\rangle|0\rangle + \langle\psi_w|\psi_i\rangle|N-1\rangle|1\rangle.$$

Since $\langle\psi_w|\psi_i\rangle = \vec{w}^* \cdot \vec{i}$, measuring that ancillary qubit finally gives the activation function already presented in Equation 2.3. Therefore, that quantum neuron framework

applies the kernel trick and activates following a non-deterministic function of the computed inner product, which was to be shown. To instantiate a quantum neuron, one only needs to realize a feature mapping $\Phi(\cdot)$ by means of the operators E and D.

2.1 BINARY-VALUED QUANTUM NEURON

That generalized framework can be first instantiated in the binary-valued quantum neuron (BVQN) (TACCHINO et al., 2019). Given m-dimensional classical vectors $\boldsymbol{\theta} = (\theta_0, \theta_1, \cdots, \theta_{m-2}, \theta_{m-1})$ and $\boldsymbol{\phi} = (\phi_0, \phi_1, \cdots, \phi_{m-2}, \phi_{m-1})$, where $\theta_j, \phi_j \in \{-1, 1\}$, the BVQN feature mapping simply normalizes those classical vectors, which leads to the following N-dimensional quantum vectors, where $N = m$:

$$\vec{i} = \Phi(\boldsymbol{\theta}) = \frac{1}{\sqrt{N}}\Big(\theta_0, \theta_1, \cdots, \theta_{m-2}, \theta_{m-1}\Big) \text{ and}$$
$$\vec{w} = \Phi(\boldsymbol{\phi}) = \frac{1}{\sqrt{N}}\Big(\phi_0, \phi_1, \cdots, \phi_{m-2}, \phi_{m-1}\Big).$$

Those quantum vectors live in the Hilbert space spanned by n qubits, where $N = 2^n$. As N equals m, the BVQN has an exponential advantage in storing the information.

Note that $|+\rangle^{\otimes n} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle$, so $E(\boldsymbol{\theta})$ only needs to flip the sign of the basis states $|j\rangle$ where $\theta_j = -1$ to encode $\vec{i}$ in $|\psi_i\rangle$. Similarly, $D(\boldsymbol{\phi})$ only needs to cancel the sign of the basis states $|j\rangle$ where $\phi_j = -1$ to decode $\vec{w}$ from $|\psi_w\rangle$. Those operations are accomplished by applying sign-flip blocks one by one or, more efficiently, by applying the hypergraph states generation subroutine (HSGS) (TACCHINO et al., 2019; ROSSI et al., 2013). However, those two strategies are exponentially expensive in terms of circuit depth. Finally, the BVQN fires with the probability presented in Equation 2.4.

$$|\vec{w}^* \cdot \vec{i}|^2 = \left| \frac{1}{N} \sum_{j=0}^{m-1} \phi_j \theta_j \right|^2 \tag{2.4}$$

2.2 CONTINUOUS-VALUED QUANTUM NEURON

A first alternative to accepting real vectors instead of only binary ones is to instantiate the framework in the continuous-valued quantum neuron (CVQN) (MANGINI et al., 2020), which implements a phase-based feature mapping. That mapping encodes each classical

component in the phase of a complex number written in the exponential form with modulus $1/\sqrt{N}$. Thus, the classical vectors are mapped to quantum vectors of equal size in the following manner:

$$\vec{i} = \Phi(\boldsymbol{\theta}) = \frac{1}{\sqrt{N}}\left(e^{i\theta_0}, e^{i\theta_1}, \cdots, e^{i\theta_{m-2}}, e^{i\theta_{m-1}}\right) \text{ and}$$
$$\vec{w} = \Phi(\boldsymbol{\phi}) = \frac{1}{\sqrt{N}}\left(e^{i\phi_0}, e^{i\phi_1}, \cdots, e^{i\phi_{m-2}}, e^{i\phi_{m-1}}\right).$$

Periodicity in that phase-based scheme and the square modulus in the activation function make the CVQN equally recognize values that are actually different. To really distinguish such values, the classical vectors are scaled to the interval $[0, \pi/2]$. Thus, the feature mapping is applied to classical vectors $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, where $\theta_j, \phi_j \in [0, \pi/2]$. The N-dimensional quantum vectors, where $N = m$, are described by n qubits, where $N = 2^n$, which represents an exponential advantage again in information storage.

To implement $E(\boldsymbol{\theta})$ and $D(\boldsymbol{\phi})$, the blocks that flip the sign in the BVQN are used here to encode each $e^{i\theta_j}$ and to decode each $e^{i\phi_j}$, respectively, by applying multi-controlled phase gates instead of multi-controlled Z gates. The phase gate is represented in the matrix form by:

$$P(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}.$$

As sign-flip blocks, phase-shift blocks are also exponentially expensive in terms of circuit depth. Finally, the CVQN fires with the probability presented in Equation 2.5.

$$|\vec{w}^* \cdot \vec{i}|^2 = \left| \frac{1}{N} \sum_{j=0}^{m-1} e^{i(\theta_j - \phi_j)} \right|^2 \tag{2.5}$$

3 PROPOSED QUANTUM NEURON

Amplitude encoding leverages the qubit capacity of storing an exponential amount of information simultaneously, but the circuit depth to encode the data in such amplitudes grows exponentially. Here, we propose to use a qubit encoding scheme under that framework of quantum neurons, differently from the BVQN and the CVQN that use amplitude encoding. In this way, an m-dimensional classical vector is encoded in m qubits, which represents less efficient storage, but the state preparation is definitely efficient in terms of circuit depth as each qubit only requires a single rotation to encode the data (GRANT et al., 2018; KONAR et al., 2022). That efficiency of qubit encoding allows us to propose the constant-depth quantum neuron (CDQN).

Instead of a qubit encoding from rotations about the y-axis (GRANT et al., 2018; KONAR et al., 2022), we apply the phase shift gate $P(\lambda)$ already used in amplitude encoding previously (MANGINI et al., 2020). Thus, the classical input vector $\boldsymbol{\theta}$ is directly encoded by the operator E as follows:

$$E(\boldsymbol{\theta}) = \otimes_{j=0}^{m-1} P(\theta_j).$$

Similarly, the classical weight vector $\boldsymbol{\phi}$ is directly decoded by the operator D in the following way:

$$D(\boldsymbol{\phi}) = \otimes_{j=0}^{m-1} P(-\phi_j).$$

As a result, U_i comes down to:

$$U_i = \otimes_{j=0}^{m-1} P(\theta_j) H^{\otimes m}.$$

In turn, U_w comes down to:

$$U_w = X^{\otimes m} H^{\otimes m} \otimes_{j=0}^{m-1} P(-\phi_j).$$

Therefore, with a circuit depth independent of the input size, the proposed quantum neuron computes the inner product between two vectors that are transformed by a tensor product of local feature mappings (STOUDENMIRE; SCHWAB, 2016). Those classical vectors are also pre-processed to the interval $[0, \pi/2]$, as for the CVQN.

Applying that $E(\boldsymbol{\theta})$ in a basis state $|s\rangle$ produces the following state:

$$E(\boldsymbol{\theta})\,|s\rangle = \prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,s,m}}\,|s\rangle\,,$$

where $b_{j,s,m}$ is the j-th bit in the binary representation of the integer value s with m bits. In this way, $b_{j,s,m}$ selects the phase shifts to be applied. For example, with $m = 4$, applying $E(\boldsymbol{\theta})$ in $|5\rangle$ produces $e^{i\theta_1}e^{i\theta_3}\,|5\rangle$, using the big-endian convention. Applying $E(\boldsymbol{\theta})$ in $|+\rangle^{\otimes m}$ stores a conditioned product of $e^{i\theta_j}$ on the amplitude of each possible state $|s\rangle$, which represents the following feature mapping, where $N = 2^m$:

$$\vec{i} = \Phi(\boldsymbol{\theta}) = \frac{1}{\sqrt{N}}\Big(\prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,0,m}}, \prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,1,m}}, \cdots, \prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,N-2,m}}, \prod_{j=0}^{m-1} (e^{i\theta_j})^{b_{j,N-1,m}} \Big)$$

and

$$\vec{w} = \Phi(\boldsymbol{\phi}) = \frac{1}{\sqrt{N}}\Big(\prod_{j=0}^{m-1} (e^{i\phi_j})^{b_{j,0,m}}, \prod_{j=0}^{m-1} (e^{i\phi_j})^{b_{j,1,m}}, \cdots, \prod_{j=0}^{m-1} (e^{i\phi_j})^{b_{j,N-2,m}}, \prod_{j=0}^{m-1} (e^{i\phi_j})^{b_{j,N-1,m}} \Big).$$

Different from the previous quantum neurons that map m-dimensional classical vectors to N-dimensional quantum vectors with $N = m$, the CDQN applies a feature mapping where $N > m$. Specifically, the CDQN maps the vectors to an exponentially larger space. Thus, the CDQN is a kernel method that takes advantage of the theorem proposed by Cover (1965), which states that mapping to feature spaces of high dimensionality improves the separating capacity. As an implication of the theorem, non-linearly separable problems are more likely to become linearly separable in a high-dimensional space. Finally, the CDQN fires with the probability presented in Equation 3.1.

$$|\vec{w}^* \cdot \vec{i}|^2 = \left| \frac{1}{N} \sum_{s=0}^{N-1} \prod_{j=0}^{m-1} (e^{i(\theta_j - \phi_j)})^{b_{j,s,m}} \right|^2 \tag{3.1}$$

We can simplify the relation presented in Equation 3.1 by observing that it is actually a combination of m local inner products. Each local inner product is computed with respect to a local feature mapping that maps a θ_j to $\vec{i_j} = \frac{1}{\sqrt{2}}(1, e^{i\theta_j})$ and a ϕ_j to $\vec{w_j} = \frac{1}{\sqrt{2}}(1, e^{i\phi_j})$. As a result, the relation presented in Equation 3.1 is equivalent to a product of m local inner products in the form $\vec{w_j}^* \cdot \vec{i_j} = \frac{1}{2}(1 + e^{i(\theta_j - \phi_j)})$. Therefore, the CDQN fires with a probability that can be simplified as presented in Equation 3.2.

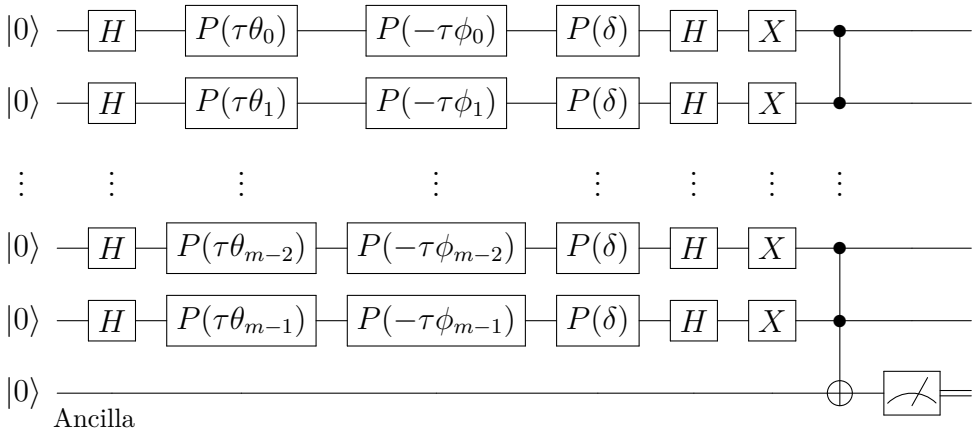
$$|\vec{w}^* \cdot \vec{i}|^2 = \left| \frac{1}{N} \prod_{j=0}^{m-1} (1 + e^{i(\theta_j - \phi_j)}) \right|^2 \tag{3.2}$$

Building on the CDQN, we propose to parametrize its activation function by a multiplicative factor τ and an additive constant δ. Those parameters act on the component-wise differences between the classical input and weight vectors. In this way, we propose the parametrized constant-depth quantum neuron (PCDQN) that fires with the probability presented in Equation 3.3. By adjusting τ and δ, the PCDQN can change its activation function shape. Thus, that parametrization generates flexibility. Such flexibility is supposed to allow the PCDQN to fit different underlying patterns.

$$|\vec{w}^* \cdot \vec{i}|^2 = \left| \frac{1}{N} \prod_{j=0}^{m-1} (1 + e^{i[\tau(\theta_j - \phi_j) + \delta]}) \right|^2 \tag{3.3}$$

At the circuit level, that parametrization is achieved by passing $\tau\boldsymbol{\theta}$ as the argument of $E(\cdot)$ and $\tau\boldsymbol{\phi}$ as the argument of $D(\cdot)$ followed by applying $P(\delta)$ to each qubit. Figure 2 shows the circuit implementation of the PCDQN. As can be seen, the PCDQN is implemented by a circuit of constant depth with a linear number of elementary single-qubit gates. The PCDQN is not only efficient but also flexible due to τ and δ.

Figure 2 – Circuit Implementation of the PCDQN



Source: The author (2022).

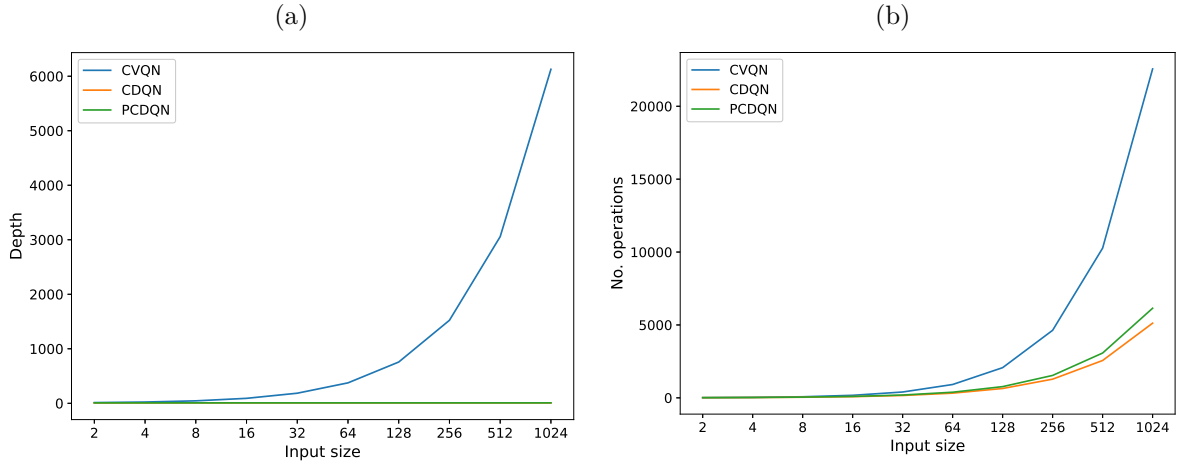
Qubit encoding provides a constant-depth circuit with a linear number of elementary single-qubit gates. Additionally, the parameters τ and δ can change the PCDQN activation function shape.

The PCDQN could be even more efficient by applying only one phase gate $P(\lambda)$ to each qubit, where $\lambda = \tau(\theta_j - \phi_j) + \delta$. Therefore, τ and δ would be included in the circuit with no additional cost. However, that circuit presented in Figure 2 increases the depth by 1 and the number of operations by m to implement the parametrization. Actually, the

parameter τ is included in the phase gates that encode the inputs and decode the weights, but the parameter δ is included in phase gates apart.

To demonstrate the efficiency of the CDQN and the PCDQN over the CVQN, Figure 3 compares the circuit growth of each quantum neuron as the input size increases. Figure 3a compares in terms of circuit depth and Figure 3b compares in terms of number of operations by using, respectively, the *depth* and *size* functions provided by Qiskit (ANIS et al., 2021). The blue curve represents the CVQN, the orange curve represents the CDQN, and the green curve represents the PCDQN.

Figure 3 – Circuit Growth of Each Quantum Neuron as the Input Size Increases



Source: The author (2022).

The blue, orange, and green curves represent the CVQN, the CDQN, and the PCDQN growths, respectively. (a) in terms of circuit depth. (b) in terms of the number of operations.

As can be seen, the CVQN grows linearly with the input size m in terms of depth and number of operations. Since $m = 2^n$ for the CVQN, that quantum neuron grows exponentially with the number of qubits n. On the other hand, the CDQN and the PCDQN have a constant depth and a lower number of operations. Additionally, the CDQN and the PCDQN are efficiently implemented with elementary single-qubit gates only, while the CVQN is more expensive, even using complex phase-shift blocks.

4 ACTIVATION FUNCTION SHAPES

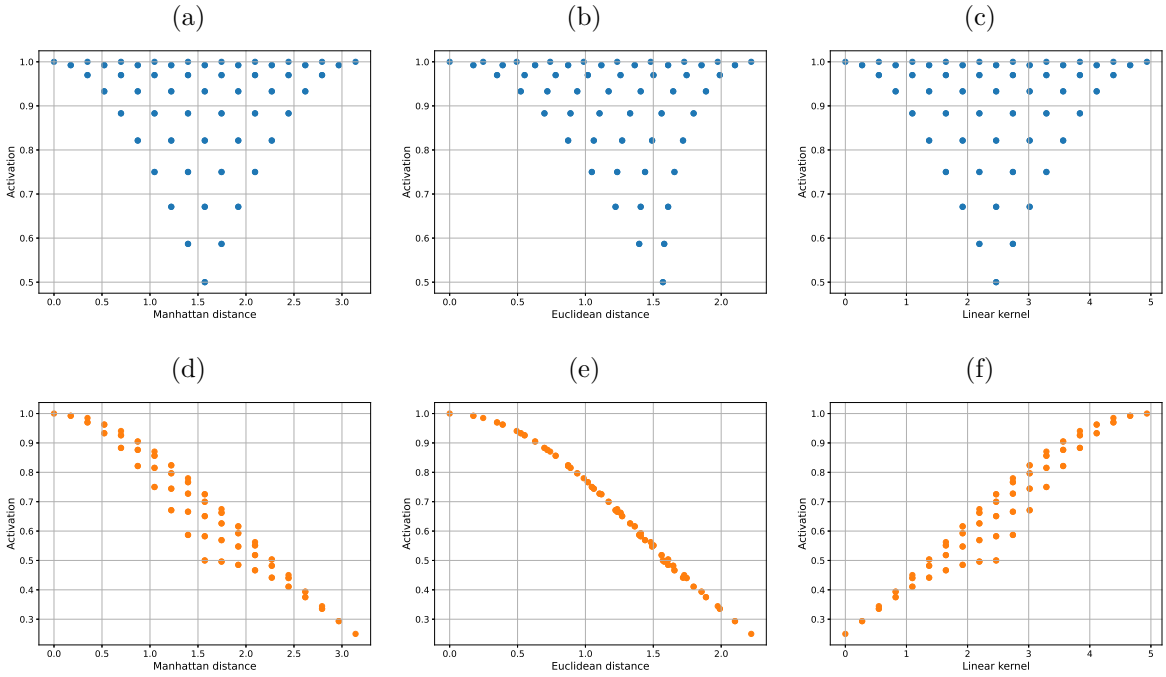
Classical neuron outputs depend on some interaction between the input and weight vectors. For example, the larger the inner product between the vectors, the larger the output of the sigmoid activation function. As another example, the larger the Euclidean distance between the vectors, the smaller the output of the radial basis activation function. On the other hand, the quantum neuron outputs depend on the inner product between the input and weight vectors in the feature space. In this chapter, we study the quantum neuron outputs as a function of some relations between the original vectors, which can be seen as an indirect way to visualize quantum activation shapes.

Then, we defined a set of two-dimensional input vectors by splitting the interval $[0, \pi/2]$ into ten equidistant values and taking all possible combinations, which gives an input set of one hundred vectors. The weight vector is fixed here as $(\pi/2, \pi/2)$. Thus, we compute the classical relations and the corresponding quantum neuron outputs for each input vector with respect to that weight vector of reference. Input vectors that give the same measure to that reference vector do not necessarily give the same neuron outputs. Specifically, we considered the following metrics provided by scikit-learn (PEDREGOSA et al., 2011): Manhattan distance, Euclidean distance, linear kernel, polynomial kernel, RBF kernel, and sigmoid kernel.

Figure 4 contrasts the CVQN activation with the CDQN activation as functions of the Manhattan distance, the Euclidean distance, and the linear kernel. The CVQN activation presents the same qualitative behavior in all cases but with a shift to the right for the Euclidean distance, as can be seen in Figure 4a, Figure 4b, and Figure 4c. In all cases, high values of activation are generated for the input vectors with extreme measures. Moving away from the extremes, the measures generate smaller and smaller values of activation in addition to the previous high values. Regarding the CDQN, its activation monotonically decreases as the Manhattan and Euclidean distances increase, as shown in Figure 4d and Figure 4e, and monotonically increases as the linear kernel also increases, as shown in Figure 4f.

In a similar way, Figure 5 contrasts the CVQN activation with the CDQN activation as functions of the polynomial kernel, the RBF kernel, and the sigmoid kernel. The CVQN activation presents relatively the same qualitative behavior already presented in Figure 4

Figure 4 – Quantum Neuron Activation as a Function of the Manhattan Distance, the Euclidean Distance, and the Linear Kernel Between the Original Input and Weight Vectors



Source: The author (2022).

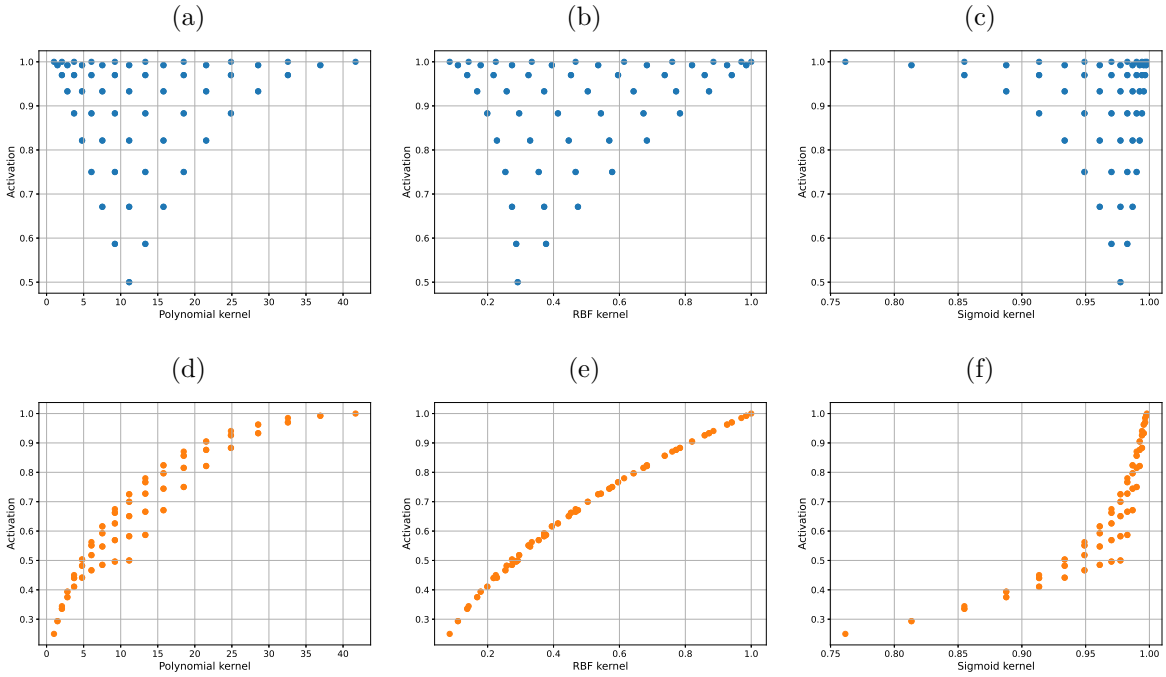
(a), (b), and (c) for the CVQN. (d), (e), and (f) for the CDQN.

but with a shift to the left for the polynomial and RBF kernels, as shown in Figure 5a and Figure 5b, and a considerable shift to the right for the sigmoid kernel, as shown in Figure 5c. The CDQN activation presents relatively the same qualitative behavior already presented in Figure 4f but with shapes that suggest a logarithmic growth in Figure 5d, a linear growth in Figure 5e, and an exponential growth in Figure 5f.

Therefore, the CVQN and the CDQN can solve classification problems where those activation shapes fit correctly. It turns out the nature of each quantum neuron is preserved for any of those metrics. Thus, we follow the analysis using only the Euclidean distance due to its intuitiveness.

Figure 6 shows the PCDQN activation as a function of the Euclidean distance for some combinations of τ and δ, which gives examples of substantial changes that are achieved by means of parametrization on that linear decay already presented in Figure 4e. For example, $(\tau = 1/4, \delta = \pi/2)$ and $(\tau = 1/2, \delta = \pi/4)$ give approximately a linear growth and a logarithmic growth, as shown in Figure 6a and Figure 6b. Figure 6c shows a chain of concave-down parabolas achieved by $(\tau = 1, \delta = \pi/4)$. An exponential decay and an exponential growth are achieved by $(\tau = 1, \delta = 3\pi/2)$ and $(\tau = 2, \delta = 5\pi/4)$ respectively,

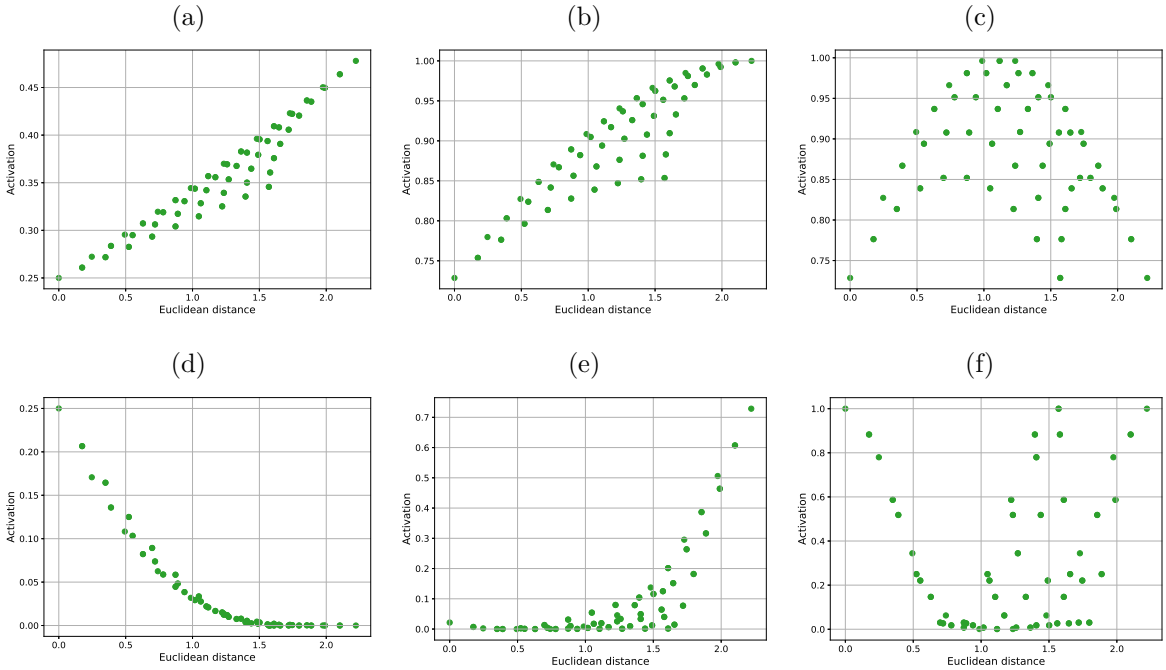
Figure 5 – Quantum Neuron Activation as a Function of the Polynomial Kernel, the RBF Kernel, and the Sigmoid Kernel Between the Original Input and Weight Vectors



Source: The author (2022).

(a), (b), and (c) for the CVQN. (d), (e), and (f) for the CDQN.

Figure 6 – PCDQN Activation as a Function of the Euclidean Distance Between the Original Input and Weight Vectors for Some Combinations of τ and δ



Source: The author (2022).

(a) for $\tau = 1/4$ and $\delta = \pi/2$. (b) for $\tau = 1/2$ and $\delta = \pi/4$. (c) for $\tau = 1$ and $\delta = \pi/4$. (d) for $\tau = 1$ and $\delta = 3\pi/2$. (e) for $\tau = 2$ and $\delta = 5\pi/4$. (f) for $\tau = 4$ and $\delta = 0$.

as shown in Figure 6d and Figure 6e. Figure 6f approaches multiple concave-up parabolas with $(\tau = 4, \delta = 0)$. In summary, the PCDQN is flexible. In practice, such flexibility is expected to enable the PCDQN to fit problems that the other neurons cannot fit correctly.

5 APPLICATION ON CLASSIFICATION PROBLEMS

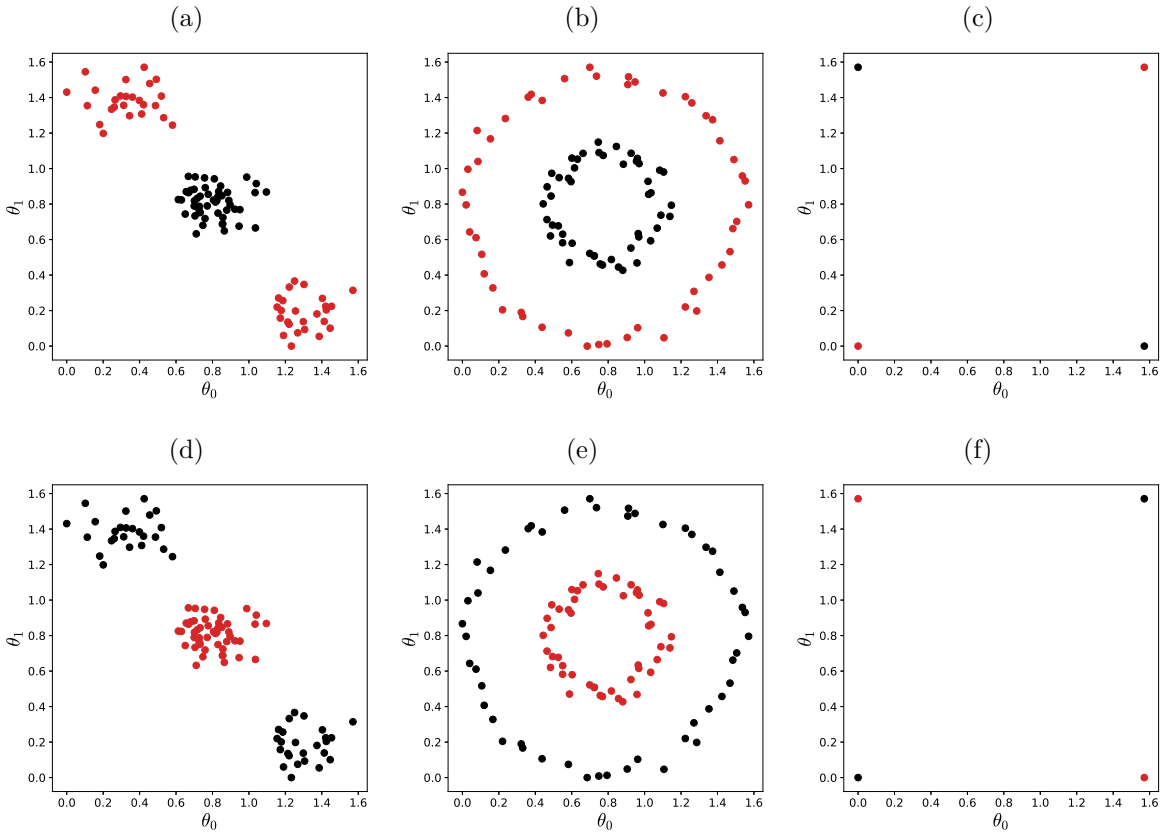
Here, we numerically compare the quantum neurons at solving some classification problems. Additionally, we study the best solutions of the quantum neurons by means of their activation function shapes applied to the problems. Finally, we demonstrate the feasibility of those numerical solutions by executing the quantum neuron circuits on a quantum simulator.

5.1 DATASETS

The three datasets used here are depicted in Figure 7a, Figure 7b, and Figure 7c. The positive class is represented by the black blobs, while the negative class is represented by the red blobs, although there is no previous indication of that labeling. Thus, we decide to swap the labels, deriving three more classification problems where the old negative classes become the new positive classes and vice versa, as depicted in Figure 7d, Figure 7e, and Figure 7f. Note that a line is not able to separate the classes of any of those two-dimensional binary classification problems. It is worth also mentioning that each input variable is scaled to that interval $[0, \pi/2]$.

We refer to Figure 7a and Figure 7d as the diagonal blobs due to the formed structure. The difference is that the target class is formed by the center blobs in Figure 7a, while the target is the corner blobs in Figure 7d. That dataset is provided by the scikit-learn *make_blobs* function, letting *centers* be [(-5,8), (0,0), (5,-8)], *cluster_std* be 1.2, and *random_state* be 0. Each class has 50 samples. Alternatively, the target is the inner circle in Figure 7b, while the target is the outer circle in Figure 7e. Those concentric circles are provided by the scikit-learn *make_circles* function, letting *noise* be 0.05, *factor* be 0.4, and *random_state* be 0. Each circle has 50 samples. Finally, the blobs in the coordinates $[(0, 0), (0, \pi/2), (\pi/2, 0), (\pi/2, \pi/2)]$ form a square, where the target is like the Exclusive OR (XOR) problem in Figure 7c, while the target is like the Not Exclusive OR (NXOR) problem in Figure 7f.

Figure 7 – Classification Problems where the Quantum Neurons are Applied



Source: The author (2022).

The positive class is represented by the black blobs, while the negative class is represented by the red ones. All inputs are in the interval $[0, \pi/2]$.

5.2 SEARCH FOR WEIGHTS AND PARAMETERS

We conduct a grid search for the best weight vector of each quantum neuron for each classification problem. The space of weight vectors is formed here by all vectors (ϕ_0, ϕ_1) where each component assumes one-hundred equidistant values in the interval $[0, \pi/2]$, which gives a ten-thousand-vector weight space. Then, we choose the first weight vector in the search that maximizes the Area Under the Receiver Operating Characteristic Curve (AUC ROC). The AUC ROC allows measuring the classification quality in a threshold-free manner. We compute that metric by means of the scikit-learn *roc_auc_score* function.

Specifically for the PCDQN, we conduct a nested grid search for the best parameter combination and weight vector for each classification problem. The space of parameters is formed here by all combinations (τ, δ), where

$$\tau \in \left\{\frac{1}{4}, \frac{1}{2}, 1, 2, 4\right\}$$

and

$$\delta \in \left\{0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}, \pi, \frac{5\pi}{4}, \frac{3\pi}{2}\right\}.$$

Actually, the combination (1, 0) is not allowed since it produces the CDQN. Consequently, there are 34 allowed combinations to search. Those combinations explore a variety of lengths and shifts along the activation function period, which is 2π. For each combination of τ and δ, we search that ten-thousand-vector weight space. Then, we choose the first values of (τ, δ) and (ϕ_0, ϕ_1) that maximize the AUC ROC together in the nested search.

5.3 RESULTS AND DISCUSSION

Table 1 shows the maximum AUC ROC that each quantum neuron achieved for each classification problem. The CVQN and the CDQN perfectly solved the Diagonal blobs when targeting the class Center. The CVQN also solved the NXOR-like Square blobs, while the CDQN performed like a random model. On the other hand, while the CVQN did not solve the Concentric circles when targeting the class Inner, the CDQN did. In summary, the CVQN and the CDQN only solved two problems each. The PCDQN produced optimal solutions for all problems, even for those problems where the other neurons performed like merely random models. Those results confirm the hypothesis that parametrization gives flexibility.

Table 1 – Maximum AUC ROC of Each Quantum Neuron for Each Classification Problem

Dataset	**Target**	**CVQN**	**CDQN**	**PCDQN**
Diagonal blobs	Center	**1.0**	**1.0**	**1.0**
	Corner	0.5	0.5	**1.0**
Concentric circles	Inner	0.8244	**1.0**	**1.0**
	Outer	0.502	0.4088	**0.9524**
Square blobs	XOR-like	0.5	0.5	**1.0**
	NXOR-like	**1.0**	0.5	**1.0**

Source: The author (2022).

The best parameter combinations and weight vectors of each quantum neuron for each classification problem are reported in Table 2. Actually, the weights have about sixteen decimal places. We report here only the first two decimal places, which is enough to retrieve the original weights of about sixteen decimal places used in the search.

Table 2 – Best Parameter Combinations and Weight Vectors of Each Quantum Neuron for Each Classification Problem

Dataset	**Target**	**CVQN**	**CDQN**	**PCDQN**	
		(ϕ_0, ϕ_1)	(ϕ_0, ϕ_1)	(τ, δ)	(ϕ_0, ϕ_1)
Diagonal blobs	Center	(0, 0)	(0.46, 0.49)	(1/4, 0)	(0.46, 0.50)
	Corner	(0, 0.87)	(0, 1.07)	(1/4, π)	(0.69, 0.76)
Concentric circles	Inner	(0, 0.03)	(0.60, 0.76)	(1/4, 0)	(0.60, 0.76)
	Outer	(0.79, 0)	(1.57, 1.45)	(1/4, π)	(0.82, 0.74)
Square blobs	XOR-like	(0, 0.79)	(0, 0)	(4, 0)	(0.15, 1.20)
	NXOR-like	(0, 0)	(0, 0)	(4, 0)	(0.04, 0.04)

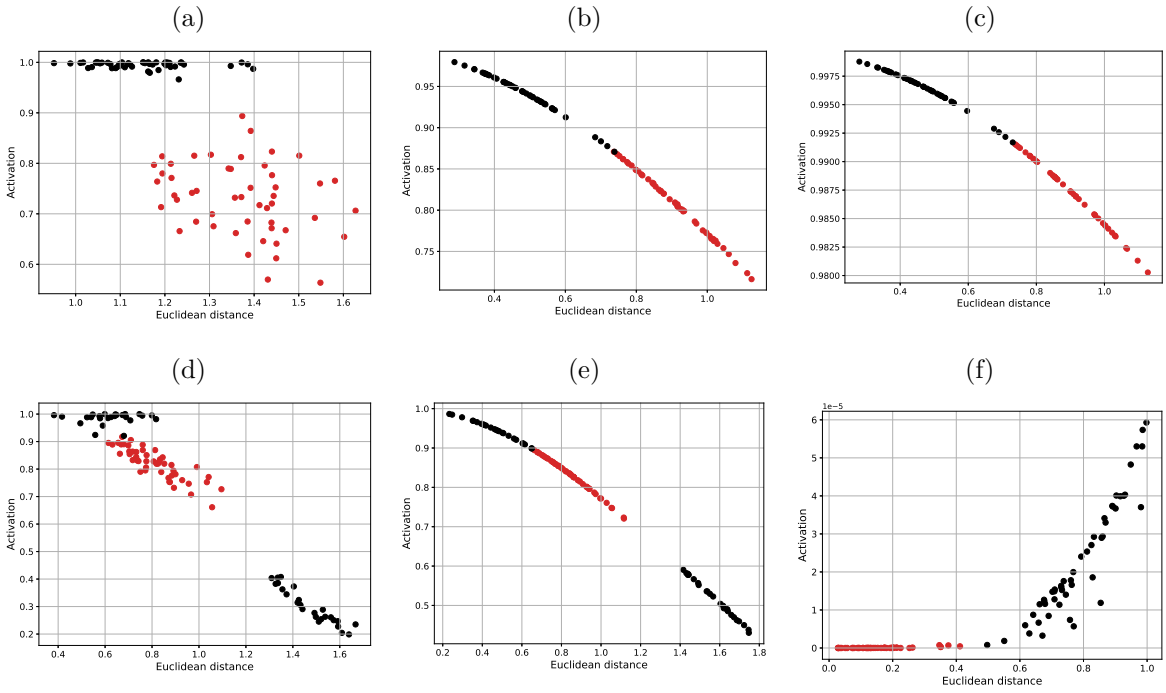
Source: The author (2022).

In the following, we use those best parameters and weights to study the quantum neuron solutions. We then plot the activation shape of each quantum neuron for each classification problem, which depends on the best parameters for the PCDQN. Those shapes are drawn as functions of the Euclidean distance from the input data to the corresponding best weight vector. A problem is solved if the neuron outputs for the black blobs are higher than the ones for the red blobs, regardless of the Euclidean distances to the best weight vector. In this way, there will be a threshold that separates the classes perfectly.

Figure 8 contrasts the quantum neuron solutions for the two problems derived from the Diagonal blobs. The CVQN solves the problem when targeting the class Center by positioning the weight vector closer to the center cluster, which gives values of activation for that cluster higher than the ones for the corner clusters, as shown in Figure 8a. That CVQN solution required the ability to also distinguish the black blobs from the red blobs at the intersection of Euclidean distances between the clusters. A similar CVQN solution correctly separated the upper left cluster from the center cluster when targeting the class Corner, although the bottom right cluster has incorrectly obtained the lowest values, as shown in Figure 8d. The CDQN and the PCDQN solve the problem when targeting the class Center by positioning the weight vector close to the target cluster, and then implementing a monotonic decay as the Euclidean distance increases in any direction, as

shown in Figure 8b and Figure 8c. A monotonic decay does not solve the problem when targeting the class Corner, as shown in Figure 8e for the CDQN. Actually, a monotonic growth from a weight vector within the center cluster solves that problem, as shown in Figure 8f for the PCDQN, which can be achieved by means of parametrization.

Figure 8 – Solutions Obtained by the CVQN, the CDQN, and the PCDQN for the Two Problems Derived from the Diagonal Blobs



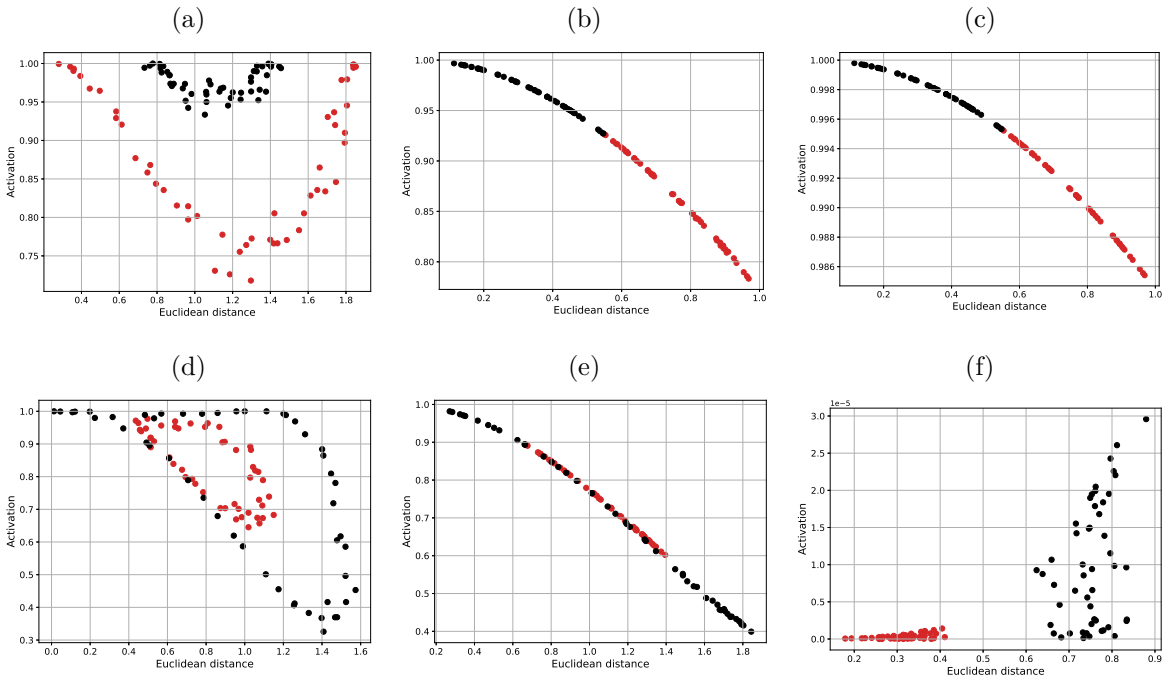
Source: The author (2022).

(a), (b), and (c) when targeting the class Center. (d), (e), and (f) when targeting the class Corner.

The quantum neuron solutions for the two problems derived from the Concentric circles are contrasted in Figure 9. The CVQN correctly separates the classes at the intersection of Euclidean distances between them when targeting the class Inner, as shown in Figure 9a. The CVQN does not solve the problem because the closest and the most distant blobs, which are of the negative class, obtained high values of activation. Figure 9d shows that the CVQN performed like a random model when targeting the class Outer because the solution basically implements a decay. However, the circle shapes inside that solution suggest the multiple neuron outputs for the same Euclidean distance depend on the distance direction. The CDQN and the PCDQN solve the problem when targeting the class Inner by implementing a monotonic decay from that target circle, as shown in Figure 9b and Figure 9c. That shape does not fit the problem when targeting the class Outer, as shown in Figure 9e for the CDQN. A monotonic growth from the center can really fit that problem,

which can be achieved by parametrization, as shown in Figure 9f for the PCDQN. That solution does not fit perfectly because far blobs obtained low values of activation. Thus, the distance direction really matters.

Figure 9 – Solutions Obtained by the CVQN, the CDQN, and the PCDQN for the Two Problems Derived from the Concentric Circles



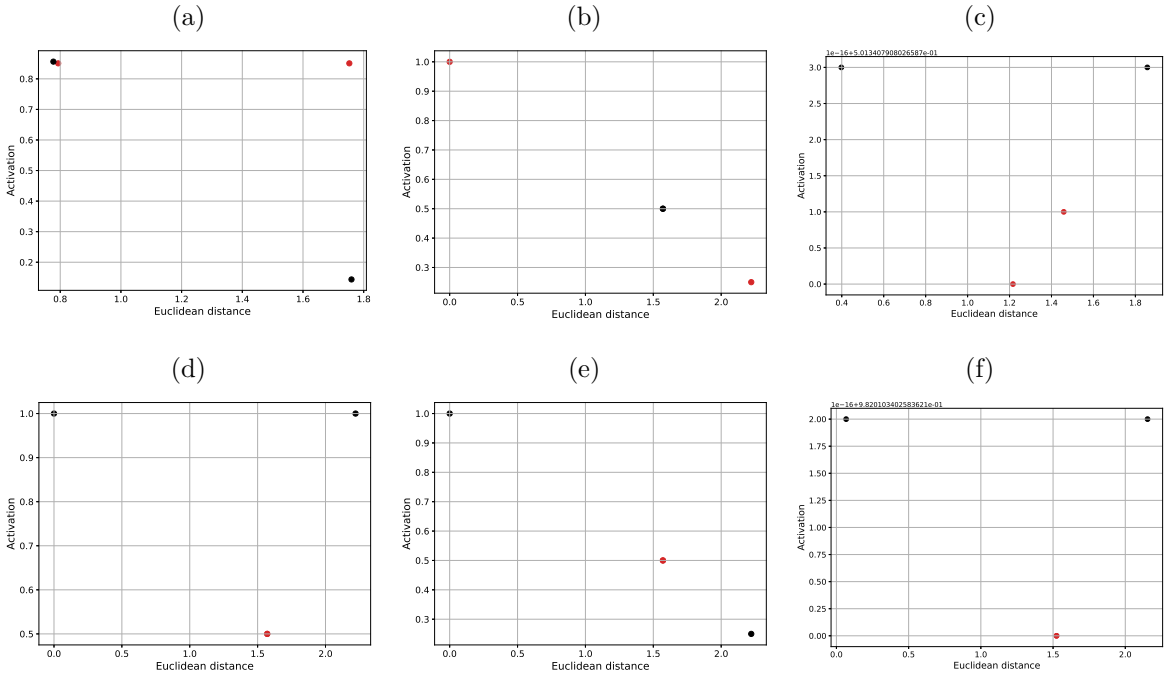
Source: The author (2022).

(a), (b), and (c) when targeting the class Inner. (d), (e), and (f) when targeting the class Outer.

Finally, Figure 10 contrasts the quantum neuron solutions for the two problems derived from the Square blobs. The CVQN performed as a random model when targeting the XOR-like problem because the solution basically implements a decay, as shown in Figure 10a. A distant point obtained a high value of activation due to its distance direction. The CVQN solves the problem perfectly when targeting the NXOR-like problem by taking advantage of all spectrum of its activation shape to generate high values of activation in the extremes and low values in the intermediate distances, as shown in Figure 10d. A monotonic decay does not fit any of those two problems, as shown in Figure 10b and Figure 10e, both for the CDQN. In contrast, a concave-up parabola can fit those two problems. By the way, the PCDQN can implement such shape, as already shown in Figure 6f. Thus, the PCDQN can solve those two problems perfectly, as shown in Figure 10c and Figure 10f.

Those results demonstrate limitations in the CVQN and in the CDQN since their

Figure 10 – Solutions Obtained by the CVQN, the CDQN, and the PCDQN for the Two Problems Derived from the Square Blobs



Source: The author (2022).

(a), (b), and (c) when targeting the XOR problem. (d), (e), and (f) when targeting the NXOR problem.

activation shapes can fit only particular problems. The PCDQN, in turn, can dynamically change its activation shape to a monotonic decay, a monotonic growth, and a concave-up parabola, for example. That list of shapes is not exhaustive. The PCDQN can implement even more shapes by exploring its real parameters, depending on the underlying pattern of the problem. Thus, the PCDQN is flexible. Such flexibility allows the PCDQN to fit complex problems that are beyond the capabilities of the other neurons.

5.4 EXECUTION ON A QUANTUM SIMULATOR

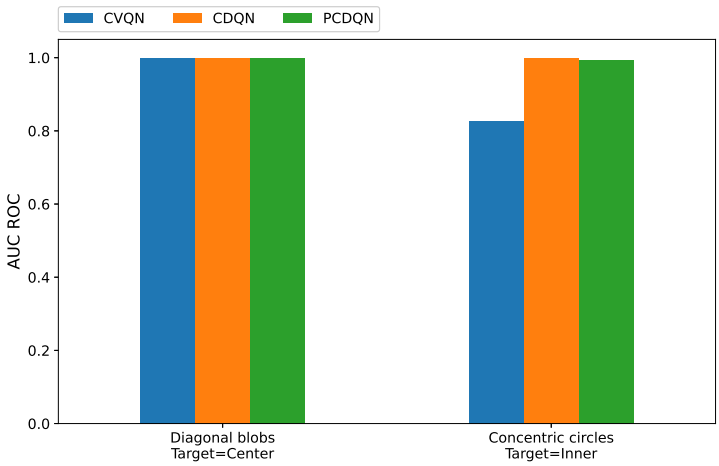
Finally, we demonstrate the physical feasibility of the previous numerical solutions through a proof-of-concept experiment on a quantum simulator. We address here the two classification problems used by Mangini et al. (2020): Diagonal blobs targeting the class Center and Concentric circles targeting the class Inner. The quantum neuron solutions for those two problems are then implemented in quantum circuits that are executed on the Qiskit QASM simulator.

To estimate the quantum neuron activation for a given pair of vectors, we take the

proportion of 1s in the circuit outputs after twenty thousand executions of the circuit. Thus, each quantum neuron is executed twenty thousand times for each input vector of each classification problem to estimate the activation with respect to the corresponding best weight vector. Twenty thousand executions of three quantum neurons applied in two classification problems of a hundred samples each give twelve million circuit executions. Nevertheless, finite numbers of execution can cause small errors in the activation estimates.

After estimating the activation, we can compute the circuit-generated AUC ROC of each quantum neuron for each classification problem on the quantum simulator. Those circuit-generated values of AUC ROC are contrasted in Figure 11. The blue bars represent the CVQN, the orange bars represent the CDQN, and the green bars represent the PCDQN. In fact, the quantum neurons reproduce the reference values reported in Table 1 for those two problems. The CDQN and the PCDQN generate optimal results, while the CVQN is optimal in the first problem but generates an AUC ROC of about 0.82 in the second problem. Therefore, the quantum neuron solutions are validated by the circuit realizations in the quantum simulator.

Figure 11 – AUC ROC Generated by the Quantum Neuron Circuits on a Quantum Simulator for the Diagonal Blobs when Targeting the Class Center and the Concentric Circles when Targeting the Class Inner



Source: The author (2022).

The blue, orange, and green bars represent the CVQN, the CDQN, and the PCDQN, respectively.

6 BENCHMARKING CLASSICAL AND QUANTUM NEURONS

The experiments of the previous chapter prototyped the quantum neurons through an initial application on a few classification problems. In this chapter, we conduct an extensive experimental comparison in order to obtain enough evidence to support solid conclusions. The present extension is based on comparing against more neurons, which includes classical ones. Classification problems derived from other datasets are also considered here. We put all this together in benchmarks from two perspectives. First, the neurons are contrasted by their maximum AUC ROC values obtained on the classification problems. Finally, the class separation on the classification problems perfectly solved by the neurons becomes the comparison criterion of the benchmark.

6.1 NEW NEURONS

During the development of this research, the continuous Rosenblatt quantum neuron (CRQN) was proposed in the literature (MARONESE; PRATI, 2021). Thus, we readily took it into account in the benchmark development. In this way, the previous neurons could be compared against one more model that can be recast in the kernel-based framework of quantum neurons proposed in this research. The CRQN provides a straightforward generalization of the BVQN (TACCHINO et al., 2019) by basically applying a feature mapping that only normalizes classical vectors. The difference from the BVQN is that the CRQN works with real classical vectors instead of binary ones. From classical vectors $\boldsymbol{\theta} = (\theta_0, \theta_1, \cdots, \theta_{m-2}, \theta_{m-1})$ and $\boldsymbol{\phi} = (\phi_0, \phi_1, \cdots, \phi_{m-2}, \phi_{m-1})$, where $\theta_j, \phi_j \in [-1, 1] \subset \mathbb{R}$, the CRQN feature mappings $\Phi_i(\cdot)$ and $\Phi_w(\cdot)$ provide the following N-dimensional quantum vectors, where $N = m + 2$:

$$\vec{i} = \Phi_i(\boldsymbol{\theta}) = \frac{1}{\sqrt{m}}(\alpha_{\boldsymbol{\theta}}, \theta_0, \theta_1, \cdots, \theta_{m-2}, \theta_{m-1}, 0) \text{ and}$$
$$\vec{w} = \Phi_w(\boldsymbol{\phi}) = \frac{1}{\sqrt{m}}(0, \phi_0, \phi_1, \cdots, \phi_{m-2}, \phi_{m-1}, \alpha_{\boldsymbol{\phi}}).$$

The components $\alpha_{\boldsymbol{\theta}}$ and $\alpha_{\boldsymbol{\phi}}$ compensate for the normalization factor $1/\sqrt{m}$ so that the vectors $\vec{i}$ and $\vec{w}$ become unit vectors. In this way, $\alpha_{\boldsymbol{\theta}}$ equals $\sqrt{m - \boldsymbol{\theta} \cdot \boldsymbol{\theta}}$, while $\alpha_{\boldsymbol{\phi}}$ equals $\sqrt{m - \boldsymbol{\phi} \cdot \boldsymbol{\phi}}$. In the inner product between $\vec{i}$ and $\vec{w}$, $\alpha_{\boldsymbol{\theta}}$ is canceled out by a zeroed

component in the respective position in $\vec{w}$. Similarly, $\vec{i}$ has a zeroed component in the respective position to cancel out α_{ϕ}. As the BVQN, the CRQN has an exponential advantage by storing the $N = m + 2$ vector components in only n qubits, where $n = \log_2 N$. If N is not an exact power of 2, the quantum vectors are filled with more $2^{\lceil \log_2 N \rceil} - N$ zeroed components.

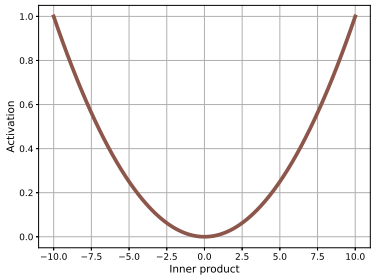
For the amplitude encoding of $\vec{i}$, the operator $E(\boldsymbol{\theta})$ first encodes each $|i_j|$ by applying a particular case for real data of an algorithm that initializes arbitrary quantum states (SHENDE; BULLOCK; MARKOV, 2006). That particular implementation for real vectors is basically composed of multi-controlled R_y gates. After the rotations along the y-axis to encode the components in absolute value, $E(\boldsymbol{\theta})$ corrects the component signs by applying the HSGS already used by the BVQN. Analogously, the operator $D(\boldsymbol{\phi})$ decodes $\vec{w}$ by first applying the HSGS to cancel negative signs, and then multi-controlled R_y gates finally decode each $|w_j|$. Therefore, the operators $E(\boldsymbol{\theta})$ and $D(\boldsymbol{\phi})$ are subdivided into two steps that are implemented by exponentially expensive circuits in terms of depth and number of operations, even using complex multi-controlled gates. Finally, the CRQN fires with the probability presented in Equation 6.1.

$$|\vec{w}^* \cdot \vec{i}|^2 = \left| \frac{1}{m} \sum_{j=0}^{m-1} \phi_j \theta_j \right|^2 \tag{6.1}$$

As can be seen, the CRQN activation depends explicitly on the inner product between the original vectors. Thus, we can unequivocally study its activation by varying only such relation between the vectors. Figure 12 shows the CRQN activation values for inner products varying from -10 to 10 with a step size equal to 0.1 and m equal to 10. This activation function shape exhibits the impacts on the CRQN outputs from changes in that interaction between classical vectors. The minimum point of the parabola is at the inner product equal to 0. Then, the activation monotonically increases as the inner product moves away from 0 in any direction. The maximum activation values occur in the extreme inner products.

However, this benchmark of neurons is not about quantum models only. Classical neurons are also contemplated here. The idea is to extensively compare standard classical neurons, which are strongly consolidated, with emerging quantum ones, which are seeking consolidation in the community of machine learning and its applications. This type of research is valuable for demonstrating whether a practical quantum advantage can exist

Figure 12 – CRQN Activation as a Function of the Inner Product Between the Original Vectors



Source: The author (2022).

By controlling the inner product directly, the CRQN activation was computed for that metric varying from -10 to 10 in steps of 0.1. The normalization occurred by a factor m equal to 10 in the CRQN activation function.

or not. In this way, three classical neurons with common activation functions are explored in the benchmark. The linear classical neuron (LCN), the logistic sigmoid classical neuron (LSCN), and the hyperbolic tangent classical neuron (HTCN) are the three classical baselines against which the quantum neurons are compared here.

The notion of feature space mappings that are implemented by quantum neurons does not hold for the classical neurons. Thus, the vectors $\vec{i}$ and $\vec{w}$ come down to the original input and weight vectors $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively. The inner product between $\vec{i} = \boldsymbol{\theta}$ and $\vec{w} = \boldsymbol{\phi}$ determines the outputs given by the activation functions of those classical neurons. The activation function of the LCN is presented in Equation 6.2. The LSCN activation function was already presented in Equation 2.2 for contextualization reasons. Equation 6.3 finally presents the HTCN activation function.

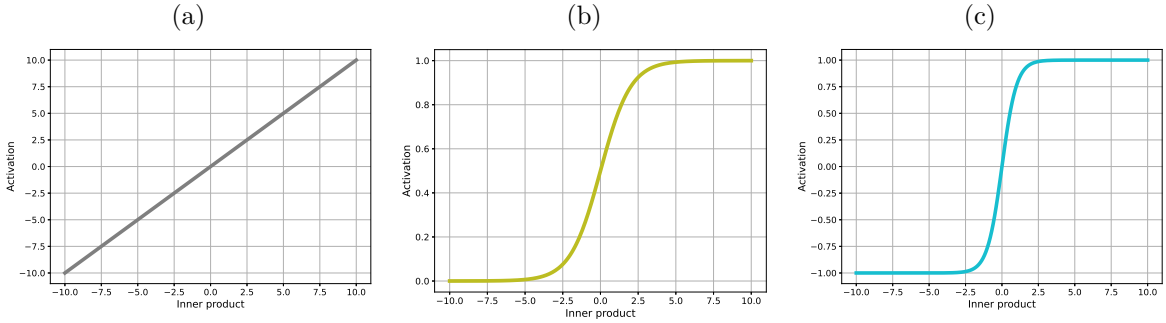
$$\varphi(\vec{w} \cdot \vec{i}) = \vec{w} \cdot \vec{i} \tag{6.2}$$

$$\varphi(\vec{w} \cdot \vec{i}) = \frac{e^{(\vec{w} \cdot \vec{i})} - e^{-(\vec{w} \cdot \vec{i})}}{e^{(\vec{w} \cdot \vec{i})} + e^{-(\vec{w} \cdot \vec{i})}} \tag{6.3}$$

Some intuition can be obtained from the shapes of those activation functions. Figure 13 shows the outputs of each classical neuron as the inner product varies. As can be seen in Figure 13a, the LCN implements a line that crosses the origin with a slope of 1 by giving outputs identical to the inputs. With shapes that look like the letter “S", the LSCN and the HTCN suppress those outputs to the intervals [0, 1] and [-1, 1], as shown in Figure 13b and Figure 13c respectively. In all cases, the neuron outputs monotonically grow with the inner product. That output growth occurs at the same ratio of the inner

product changes in the LCN. In turn, the derivatives vary in the LSCN and the HTCN. Firstly, the derivatives are low but smoothly increasing for small inner products, where the neuron outputs are also small. Then, the derivatives achieve their highest values around inner products of 0, where the neuron outputs grow rapidly from changes in the inputs. Finally, the derivatives become smoothly low again for high inner products, where the neuron outputs are also high. It is worth mentioning that, in the following benchmark experiments, the classical neurons work with vector entries pre-processed to the interval [0, 1] as a common practice to avoid scaling issues.

Figure 13 – Classical Neuron Activation as a Function of the Inner Product Between Vectors



Source: The author (2022).

By controlling the inner product directly, the classical neuron activation was computed for that metric varying from -10 to 10 in steps of 0.1. (a) for the LCN. (b) for the LSCN. (c) for the HTCN.

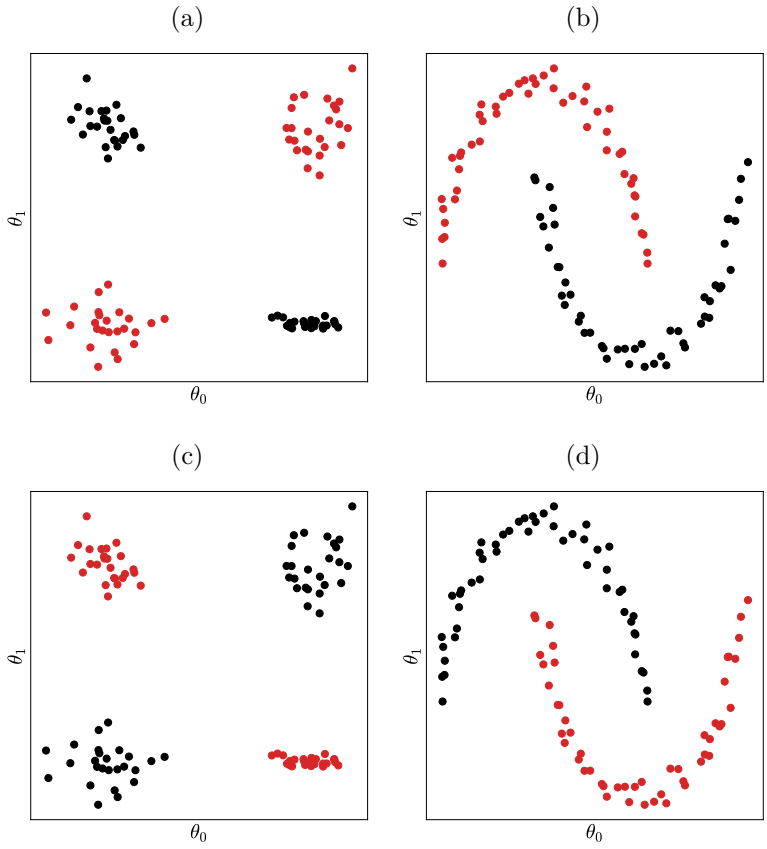
6.2 NEW DATASETS

To benchmark the classical and quantum neurons, a greater variety of classification problems is needed in order to contemplate more situations and levels of complexity. Thus, more datasets are required for an extensive experimental comparison between classical and quantum neurons. In this way, an old dataset was modified and other three new datasets were added to the experiments. The two classification problems derived from the Diagonal blobs, which were already showed in Figure 7a and Figure 7d, continue for the present benchmark. In the same manner, this benchmark also consider the two classification problems derived from the Concentric circles already presented in Figure 7b and Figure 7e.

Figure 7c and Figure 7f already presented the two problems addressed here that were derived from a square with a blob in each of its corners. The modification in that old

dataset for the benchmark was to fill the corners of a square with clusters of blobs instead. Figure 14a shows the obtained classification problem when organizing the Square clusters as the XOR problem. Conversely, targeting the Square clusters like the NXOR problem generates the classification problem presented in Figure 14c. This new dataset can be obtained by the scikit-learn *make_classification* function, letting *class_sep* be 5.0 and *random_state* be 0. The cluster of each corner has 25 samples, which implies 50 samples per class. However, the first really new dataset considered for the benchmark was the Moons. Two interleaved moons are obtained from two half circles where one is reflected and shifted with respect to the other. Figure 14b shows the problem of targeting the moon that is like the $\cup$ symbol. The other moon, which is like the $\cap$ symbol, is the target class in Figure 14d. That dataset of moons can be produced by the scikit-learn *make_moons* function, letting *noise* be 0.05 and *random_state* be 0. Each moon is formed by 50 samples.

Figure 14 – Classification Problems Derived from the Two New Datasets of Two Classes Contemplated in the Benchmark of Classical and Quantum Neurons



Source: The author (2022).

The positive class is represented by the black blobs, while the negative class is represented by the red ones. Intentionally, the data scale is omitted because the neurons work in different input intervals.

The first dataset of 3 classes contemplated in this study was the Gaussian quantiles.

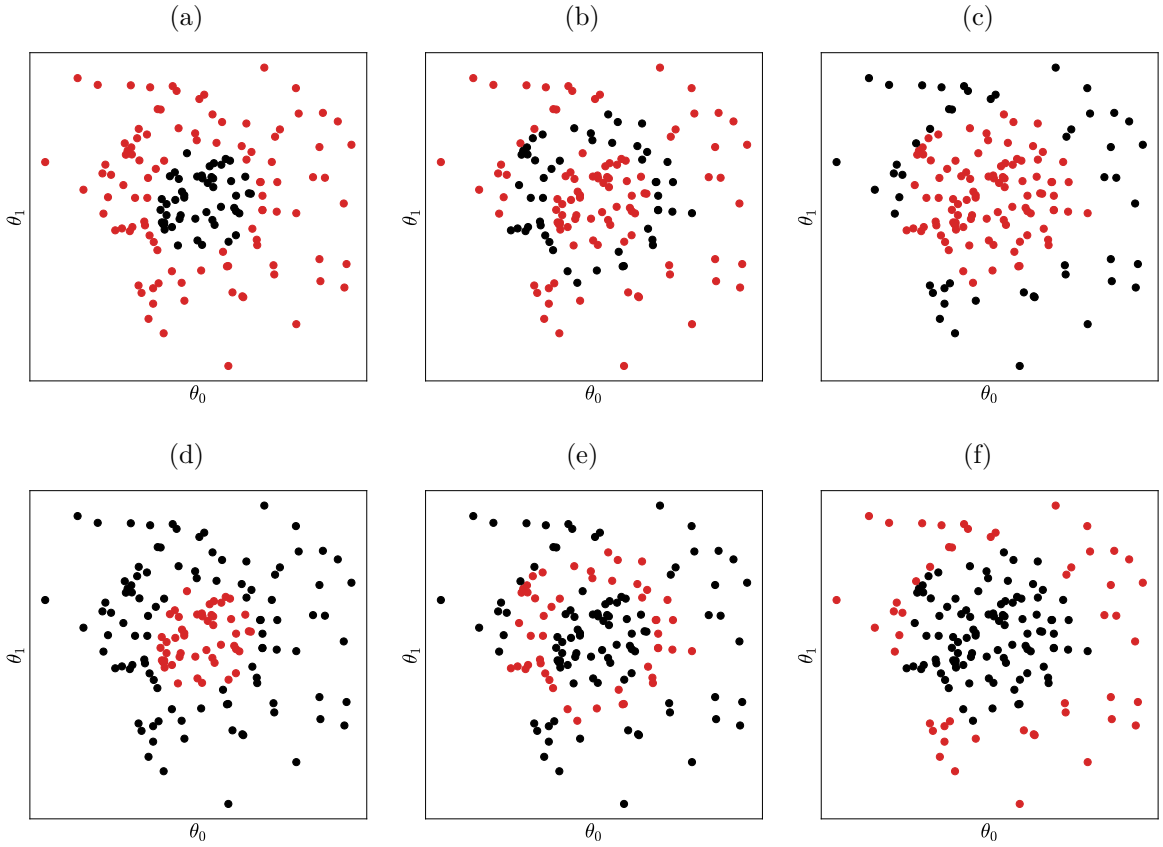
This dataset is constructed here by sampling 150 points from a 2-dimensional normal distribution. Then, the 50 points closest to the origin are labeled as the class Inner. The class Middle is formed taking out the 50 points closest and the 50 points more distant from the origin. Those 50 points that are more distant from the origin are labeled as the class Outer. Applying a one-vs-all approach, the Gaussian quantiles derive three classification problems, where the first one is to target the class Inner, the second one is to target the class Middle, and the third one is to target the class Outer, as shown respectively in Figure 15a, Figure 15b, and Figure 15c. For consistency reasons, we also applied in this dataset an inverted one-vs-all approach, where the class at hand is the negative class of the problem and the others form the positive class. Figure 15d shows the problem of targeting all classes other than the class Inner. The class Middle is the negative class of the problem in Figure 15e. Finally, in Figure 15f, the class Inner and the class Middle form the positive class, while the class Outer is the negative class. The scikit-learn *make_gaussian_quantiles* function, letting *random_state* be 0, provide the Gaussian quantiles used here.

Different from the previous problems, the last dataset added to the benchmark consists of real-life data. The Iris is a well-known 4-dimensional dataset of 3 classes, with 50 instances per class. The scikit-learn *load_iris* function provide the data. The classes refer to 3 species of flowers, which are Setosa, Versicolor, and Virginica. Again, by applying both a one-vs-all approach and an inverted one-vs-all approach, six classification problems can be derived from this dataset:

1. Setosa as the positive class, while Versicolor and Virginica as the negative class;
2. Setosa as the negative class, while Versicolor and Virginica as the positive class;
3. Versicolor as the positive class, while Setosa and Virginica as the negative class;
4. Versicolor as the negative class, while Setosa and Virginica as the positive class;
5. Virginica as the positive class, while Setosa and Versicolor as the negative class;
6. Virginica as the negative class, while Setosa and Versicolor as the positive class.

In total, from 4 datasets of 2 classes and 2 datasets of 3 classes, we derive 20 classification problems in which the classical and quantum neurons are applied in the benchmark.

Figure 15 – Classification Problems Derived from a Dataset of Three Classes Contemplated in the Benchmark of Classical and Quantum Neurons



Source: The author (2022).

The positive class is represented by the black blobs, while the negative class is represented by the red ones. Intentionally, the data scale is omitted because the neurons work in different input intervals.

Those 20 problems are addressed here in the original form and with the presence of a bias, summing 40 problems to apply the neurons. The bias itself means a new degree of freedom for the weight vectors. Consequently, a neutral element is attached to the input vectors. Attaching the neutral element of the addition or the neutral element of the multiplication depends on the neuron that is applied. The way that weight values operate in the activation function determines the attached input value. For the quantum neurons already applied in the previous chapter, the attached input value is 0. For the neurons added in this chapter, the attached input value is 1. Regardless of bias, the applied neuron also determines the input scale. As already seen in the previous chapter, those quantum neurons require a pre-processing stage in which the data is scaled to the interval $[0, \pi/2]$. A rescaling of data is necessary for the neurons introduced in this chapter. The CRQN works in the interval $[-1, 1]$, while the classical neurons operate in the interval $[0, 1]$. We suppressed the scale in Figure 14 and Figure 15 because of that neuron dependency.

6.3 BENCHMARKING THROUGH THE MAXIMUM AUC ROC

The maximum values of AUC ROC that we reported before were obtained by searching a weight space of ten thousand two-dimensional vectors. For each vector component, the allowed interval $[0, \pi/2]$ was subdivided into one-hundred equidistant values. The weight space was then constructed by taking the Cartesian product of those sets of vector component values. Now, we are also addressing four-dimensional problems and allowing the presence of a bias. Thus, a different weight space is required depending on the number of freedom degrees of the problem. All weight spaces used here have at least ten thousand vectors. The number of equidistant parts in which the interval $[0, \pi/2]$ is subdivided is the smaller integer that guarantees the required weight space size when the Cartesian product is taken.

Specifically, two-dimensional problems and four-dimensional problems, both without a bias, have a weight space of 10000 vectors by subdividing that interval into 100 parts and 10 parts, respectively. When a bias is added, the weight space size becomes 10648 by subdividing that interval into 22 parts for two-dimensional problems and becomes 16807 by subdividing that interval into 7 parts for four-dimensional problems. The rule is to subdivide that interval in a number of parts n_p such that $n_p = \lceil \sqrt[d]{10000} \rceil$, where d is the number of freedom degrees. In that way, a weight space with n_v vectors is obtained, where $n_v = (n_p)^d \geq 10000$. Note that there is a trade-off here when it comes to adding a bias. One more degree of freedom increased n_v but decreased n_p. Thus, there are more trials to maximize the AUC ROC when a bias is added, but the steps are less refined in the grid.

The weight space depends not only on the number of freedom degrees but also on the neuron that is applied due to scale reasons. Not all neurons work with vector entries in the interval $[0, \pi/2]$. Actually, only the neurons of the previous chapter work in this interval. The neurons that we included now work in another interval of vector entries. The CRQN works in the interval $[-1, 1]$, while the classical neurons work in the interval $[0, 1]$. In this way, the weight space constructed for a problem may be rescaled to attend to the interval of values required for the neuron at hand. Therefore, we conducted a fair comparison between the neurons in terms of the number of trials and the refinement of the steps, with each neuron working in its appropriate interval of vector entries.

Here, that distinction between the CDQN and the PCDQN in order to make explicit the gains obtained by the parametrization is no longer necessary. Allowing the combination

($\tau = 1, \delta = 0$) in the parameter space enables the PCDQN to instantiate the CDQN when appropriate. Thus, there are 35 parameter combinations to try when adjusting the PCDQN for a given problem. As before, for each parameter combination of the PCDQN, we also search the space of weight vectors, which depends on the freedom degrees of the problem. In order to avoid overly fine decision boundaries, we round the outputs of all neurons to two decimal places. Finally, we compute the AUC ROC obtained by each trial and then report the maximum value found for each neuron when solving each classification problem without a bias and with a bias.

Table 3 benchmarks the neurons through their best values of AUC ROC on the sixteen classification problems derived from the datasets of two classes. When there is no bias in the problem, we indicate by “N" in the column named “Bias”. Alternatively, “Y” indicates that there is a bias in the problem. In each problem, we compared the quantum neurons and the classical neurons separately. The maximum AUC ROC value obtained within each group of neurons is marked in bold. Then, we compared the best results between the two groups of neurons. The best overall result is further marked by an underscore.

The first problem addressed in this extensive comparison between neurons is the Diagonal blobs when targeting the class Center. The CVQN and the PCDQN performed like an optimal model, while the CRQN, the LCN, and the HTCN performed like a random model in that problem. The LSCN, in turn, performed as a middle ground between the random and the optimal models. When a bias was added, the CRQN achieved that intermediate performance also obtained by the LSCN. Targeting the class Corner made the CVQN perform as a random model while allowing the CRQN to achieve the optimal value of AUC ROC. The addition of a bias, however, did not produce significant changes in the neuron performances in that problem.

Regarding the Concentric circles when targeting the class Inner, only the PCDQN performed as an optimal model. The CVQN performed as a middle ground between the PCDQN performance and the performance of all other neurons, which approached the random model with the LSCN being slightly better. The addition of a bias allowed the CVQN to match the PCDQN performance and the CRQN to match the LSCN performance. Changing the target to the class Outer made the CVQN perform like a random model and the CRQN approach an optimal model. Even with a slight deterioration in the performance, the PCDQN was the best neuron again. Finally, the neurons did not take advantage of the bias when targeting the class Outer.

Table 3 – Benchmarking Classical and Quantum Neurons through the Maximum AUC ROC Values Obtained on Datasets of Two Classes

Dataset	Target	Bias	CVQN	PCDQN	CRQN	LCN	LSCN	HTCN
Diagonal blobs	Center	N	**1.0**	**1.0**	0.53	0.57	**0.75**	0.57
		Y	**1.0**	**1.0**	0.75	0.5664	**0.75**	0.628
	Corner	N	0.5	**1.0**	**1.0**	0.58	**0.75**	0.58
		Y	0.5	**1.0**	**1.0**	0.5332	**0.75**	0.6518
Concentric circles	Inner	N	0.832	**1.0**	0.5	0.5394	**0.68**	0.5394
		Y	**1.0**	**1.0**	0.66	0.52	**0.67**	0.563
	Outer	N	0.5036	**0.93**	0.8596	0.5288	**0.69**	0.5204
		Y	0.4604	**0.9202**	0.8554	0.5304	**0.67**	0.5484
Square clusters	XOR-like	N	0.5	0.75	**1.0**	0.5604	**0.74**	0.5604
		Y	0.5056	0.75	**1.0**	0.5692	**0.66**	0.5618
	NXOR-like	N	**1.0**	0.75	**1.0**	0.536	**0.73**	0.5348
		Y	0.965	0.75	**1.0**	0.544	**0.6814**	0.546
Moons	$\cup$-like	N	0.9478	**0.9568**	0.5932	**0.826**	**0.826**	**0.826**
		Y	0.9546	0.957	**0.9604**	0.8186	**0.8434**	**0.8434**
	$\cap$-like	N	0.9472	**0.9516**	0.6	0.941	**0.9418**	0.9398
		Y	0.9474	0.9508	**0.959**	0.943	0.944	**0.9472**

Source: The author (2022).

Since each dataset provides two classification problems that are addressed both without a bias and with a bias, four datasets provided sixteen classification problems in which the neurons were contrasted. In the column named “Bias", “N" and “Y" refer to the absence and the presence of a bias, respectively. The best results among the quantum neurons are highlighted in bold. The best results among the classical neurons are also highlighted in bold. The best overall results are underlined.

The Square clusters were perfectly separated only by the CRQN when the target class was like the XOR problem. The CVQN, the LCN, and the HTCN performed like random models on that problem. The PCDQN and the LSCN obtained intermediate performances between those extremes. With a bias, the only difference that emerged was the deterioration in the LSCN performance, making that neuron approach a random model. Organizing the problem like the NXOR problem allowed the CVQN to perform optimally, while the other neurons performed as before without a bias, which includes the CRQN performing also optimally. The addition of a bias deteriorated not only the LSCN again but also the CVQN this time.

The last two-class dataset addressed here was the Moons. When targeting the moon that is like the $\cup$ symbol, the CVQN and the PCDQN approached optimal models, with the PCDQN being slightly better. The CRQN, in turn, approached a random model. All

classical neurons obtained intermediate performances. With a bias, the CRQN became the best neuron to address that problem, which represented one of the most significant improvements observed in the experiments due to the addition of a bias. By targeting the moon that is like the ∩ symbol, although the PCDQN continued as the best neuron followed by the CVQN when a bias is not used, all classical neurons obtained competitive performances that approach the performances of those quantum neurons. One more time, the CRQN performed like a random model, and the expressive gains from the addition of a bias made it the best of all neurons for the problem at hand.

The results for the two datasets of three classes addressed here, which are the Gaussian quantiles and the Iris, are reported in Table 4. There are twenty-four classification problems derived from those datasets in which the maximum AUC ROC values obtained by the neurons are contrasted. Regarding the Gaussian quantiles, the first problem was to target the class Inner, where only the PCDQN solved perfectly. The CVQN approached an optimal model, while the other neurons performed like random models with the LSCN performing moderately better. The CVQN approached the PCDQN and the CRQN approached the LSCN when a bias was added. When the target class is formed by all classes other than the class Inner, the CRQN approached an optimal model but not as much as the PCDQN that continued as the best neuron. The other neurons performed like random models, with the LSCN relatively better again. That LSCN performance was approximated by the HTCN making use of a bias.

Separating the class Middle from the other classes of the Gaussian quantiles was the most challenging task because all neurons performed basically as random models, independent of whether or not a bias was used and whether or not the class Middle was the positive class of the problem. Nevertheless, the PCDQN can be highlighted since it obtained the best performances in all those cases, approaching the middle ground performance between the random and the optimal models, especially when a bias was used to target the class Middle as the positive class. Targeting the class Outer, independent of using a bias, was another case where the PCDQN obtained the best results by approaching an optimal model. Among the remaining neurons, the CRQN achieved the best values of AUC ROC by also approaching an optimal model. The other neurons performed like random models. When the class Outer was the negative class, the PCDQN solved the problem perfectly, the CVQN approximated an optimal model, and all other neurons performed like random models. The CVQN actually performed as an optimal model when

Table 4 – Benchmarking Classical and Quantum Neurons through the Maximum AUC ROC Values Obtained on Datasets of Three Classes

Dataset	Target	Bias	CVQN	PCDQN	CRQN	LCN	LSCN	HTCN
Gaussian quantiles	Inner	N	0.8764	**1.0**	0.5	0.5536	**0.685**	0.5536
		Y	0.9943	**1.0**	0.67	0.5528	**0.685**	0.5782
	Not Inner	N	0.5174	**0.9328**	0.8824	0.59	**0.67**	0.59
		Y	0.4905	**0.9168**	0.8708	0.5639	**0.66**	0.6462
	Middle	N	0.6264	**0.695**	0.5859	0.5382	**0.573**	0.5382
		Y	0.6115	**0.735**	0.6185	0.5377	**0.5779**	0.5728
	Not Middle	N	0.5535	**0.6508**	0.6104	0.5766	**0.615**	0.5766
		Y	0.5474	**0.6588**	0.63	0.5829	**0.6438**	0.5948
	Outer	N	0.5375	**0.9182**	0.8688	0.62	**0.66**	0.62
		Y	0.5012	**0.9159**	0.8638	0.5519	**0.66**	0.6083
	Not Outer	N	0.8575	**1.0**	0.5	0.525	**0.655**	0.525
		Y	0.9842	**0.9996**	0.63	0.5	**0.655**	0.5185
Iris	Setosa	N	**1.0**	**1.0**	0.9825	0.8796	**0.8802**	0.8796
		Y	**1.0**	**1.0**	**1.0**	0.8796	0.8802	**0.8812**
	Not Setosa	N	**1.0**	**1.0**	0.716	**1.0**	**1.0**	**1.0**
		Y	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
	Versicolor	N	0.9422	**0.9955**	0.6808	0.5804	**0.6395**	0.5785
		Y	**0.9955**	0.9941	0.8306	0.5804	**0.6258**	0.5867
	Not Vers.	N	0.7163	0.9946	**0.9989**	0.7942	**0.7972**	0.7942
		Y	0.6606	0.9928	**0.9985**	0.7956	**0.7977**	0.7974
	Virginica	N	0.9452	**0.9931**	0.7511	**0.9979**	0.9973	0.9974
		Y	0.9891	0.9928	**0.9986**	0.9976	0.9976	**0.9978**
	Not Virg.	N	0.9549	**0.999**	0.6251	0.5856	**0.5872**	0.5854
		Y	0.9988	**0.9991**	0.9955	0.5935	0.5946	**0.5947**

Source: The author (2022).

Since each dataset provides six classification problems that are addressed both without a bias and with a bias, two datasets provided twenty-four classification problems in which the neurons were contrasted. In the column named “Bias", “N" and “Y" refer to the absence and the presence of a bias, respectively. The best results among the quantum neurons are highlighted in bold. The best results among the classical neurons are also highlighted in bold. The best overall results are underlined.

using a bias. In that case, although a minor deterioration occurred, the PCDQN continued with the best performance.

Finally, considering the problems derived from the Iris, the first one is to target the class Setosa, where the CVQN and the PCDQN achieved optimal results. The classical neurons approximated an optimal model but not as close as the CRQN, which actually performed perfectly when a bias was added. All neurons perfectly solved the problem

when the class Setosa was the negative class, with the exception of the CRQN that only achieved that performance with a bias again. When targeting the class Versicolor, the CVQN approached an optimal model, although the PCDQN approached this ideal result even more. The other neurons performed as random models, with the CRQN being slightly better. The addition of a bias allowed enough improvement in the CVQN performance to overcome the PCDQN performance, which suffered a minor deterioration. A reasonable improvement also occurred in the CRQN performance by the use of a bias. On the other hand, the neurons did not take advantage of a bias when the class Versicolor was the negative class. In that case, the PCDQN and the CRQN really approximated the ideal value of AUC ROC, with the CRQN being smoothly superior. All other neurons obtained intermediate performances.

The problem of targeting the class Virginica was the only case where the classical neurons outperformed the quantum neurons, although the PCDQN obtained a really competitive result. The CVQN also approached an optimal model but not enough to overcome the PCDQN or even the classical neurons. The CRQN obtained a middle ground between the random and the ideal results. When a bias was added, the CVQN improved but still not enough to overcome the other neurons. However, the CRQN took advantage of the bias significantly, overcoming the other neurons and becoming the best neuron for that problem. Ultimately, the CRQN and the classical neurons, which have performed near optimally, became random models by only inverting the problem in order to turn the class Virginica into the negative class. Then, the CRQN experienced the most significant improvement observed in all experiments conducted in this study due to the addition of a bias. The CRQN almost solved the problem perfectly with a bias. The CVQN really approached an optimal model, especially with a bias. The PCDQN, in turn, practically achieved the ideal result, which made it the best neuron for the problem one more time.

In summary, the quantum neurons outperformed the classical ones in all cases. Generally, the quantum neurons performed like optimal models, and the classical neurons obtained poor performances. The quantum neurons demonstrated a superior capacity in hard problems for classical neurons since the problems are not linearly separable. The LSCN was the classical neuron with the best performance, although its performance did not match the high performances of the quantum neurons. Specifically, the PCDQN was the best neuron in most cases. Those results provided even more evidence that the parametrization really equips the PCDQN with the required flexibility to fit a wide range

of problems. Regarding the use of a bias, when it was beneficial, the CVQN and mainly the CRQN were the neurons that could take advantage of the new degree of freedom. In the amplitude encoding, the bias is stored in a separate component of the quantum vector, which enables direct control of its impacts. Differently, in the qubit encoding of the PCDQN, the bias affects half of the components of the quantum vector, which complicates its exploration strategically due to those multiple dependencies. For the classical neurons, the problems are beyond their capacities even with a bias.

6.4 BENCHMARKING THROUGH THE SEPARATION OF CLASSES

In this section, the benchmark between classical and quantum neurons proceeds to another perspective, which is focused on the cases where the AUC ROC achieved by the neurons equals 1. The idea is to measure the quality of those solutions in some way, even though the performances are already perfect from the AUC ROC viewpoint. An AUC ROC equal to 1 means that some threshold separates the positive class from the negative class. That is, the neuron outputs for the examples of the positive class are higher than the neuron outputs for the examples of the negative class. However, there is no information regarding the quality of that separation between the classes. The greater the separation, the greater the margin, which represents the robustness of the solution to errors. From this viewpoint, it is possible to further compare neurons that achieved AUC ROC values of 1.

In this way, we calculated a measure of class separation by taking the difference between the lowest neuron output for an example of the positive class and the highest neuron output for an example of the negative class. The distance of those neuron outputs reveals the tolerance of that neuron solution. The greater the distance, the greater the separation of classes. However, the neuron can separate the classes of the problem with multiple weight vectors searched in the grid. Those best weight vectors generate solutions that differ from each other by the metric of class separation that we defined. With some weight vectors, the classes may be separated at the limit of separation. With others, the classes may be far from each other. Thus, the solution tolerance varies with the weight vector that is used, even though all those best weight vectors are equal from the AUC ROC viewpoint since the problem is perfectly solved using any of them. Therefore, due to that variance, we contrast the neurons by the mean value of the class separations obtained

from all weight vectors that solved the problem. Also, we compare the proportion of those best weight vectors over the whole weight space.

An ideal neuron is the one that maximizes the class separation mean while minimizing the standard deviation with a higher proportion of successful weight vectors. By maximizing the class separation mean, such neuron is expected to generate the most robust solutions. The minimization of the standard deviation guarantees the stability of that robustness. A high proportion of successful vectors means that, on average, less time is required to find an optimal vector in the weight space, i.e., the convergence is faster. Before using those criteria to compare the classical and quantum neurons, we first used them to select the best instance of the PCDQN that solved the problem perfectly. Among the parameter combinations that achieved an AUC ROC of 1, we report the results for the PCDQN instance that obtained the higher class separation mean. If that criterion was not enough to select the best instance, we selected the one with the lowest standard deviation among the PCDQN instances with the maximum class separation mean. The maximization of the proportion of successful weight vectors was used as the last criterion whenever necessary. Once the best PCDQN results are taken, we can discuss the performance of all neurons according to those same criteria that lead to an ideal neuron.

Table 5 benchmarks the quantum neurons through their class separation mean values obtained on the classification problems perfectly solved by those neurons. The mean values are followed by their respective standard deviations, which are reported inside the parentheses. The results are reported for the problems where at least one quantum neuron provided optimal solutions. If not all quantum neurons perfectly solved a problem, the non-optimal neurons are filled with a dash. As can be seen, quantum neurons separated the classes perfectly in seventeen of the forty classification problems addressed here previously. On the other hand, classical neurons separated the classes perfectly in only two of those classification problems, as reported in Table 6.

The first perfectly solved problem was the Diagonal blobs when targeting the class Center, where only the CVQN and the PCDQN provided optimal solutions. The PCDQN obtained a margin three times greater than the CVQN one on average. The CVQN solutions, however, deviated from the respective mean value four times less than the PCDQN solutions. When a bias was added, the mean margin of the CVQN became half of the previous margin, but the solutions became three times more stable. The PCDQN, in turn, separated the classes with two-thirds of the previous mean margin. Comparing the two

Table 5 – Benchmarking through the Separation of Classes Obtained on the Classification Problems that the Quantum Neurons Solved Perfectly

Dataset	Target	Bias	CVQN	PCDQN	CRQN
Diagonal blobs	Center	N	0.06 (±0.03)	**0.18 (±0.12)**	-
		Y	0.03 (±0.01)	**0.12 (±0.11)**	-
	Corner	N	-	**0.07 (±0.05)**	0.04 (±0.03)
		Y	-	**0.05 (±0.04)**	0.02 (±0.01)
Concentric circles	Inner	N	-	**0.18 (±0.12)**	-
		Y	0.02 (±0.01)	**0.11 (±0.09)**	-
Square clusters	XOR-like	N	-	-	**0.04 (±0.03)**
		Y	-	-	**0.03 (±0.02)**
	NXOR-like	N	**0.06 (±0.03)**	-	0.06 (±0.05)
		Y	-	-	**0.03 (±0.02)**
Gaussian quantiles	Inner	N	-	**0.03 (±0.01)**	-
		Y	-	**0.01 (±0.00)**	-
	Not Outer	N	-	**0.01 (±0.00)**	-
Iris	Setosa	N	0.03 (±0.02)	**0.09 (±0.06)**	-
		Y	0.03 (±0.02)	**0.07 (±0.05)**	0.05 (±0.04)
	Not Setosa	N	0.02 (±0.01)	**0.10 (±0.06)**	-
		Y	0.06 (±0.04)	**0.09 (±0.06)**	0.01 (±0.01)

Source: The author (2022).

The mean values of class separation followed by the standard deviations obtained by the quantum neurons are reported. A dash indicates that the respective quantum neuron did not solve the problem perfectly. The best results among the quantum neurons are highlighted in bold and also underlined if the result remains the best of all when compared with the classical neurons. The quantum neurons provided optimal solutions for seventeen out of forty classification problems. In the column named “Bias", “N" and “Y" refer to the absence and the presence of a bias, respectively.

quantum neurons with a bias, the PCDQN solutions were four times more robust on average, although the instability of the PCDQN is evident. When the labels were inverted, the CVQN no longer solved the problem, the opposite effect occurred with the CRQN, and the PCDQN continued providing optimal solutions. In that case, the PCDQN was almost two times more robust than the CRQN on average, although the PCDQN was almost two times more unstable. The bias addition decreased the PCDQN mean margin but not as much as the CRQN one that became half of before with also three times more stability than before. With the bias, the PCDQN obtained a mean margin almost three times greater than the CRQN one, although the PCDQN instability is evident again.

The Concentric circles were perfectly separated only when the target was the class

Table 6 – Benchmarking through the Separation of Classes Obtained on the Classification Problems that the Classical Neurons Solved Perfectly

Dataset	Target	Bias	LCN	LSCN	HTCN
Iris	Not Setosa	N	**0.15 (±0.1)**	0.04 (±0.02)	0.12 (±0.08)
		Y	**0.15 (±0.1)**	0.03 (±0.02)	0.07 (±0.06)

Source: The author (2022).

The mean values of class separation followed by the standard deviations obtained by the classical neurons are reported. The best results among the classical neurons are highlighted in bold and also underlined because those results remain the best of all when also compared with the quantum neurons. The classical neurons provided optimal solutions for two out of forty classification problems. In the column named “Bias", “N" and “Y" refer to the absence and the presence of a bias, respectively.

Inner, and only quantum neurons were able to achieve that complete separation of classes. Particularly, the CVQN solved the problem exclusively with a bias. On average, the CVQN solutions worked at the limit of separation, which represents a low error tolerance. Conversely, the class distances were almost six times greater with the PCDQN using a bias in mean terms. Without a bias, the PCDQN class distances were even greater. Despite the instability, only the PCDQN provided robust solutions for that problem. The problems derived from the Square clusters were exclusively solved by quantum neurons again, with only the CRQN providing optimal solutions for all those cases. The addition of a bias, however, approximated the CRQN solutions to the limit of separation between classes, independent of the positive class. The CVQN solved the problem especially when the target class was like the NXOR problem. In that case, the CVQN mean margin equaled the CRQN mean margin, although the CVQN was more stable, which makes the CVQN the best neuron for that problem. The bias addition, though, compromised sufficiently that the CVQN no longer separated the classes of that problem.

Proceeding to the problems derived from the Gaussian quantiles, only the PCDQN separated the classes when the target was the class Inner. That separation occurred in the lowest allowed margin between classes when a bias was used. Without the bias, the PCDQN solutions were more error tolerant. When the positive class was formed by the class Inner and also the class Middle, which is equivalent to all classes other than the class Outer form the positive class, the PCDQN continued solving the problem solely but only without a bias and working at the separation limit.

Regarding the Iris, the only task that the neurons accomplished perfectly was to separate the class Setosa from the other classes, independent of whether that class was the

positive or the negative class of the problem. Particularly, the classical neurons solved the problem merely when the class Setosa was the negative class. With the class Setosa as the positive class, only the CVQN and the PCDQN provided optimal solutions, with the mean margin of the PCDQN solutions being three times greater than the CVQN ones. At the same time, the CVQN solutions were three times more stable than the PCDQN ones. When a bias was added, the CVQN was not affected, the PCDQN became less error-tolerant and more stable, and the CRQN entered the competition by also providing optimal solutions. Comparing the quantum neurons, the PCDQN obtained the most robust solutions, followed by the CRQN and finally by the CVQN. The quantum neurons can be ordered in the same way from the most unstable to the most stable in that problem.

Letting the class Setosa be the negative class of the problem made the CVQN work nearly to the separation limit, while the PCDQN provided a mean margin five times greater than the CVQN one, although some instability exists. In that case, the classical neurons also separated the classes successfully. The LSCN, specifically, obtained a mean margin two times greater than the CVQN one, although the CVQN was two times more stable. The LCN and the HTCN, in turn, obtained mean margins even greater than the PCDQN one. At the same time, those two classical neurons were also more unstable. The LCN generated the most error-tolerant solutions on average, although the instability is evident. Using a bias triplicated the CVQN mean margin and allowed the CRQN to solve the problem, even though the CRQN class separation occurred at the separation limit. The CVQN also became more unstable with a bias. The use of a bias slightly reduced the PCDQN mean margin, but that neuron continued as the best neuron among the quantum ones. With a bias, the LSCN and the HTCN obtained mean margins lower than the previous ones. The HTCN also became more stable than before. Finally, the LCN was not impacted by the bias, which preserved the best results not only among the classical neurons but also when compared with the quantum ones.

This study ends by benchmarking the neurons through their proportions of successful weight vectors in Table 7. Firstly, as can be seen, the CVQN solved the Diagonal blobs when targeting the class Center with more than a quarter of the weight space. The PCDQN had about three and a half times fewer weight vectors of success in the same problem. With a bias, the CVQN proportion of successful weight vectors reduced to less than half of before, which is still larger than the PCDQN proportion. Targeting the

class Corner was a task that the CRQN solved with almost half of the weight space. The PCDQN proportion of successful weight vectors was about ten and a half times lower. The bias usage, however, impacted negatively the CRQN proportion in a way that practically equaled that low PCDQN proportion.

Table 7 – Benchmarking through the Proportion of Successful Weight Vectors Obtained on the Classification Problems that the Neurons Solved Perfectly

Dataset	Target	Bias	CVQN	PCDQN	CRQN	LCN	LSCN	HTCN
Diagonal blobs	Center	N	**0.2584**	0.0733	-	-	-	-
		Y	**0.1129**	0.0742	-	-	-	-
	Corner	N	-	0.0437	**0.4622**	-	-	-
		Y	-	0.046	**0.0565**	-	-	-
Concentric circles	Inner	N	-	**0.0423**	-	-	-	-
		Y	0.0172	**0.0289**	-	-	-	-
Square clusters	XOR-like	N	-	-	**0.1938**	-	-	-
		Y	-	-	**0.025**	-	-	-
	NXOR-like	N	**0.2482**	-	0.2476	-	-	-
		Y	-	-	**0.0344**	-	-	-
Gaussian quantiles	Inner	N	-	**0.0002**	-	-	-	-
		Y	-	**0.0007**	-	-	-	-
	Not Outer	N	-	**0.0001**	-	-	-	-
Iris	Setosa	N	**0.0911**	0.07	-	-	-	-
		Y	0.0548	0.088	**0.0915**	-	-	-
	Not Setosa	N	0.0592	**0.1794**	-	**0.2151**	0.202	0.2146
		Y	0.1142	**0.1789**	0.0253	**0.2265**	0.2096	0.2183

Source: The author (2022).

The proportions of weight vectors that allowed the neurons to solve the classification problems perfectly are reported. A dash indicates that the respective neuron did not solve the problem perfectly. The best results among the quantum neurons are highlighted in bold. The best results among the classical neurons are also highlighted in bold. The best overall results are underlined. The quantum neurons solved seventeen out of forty classification problems, while the classical neurons solved only two of them. In the column named “Bias", “N" and “Y" refer to the absence and the presence of a bias, respectively.

When targeting the class Inner of the Concentric circles, the addition of a bias made the PCDQN proportion even smaller. Nevertheless, the PCDQN proportion was almost double the CVQN proportion for the same problem with a bias. Regarding the Square clusters with the target like the XOR problem, practically a fifth of the weight space was successful when used by the CRQN. That proportion was reduced almost ten times only by the bias addition. The CVQN and the CRQN solved the problem with about a quarter of the weight space when the target was like the NXOR problem, with the CVQN

proportion slightly higher. Again, the bias addition reduced the CRQN proportion of successful weight vectors by an order of magnitude. Conversely, the bias addition increased the PCDQN proportion when targeting the class Inner of the Gaussian quantiles, even though that PCDQN proportion continued extremely low. Even more extreme was the PCDQN proportion for the Gaussian quantiles with the class Outer as the negative class. In that case, only one weight vector in the whole weight space allowed the PCDQN to solve the problem.

Finally, the neurons are contrasted by their proportions of successful weight vectors in Iris-derived problems. When targeting the class Setosa, the CVQN proportion approximated a tenth of the whole weight space. The PCDQN proportion, in turn, was inferior but close. Those CVQN and PCDQN proportions, respectively, decreased and increased with the bias addition, making the PCDQN one the highest between them. However, the PCDQN proportion with the use of a bias was not higher than the CRQN proportion, which also approximated a tenth of the weight space.

The PCDQN finally obtained the best proportion among the quantum neurons when the class Setosa was the negative class of the problem. In that case, the PCDQN proportion approximated a fifth of the weight space. That PCDQN result was three times higher than the CVQN one. On the other hand, the classical neurons obtained proportions that exceeded a fifth of the weight space. Although all classical neurons obtained similar results that are higher than the PCDQN result, the LCN solved the problem with the best proportion. When a bias was added, the CVQN almost doubled its proportion, reducing the difference from the PCDQN proportion, which continued as the best among the quantum neurons. The CRQN proportion with a bias was an order of magnitude lower than the CVQN and the PCDQN ones. The bias addition slightly improved the classical neuron proportions, with the LCN providing the best proportion not only among the classical neurons but also when compared with the quantum ones again.

In summary, those results demonstrated that different from the classical neurons that only solved two problems, the quantum neurons solved seventeen problems with robust solutions and significant proportions of successful weight vectors in most cases. Specifically, the parametrization equipped the PCDQN with the ability to also optimize the margin between classes. Before, the PCDQN optimized the AUC ROC in the majority of the cases. Here, its flexibility could be also explored in order to generate the best solutions in terms of class separation in the majority of the cases again. That quality in the

separation of classes complicated the problems even more. In this way, the proportion of successful weight vectors naturally decreased for the PCDQN solutions. In some cases, nevertheless, the PCDQN maximized not only the robustness of the solutions but also the proportion of optimal weight vectors.

In the specific problems that the classical neurons solved, their class separations and proportions of successful weight vectors were the highest. The high separations of the LCN are explained by the lack of output bounds in the linear activation. Thus, there is a propensity to have higher distances in the LCN outputs. The results also demonstrated that high mean margins are associated with more instability of the neuron in providing such robust solutions. Those results suggest that, when the mean margins increase, the successful weight space regions enlarge. As the grid search strategy causes gradual movements in the weight space, the neurons experience a variety even greater of smooth variations in the solution robustness, which increases the deviations from the mean values. Regarding the use of bias, generally, the impacts were negative, even for the CVQN and the CRQN, which took advantage of the new degree of freedom in the AUC ROC experiment. Here, focusing only on the problems perfectly solved, the bias compromised the solution robustness and the proportion of successful weight vectors of the neurons. Adding a bias in the cases already solved without it only complicates the problems, affecting negatively the neuron performance measures.

7 CONCLUSIONS

Here, we conclude by summarizing the contributions of this research as well as the corresponding implications for the state of the art. Future directions for related works are also presented.

7.1 CONTRIBUTIONS AND IMPLICATIONS

In this research, we proposed a generalized framework to build quantum neurons that apply the kernel trick. We highlight that the feature space mapping is what determines each quantum neuron under the proposed framework. Each particular case only instantiates the transformation that is applied to the input and weight vectors. It makes room to define countless quantum neurons, including for actual quantum devices, as long as the feature mappings can be implemented in quantum circuits. Feature mappings hard to compute classically may solve practical problems more efficiently, which will represent a quantum advantage achieved by that quantum neuron framework.

Under that framework, we presented a quantum neuron with a parametrized activation function and an efficient circuit implementation. That circuit implementation further complies with the constraints of actual quantum devices compared to the existing quantum neurons. Firstly, a circuit with constant depth and a linear number of operations mitigates the errors by decoherence times and gate infidelities. Qubit encoding generates a straightforward state preparation. Connectivity is not a problem since that encoding strategy generates separable states. Circuit rewriting is facilitated since the circuit uses elementary single-qubit gates. Additionally, the proposed neuron is still flexible due to the parametrization of its activation function.

Each quantum neuron implements a fundamentally different activation function. Here, we showed the neuron activation shapes as functions of some relations between the input and weight vectors. While the existing quantum neurons are restricted, the parametrized neuron that we proposed can fit different problem structures. As a first demonstration, we showed the parametrized neuron producing optimal solutions for six classification problems that an existing quantum neuron was able to solve only two of them. The proposed quantum neuron without parametrization also solved only two problems. Thus,

parametrizing really improves the neuron capabilities. We finished that initial demonstration with a feasibility experiment in a quantum simulator, where the numerical solutions were successfully reproduced. Validating those numerical solutions on simulated quantum circuits is a primary step toward taking advantage of quantum effects on actual quantum devices. There will be a moment when those devices will allow us to deal with impracticable problems for classical simulations.

Currently, however, quantum computing is still proving itself as a solid alternative to classical computing from multiple perspectives. We contributed in this direction by conducting a massive comparison between classical and quantum neurons at solving several non-linearly separable problems. Fundamentally, single classical neurons are linear classifiers. Thus, the classical neurons obtained the expected poor performances in those classification problems. On the other hand, the quantum neurons provided higher values of AUC ROC in all cases, approaching optimal performances in the majority of the problems, especially by the quantum neuron that we proposed. That benchmark demonstrated a quantum superiority with regard to neuron capabilities. Nevertheless, those limited classical neurons are the fundamental building blocks of sophisticated classical models that are responsible for the success of countless data-driven applications in multiple domains. Alternatively, due to the demonstrated quantum superiority, the application of learning models built from those quantum neurons is expected to achieve even more remarkable results on problems of practical interest.

Actually, the positive class and the negative class can be separated by a hyperplane in two problems addressed here. Exclusively, the classical neurons separated the classes of those two problems perfectly. Apart from those two problems, the quantum neurons solved fifteen problems that a hyperplane does not separate the classes in any way. In particular, the proposed quantum neuron produced the most robust solutions by maximizing the mean separation of classes. Better margins between classes can prevent errors when classifying unseen data of space regions that the training samples may not cover. Additionally, even if the training samples represent the data population, better margins provide more tolerance to the multiple sources of noise present in the real quantum hardware. However, high-quality solutions were generally obtained by a low number of weight vectors. In some cases, nevertheless, the proposed quantum neuron also maximized the proportion of successful weight vectors, which represents more global minima in the error surface. Therefore, faster convergence times can be eventually obtained in training phases.

7.2 FUTURE WORKS

Future works can discover other quantum neurons under the proposed framework by using other gates, combining gates, applying heterogeneous gates, or even evolving some evolutionary approach to learn the optimal feature mapping for a given learning task. A theoretical study can define the bounds of that framework or even prove that it is universal. Subsequent works can compare the quantum neuron performances when executed on real quantum devices. Applying the quantum neurons to practical classification problems is another possible direction since the quantum neuron framework can eventually discover practical feature mappings. Instead of a grid search strategy, the weight vectors can be adjusted by variational quantum algorithms (CEREZO et al., 2021; HUANG; TAN; XU, 2022). That variational strategy can be explored to also optimize the parameter combination in the quantum neuron proposed here, which will lead to an efficient quantum neuron with a trainable activation function shape.

A natural future step is to construct networks composed of quantum neurons under the proposed framework. However, that step brings multiple possibilities. First, different neural network architectures can be explored. For example, the quantum neurons can constitute a feedforward network. Alternatively, a recurrent structure can be obtained by using an auxiliary quantum register to store and transfer past information over the iterations (BAUSCH, 2020; TAKAKI et al., 2021). Another point to explore is the composition of neurons within the network. Networks can be formed by a specific quantum neuron only or different quantum neurons can be used in each layer. In this way, fair comparisons will require classical neural networks and other quantum neural networks in future benchmarks. Finally, the developed quantum neural network can be used in learning tasks other than classification, such as regression and time series forecasting.

REFERENCES

ACAMPORA, G.; VITIELLO, A. Implementing evolutionary optimization on actual quantum processors. *Information Sciences*, Elsevier, v. 575, p. 542–562, 2021.

ACASIETE, F.; AGOSTINI, F. P.; MOQADAM, J. K.; PORTUGAL, R. Implementation of quantum walks on ibm quantum computers. *Quantum Information Processing*, Springer, v. 19, n. 426, 2020.

ANIS, M. S.; ABBY-MITCHELL; ABRAHAM, H.; ADUOFFEI; AGARWAL, R.; AGLIARDI, G.; AHARONI, M.; AKHALWAYA, I. Y.; ALEKSANDROWICZ, G.; ALEXANDER, T.; AMY, M.; ANAGOLUM, S. et al. *Qiskit: An Open-source Framework for Quantum Computing*. 2021.

BAUSCH, J. Recurrent quantum neural networks. In: LAROCHELLE, H.; RANZATO, M.; HADSELL, R.; BALCAN, M.F.; LIN, H. (Ed.). *Advances in Neural Information Processing Systems (NeurIPS Proceedings)*. [S.l.]: Curran Associates, Inc., 2020. v. 33, p. 1368–1379.

BISHOP, C. M. *Pattern Recognition and Machine Learning*. New York, USA: Springer, 2006.

BRAVYI, S.; GOSSET, D.; KÖNIG, R. Quantum advantage with shallow circuits. *Science*, American Association for the Advancement of Science, v. 362, n. 6412, p. 308–311, 2018.

BRAVYI, S.; GOSSET, D.; KÖNIG, R.; TOMAMICHEL, M. Quantum advantage with noisy shallow circuits. *Nature Physics*, Nature Publishing Group, v. 16, p. 1040–1045, 2020.

BUHRMAN, H.; CLEVE, R.; WATROUS, J.; WOLF, R. Quantum fingerprinting. *Physical Review Letters*, APS, v. 87, n. 16, p. 167902, 2001.

CARVALHO, J. H. A.; PAULA NETO, F. M. Parametrized constant-depth quantum neuron. *arXiv preprint arXiv:2202.12496*, 2022.

CARVALHO, J. H. A.; SOUZA, L. S.; PAULA NETO, F. M.; FERREIRA, T. A. E. Impacts of multiple solutions on the lackadaisical quantum walk search algorithm. In: CERRI, R.; PRATI, R. C. (Ed.). *Intelligent Systems. BRACIS 2020*. [S.l.]: Springer, 2020. (Lecture Notes in Computer Science. LNAI, v. 12319), p. 122–135.

CARVALHO, J. H. A.; SOUZA, L. S.; PAULA NETO, F. M.; FERREIRA, T. A. E. On applying the lackadaisical quantum walk algorithm to search for multiple solutions on grids. *Information Sciences*, Elsevier, v. 622, p. 873–888, 2023.

CEREZO, M.; ARRASMITH, A.; BABBUSH, R.; BENJAMIN, S. C.; ENDO, S.; FUJII, K.; MCCLEAN, J. R.; MITARAI, K.; YUAN, X.; CINCIO, L.; COLES, P. J. Variational quantum algorithms. *Nature Reviews Physics*, Nature Publishing Group, v. 3, p. 625–644, 2021.

CESCHINI, A.; ROSATO, A.; PANELLA, M. Design of an lstm cell on a quantum hardware. *IEEE Transactions on Circuits and Systems II: Express Briefs*, IEEE, v. 69, n. 3, p. 1822–1826, 2022.

CONG, I.; CHOI, S.; LUKIN, M. D. Quantum convolutional neural networks. *Nature Physics*, Nature Publishing Group, v. 15, p. 1273–1278, 2019.

COVER, T. M. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Transactions on Electronic Computers*, IEEE, EC-14, n. 3, p. 326–334, 1965.

DALLAIRE-DEMERS, P.-L.; KILLORAN, N. Quantum generative adversarial networks. *Physical Review A*, APS, v. 98, n. 1, p. 012324, 2018.

DEVITT, S. J.; MUNRO, W. J.; NEMOTO, K. Quantum error correction for beginners. *Reports on Progress in Physics*, IOP Publishing, v. 76, n. 7, p. 076001, 2013.

DING, C.; BAO, T.-Y.; HUANG, H.-L. Quantum-inspired support vector machine. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 33, n. 12, p. 7210–7222, 2022.

GRANT, E.; BENEDETTI, M.; CAO, S.; HALLAM, A.; LOCKHART, J.; STOJEVIC, V.; GREEN, A. G.; SEVERINI, S. Hierarchical quantum classifiers. *npj Quantum Information*, Nature Publishing Group, v. 4, n. 65, 2018.

HAVLÍČEK, V.; CÓRCOLES, A. D.; TEMME, K.; HARROW, A. W.; KANDALA, A.; CHOW, J. M.; GAMBETTA, J. M. Supervised learning with quantum-enhanced feature spaces. *Nature*, Nature Publishing Group, v. 567, p. 209–212, 2019.

HAYKIN, S. *Neural Networks and Learning Machines*. New Jersey, USA: Pearson, 2009.

HUANG, R.; TAN, X.; XU, Q. Learning to learn variational quantum algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, 2022. Early Access.

KANDALA, A.; TEMME, K.; CÓRCOLES, A. D.; MEZZACAPO, A.; CHOW, J. M.; GAMBETTA, J. M. Error mitigation extends the computational reach of a noisy quantum processor. *Nature*, Nature Publishing Group, v. 567, p. 491–495, 2019.

KONAR, D.; BHATTACHARYYA, S.; PANIGRAHI, B. K.; BEHRMAN, E. C. Qutrit-inspired fully self-supervised shallow quantum learning network for brain tumor segmentation. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 33, n. 11, p. 6331–6345, 2022.

LAROSE, R. Overview and comparison of gate level quantum software platforms. *Quantum*, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 3, p. 130, 2019.

LEYMANN, F.; BARZEN, J. The bitter truth about gate-based quantum algorithms in the nisq era. *Quantum Science and Technology*, IOP Publishing, v. 5, n. 4, p. 044007, 2020.

LI, Y.; BENJAMIN, S. C. Efficient variational quantum simulator incorporating active error minimization. *Physical Review X*, APS, v. 7, n. 021050, 2017.

LIU, Y.; ARUNACHALAM, S.; TEMME, K. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, Nature Publishing Group, v. 17, p. 1013–1017, 2021.

MANGINI, S.; TACCHINO, F.; GERACE, D.; MACCHIAVELLO, C.; BAJONI, D. Quantum computing model of an artificial neuron with continuously valued input data. *Machine Learning: Science and Technology*, IOP Publishing, v. 1, n. 045008, 2020.

MARONESE, M.; DESTRI, C.; PRATI, E. Quantum activation functions for quantum neural networks. *arXiv preprint arXiv:2201.03700*, 2022.

MARONESE, M.; PRATI, E. A continuous rosenblatt quantum perceptron. *International Journal of Quantum Information*, World Scientific, v. 19, n. 04, p. 2140002, 2021.

NGUYEN, N. H.; BEHRMAN, E. C.; MOUSTAFA, M. A.; STECK, J. E. Benchmarking neural networks for quantum computations. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 31, n. 7, p. 2522–2531, 2020.

NIELSEN, M. A.; CHUANG, I. L. *Quantum Computation and Quantum Information*. New York, USA: Cambridge University Press, 2010.

PAULA NETO, F. M.; LUDERMIR, T. B.; OLIVEIRA, W. R.; SILVA, A. J. Implementing any nonlinear quantum neuron. *IEEE Transactions on Neural Networks and Learning Systems*, IEEE, v. 31, n. 9, p. 3741–3746, 2020.

PEDREGOSA, F.; VAROQUAUX, G.; GRAMFORT, A.; MICHEL, V.; THIRION, B.; GRISEL, O.; BLONDEL, M.; PRETTENHOFER, P.; WEISS, R.; DUBOURG, V.; VANDERPLAS, J.; PASSOS, A. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, n. 85, p. 2825–2830, 2011.

PRESKILL, J. Quantum computing and the entanglement frontier. *arXiv preprint arXiv:1203.5813*, 2012.

PRESKILL, J. Quantum computing in the nisq era and beyond. *Quantum*, Verein zur Förderung des Open Access Publizierens in den Quantenwissenschaften, v. 2, p. 79, 2018.

ROSSI, M.; HUBER, M.; BRUß, D.; MACCHIAVELLO, C. Quantum hypergraph states. *New Journal of Physics*, IOP Publishing, v. 15, n. 113022, 2013.

SCHULD, M.; KILLORAN, N. Quantum machine learning in feature hilbert spaces. *Physical Review Letters*, APS, v. 122, n. 4, p. 040504, 2019.

SCHULD, M.; SWEKE, R.; MEYER, J. J. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, APS, v. 103, n. 032430, 2021.

SHENDE, V. V.; BULLOCK, S. S.; MARKOV, I. L. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 25, n. 6, p. 1000–1010, 2006.

SOUZA, L. S.; CARVALHO, J. H. A.; FERREIRA, T. A. E. Lackadaisical quantum walk in the hypercube to search for multiple marked vertices. In: BRITTO, A.; DELGADO, K. V. (Ed.). *Intelligent Systems. BRACIS 2021*. [S.l.]: Springer, 2021. (Lecture Notes in Computer Science. LNAI, v. 13073), p. 249–263.

SOUZA, L. S.; CARVALHO, J. H. A.; FERREIRA, T. A. E. Classical artificial neural network training using quantum walks as a search procedure. *IEEE Transactions on Computers*, IEEE, v. 71, n. 2, p. 378–389, 2022.

STOUDENMIRE, E. M.; SCHWAB, D. J. Supervised learning with tensor networks. In: LEE, D.; SUGIYAMA, M.; LUXBURG, U.; GUYON, I.; GARNETT, R. (Ed.). *Advances in Neural Information Processing Systems (NeurIPS Proceedings)*. [S.l.]: Curran Associates, Inc., 2016. v. 29.

TACCHINO, F.; BARKOUTSOS, P.; MACCHIAVELLO, C.; TAVERNELLI, I.; GERACE, D.; BAJONI, D. Quantum implementation of an artificial feed-forward neural network. *Quantum Science and Technology*, IOP Publishing, v. 5, n. 4, p. 044010, 2020.

TACCHINO, F.; MACCHIAVELLO, C.; GERACE, D.; BAJONI, D. An artificial neuron implemented on an actual quantum processor. *npj Quantum Information*, Nature Publishing Group, v. 5, n. 26, 2019.

TAKAKI, Y.; MITARAI, K.; NEGORO, M.; FUJII, K.; KITAGAWA, M. Learning temporal data with a variational quantum recurrent neural network. *Physical Review A*, APS, v. 103, n. 052414, 2021.

TEMME, K.; BRAVYI, S.; GAMBETTA, J. M. Error mitigation for short-depth quantum circuits. *Physical Review Letters*, APS, v. 119, n. 180509, 2017.

YANOFSKY, N. S.; MANNUCCI, M. A. *Quantum Computing for Computer Scientists*. New York, USA: Cambridge University Press, 2008.

ZOUFAL, C.; LUCCHI, A.; WOERNER, S. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, Nature Publishing Group, v. 5, n. 103, 2019.

ZULEHNER, A.; PALER, A.; WILLE, R. An efficient methodology for mapping quantum circuits to the ibm qx architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, IEEE, v. 38, n. 7, p. 1226–1236, 2019.