UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Industriales

Automatic Generation of Optimized Quantum Circuits for Real-world Machine Learning Applications

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Sergio Altares López
MSc. in Business Analytics and Big Data

Madrid, 2025

UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Industriales

Doctoral Degree in Automatic Control and Robotics

Automatic Generation of Optimized Quantum Circuits for Real-world Machine Learning Applications

D O C T O R A L T H E S I S

Submitted for the degree of Doctor by:

Sergio Altares López
MSc. in Business Analytics and Big Data

Under the supervision of:
Dr. Ángela Ribeiro Seijas
Dr. Juan José García Ripoll

Madrid, 2025

Title: Automatic Generation of Optimized Quantum Circuits for Real-world Machine Learning Applications

Author: Sergio Altares López

Doctoral Programme: Automatic Control and Robotics

Thesis Supervision:

Dr. Ángela Ribeiro Seijas, Scientist researcher, Centre for Automation and Robotics (CAR), Consejo Superior de Investigaciones Científicas (CSIC).

Dr. Juan José García Ripoll, Scientist researcher, Instituto de Física Fundamantal (IFF), Consejo Superior de Investigaciones Científicas (CSIC).

External Reviewers:

Thesis Defense Committee:

Thesis Defense Date:

This thesis has been supported by Ministerio de Asuntos Economicos y Transformacion Digital, project - MIA.2021.M01.0004.

To my parents, M^{a}Paz and Luis Manuel...

A mis padres, M^{a}Paz y Luis Manuel...

Acknowledgement

Because dreams come true when pursued tirelessly, because curiosity is the true engine that moves this senseless universe, because where there's a will, there's a way, because we are the sum of all the good and bad that has happened to us in life, and the present is nothing more than a simple differential of one of the possible uncertain futures of this universe.

Because impossible is just an opinion, because things that were possible, at first seemed impossible, because the word *error* should be replaced in the dictionary for *learning*, because, paraphrasing Dr. Prof. A. Einstein, *Everything is relative*, and paraphrasing Dr. Prof. S. Hawking, *Intelligence is the ability to adapt to changes*. **THANKS TO ALL THE PEOPLE WHO HAVE HELPED ME TO HAVE THIS THINKING WAY**.

A mis padres, por su apoyo, paciencia, confianza y aguante. Sin ellos, esto habría sido **IMPOSIBLE**. No hay palabras para agradeceros tanto.

GRACIAS a mi **madre**, *Mª Paz López Iglesias*, por haber sabido transmitirme los mejores valores, ética y educación, pilares fundamentales en mi vida. Sin ellos, el mayor de los conocimientos quedaría vacío. Gracias por tu paciencia infinita, por tus sacrificios diarios, y por estar siempre ahí, no solo como guía, sino también como ejemplo de fortaleza y bondad. Cada paso que doy en mi camino está inspirado por todo lo que me has enseñado, y tus palabras de apoyo, siempre llenas de sabiduría, me han dado fuerzas en los momentos más difíciles. Por todo lo que eres y todo lo que haces, gracias desde lo más profundo de mi corazón.

GRACIAS a mi **padre**, *Luis Manuel Altares Sampedro*, por haber sembrado en mí, desde pequeño, la curiosidad, el asombro y el amor por la ciencia. Tu manera de explicar el mundo, con paciencia y entusiasmo, encendió en mí esa chispa que hoy sigue guiando mi camino. Gracias por incentivarme siempre a hacer preguntas, a no conformarme con respuestas simples, y a buscar siempre más allá. Has sido un ejemplo constante de dedicación, pasión y rigor, y esos valores me han acompañado en cada etapa de mi vida. Por todas las conversaciones, las enseñanzas y el apoyo incondicional, gracias de corazón.

Aunque no suelo expresar mis sentimientos con palabras, quiero aprovechar este documento, de forma que quede grabado para siempre: *os quiero*.

A mis abuel@s: *Angelines, Pacita, Antonio* y *Manuel*, porque siempre que estoy perdido, mirar hacia atrás y recordar de dónde vengo me da la claridad para avanzar hacia dónde quiero ir. Gracias. También quiero agradecer al resto de mi familia su apoyo, especialmente a mi *tío Juanqui* y a mi *tía Mari*, por estar siempre ahí en los momentos importantes.

GRACIAS a mis amig@s, mi otra *familia*, *Leidy Arrieta, Irene Molina, Eduardo M. Jara Galán, Miriam Sánchez, Alberto R. Rodríguez, Carlos Gilarranz, Enrique Bocanegra, Miriam Pareja* y *Edwin X. Naranjo*, entre otros, por estar ahí siempre que les he necesitado, la paciencia de escucharme y por los innumerables y sabios consejos, que algún día terminaré de entender y poner en práctica. Como suelo decir, *algún día aprenderé; mientras tanto, seguiré aprendiendo...* Tambien quiero dar las gracias a *Sara Castillejo*, porque gracias a ella mis meses de estancia en Munich han sido aún más especiales. Por todos los viajes, schnitzels

y, sobre todo, por todos los momentos compartidos. De verdad, no podría sentirme más afortunado de teneros en mi vida.

GRACIAS a mis compañer@s y ya amig@s del grupo de percepción artificial (GPA) del CAR-CSIC donde he realizado la presente tesis doctoral. Por las risas, los cafés con churros y muchas pizarras llenas de ideas.

I would especially like to thank my two directors, *Dr. Ángela Ribeiro Seijas* and *Dr. Juan José García Ripoll*, two people I admire both academically and personally, for giving me the great opportunity to develop this doctoral thesis under their expert guidance, introducing me to the field of research, and guiding me on this exciting journey that this work has been for me, Friday after Friday in our weekly meetings at 16:00, and which started with a simple card. I will miss our weekly meetings!

I would like to express my sincere gratitude to the AgrarIA Project: Artificial Intelligence Applied to the Chain of Agricultural Production 2050 (MIA.2021.M01.0004), funded by the Ministry for Digital Transformation. Their support has been essential for developing this work, advancing knowledge, and applying artificial intelligence in agriculture. I would also like to thank IQM Quantum Computers in Munich, Germany, for hosting me during my stay, particularly to *Inés, Tom, Jirka, Mario,* and *Martin*. Vielen Dank!

Lastly, and certainly not less important, I would like to express my gratitude to **ALL** the scientists who have dedicated their lives to advancing science and technology, without whom our current society would not be possible. Thank you.

This dream has just begun...

Abstract

Quantum machine learning combines principles of quantum computing and machine learning (a subdiscipline of artificial intelligence), to leverage the capabilities of quantum systems for efficient data processing and analysis. Quantum machine learning models are often based on quantum circuits, which consist of sequences of operations applied to qubits or quantum bits. However, the design of these circuits is challenging given the limitations of these models in handling large volumes of data and the size of the circuits. These large circuits can lead to inefficiencies during the training process, while smaller circuits often have expressivity issues and may not capture the complexities of the task at hand. Consequently, identifying the optimal balance between a circuit's trainability and its expressive power represents a critical challenge in quantum machine learning.

This research focuses on the design of quantum algorithms applied to machine learning, creating a new global optimization framework for generating quantum circuit-based models using multi-objective genetic algorithms. The techniques developed allow the automatic generation of optimal quantum circuits, offering full flexibility in selecting structure, operators, and parameters and considering critical factors such as accuracy and model size. The use of a gradient-free optimization method avoids common problems such as *barren plateaus*. Using the methodology proposed in this thesis, several experiments and use cases of increasing complexity in terms of input data types are carried out. These experiments cover applications as diverse as binary and multi-class classification of tabular data, grayscale image classification, multiband image segmentation, and time-series based prediction of satellite imagery from historical data sets. In each of these applications, processing strategies are developed that enable quantum models to achieve very promising results, which underlines the versatility and robustness of the proposed framework.

The quantum models discovered by this methodology stand out for their high representational capacity with minimal use of qubits, which directly addresses the challenges inherent in gradient-based training methods, especially when applied to higher dimensionality data. This approach reduces common scalability and efficiency limitations, promoting stable performance even for large models. In addition, the framework drives the discovery of circuits with high expressivity and reduced correlation, thus laying the foundation for creating machine learning models inspired by quantum computing as classically simulatable models that can be used in both academia and industry.

Resumen

El aprendizaje automático cuántico combina principios de la computación cuántica y el aprendizaje automático (una subdisciplina de la inteligencia artificial), con el objetivo de aprovechar las capacidades de los sistemas cuánticos para el procesamiento y análisis eficiente de datos. Los modelos de aprendizaje automático cuántico suelen basarse en circuitos cuánticos, que consisten en secuencias de operaciones aplicadas a qubits o bits cuánticos. Sin embargo, el diseño de estos circuitos es un reto dadas las limitaciones de estos modelos a la hora de manejar grandes volúmenes de datos y el tamaño de los circuitos. Estos circuitos de gran tamaño pueden dar lugar a ineficiencias durante el proceso de entrenamiento, mientras que los circuitos más pequeños suelen tener problemas de expresividad y pueden no captar las complejidades de la tarea en cuestión. En consecuencia, identificar el equilibrio óptimo entre la entrenabilidad de un circuito y su poder expresivo representa un reto crítico en el campo del aprendizaje automático cuántico.

Esta investigación se centra en el diseño de algoritmos cuánticos aplicados al aprendizaje automático, creando un nuevo marco de optimización global para generar modelos basados en circuitos cuánticos mediante algoritmos genéticos multiobjetivo. Las técnicas desarrolladas permiten la generación automática de circuitos cuánticos óptimos, ofreciendo total flexibilidad en la selección de estructura, operadores y parámetros y considerando factores críticos como la precisión y el tamaño del modelo. El uso de un método de optimización sin gradiente evita problemas comunes como las *mesetas estériles*. Con la metodología propuesta en esta tesis, se llevan a cabo diversos experimentos y casos de uso de complejidad creciente en cuanto a los tipos de datos de entrada. Estos experimentos abarcan aplicaciones tan diversas, como la clasificación binaria y multiclase de datos tabulares, la clasificación de imágenes en escala de grises, la segmentación de imágenes multibanda y la predicción basada en series temporales de imágenes de satélite a partir de conjuntos de datos históricos. En cada una de estas aplicaciones se desarrollan estrategias de procesamiento que permiten a los modelos cuánticos alcanzar resultados muy prometedores, lo que subraya la versatilidad y robustez del marco propuesto.

Los modelos cuánticos descubiertos por esta metodología, destacan por su alta capacidad de representación con un uso mínimo de qubits, lo que aborda directamente los retos inherentes a los métodos de entrenamiento basados en gradientes, especialmente cuando se aplican a datos de mayor dimensionalidad. Este enfoque reduce las limitaciones comunes de escalabilidad y eficiencia, promoviendo un rendimiento estable incluso para modelos de gran tamaño. Además, el marco impulsa el descubrimiento de circuitos con alta expresividad y una correlación reducida, sentando así las bases para crear modelos de aprendizaje automático inspirados en la computación cuántica como modelos clásicamente simulables que puedan ser utilizados tanto en la academia como en la industria.

Table of Contents

List of Figures

List of Tables

Abbreviations and acronyms

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
RGB	RED, Green and Blue
NISQ	Near Term Intermediate-Scale Quantum computing
QML	Quantum Machine Learning
GA	Genetic Algorithms
SVM	Support Vector Machine
QSVM	Quantum Support Vector Machine
ANN	Artificial Neural Network
AE	Autoencoder
CNN	Convolutional Neural Network
CAE	Convolutional Autoencoder
VQC	Variational Quantum Circuit
PQC	Parameterized Quantum Circuit
MLP	Multilayer Perceptron
PCA	Principal Components Analysis
HQNN	Hybrid Quantum Neural Network
QNN	Quantum Neural Network
NDVI	Normalized Difference Vegetation Index
NAS	Neural Architecture Search
CNOT	Controlled-NOT
H	Hadamard

Chapter 1

Introduction

1.1 Motivation

Quantum computing combines computer science, physics, and mathematics, exploring using quantum physical systems to store, communicate, and process information. It emerges as a response to the inherent limitations of classical computing, leveraging the principles of quantum physics that govern the behavior of subatomic particles. While technology advances, classical devices, which operate according to the classical physics laws, face fundamental constraints regarding miniaturization and efficiency in solving complex problems. Richard Feynman, one of the godfathers of quantum computing, stated a fundamental problem (Feynman, 1982): classical computers struggled when simulating quantum systems because of their bit-based structure. He argued that since quantum physics fundamentally differs from classical physics, a computational approach was needed to reflect this difference. He expressed that a computer based on quantum mechanical principles could overcome the limitations of classical machines and take advantage of quantum phenomena to perform more efficient simulations. In 1985, David Deutsch advanced this fundamental idea by developing the concept of a *universal quantum machine* (Deutsch, 1985). He demonstrated theoretically that a quantum machine, using quantum physics principles, could perform calculations far more complex than any classical computer, thus laying the foundation for general quantum computing.

These ideas led to the development of new quantum algorithms, such as the one developed by the mathematician Peter Shor (Shor, 1994) for integer factorization and the discrete logarithm problem that demonstrated that quantum computers could perform these calculations exponentially more efficiently than any Turing machine (Turing, 1936). This breakthrough showed that quantum computers could tackle problems that, even with today's technologies, remain intractable for classical machines due to time and resource constraints. At the same time, Benjamin Schumacher formulated a principle similar to Shannon's noiseless coding theorem in classical information theory (Shannon, 1948), but the equivalent with noise remains unsolved. He refers to the term for the basic information units of quantum computation: *quantum-bit* or *qubit* (Schumacher, 1995). In the 1990s, research continued to demonstrate the advantages of quantum systems over classical ones, as evidenced by the algorithm presented

by Lov Grover (Grover, 1996) for searching unstructured databases, which offers a quadratic advantage over classical search methods. This breakthrough provided evidence that quantum computers can significantly speed up certain types of computations, thus broadening the scope of quantum computing and demonstrating its ability to improve efficiency in various practical problems beyond factorization. These milestones led many researchers to explore further quantum algorithms, such as the HHL algorithm developed by Harrow, Hassidim, and Lloyd (Harrow et al., 2009), which demonstrates that quantum computers can solve systems of linear equations exponentially faster than classical methods.

The progress in quantum algorithm development has motivated a strong effort in designing and fabricating quantum computing hardware, which still exhibits an exponential trend. This advancement is manifested in the increasing number of qubits in quantum computers, currently ranging from 27 to 443 qubits, pushing the limits of scalability. Simultaneously with the development of more advanced and higher-capacity quantum devices, John Preskill proposed a new paradigm known as NISQ (Near-Term Intermediate-Scale Quantum) computing to leverage existing quantum systems (Preskill, 2018). Although they currently do not surpass classical computers, they are valuable for tackling real-world problems and pushing forward algorithm research in essential areas like optimization, simulation of physical systems, or quantum chemistry, even with noise and limited qubits.

In recent years, the field of *Artificial Intelligence* (AI) research has experienced an exponential surge, in parallel with advances in quantum technology (LeCun et al., 2015). In this context, we find *Machine Learning* (ML), a branch of AI focused on developing algorithms that enable computers to learn from data and make data-driven decisions (Bishop, 2006). ML enables computers to improve their performance on specific tasks by analyzing large amounts of data without being explicitly programmed. Building on this, *Quantum Machine Learning* (QML) merges quantum computing principles with these machine learning techniques (Biamonte et al., 2017; Schuld et al., 2017). It transitions machine learning models to a quantum computing framework, utilizing *qubits* to store information and *quantum gates* to manipulate it, providing an alternative method for data processing. By utilizing quantum mechanical phenomena such as *superposition* and *entanglement*, QML has the potential to perform computations that may exceed the capabilities of classical computing methods. Central to QML are *quantum circuits*, which represent computational models for executing advanced quantum algorithms. These circuits aim to enhance efficiency and improve data processing capabilities for machine learning tasks.

When considering a machine learning model, it is important to consider the *expressiveness*, which is the ability to represent complex functions, and the *trainability*, which shows how efficiently the model parameters can be adjusted to optimize performance. The literature suggests that classical machine learning models adequately balance model size, expressiveness, and trainability. In contrast, quantum machine learning models are typically moderate in size and do not have the same robustness in training as classical models. It results in specific challenges, such as *vanishing gradient* and *barren plateaus*, which can make the optimization and training process considerably more problematic (Arrasmith et al., 2021; Cerezo et al., 2021). Finding a balance between expressivity and trainability in QML models is particularly challenging. Excessively complex quantum models can be complicated to train effectively,

resulting in suboptimal performances. On the other hand, more straightforward may not be able to capture complex patterns in the data, which also negatively affects the global performance (Holmes et al., 2022; McClean et al., 2018).

The literature has dealt with the QML model adaptation problem from different approaches. On the one hand, a strategy based on the specific design of hardware-adapted circuits has been developed, where simulatable circuits are not subject to optimization, which may result in *barren plateaus* effects. On the other hand, there are strategies based on classical methods for the adaptation and optimization of these circuits, such as adaptive initialization (Grant et al., 2019), circuit pruning (Sim et al., 2021a), and the use of density matrices and random features for approximating distributions (González et al., 2021). These strategies provide partial solutions, but no global strategy covers the problem.

These circuit design issues are more challenging when applied to large data sets and high-dimensional problems like images. Researchers in classical machine learning tackle these problems using deep learning techniques, such as convolutional networks, without imposing size constraints. One crucial challenge is to explore whether some of the ideas and methods of QML can be effectively extended and adapted to this particular domain. Some approaches to addressing this problem are the development of quantum convolutional neural networks (Cong et al., 2019), hybrid classical-quantum systems using classical and quantum networks (J. Liu et al., 2021), or *transfer learning* applying pre-trained classical models to the data as preprocessing before embedding the feature vector in quantum neural networks (Mari et al., 2020).

This thesis aims to tackle the problem of creating quantum machine learning models in domains of practical utility and increasing complexity. For this purpose, we address two main challenges facing QML: (i) the automatic design of quantum algorithms for quantum machine learning considering all its components and (ii) the specialization of QML algorithms to handle different data sets of varying data types, enabling their applications in real-world scenarios. It requires developing optimization tools that allow the design of optimal models that maximize expressiveness while ensuring adequate trainability.

1.2 Objectives and Scope

The present research aims to advance the field of quantum machine learning by addressing several essential challenges related to the design of quantum circuits, *barren plateaus* issues arising in the training phase, and applications to high-dimensional data in classification and regression tasks.

- **O1. Design of quantum circuits:** Develop a global optimization strategy that allows the automatic generation of ad-hoc quantum circuits for real quantum machine learning applications.
- **O2. Addressing *barren plateaus* issues:** Implement strategies to mitigate *barren plateaus* during the training phase of quantum circuits, utilizing advanced parameterization techniques.

- **O3. Applications to high-dimensional data:** Design, develop, and validate strategies for applying quantum circuits for high-dimensional data processing, explore quantum models in single-channel classification, multiband image segmentation, and time series modelling for satellite images.
- **O4. Reduction of the complexity of quantum models:** In terms of operators, achieving quantum-inspired models with low complexity and possibly being programmed in classical computers: classically simulatable models.
- **O5. Real-world applications and use cases:** Show the use of quantum computing in practical applications such as crop yield prediction, medical image analysis, and environmental monitoring. Evaluate quantum-classical hybrid approaches for classification, regression, and forecasting tasks.

Thus, this research work contributes to advancing quantum machine learning by addressing key challenges and exploring innovative applications of quantum machine learning models, considering efficiency, accuracy, and scalability to solve real-world problems. Also, this research investigates several aspects of quantum machine learning, including the design of optimized circuits to reduce current training issues such as *barren plateaus*, the development of novel parameterization techniques, and the exploration of quantum-classical hybrid approaches for solving classification and prediction tasks.

1.3 Contributions

1.3.1 Publications Directly Related

- **C1. Altares-López, Sergio**, Angela Ribeiro, and Juan José García-Ripoll. *Automatic design of quantum feature maps.* Quantum Science and Technology 6.4 (2021): 045015. DOI: https://doi.org/10.1088/2058-9565/ac1ab1.
- **C2. Altares-López, S.**, García-Ripoll, J. J., & Ribeiro, A. (2023). *AutoQML: Automatic generation and training of robust quantum-inspired classifiers using evolutionary algorithms on grayscale images.* Expert Systems with Applications, 122984. DOI: https://doi.org/10.1016/j.eswa.2023.122984.
- **C3. Altares-López, S.**, García-Ripoll, J. J., & Ribeiro, A. (2024). *Optimal quantum circuit generation for pixel segmentation in multiband images.* Applied Soft Computing, 112175. DOI: https://doi.org/10.1016/j.asoc.2024.112175.
- **C4. Altares-López, Sergio**, Juan José García-Ripoll, and Angela Ribeiro. *Hybrid quantum-classical neural networks for crop yield prediction.*
- **C5. Altares-López, Sergio**, Juan José García-Ripoll, and Angela Ribeiro. *Automatic design of quantum feature maps.* 2nd European Quantum Technologies Virtual Conference (EQTC). Dublin, Ireland (2021). Poster Presented.

1.3.2 Publications Partially Related

- **C6. Altares-López, Sergio**, Juan José García-Ripoll, and Angela Ribeiro. *Robótica cuántica: Elementos principales.* National Robotics and Bioengineering Conference, Madrid (2023). Presentation. DOI: https://doi.org/10.20868/UPM.book.74896, (Altares-López et al., 2023b).

1.3.3 Software

- **C7. Registered Software.** *Auto-Generated Quantum-Inspired Kernels by Using Genetic Algorithms (AQUIK)* - CSIC 929840. May 15th 2023. **Altares-López, S.**, García-Ripoll J.J., Ribeiro, A. https://github.com/sergio94al/Automatic_design_of_quantum_feature_maps_Genetic_Auto-Generation, (Altares-López et al., 2022).
- **C8. Code Ocean - Reproducibility Badge. Sergio Altares-López**, Juan José García-Ripoll, Angela Ribeiro (2023) *AutoQML: Quantum Inspired Kernels by Using Genetic Algorithms for Grayscale Images*, https://doi.org/10.24433/co.3955535.v1.

Research Projects

- **AgrarIA.** Artificial intelligence applied to the agricultural production value chain 2050 has been funded by the Ministry for Digital Transformation and Civil Service through the R&D Missions in Artificial Intelligence 2021 Program, within the framework of the Spain Digital Agenda 2025 and the National Artificial Intelligence Strategy, with European funding through the Recovery, Transformation and Resilience Plan. This competitive project seeks to use Artificial Intelligence and other Industry 4.0 technologies to improve Spanish agriculture. It is composed of 24 public-private partners. Project - MIA.2021.M01.0004.

1.4 Thesis Structure

This document is organized into six chapters: one that introduces basic concepts and methodologies of quantum computing and quantum machine learning, four chapters that describe the original contributions of the research, and a final chapter with general conclusions and lines of future work.

Chapter 2. Quantum Machine Learning Fundamentals: This chapter describes basic concepts of quantum computing, such as qubits, quantum operators, and quantum circuits, to define quantum machine learning methods.

Chapter 3. Generation of Optimal Quantum Circuits for Tabular Data: This chapter introduces the global optimization method proposed that uses genetic algorithms to generate quantum circuits for automatic tabular data classification. It also includes an introduction to genetic algorithms and their operators.

Chapter 4. Quantum Circuits for Single-channel Image Classification: This chapter describes the proposed adaptation of the above method to address the classification

of grayscale images. These techniques are applied to medical image datasets.

Chapter 5. Quantum Pixel Segmentation in Multiband Images: This chapter explains the new method developed for processing multiband images using quantum circuits. The primary emphasis of this work is on the segmentation task. The method is applied to vegetation cover and solar panel identification.

Chapter 6. Hybrid Neural Networks for Processing Sequential Satellite Images: It presents the method developed for using hybrid models to predict continuous values from satellite image sequences. It details a double optimization step for parameter tuning.

Chapter 7. Conclusions: This chapter presents the key findings of this original research and discusses them in the current state of the art, highlighting how they contribute to existing knowledge.

Chapter 2

Quantum Machine Learning Fundamentals

This chapter provides an introduction to the fundamental principles of quantum computation and the application of quantum machine learning methods. It defines qubits and quantum states, the fundamental units of information in quantum computing, and how they can be transformed by universal operators and sets of operations essential for developing algorithms. It also discusses the architecture of quantum circuits, which combine gates and measurements to manipulate information, and introduces the concept of variational quantum circuits. Finally, two essential methods of quantum machine learning are explored: quantum support vector machines and quantum neural networks, showing their potential for solving complex problems.

2.1 Qubits and Quantum States

In Boolean computation, the most basic unit of information is the *bit*, a binary integer with two possible and mutually exclusive states, 0 or 1. Analogous to this classical bit, a *quantum-bit* or *qubit* is the elemental unit of quantum information processing in a quantum system (Nielsen and Chuang, 2002). A qubit is represented as a vector in a two-dimensional Hilbert space $\mathcal{H}$, providing the framework for describing quantum states. For a qubit, this Hilbert space has two distinctive orthogonal basis states, which can be expressed as,

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.1}$$

Unlike classical bits, which can only be in one of two possible states (see Figure 2.1a), qubits have properties derived from *quantum mechanics*. One of these features is *quantum superposition*, which allows a qubit to exist simultaneously in multiple states, in any possible combination of basis states as,

$$|\psi\rangle = \psi_0 |0\rangle + \psi_1 |1\rangle . \tag{2.2}$$

In this equation, the amplitudes $|\psi_0|$ and $|\psi_1|$ are complex values and are normalized, satisfying $|\psi_0|^2 + |\psi_1|^2 = 1$. The values $|\psi_0|^2$ and $|\psi_1|^2$ correspond to the probabilities of detecting the states $|0\rangle$ and $|1\rangle$, respectively.

General quantum states with two or more qubits are constructed as tensor product representations of the Hilbert spaces of individual qubits. For instance, a general state of two qubits is a linear combination of the 2^2 basis states,

$$|\psi\rangle = \psi_{00}\,|00\rangle + \psi_{01}\,|01\rangle + \psi_{10}\,|10\rangle + \psi_{11}\,|11\rangle\,. \tag{2.3}$$

As before, $|\psi_{mn}|^2$ represents the probability of obtaining the bits m and n, where m and n in $\{0, 1\}$, by measuring the both qubits. The sum of all these probabilities is normalized to one $\sum_{mn} |\psi_{mn}|^2 = 1$. In these systems with two or more qubits, another fundamental property of quantum mechanics appears: *quantum entanglement*. This quantum feature allows qubits to establish dependencies between them, such that the state of one directly affects the state of the other. These interdependent relations lead to strong correlations that can not be found in classical systems.

Due to the composition of Hilbert spaces, the number of complex coefficients used to represent a quantum state with N qubits grows exponentially as 2^N. On the one hand, this implies that a general-purpose quantum computer with 40 or more qubits cannot be classically simulated on any supercomputer. On the other hand, this relation suggests that large feature spaces can be used to implement sophisticated algorithms of *machine learning*.

The combination of superposition, entanglement and the exponential growth of Hilbert space provides a significant advantage in information processing, opening new perspectives in developing a new computing paradigm.

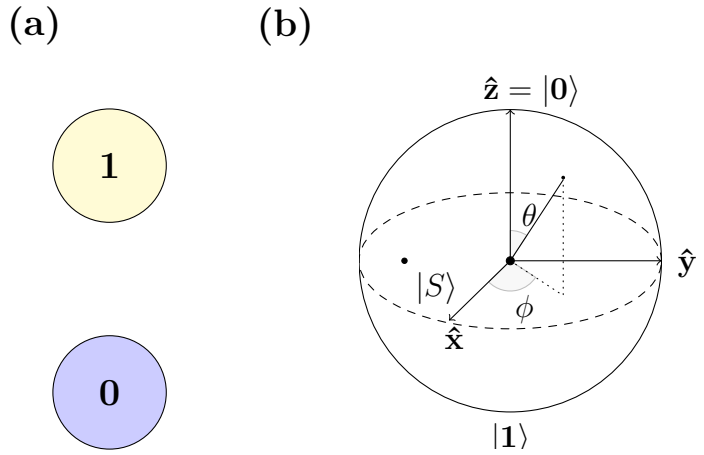


Figure 2.1: (a) Classical bit states 0 and 1. (b) Bloch sphere representation. The $|S\rangle$ state is a superposition state.

The *Bloch sphere* is a three-dimensional model frequently used to represent pure quantum states (Bloch, 1946) (see Figure 2.1b). In this model, each point on the sphere corresponds

to a unique quantum state. The Bloch sphere provides an intuitive way to understand how the states of a single qubit change under different quantum operations. However, this representation is not applicable to generic quantum states such as multiqubit or higher dimensional systems.

The state of a qubit is described by two angles: θ, which is the polar angle that varies from 0 to π and ϕ, the azimuthal angle that varies from 0 to 2π. These angles define the position of the qubit state on the sphere as,

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,. \tag{2.4}$$

Here, θ controls the relative amplitude between the $|0\rangle$ and $|1\rangle$ states, while ϕ determines the relative phase between them. In the Bloch sphere, basis states $|0\rangle$ and $|1\rangle$ correspond to the north and south pole, respectively, as shown in Figure 2.1b. Quantum superpositions of a qubit's states can be visualized as points on the surface of the Bloch sphere that are not at the poles, such as the state $|S\rangle$ depicted in Figure 2.1. This $|S\rangle$ state is equidistant from the north and south poles, indicating that the probability of measuring $|0\rangle$ or $|1\rangle$ is equal, $(\frac{1}{\sqrt{2}})^2$.

2.2 Quantum Gates

Logic gates are the basic tool of Boolean logic to decompose arbitrary computations in terms of operators that are easily implementable with hardware. These operators are irreversible since it is impossible to determine the original input state from its output state. The most general logic operator in Boolean logic is the NAND gate, which can be used to construct any other logic gate, making it a universal operator.

By analogy, arbitrary computations can be decomposed using *quantum gates* in quantum computing. These gates act on one or more qubits, transforming them in a *reversible* way, and are represented as unitary matrices in the basis states of a qubit $|0\rangle$ and $|1\rangle$. The system is described in a higher-dimensional space when multiple qubits are involved in a quantum gate. For a quantum register consisting of N qubits, the state space is represented by a 2^N-dimensional Hilbert space. Each possible state of the quantum register corresponds to a basis vector in this space. These basis vectors are denoted as $|000...0\rangle$, $|100...0\rangle$, $|010...0\rangle$, and so on, up to $|111...1\rangle$, where each vector represents a unique combination of qubit states. Unitarity is a property of these quantum operators and their matrix representation in the basis. A unitary matrix U transforms a state vector into another complex state vector $|\psi_2\rangle = U|\psi_1\rangle$, with its adjoint $U^\dagger$ serving as the inverse operation, such that $UU^\dagger = I$.

In quantum computation, similar to Boolean logic, where arbitrarily complex computations can be decomposed in terms of elementary NAND operations, any computation can be decomposed into a universal set of operations and measurements, no matter how complex. This universal set can be constructed using one and two-qubit gates. There are two ways to build these universal sets: (i) a discrete set of single-qubit gates that can be combined and a two-qubit gate such as the CNOT, or (ii) arbitrary parameterized single-qubit unitaries combined with a two-qubit gate.

2.2.1 Universality by Discrete Set

A universal set of quantum gates requires approximating arbitrary single-qubit unitary operations and implementing a maximally entangling two-qubit gate. In the following, we discuss both types of operations, introducing single and two-qubit gates and explaining which discrete single-qubit and entangling gates are usually selected to build such sets.

In the field of quantum mechanics, Wolfgang Pauli introduced operators in matrix form to represent particles with spin $\frac{1}{2}$, such as electrons (Pauli, 1933). These operators, known as *Pauli matrices*, correspond to the projection of the spin along each of the Cartesian axes (*x, y, z*) and are essential for describing the spin states and their transformations (Ballentine, 2014). Pauli matrices and the identity matrix (Equation 2.5) define fundamental rotations in the qubit's state space, acting as quantum gates and forming the basis for constructing more complex quantum operations. An appealing way to visualize these single-qubit rotations is through three-dimensional rotations on the Bloch sphere (see Figure 2.1b), where each Pauli matrix induces a 180-degree rotation around its respective axis: σ_x around the x-axis, σ_y around the y-axis, and σ_z around the z-axis. Thus, any rotation can be expressed as a combination of Pauli matrices (Nielsen and Chuang, 2002),

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.5}$$

This set includes the quantum equivalent of the Boolean NOT operation $X = \sigma_x$, which flips the state of a qubit. The matrix form of the operator is displayed in Figure 2.2, along with its symbol in quantum models and the operator's truth table,

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

Input($\|\psi\rangle$)	Output($X\|\psi\rangle$)
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$

Figure 2.2: Description of X-gate.

The application of this quantum gate to a qubit in each of the basis states $|0\rangle$ and $|1\rangle$ is expressed in Equation 2.6,

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle, \quad X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle. \tag{2.6}$$

However, the two Pauli matrices σ_y and σ_z do not have a Boolean analog since they act on the phase information of the quantum state. The Z-gate introduces a sign change operation in the component $|1\rangle$, while the component $|0\rangle$ is unaffected by applying the operator Z $=\sigma_z$, performing a 180-degree rotation around the Z-axis. The Y$= \sigma_y$ performs a 180-degree rotation around the Y axis, introducing complex phase factors into the coefficients of the basis states. In Figure 2.3 are presented their matrix form and truth tables for basis states,

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad —\boxed{Z}—$$

Input($\|\psi\rangle$)	Output($Z\|\psi\rangle$)
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$-\|1\rangle$

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad —\boxed{Y}—$$

Input($\|\psi\rangle$)	Output($Y\|\psi\rangle$)
$\|0\rangle$	$-i\|1\rangle$
$\|1\rangle$	$i\|0\rangle$

Figure 2.3: Description of Y and Z quantum gates.

Some operators used in the quantum paradigm allow physical properties such as superposition and entanglement to be used. The *Hadamard*-gate or `H`-gate projects a basis state to the equator of the Bloch sphere to create quantum superposition among them, performing a 90° rotation around the y-axis. Figure 2.4 shows the matrix form, the symbol used for its representation on quantum algorithms, and its truth table for both states.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad —\boxed{H}—$$

Input($\|\psi\rangle$)	Output($H\|\psi\rangle$)
$\|0\rangle$	$\frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$
$\|1\rangle$	$\frac{\|0\rangle-\|1\rangle}{\sqrt{2}}$

Figure 2.4: Representation of Hadamard operator.

When this operator is applied to the basis states, it creates a superposition expressed as,

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H|1\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{2.7}$$

One notable feature of quantum computing is its capacity to establish strong correlations between qubits. This correlation is established by using the Controlled-NOT (CNOT) operator, which acts on two qubits to generate quantum entanglement between them, as shown in Equation 2.13. This operator works on the basis states as illustrated in the Equations 2.8,

$$
\begin{aligned}
\text{CNOT}|00\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = |00\rangle \\
\text{CNOT}|01\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = |01\rangle \\
\text{CNOT}|10\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = |11\rangle \\
\text{CNOT}|11\rangle &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = |10\rangle.
\end{aligned}
\tag{2.8}
$$

As can be seen in Equation 2.8, the first qubit —called the *control* qubit— determines whether the second qubit —known as *target* qubit— should be inverted. If the control qubit is in the $|1\rangle$ state, the state of the target qubit is inverted, but if the control qubit is in the $|0\rangle$ state, the state of the target qubit remains unchanged. Figure 2.5 describes the operator in matrix form, its symbol used for quantum algorithms implementations, and the truth table for basis states,

$$
CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}
$$

Input$\|\psi\rangle$	Output(CNOT$\|\psi\rangle$)
$\|00\rangle$	$\|00\rangle$
$\|01\rangle$	$\|01\rangle$
$\|10\rangle$	$\|11\rangle$
$\|11\rangle$	$\|10\rangle$

Figure 2.5: CNOT operator characterization.

On the other hand, additional operators allow phase rotations and, therefore, do not have Boolean analogs, such as T-gate, which performs a phase rotation of $e^{i\pi/4}$ around the Z-axis. It affects the $|1\rangle$ state with a global phase of $e^{i\pi/4}$ without impacting the $|0\rangle$ state. Similarly, the S-gate performs a phase rotation of $\frac{\pi}{2}$ on the same axis, introducing a global phase of i into the $|1\rangle$ state while leaving the $|0\rangle$ state unchanged. Since both are phase-rotation gates, the S-gate can be viewed as the successive application of two T-gates: by applying the T-gate twice, a total rotation of $\frac{\pi}{2}$ radians is achieved, equivalent to the rotation performed by the S-gate (Figure 2.6). In this way, larger phase rotations can be constructed from smaller

rotations. Both operators are presented in Figure 2.6 with their truth tables and the relation between them.

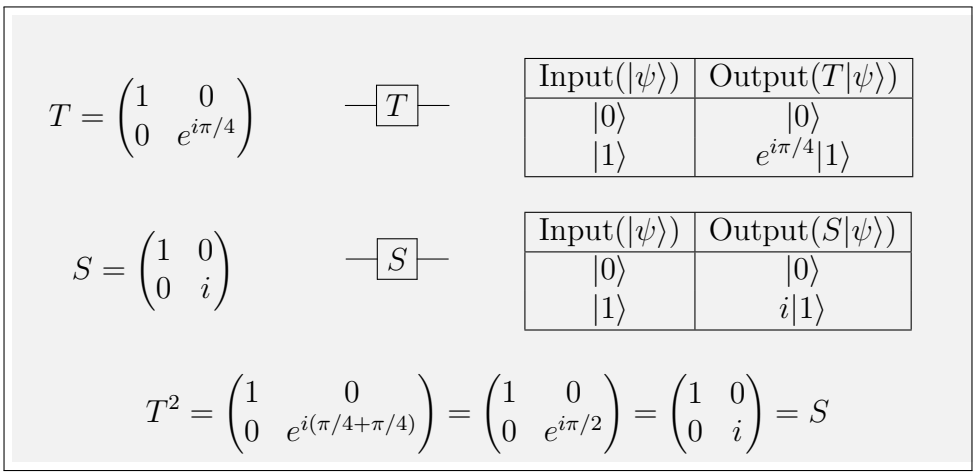


Figure 2.6: Description and relationship between T and S phase operators.

The discrete version requires the combination of all these gates, H, T, S, and CNOT, to reach universality and approximate parameterized gates. Next, these parameterized operators are described as universal sets.

2.2.2 Universality by Parametrized Gates

A universal set of gates can also be constructed using operators that allow rotations parameterized by real angles θ along the three axes (Nielsen and Chuang, 2002). These rotations are defined using the Pauli matrices on each of the axes:

- Rotations around the X-axis are represented by the gate $R_x(\theta)$, which induces a rotation in the YZ-plane as shown in Figure 2.7a. This gate allows adjusting the state of the qubit along the X-axis by changing the amplitude of the state components in the Y and Z-axes. Mathematically, this rotation is described by the rotation matrix in Equation 2.9,

$$R_x(\theta) = e^{-i\theta\sigma_x/2} = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\sigma_x\right) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}. \tag{2.9}$$

- Rotations around the Y-axis are expressed by the gate $R_y(\theta)$, which causes a rotation in the XZ-plane as shown in Figure 2.7b. This gate adjusts the qubit's state along the Y-axis by altering the amplitudes of the state components in the X and Z-axes. This procedure is performed by applying the operator in Equation 2.10,

$$R_y(\theta) = e^{-i\theta\sigma_y/2} = \cos\left(\frac{\theta}{2}\right) I - i\sin\left(\frac{\theta}{2}\sigma_y\right) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}. \tag{2.10}$$

- Rotations around the Z axis are denoted by the gate $R_z(\theta)$, which entails a rotation in the XY-plane as shown in Figure 2.7c. This gate allows adjusting the relative phase between the $|0\rangle$ and $|1\rangle$ states without changing the probability amplitudes expressed as the Equation 2.11,

$$R_z(\theta) = e^{-i\theta\sigma_z/2} = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\sigma_z\right) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}. \tag{2.11}$$

Combining these parameterized rotations and a two-qubit gate such as CNOT allows approximating any arbitrary operation or computation. Because of this approximation property, this selection of rotations and two-qubit gates forms a universal set.

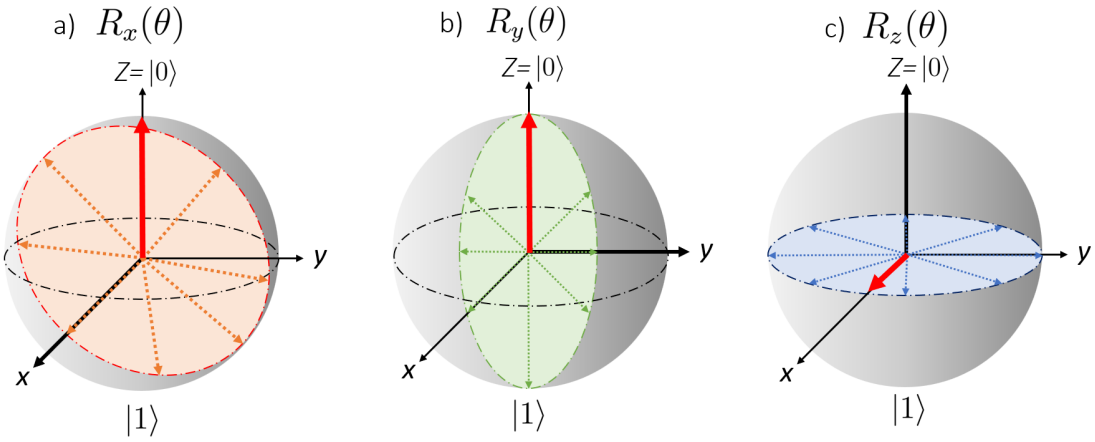


Figure 2.7: Bloch sphere representation of quantum gates. Visualization of the resulting vectors after applying parameterized rotations $R_x(\theta)$, $R_y(\theta)$ and $R_z(\theta)$ on an initial state on the Bloch sphere. a) Rotation around the x-axis with $R_x(\theta)$. b) Rotation around the y-axis with $R_y(\theta)$. c) Rotation around the z-axis with $R_z(\theta)$. The red arrow represents the initial state.

2.3 Quantum Circuits

Any calculations on a quantum computer must be decomposed into quantum gates and measurements. The execution of these quantum gates is usually represented as a *quantum circuit* as shown in Figure 2.8, which determines the gates, the qubits on which they act, and a possible order of execution. In these circuits, the horizontal lines represent the qubit wires, and the quantum gates are placed as blocks in the qubit on which they act. This representation is read from left to right, so the order of the operations influences the evolution of the quantum states.

The circuit in Figure 2.8 comprises two qubits represented by wires, both initially in the state $|0\rangle$. Thus, the initial state of the system is $|\psi_0\rangle = |0\rangle \otimes |0\rangle = |00\rangle$. The first gate that we find in the circuit is a Hadamard gate (see Figure 2.4) applied to the first qubit, creating

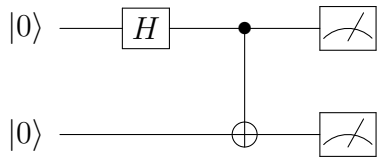


Figure 2.8: Quantum circuit with two qubits composed of an H-gate, an entanglement gate, and measurements.

a superposition state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The state changes from $|\psi_0\rangle$ to $|\psi_1\rangle$ after this operation,

$$|\psi_1\rangle = H|0\rangle \otimes |0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle). \tag{2.12}$$

Afterward, we apply a CNOT gate (see Figure 2.5), using the first qubit as the control and the second qubit as the target. This gate changes the state of the target qubit only if the control qubit is in state 1. When the CNOT gate is applied to the previous state $|\psi_1\rangle$, it is transformed to the state $|\psi_2\rangle$,

$$|\psi_2\rangle = \text{CNOT}\left(\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle)\right) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{2.13}$$

Finally, we perform a *measurement* on both qubits using a computational basis. A measurement is represented as a *gauge*, and it is a probabilistic process that determines one of the possible outcomes of the quantum system. If we measure the qubit's polarization, represented here by the observable $\sigma^z = |1\rangle\langle 1| - |0\rangle\langle 0|$, the outcome of the measurement will be 0 with probability $|\langle 0|\psi\rangle|^2$, and 1 with probability $|\langle 1|\psi\rangle|^2$, as anticipated above. The potential measurement outcomes from the circuit in Figure 2.8 are $|00\rangle$ and $|11\rangle$, each with a 50% probability. This outcome demonstrates the perfect correlation between the entangled qubits, indicating that we cannot measure $|01\rangle$ or $|10\rangle$ from this circuit. Figure 2.8 represents a simple sequence of gates in the circuit; however, decomposing generic algorithms into elementary gates and circuit representations is exponentially complex.

2.4 Quantum Machine Learning Models

Machine learning is a subfield of artificial intelligence focused on developing algorithms that enable machines to learn from data. By analyzing datasets, these algorithms identify patterns and correlations that allow machines to make predictions or decisions without being explicitly programmed for specific tasks. This process involves feeding data into the model, which then uses statistical analysis to predict the results, making machine learning an essential part of creating adaptive and intelligent systems (Bishop, 2006). Typically, ML can have four learning methods: supervised, unsupervised, generative, and reinforcement learning. This thesis focuses on supervised learning that needs labeled data sets to perform training. Taking into account a *training dataset* $\{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$ consisting of normalized feature vectors $\mathbf{x}_i \in \mathbb{R}^d$

and binary classes $y_i \in \{+1, -1\}$, the main goal is to design a model $f(\mathbf{x})$ that can accurately predict the class of any point, whether it belongs to the training set or unseen data.

The universal nature of parameterized sets of gates implies, intuitively, that a sufficiently complex quantum circuit can take the state $|000...\rangle$ and bring it close to any other quantum state in the Hilbert space. Naturally, the more gates and parameters the circuit has, the easier and better this approximation is. This property of parameterized circuits leads us to consider two applications: The first is engineering feature maps, which associate a set of parameters (the angles of the gates) to states in the Hilbert space. The second one is engineering discriminators, whereby a quantum circuit transforms a quantum state, and the outcome of the transformation —e.g., the overlap with the $|0000...\rangle$ state— allows us to determine the nature of the input state. The integration of these two concepts gives rise to various families of machine-learning protocols that leverage the quantum domain as a foundational space for information processing and manipulation.

Quantum machine learning often uses *Variational Quantum Circuits* (VQC) as models (Benedetti et al., 2019). These circuits consist of gates with adjustable parameters and measurements, allowing us to optimize specified objectives or cost functions. The variational circuit in Figure 2.9 is comprised of two qubits with rotation gates $R_y(\theta)$ that integrate the angles θ_1, θ_2, θ_3, and θ_4 as parameters to be fixed, an entanglement gate, and measurements. In the following sections, the focus will be on quantum machine learning methods that utilize variational circuits and their adjustable parameters for specific tasks.

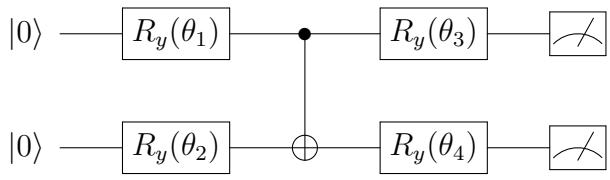


Figure 2.9: Two-qubit quantum variational circuit. It consists of an entanglement gate and four parameterized gates with free angles for optimization, followed by measurements.

2.4.1 Quantum Kernel Method

One of the earliest techniques for binary classification is the *Support Vector Machine* (SVM) (Cortes and Vapnik, 1995; Stitson et al., 1996). This method is explicitly developed for linearly separable data constructing a *hyperplane* with an average vector $\mathbf{w}$ and a displacement value b. The main idea is to locate the hyperplane so that the two classes $y = +1$ and $y = -1$ are on opposite sides. The classifier takes a simple form, represented by the sign function in Equation 2.14,

$$f(\mathbf{x}) = \text{sign}\left(\mathbf{w}^T \cdot \mathbf{x} + b\right). \tag{2.14}$$

The classifier hyperplane uses a few *support vectors* from the training set. They are selected to maximize the margin between them and the hyperplane, as shown in Figure 2.10. Mathematically, the normal vector $\mathbf{w}$ is obtained by summing the products of the coefficients β_i,

which are the Lagrange multipliers associated with each data point, and determining their importance in the formation of the vector $\mathbf{w}$, together with the corresponding class labels y_i and the feature vectors $\mathbf{x}_i$,

$$\mathbf{w} = \sum_i \beta_i y_i \mathbf{x}_i \,. \tag{2.15}$$

By maximizing the margin, SVM seeks a clear separation between classes and aims to create the largest possible distance between the separation hyperplane and the closest data points in each class. It reduces the probability of error in classifying new data, since a wider margin provides a more remarkable ability to generalize correctly from the training data, allowing more accurate classification of future data points.

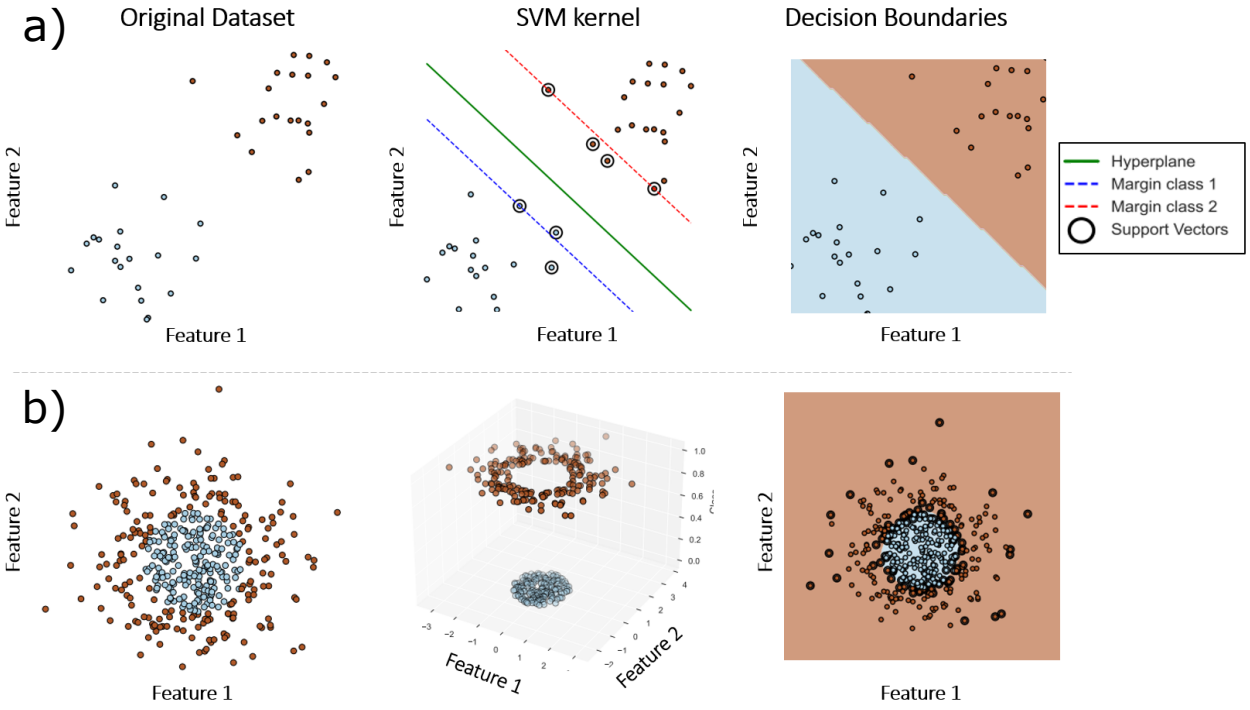


Figure 2.10: SVM. a) two classes are linearly separable, and SVM produces the decision boundary. b) Non-linear dataset and decision boundaries produced by SVM after increasing dimensionality.

Multiple techniques can transform SVMs into a universal classifier for handling non-linearly separable data. One technique involves creating additional features or variables based on the original vectors, thereby increasing the dimensionality of the classification space, known as the *kernel trick* (Cortes and Vapnik, 1995). This is accomplished by defining $\tilde{\mathbf{x}}_i := \mathbf{\Phi}(\mathbf{x}_i) \in \mathbf{R}^r$, where r is much larger than the original dimension d. A *feature map* can transform the problem into a linearly separable one by expanding the dimensionality, as can be seen in Figure 2.10b.

Interestingly, one can derive the classifier from a kernel function, which captures the scalar product between the new features. The definition of the kernel function is:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{x}'), \tag{2.16}$$

where $\mathbf{x}$ and $\mathbf{x}'$ are two input vectors, and $\mathbf{\Phi}(\mathbf{x})$ and $\mathbf{\Phi}(\mathbf{x}')$ are their projections in a higher-dimensional feature space. The inner product $\mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{x}')$ calculates the similarity between these projections, allowing us to work in the transformed space without explicitly computing the projections. The expression of the hyperplane in terms of the new features and how it relates to the final classifier can be understood through Equation 2.17,

$$\begin{aligned} f(\mathbf{x}) &= \text{sign}\left(\sum_i \beta_i y_i \mathbf{\Phi}(\mathbf{x}_i)^T \mathbf{\Phi}(\mathbf{x}) + b\right) \\ &= \text{sign}\left(\sum_i \beta_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right) \in \{+1, -1\}. \end{aligned} \tag{2.17}$$

In this equation, $\mathbf{x}$ is the input vector for making a classification. The decision function uses the support vectors $\mathbf{x}_i$ and their labels y_i, along with the coefficients β_i obtained during training, to compute a linear combination in the feature space. The projection $\mathbf{\Phi}(\mathbf{x})$ of $\mathbf{x}$ and the projection $\mathbf{\Phi}(\mathbf{x}_i)$ of the support vectors are used, simplified through the kernel function $K(\mathbf{x}_i, \mathbf{x})$, while the term b is the bias that adjusts the position of the decision hyperplane. In this context, the sign of the function $f(\mathbf{x})$ determines the class of the input data x. Also, it is essential to consider *Mercer's theorem* (Géron, 2022; Mercer, 1909), which demonstrates that it is unnecessary to know the exact form of the feature map, which could even be an infinite-dimensional function. Instead, it is sufficient to have a kernel function $K(\mathbf{x}, \mathbf{x}')$ that possesses the necessary positive properties to encode a scalar product. Figure 2.11 shows the iterative process of classical SVM training.

1: **Input:** Training data, kernel function
2: **Output:** Support vector coefficients and support vectors
3: Initialize support vector coefficients to zero
4: **while** the model has not converged **do**
5: **for** each training sample **do**
6: Compute the kernel product $K(x_i, x_j)$ between the sample and the support vectors
7: Classify the sample: $\hat{y}_i = \text{sign}(\sum \beta_j K(x_i, x_j))$
8: Check if the sample is correctly classified
9: **if** the sample is misclassified **then**
10: Adjust the support vector coefficients: $\beta_i \leftarrow \beta_i + \Delta\beta_i$
11: **end if**
12: **end for**
13: **end while**

Figure 2.11: Classical support vector machine training

A *Quantum Support Vector Machine* (QSVM) is a classification algorithm where the features are quantum states in a Hilbert space $|\Phi(\mathbf{x})\rangle$. The algorithm remains almost unchanged,

replacing the scalar product among vectors with the adequate product in the Hilbert space, as can be seen in Figure 2.12. In the particular case that we are interested in, the feature states are constructed by applying a parameterized quantum circuit. This feature map can take on a simple form, such as encoding the data into quantum register states or quantum amplitudes — the so-called *data encoding* as shown in Figure 2.15a. However, it is more commonly a parameterized unitary transformation constructed using quantum gates that depend on the input features $|\Phi(\mathbf{x})\rangle := \mathcal{U}(\mathbf{x};\boldsymbol{\theta})\,|0\rangle^n$, as well as additional controls $\boldsymbol{\theta}$ as shown in 2.15b. Nevertheless, it is possible to utilize these circuits solely for evaluating a kernel function (Havlíček et al., 2019; Schuld, 2021; Schuld and Killoran, 2019) such as follows,

$$K(\mathbf{x},\mathbf{x}') = |\langle\Phi(\mathbf{x})|\Phi(\mathbf{x}')\rangle|^2 = \left|\langle 0^n|\mathcal{U}(\mathbf{x};\boldsymbol{\theta})^\dagger\mathcal{U}(\mathbf{x}';\boldsymbol{\theta})|0^n\rangle\right|^2, \tag{2.18}$$

where the kernel $K(\mathbf{x},\mathbf{x}')$ is defined as the similarity between inputs $\mathbf{x}$ and $\mathbf{x}'$ in a quantum space. $|\Phi(\mathbf{x})\rangle$ and $|\Phi(\mathbf{x}')\rangle$ are quantum states generated by applying the unitary operator $\mathcal{U}(\mathbf{x};\boldsymbol{\theta})$ to the basis state $|0^n\rangle$. The kernel is the squared absolute value of the inner product between these states. The SVM classifier $f(\mathbf{x};\boldsymbol{\theta})$ can be obtained using this kernel to achieve the classification outcome. Furthermore, in this thesis, we use a different quantum kernel described in the Equation 2.19,

$$K(\mathbf{x},\mathbf{x}') = \mathrm{Re}\,\langle\Phi(\mathbf{x})|\Phi(\mathbf{x}')\rangle = \mathrm{Re}\,\langle 0^n|\mathcal{U}(\mathbf{x};\boldsymbol{\theta})^\dagger\mathcal{U}(\mathbf{x}';\boldsymbol{\theta})|0^n\rangle\,. \tag{2.19}$$

The function maintains the original purpose of $K(\mathbf{x},\mathbf{x}')$ by avoiding the squaring of the scalar product between vectors. The quantum kernel method is commonly combined with an iterative update of the parameters θ to maximize a cost function that incorporates the model's accuracy and additional regularization.

Input: Training data, quantum kernel function
Output: Quantum support vector coefficients and support vectors
Initialize support vector coefficients to zero
while the model has not converged **do**
    for each training sample **do**
        Encode the sample x_i into a quantum state $|\phi(x_i)\rangle$
        Compute the quantum kernel product $K(x_i, x_j) = |\langle\phi(x_i)|\phi(x_j)\rangle|$ between the sample and the support vectors
        Classify the sample: $\hat{y}_i = \mathrm{sign}\left(\sum \beta_j K(x_i, x_j)\right)$
        Check if the sample is correctly classified
        if the sample is misclassified **then**
            Adjust the support vector coefficients: $\beta_i \leftarrow \beta_i + \Delta\beta_i$
        end if
    end for
end while

Figure 2.12: Quantum support vector machine training

The success of this method depends on the strength of the underlying feature map, which can create challenges such as *barren plateaus* and other barriers that impede effective training (Cerezo et al., 2021; McClean et al., 2018). The *barren plateaus* effect occurs when there are issues in the gradient optimization process in quantum systems with numerous qubits and in medium-to-large classical neural networks. Increasing the number of qubits in the quantum model's structure can result in an exponential decrease in the gradient. This means that as the circuit grows, it becomes challenging to train the model efficiently because the parameter updates become insignificant for an accurate classification.

2.4.2 Quantum Neural Networks

Classical artificial neural networks (ANN) are bio-inspired algorithms based on the brain. This type of algorithm are based on small units called *neurons* that can process information with some features denoted as $X = \{x_1, x_2, \dots x_n\}$ using the Equation 2.20,

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right). \tag{2.20}$$

Here, y is the neuron's output, and $\sigma(\cdot)$ represents the activation function that introduces non-linearity. The weighted sum of input features, $\sum_{i=1}^{n} w_i x_i$, shaped by corresponding weights w_i, reflects how the neuron evaluates and combines the inputs (LeCun et al., 2015; McCulloch and Pitts, 1943). b is the bias term, a constant used to adjust the response threshold for effective learning and generalization. As observed in Figure 2.13, these networks are organized in interconnected layers of neurons.

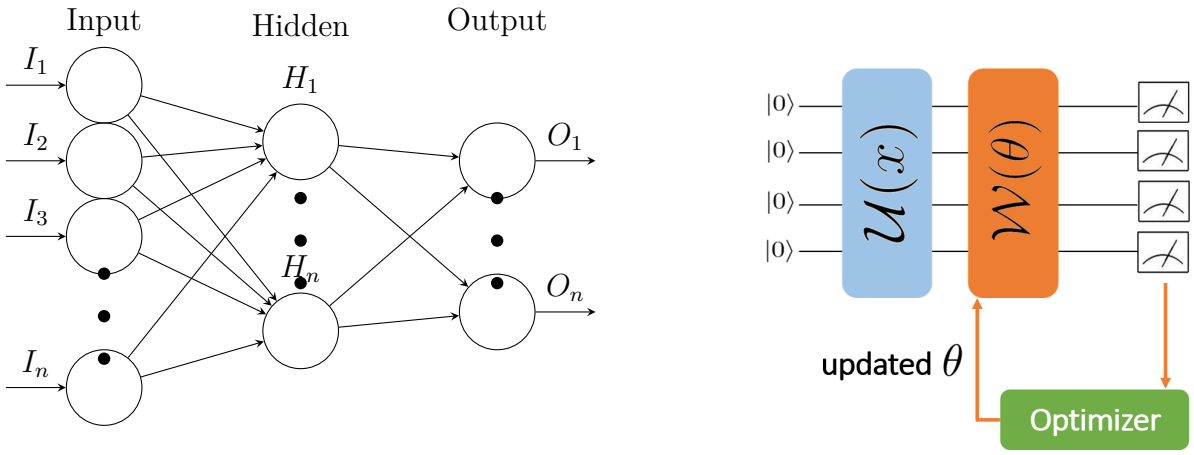


Figure 2.13: Classical and quantum neural networks architecture.

These models are trained by iteratively adjusting w_i and b to minimize a loss function defined using *backpropagation* (S. Amari, 1967; LeCun et al., 1988; Rosenblatt, 1961). In this algorithm, the partial derivatives of the cost function L are computed with respect to the weights w and biases b of the neural network, using the chain rule to update them and minimize the cost function. During the training of a neural network, the system iteratively adjusts each weight and bias until certain stopping conditions are met, which typically include

reaching a predetermined number of training epochs or reaching a plateau in the improvement of the *loss function* (also known as *cost function*). This function quantitatively measures the model error on the training data. A minimum value of this function indicates a lower model error, and consequently, the network outputs are more accurate. When the improvement in the loss function stabilizes, i.e., reaches a plateau, a typical technique called *early stopping* is applied, which stops the training to avoid *overfitting*. This effect occurs when the model fits too closely to the specific details of the training set instead of learning general patterns, which reduces its ability to generalize on unseen data (LeCun et al., 2015).

The *Quantum Neural Network* (QNN) operates on the same principles. Typically, it is built on quantum circuits that contain adjustable parameters and measurements. These circuits consist of three components with different functions: a feature map $\mathcal{U}(x)$, a variational form $W(\theta)$, and measurements as shown in Figure 2.13b (Abbas et al., 2021). The *feature map* encodes classical data in a quantum register using an approach similar to QSVM (Schuld and Killoran, 2019; Schuld et al., 2021a). The *variational* form consists of a series of parameterized gate layers with free parameters or weights θ.

1: **Input:** Training data, Feature map, Initial weights
2: **Output:** Optimized weights
3: Initialize adjustable weights $\boldsymbol{\theta}$
4: **while** not converged **do**
5: Transform input data using the feature map: $\mathbf{X} \rightarrow \phi(\mathbf{X})$
6: Apply the variational quantum circuit with parameters $\boldsymbol{\theta}$: $|\psi(\boldsymbol{\theta})\rangle = U(\boldsymbol{\theta})|\psi_0\rangle$
7: Measure and evaluate circuit output
8: Adjust weights to minimize the cost function: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla C(\boldsymbol{\theta})$
9: **end while**

Figure 2.14: Quantum neural network training

Training a quantum neural network is similar to training a classical neural network, with the objective of adjusting the parameters to minimize a cost function. The optimization of these weights is performed using commonly classical optimization algorithms. The training procedure consists of four iterative steps: **Step 1.** Initial values are selected for the free θ parameters of the circuit in a random way (Figure 2.14 line 3). **Step 2.** Input data are transformed using the feature map and passed through the quantum circuit (Figure 2.14 line 5-6). **Step 3.** A measurement is performed on the resulting quantum state, and the circuit output is evaluated to compute the cost function (Figure 2.14 line 7). **Step 4.** The QNN parameters θ are adjusted to minimize the cost function using classical optimization techniques such as *Stochastic Gradient Descent* (SGD) (S.-i. Amari, 1993) (Figure 2.14 line 8). **Steps 2-4** are iteratively repeated until a predefined convergence criterion is met or the desired model performance is achieved. Note that in Figure 2.14, these steps are explained in pseudocode for better understanding.

The design of these circuits poses challenges in achieving optimal performance. Established circuits for both feature maps and variational circuits can be found in platforms like Qiskit

or Pennylane (Bergholm et al., 2018; Qiskit contributors, 2023). In Figure 2.15a, the ZZ feature map is illustrated, showcasing its application on the variables x_0, x_1, and x_2 from the set X. This map is commonly used in the state of the art to encode classical information into a quantum system. Figure 2.15b displays a circuit structure comprising a data encoding block and two free parameter layers utilizing the RealAmplitude variational form. The circuit structures mentioned are efficient but lack adaptability and flexibility for specific use cases. Additionally, as the number of encoded variables increases, the number of qubits in these models also increases, resulting in high-dimensional arrays that require many qubits. As a result, training can become expensive and may lead to barren plateaus (McClean et al., 2018).

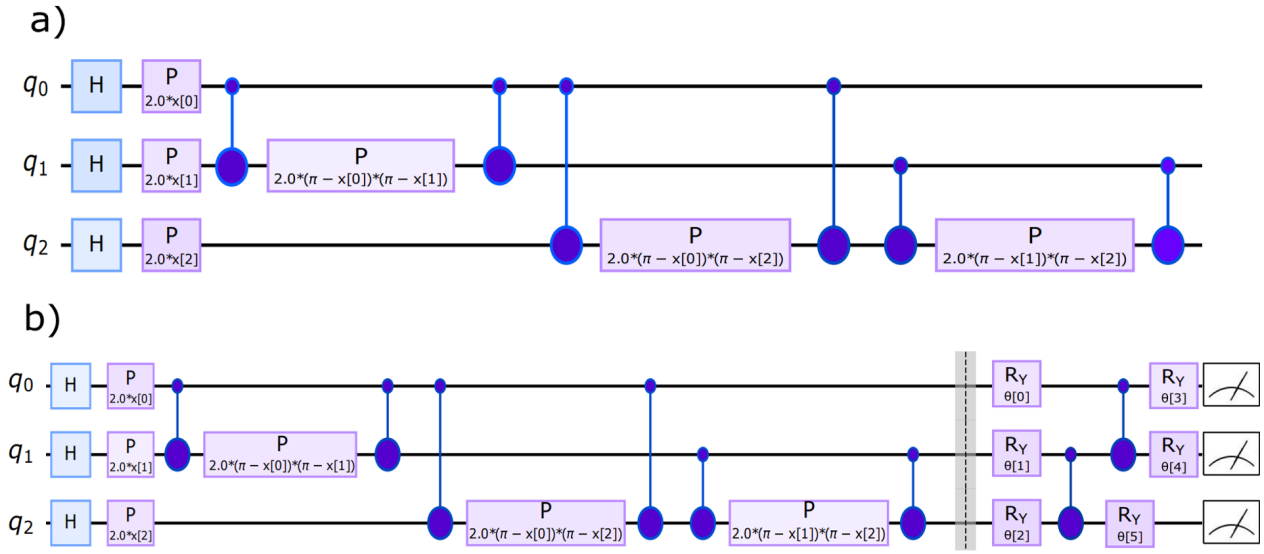


Figure 2.15: Standard quantum circuit (Havlíček et al., 2019; Qiskit contributors, 2023). a) ZZ-Feature map. b) Complete quantum neural network with ZZ-feature map and RealAmplitudes VQC for trainable parameters from qiskit.

Chapter 3

Generation of Optimal Quantum Circuits for Tabular Data

The design of quantum circuits for machine learning requires defining the circuit structure, the gates to be executed in that structure, and any possible parameterization of the gates. As commented in chapter 2, highly expressive circuits can better explore the Hilbert space and encode probability distributions accurately. However, these models face problems such as local minima and *barren plateaus*, which make optimization challenging (See Section 2.4.1). Currently, there are no generic methods for designing quantum circuits, and the process is often manual or based on predefined circuits that do not adapt to specific data sets (Bergholm et al., 2018; Qiskit contributors, 2023).

This chapter describes a methodology for the automatic generation of optimized quantum circuits for quantum machine learning, using a mechanism for constructing optimal solutions known as genetic algorithms, the basics of which are discussed in the next section. Our algorithm provides total flexibility in the configuration of quantum circuits, allowing them to dynamically adjust their topology, gate types, parameters, and coding of classical variables. This customization capability allows each circuit component to be precisely adapted to the particular characteristics of the data and the problem to be solved ad-hoc, as is the case of the low-dimensional tabular data used in this chapter. The outcome are small and accurate feature maps specifically designed for quantum models of support vector machines, which increases their efficiency and adaptability.

The algorithm is presented in two parts. Section 3.1 introduces the basics of genetic algorithms. Section 3.2 explains a methodology to globally optimize quantum circuits using genetic algorithms. Section 3 tests the methods' performance by applying them to different tabular data sets. Finally, section 4 summarizes the main conclusions of this chapter and potential applications.

3.1 Optimization Method: Genetic Tool

A quantum machine learning pipeline consists of two high-level steps: first, the definition of the quantum model (the circuit, disposition of gates, possible parameterizations, the introduction of features, etc.), and second, the training of this model (i.e., learning the free parameters) to perform the desired task. Two different but interrelated optimizations are required: (i) the continuous parameters, which refer to the angles of the quantum gates, and (ii) the discrete properties, which include both the types of quantum gates used and their specific locations within the circuit.

The mixed nature of the problem, which combines continuous and discrete optimizations, immediately rules out gradient-based methods since these methods cannot effectively handle discrete properties, such as the type and location of quantum gates. The alternative is the introduction of global optimization strategies capable of handling both the continuous and discrete aspects of the problem. *Genetic algorithms* use a solution-finding strategy inspired by natural evolution using operators such as *selection*, *crossover*, and *mutation* to effectively explore and optimize combinations of continuous and discrete parameters. These algorithms have the ability to iteratively improve the solutions found, allowing them to examine a diverse set of potential options over several generations. This ability to explore and exploit the search space ensures that genetic algorithms can adapt to changing environments and requirements, evidencing their versatility and robustness, exploring large solution spaces and avoiding local minima through their operators (Goldberg, 2013).

3.1.1 Genetic Algorithms

John Holland introduced a family of biologically inspired optimization algorithms now known as genetic algorithms (Holland, 1975, 1992). These methods perform a guided exploration within a solution space, systematically refining a set of potential solutions to optimize a specific objective function or cost function through iterative processes. These algorithms are based on the *schema theorem*, which guarantees the iterative improvement of solutions by exploiting these underlying structures, thus optimizing the objective function over time.

The genetic procedure starts from a set of possible solutions called *initial population*, composed of *individuals* or *chromosomes*. The genetic composition of each *chromosome* (or *genotype*) is defined by the *genes* that distinguish each unique solution. This genotype encodes the *phenotype*, which is the real representation of a solution (see Figure 3.1b). Each individual represents a possible solution to the problem and is evaluated by a *fitness function*, which measures how good that solution is. Individuals with the best fitness values have a high probability of being selected to form the next generation. However, an important feature of this type of algorithm is to maintain *genetic diversity*: not only the fittest individuals can be selected, but also others with lower fitness, which allows a wider variety of solutions to be explored and avoids premature convergence. Once selected, two genetic operators are applied to create new individuals: crossover and mutation. Crossover combines parts of two parents to form new individuals, simulating sexual reproduction in biology. At the same time, mutation introduces random changes in the genes of the new individuals to maintain genetic diversity and prevent the algorithm from being trapped in suboptimal solutions. After

selection, crossover, and mutation, the new individuals are introduced into the population, and fitness is calculated for the new population; if the stop criterion is not met, the genetic operators are iterated through multiple generations, as shown in Figure 3.1a.

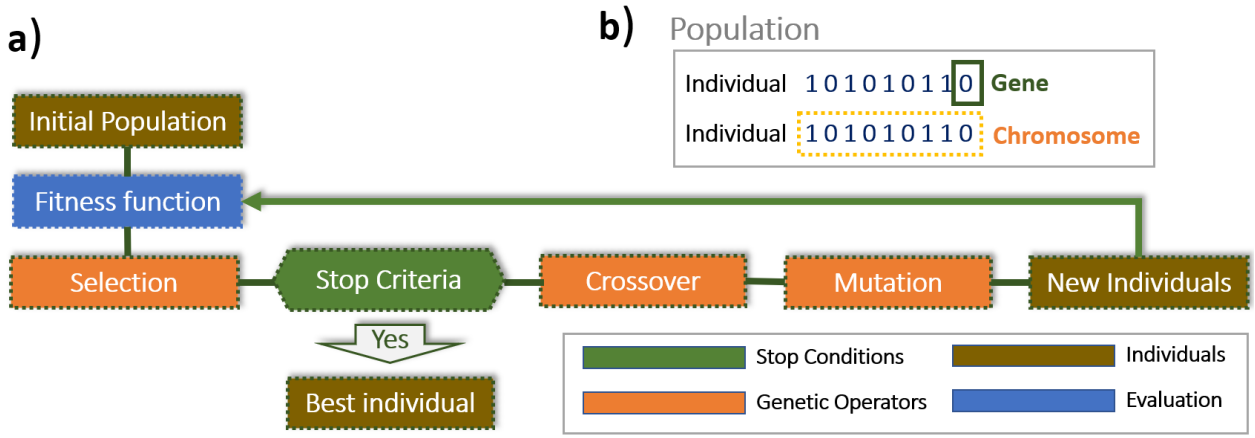


Figure 3.1: Genetic algorithm. a) General scheme of evolution in an evolutionary algorithm. The process begins with the generation of an initial population of solutions. In each generation, the solutions are evaluated, selected, and recombined by crossover and mutation. The fittest individuals advance to the next generation, repeating the loop until the stop conditions are reached, providing then the best solution. b) Definition of population, individuals, and genes.

In general, the application of genetic algorithms (See Figure 3.2) involves the next steps: **Step 1**. An initial population of random solutions (or individuals), thus establishing the starting point for the algorithm. **Step 2**. Each individual in the population is evaluated using a fitness function, assigning a value that indicates how good its solution is with respect to the problem to be solved. **Step 3**. The best individual is selected from the population based on its fitness value, which will be tracked throughout the algorithm. This ensures that the best solution is always identified for further processing. **Step 4**. A loop is started that will continue to execute as long as a specific end condition, such as a maximum number of generations or a desired fitness level, is not met. **Step 5**. Individuals are selected from the population to act as parents in the next generation based on their fitness values so that individuals with higher scores are more likely to be chosen. **Step 6**. The selected parents are crossed to create new individuals (offspring), mixing parental features to produce new solutions. **Step 7**. A mutation operator is applied to the generated offspring, introducing random variations that help maintain genetic diversity and explore new areas of the solution space. **Step 8**. The offspring is evaluated using the fitness function, assigning fitness values to the new individuals. **Step 9.** After evaluating the offspring, the algorithm compares the new evaluated individuals with the best individual found so far. If any of the new individuals have a higher fitness value than the current best, the best individual is updated to reflect this improvement. **Step 10**. Check if termination criteria are met; if not, repeat from **Step 5**. Finally, once the stop conditions are met, the algorithm returns the best individual found during the process, representing the best solution to the problem.

Encoding individuals is one of the key stages in a genetic algorithm. In simple terms,

```
1: population ← InitializePopulation(size)
2: Evaluate(population)
3: best ← SelectBest(population)
4: while not Stop Condition do
5:     parents ← ParentsSelection(population)
6:     offspring ← Crossover(parents)
7:     offspring ← Mutation(offspring)
8:     Evaluate(offspring)
9:     population ← offspring
10:    best ← CompareAndSelectBest(best, population)
11: end while
12: return Best individual
```

Figure 3.2: General Genetic Algorithm Scheme

encoding defines how each possible solution is represented within the system. Each individual or possible solution is expressed in the form of a genotype, which is a structure that encodes the essential information of a solution. This representation is crucial because it allows the genetic algorithm to work with solutions on an internal, manipulable level, regardless of the problem's complexity or in another domain. Generally, the genotype is represented using a low-cardinality alphabet; in other words, a limited set of symbols facilitates the operations. A classic example is the binary alphabet (0, 1), in which each individual is represented as a string of bits. This binary scheme is simple and powerful: it allows the manipulation of solutions by binary logic operations, facilitating the implementation of genetic operators. This type of binary encoding is particularly suitable when the search space is discrete and can be represented directly in 0s and 1s. However, in problems where the solutions are continuous or require a different representation, other encoding options exist, such as integers or real numbers. The choice of the appropriate encoding depends on the specific problem and the requirements of the genetic algorithms to maintain effective exploration of the solution space.

Once the individuals have been generated, their quality or performance in relation to the problem to be solved must be evaluated. For this, a fitness function is used, which indicates how good a solution (individual) is in the context of the problem. However, there is a key step here: before calculating the fitness, it is necessary to convert the genotype to the phenotype. The phenotype is the real solution, i.e., how the individual manifests in the context of the problem. Conversion from genotype to phenotype involves translating the internal representation (such as a bit string) into a structure or value that has meaning in the real world. After performing this conversion, the phenotype of each individual can be evaluated with the fitness function. This evaluation assigns a numerical value that indicates the quality of the solution. Higher fitness values represent better solutions to the problem. They are essential to the selection process, as individuals with higher fitness have a higher probability of being chosen to interbreed and generate the next generation.

3.1.2 Genetic Operators

The use of genetic operators introduces diversity and enables the exploration of a broader solution space. These operators mimic biological evolutionary processes, such as selection, crossover, and mutation, to generate new solutions (*offspring*) within a search space (M. Srinivas and Patnaik, 1994). The following sections describe the genetic operators used in this work, discussing various implementation alternatives.

3.1.2.1 Selection

This operator determines which individuals are selected from the population as parents to create the offspring of the next generation. It allows the propagation of desirable genes and gradually improves the quality of solutions over generations. Each operator applies selective pressure to favor the fittest individuals (Holland, 1992). High selective pressure accelerates convergence by selecting the best individuals more frequently, while low pressure slows it down, requiring more generations to adequately explore the search space. In some cases, *elitism* is incorporated, where the best individuals from the current generation are directly passed to the next, ensuring that high-quality solutions are preserved. This helps maintain the strength of the population while still allowing for diversity and exploration through other selection methods. The most common selection operators are presented:

- **Roulette selection**: Each individual in the population has a probability $P(i)$ of being selected proportional to their fitness $f(i)$. If one individual dominates the population (i.e., has significantly higher fitness than others), scaling or normalization is applied to prevent it from monopolizing the selection process. Pairs are formed by randomly selecting individuals but always based on these probabilities. However, a drawback is that individuals with higher fitness can quickly dominate the population, leading to a loss of genetic diversity in just a few generations. This operator is calculated as,

$$P(i) = \frac{f(i)}{\sum_{j=1}^{N} f(j)}. \tag{3.1}$$

- **Tournament:** A set of individuals is randomly selected from the population to form k groups. Within each group, individuals compete against each other, and the one with the best fitness value is the *winner of the tournament*, being selected as the parent of the next generation's offspring. The effectiveness of this method depends heavily on the tournament size. A large tournament size introduces high selective pressure, accelerating convergence toward local optima and reducing diversity in the population. Tournament selection tends to reduce variability quickly, as selecting high-fitness individuals repeatedly makes the solutions increasingly similar, limiting exploration of other solution space areas. The tournament size is crucial: small tournaments slow convergence but allow for better exploration of the solution space, while larger tournaments speed up convergence but increase the risk of premature convergence. When high-fitness individuals dominate, they are selected repeatedly, potentially excluding less fit individuals with valuable traits, leading to over-exploitation and stagnation. By favoring the best individuals in each tournament, tournament selection can cause the population to concentrate around

a local optimum, making it difficult to explore other regions of the solution space due to a lack of genetic diversity (Goldberg et al., 1991).

- **Rank-based:** This operator orders the individuals according to the fitness values obtained so that if the objective is maximizing, the individuals are ordered in increasing order of fitness. After sorting the solutions, a selection probability value is associated with each individual based on their position in the ranking. The individuals at the top of the ranking, with higher fitness, are assigned higher selection probabilities, while those at the bottom, with lower fitness, are assigned lower probabilities. The probability distribution is typically designed such that there is a greater chance of selecting individuals with higher fitness, but without giving them an overwhelming advantage. This helps to maintain diversity in the population by preventing premature convergence to a single optimal solution. The selection process ensures that individuals are chosen in proportion to their rank rather than their exact fitness value, which can reduce the impact of outliers and the dominance of exceptionally fit individuals, leading to a more balanced exploration of the solution space (Baker et al., 1987).

3.1.2.2 Crossover

This operator combines select individuals to produce new offspring, mimicking sexual reproduction and natural evolution). The main objective is to efficiently explore solution spaces by taking advantage of the best genes found in current individuals to generate potentially superior offspring. The crossover operator mixes the genome representation of the parents to create an offspring that inherits traits from both parents. Different types of operators vary according to their execution and the individuals' coding. The most commonly used ones are explained below:

- **Single-point:** Two parents are selected, and subsequently, a random crossover point on their chromosomes is chosen (Holland, 1975). The genetic information is exchanged among them, generating an offspring that combines parts of the parental chromosomes. Being the length of a solution L and C the crossover point, the operation could be expressed as,

$$\begin{aligned} \text{Offspring}_1 &= \text{Parent}_1[1:C] + \text{Parent}_2[C+1:L] \\ \text{Offspring}_2 &= \text{Parent}_2[1:C] + \text{Parent}_1[C+1:L] \end{aligned} \tag{3.2}$$

- **Two-point:** This method shares similarities with the previous strategy but involves two random crossover points in the parental solutions. The segments between the two points are then exchanged between the parents to produce new descendant solutions. This process introduces variations in genetic material at multiple points, enhancing the exploration capability of the algorithm (De Jong, 1975). For an individual with length L, and being C_1 and C_2 the two random crossover points (with $C_1 < C_2$), the crossover operation would be performed as,

$$\begin{aligned} \text{Offspring}_1 &= \text{Parent}_1[1:C_1] + \text{Parent}_2[C_1+1:C_2] + \text{Parent}_1[C_2+1:L] \\ \text{Offspring}_2 &= \text{Parent}_2[1:C_1] + \text{Parent}_1[C_1+1:C_2] + \text{Parent}_2[C_2+1:L] \end{aligned} \tag{3.3}$$

- **Uniform Crossover:** Each component of the offspring is randomly selected from one

of the parents with an equal probability of $\frac{1}{2}$ (Syswerda et al., 1989), ensuring an equal contribution of genetic material from both parents in each offspring, using a binary vector that is generated randomly P of the same length as the solutions. Each bit in P determines from which parent each component of the offspring is inherited. Given two selected individuals, $Parent_1$ and $Parent_2$, and a randomly generated binary vector, the offspring would be,

$$\text{Offspring}_1[i] = \begin{cases} \text{Parent}_1[i] & \text{if } P[i] = 0 \\ \text{Parent}_2[i] & \text{if } P[i] = 1 \end{cases} \tag{3.4}$$

3.1.2.3 Mutation

This operator simulates the natural process of genetic mutation in an individual's genes. It introduces genetic variability in the population of solutions while maintaining genetic diversity. It prevents premature convergence towards suboptimal solutions (Syswerda, 1991), avoiding local minima within the space (Zhao et al., 2010a, 2010b). Although previous genetic operators naturally generate offspring that inherit their parents's characteristics, mutation introduces small random changes in individuals to explore new areas of the solution space. It is also essential to consider the individuals' representation format to ensure the operators' correct operation. The most common mutation operators are listed below:

- **Flip Bit:** This mutation method is used in the binary encoding of individuals, where a gene is randomly selected from the solution, and its value is altered with a random probability p_{mut}. A random number is generated in the interval [0,1]. If the generated number is below the mutation probability (p_{mut}), then the value is changed at the position i,

$$\text{Inital} =< 1, 0, 0, 1, 1, 0 >; \quad \text{Mutated} =< 1, 0, \mathbf{1}, 1, \mathbf{0}, 0 > \tag{3.5}$$

- **Insert**: This method selects two genes randomly and joins them by moving the intervening genes in the opposite direction. Its operation modifies the order of the genes without altering their values (in bold the changed positions),

$$\text{Initial} =< 3, \mathbf{1}, 4, \mathbf{2} >; \quad \text{Mutated} =< 3, \mathbf{1,}\ \mathbf{2}, 4 > \tag{3.6}$$

- **Scramble:** It selects two genes in a random way within the chromosome and shuffles the genes between them randomly, while genes outside that segment remain unchanged,

$$\text{Initial} =< 3, \mathbf{1}, 4, 2, \mathbf{7}, 9 >; \quad \text{Mutated} =< 3, \mathbf{7}, \mathbf{2}, \mathbf{4}, \mathbf{1}, 9 > \tag{3.7}$$

- **Swap**: In this mutation method, two genes are selected at random within a chromosome, and their positions are exchanged with no changes to the rest of the chromosome,

$$\text{Initial} =< 3, \mathbf{1}, 4, \mathbf{2} >; \quad \text{Mutated} =< 3, \mathbf{2}, 4, \mathbf{1} > \tag{3.8}$$

- **Inversion**: This method selects two genes randomly within a chromosome and reverses the order for all genes between them, including the selected genes (Holland, 1975),

$$\text{Initial} =< 3, \mathbf{1}, 4, 2, \mathbf{7}, 9 >; \quad \text{Mutated} =< 3, \mathbf{7}, \mathbf{2}, \mathbf{4}, \mathbf{1}, 9 > \tag{3.9}$$

3.1.3 Multiobjective Genetic Algorithms

Multiobjective genetic algorithms enable the search for solutions that strike a balance between different objectives rather than searching for a single optimal solution. The operation is similar to single-objective genetic algorithms, with the distinction that the fitness function evaluates two or more objectives in order to improve the fitness values. When assessing at least two goals, individuals are selected considering all quality metrics, which is achieved by techniques such as Pareto dominance and selection of non-dominated solutions (Schaffer, Grefenstette, et al., 1985; N. Srinivas and Deb, 1994).

An optimization curve that balances both objectives is obtained when there are two objectives. This curve is called the *Pareto front*. In Figure 3.3, the different efficient frontiers can be seen taking into account the optimization tasks of the objectives. Each point on this curve represents a solution that cannot be improved in at least one objective without sacrificing another. These individuals are commonly referred to as *non-pareto dominated* solutions. During the evolution, the Pareto front involves a continuous and dynamic evolution of the best solutions.

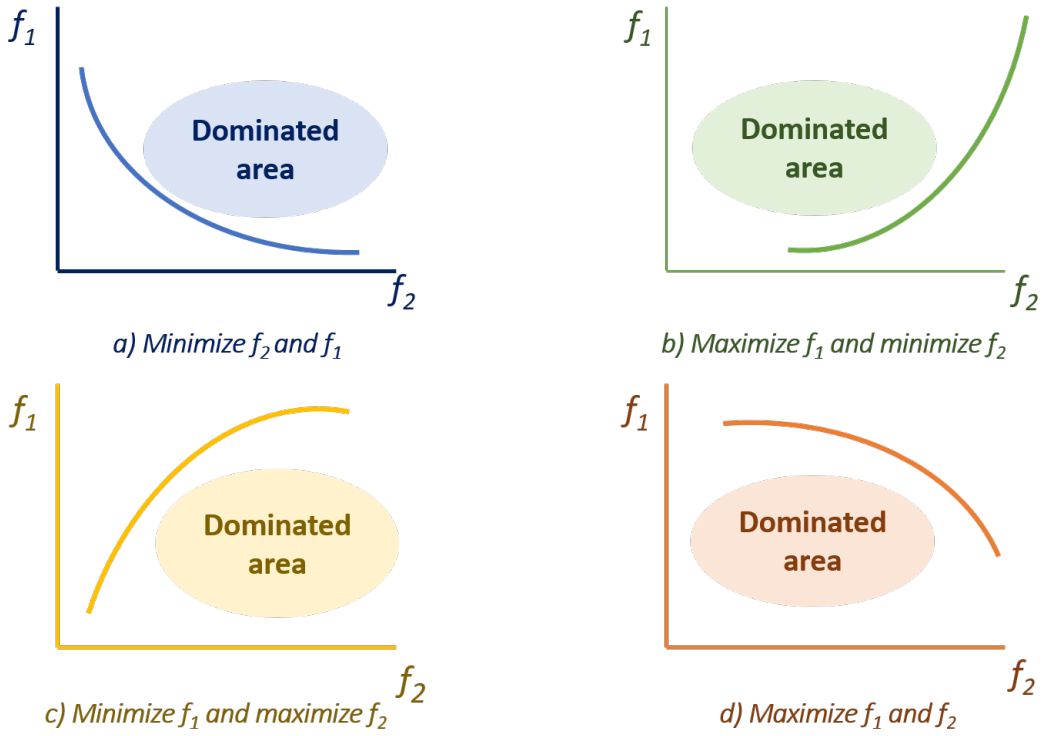


Figure 3.3: Pareto front curves considering the optimization objectives.

To mathematically define the Pareto front, let's consider a set X of possible solutions and an objective function $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$ that assigns each solution $x \in X$ a vector of objective values. A solution $x \in X$ belongs to the Pareto front, denoted as PF, if there does not exist another solution $x' \in X$ such that $F(x')$ dominates $F(x)$, so x is non-dominated if, for each objective i, there is no other solution x' in X that is better or equal in all objectives and strictly better in at least one. Mathematically, we can express this

as,

$$\text{PF} = \left\{ x \in X \mid \nexists x' \in X \text{ such that } \begin{array}{c} \forall i \in \{1, 2, \ldots, n\},\ f_i(x') \geq f_i(x) \\ \text{and} \\ \exists j \in \{1, 2, \ldots, n\},\ f_j(x') > f_j(x) \end{array} \right\}. \tag{3.10}$$

.

During an evolutionary or optimization process, the goal is to find and maintain diverse solutions that cover the Pareto front, as each represents an optimal compromise among the problem's objectives. These solutions are graphically represented on the Pareto front, clearly visualizing the potential optimal solutions as shown in Figure 3.3.

However, not all solutions within the Pareto front are equivalent. Although all are equally valid in that it cannot improve at one objective without worsening at another, each solution represents a different balance between the conflicting objectives. On the other hand, there is not necessarily a single solution that is the best in all cases. The optimal solution depends on the use case and specific target priorities. In this doctoral thesis, the objectives to be evaluated are the models' accuracy and complexity, prioritizing the accuracy and, within the best accuracy, the model with the lowest complexity.

3.2 Genetically Generated Quantum Circuits for QML

We have developed a new quantum circuit optimization algorithm that has been used in tabular data. The solution is based on the genetic algorithms described in section 2.1, with new ingredients detailed in the next section, consisting of (i) a flexible genetic encoding that describes both the structure and the components of a circuit, (ii) a fitness model that addresses applications in quantum machine learning and (iii) the set of specific genetic operators used in this framework that allow the exploration of a solution space formed by generic quantum circuits.

3.2.1 Quantum Circuits DNA and Genetic Code

To apply a genetic algorithm to the design of quantum machine models, we need encoding of the models as genes that can be selected, propagated, and mutated. We rely on a binary genotype that can be translated into a quantum circuit phenotype, determining the gates and parameterizations in a way that allows for an efficient evaluation of the *fitness*.

The main idea is to encode a quantum circuit into a binary string so that when it reaches the fitness function, the binary genotype can be translated into a quantum circuit phenotype for its evaluation. To ensure that the whole circuit is encapsulated within the chain, it's necessary to include several aspects, such as the quantum gates, their parameterization, and the qubits on which they act. The representation of a circuit is based on an MxN matrix, where M is the number of qubits and N is the maximum number of layers, as shown in Figure 3.4a. Each position in this matrix has an operator associated with it from a predefined set, indicating whether that operator depends on any of the variables in the model. Empty slots (i.e., have the identity associated with them) do not involve any operation and allow the circuit size to be reduced—but never exceed the predefined MxN number of gates.

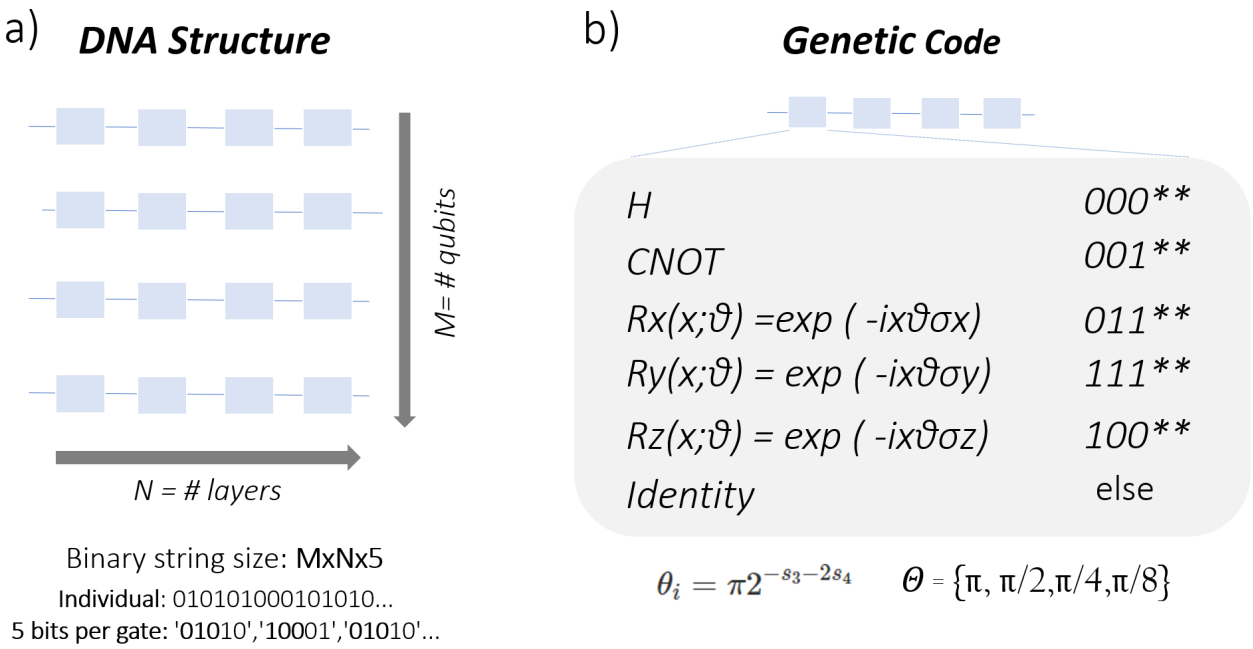


Figure 3.4: Selected encoding for quantum circuits. a) Circuit assembling. b) Genetic code representing each gate.

The genetic encoding maps a binary DNA into a quantum circuit, dividing the DNA into units of five bits $s_0s_1s_2s_3s_4$ that select a quantum gate and a parameter, as described in Figure 3.4b. The first three bits $s_0s_1s_2$ determine the gate type (Hadamard, CNOT, or a local rotation based on input data) and help select the rotation axis. The rotation is defined by $R_\alpha(\theta_i x_k) = \exp(-i\theta_i x_k \sigma_k^\alpha)$ as shown in Section 2.2.2 . The last two bits s_3s_4 encode a proportionality parameter denoted as θ_i, calculated using the formula $\theta_i = \pi 2^{-s_3-2s_4}$. In the case of a CNOT gate, it acts on consecutive qubits, specifically qubits j and $j+1$ mod M. The bit combinations in Figure 3.5b represent operators that can modify the quantum state. However, three combinations do not appear in the figure, which encodes identity operators, allowing the number of gates to be reduced.

For simplicity, the genes are used in sequential order: the i-th gene operates on the j-th qubit of the quantum register and potentially depends on the k-th variable from the input data $\mathbf{x} \in \mathbb{R}^d$, with $j = i$ mod M and $k = i$ mod d. This framework includes a combination of Clifford gates (Bravyi and Kitaev, 2005) and a random selection of rotations.

This genetic encoding enables:

- **Feature selection** is enabled by mutations that change the bits that determine which variable a gate depends on.
- **Gate elimination** is made possible by mutations that set the bits of a given gate to the value of the identity.
- **Adjusting feature weights** is done by scaling the features using the adjustable weights θ_i, enabling the control over the impact of each feature on the circuit, thereby refining its behavior.

- **Combining different features on the same qubits** is facilitated by strategic positioning of gates and parameters, creating intricate nonlinear dependencies among the features (see Figure 3.15).

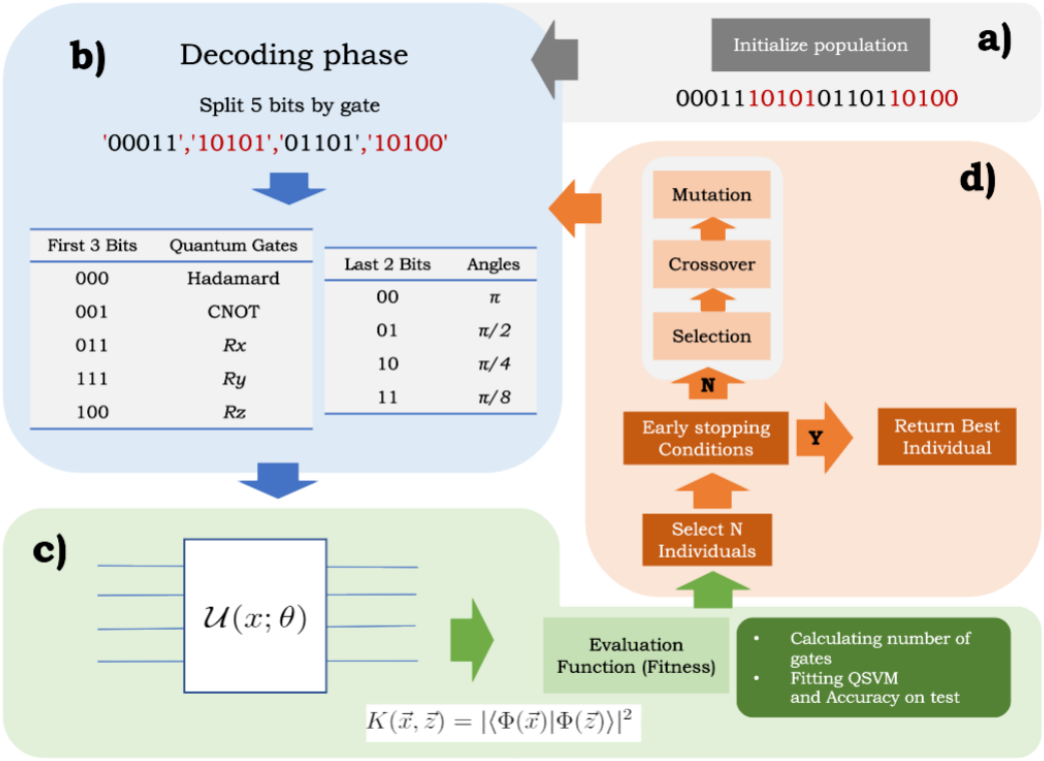


Figure 3.5: Technique scheme. (a) The initial population is initialized. (b) The decoding process is based on five bits per quantum gate and angle. (c) This individual is used in the fitness function as a feature map for QSVM to calculate accuracy and number of gates. (d) The genetic algorithm process continues with phases already seen until the early stopping condition so that the most optimized ansatz for this dataset is provided.

3.2.2 Genetically Designed Quantum Circuits

Our algorithm explores a space of parameterized quantum circuits, which act as feature maps in QSVM as explained in Section 2.4.1. The goal is to generate circuits that achieve the highest accuracy and simultaneously minimize the complexity of the solutions, which is measured in terms of circuit depth and the types of operators used (Altares-López et al., 2022; Altares-López et al., 2021). Our technique is based on a method widely used for its proven robustness in several application areas and for its compatibility with multi-objective optimization: Non-dominated Genetic Algorithm II (NSGA-II) (Deb et al., 2002). This algorithm uses a tournament selection method where individuals compete based on their fitness values (See Section 3.1.2), ensuring that individuals with superior performance (higher accuracy and lower complexity) are chosen as the parents of the next generations (Deb et al., 2000). This selection approach facilitates elitism by prioritizing the survival of the fittest individuals while allowing genetic variability from less optimal solutions to contribute to

evolution.

The process starts with the random generation of an initial population of individuals, represented by binary strings, as shown in Figure 3.5. Each individual is converted from a binary chain *genotype* to a quantum circuit *phenotype* as shown in Figure 3.6. After decoding the quantum circuit, the fitness function measures the aptitude for each objective. In this function, the model is trained. As in any supervised learning model, dividing the dataset into training and test sets is essential to ensure that the models are generalizable to new data.

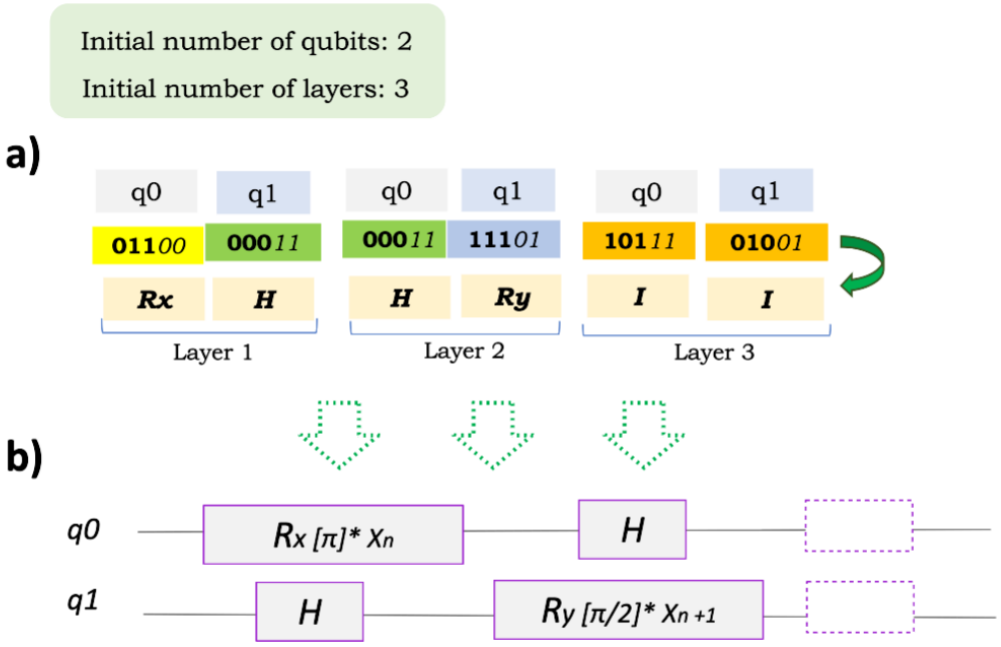


Figure 3.6: Scheme of ansatz decoding. **a)** An initial number of qubits and some layers predefined by the user. A binary chain is created as M x N x 5. Each five bits build a gate. The first three bits decode the quantum gate type, and the last two bits are the angle. As it can be seen, *layer 3* is composed of combinations of bits which code identity operators and allow for reduction of the size of the circuit. **b)** After the decoding phase, the resultant circuit is based on Figure 1b for this binary chain.

The quantum circuit and the training set are used to compute the classifier $f(\mathbf{x})$ in the quantum kernel SVM, as explained in section 2.4.1. Next, the accuracy of the model $f(\mathbf{x})$ is estimated over the test set as the fraction of correctly classified data points. It is essential to consider that accuracy is calculated by comparing the predicted classes with the actual classes from the test set, data not seen during the training phase. At the same time, the complexity of the circuit is calculated based on the number and type of gates. When all individuals in the population have been evaluated, and the stop conditions are not met, they are subjected to different genetic operations, such as selection, crossover, and mutation, considering that those with better metrics are more likely to be selected and create the next generation. The crossover operator used is the *two-point* crossover (See Section 3.1.2), as depicted in Figure 3.7c. Two offspring are obtained from these two selected parents with a mixture of genetic

material. These new solutions undergo a mutation operation to vary their genetic material. Since the quantum circuits are coded into binary strings, it is used the *bit flip* operator (See Section 3.1.2) as can be observed in Figure 3.7b. It allows avoiding local minima in the search for new solutions, where a slight change in the DNA of the circuit, for example, a single bit, can result in an individual with a completely different structure or gate type, greatly extending the search range. This approach employs elitist strategies using the $\mu + \lambda$ method. It alters how the next generation is generated, establishing a competition between the current population (parents, size μ) and the offspring (λ) obtained through genetic operators, as illustrated in Figure 3.7a (Clymer, 1999; Ha et al., 2020). This competition ensures genetic diversity while preserving the best individuals obtained through the evolutionary algorithm (Fortin et al., 2012).

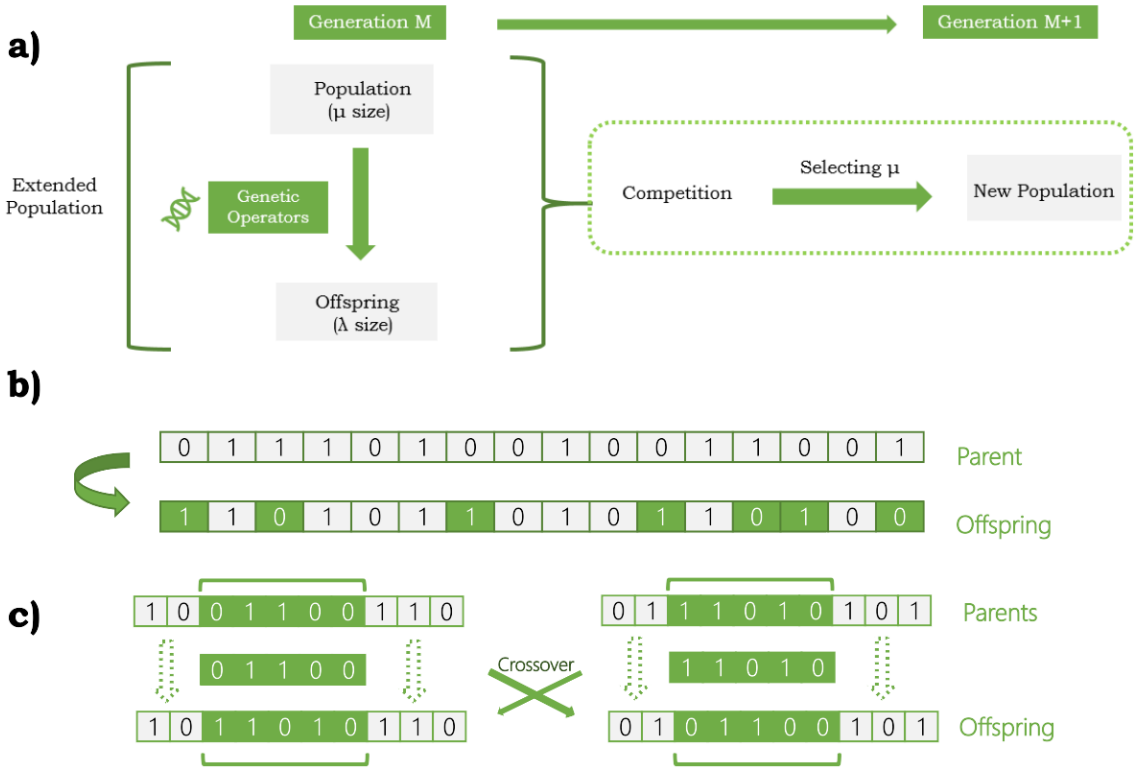


Figure 3.7: Genetic operators. a) Genetic algorithm $\mu + \lambda$ strategy. b) Mutation. A bit of a flip mutation. c) Crossover. (Two-point crossover).

The design of quantum machine learning models requires a trade-off between complexity, measured by the circuit's size and accuracy, with further considerations to the challenges of training large circuits. This problem is addressed by using a multiobjective optimization that uses two concepts, accuracy, and weight control, as metrics to create a Pareto front. The *Complexity Factor* (CF) (Equation 3.11) assigns different costs to the number of local gates N_{local} and the number of entanglement gates N_{CNOT} since the latter have a higher computational cost. In this way, not only is the number of global operators in the circuit

reduced, but also its computational cost.

$$\text{Complexity Factor (CF)} = \frac{N_{\text{local}} + 2N_{\text{CNOT}}}{N_{\text{qubits}}}. \tag{3.11}$$

Complexity and accuracy have an inversely proportional relationship, so an excessive reduction in the number of gates results in a significant loss of information, translating into a decrease in accuracy. To tackle this issue, a dynamic fitness function has been developed to augment the significance of size as the accuracy nears its maximum value of 1. This ensures a suitable balance between circuit complexity and accuracy (See Equation 3.12). This is obtained by,

$$\text{Objective Balance (OB)} = \text{CF} + \text{CF} \times {accuracy_{test}}^2. \tag{3.12}$$

The complete algorithm is described in nine steps to simplify follow-up and reproducibility. The pseudocode for Figure 3.8 is included. **Step 1.** A population of random binary strings with M × N × 5 bits is generated, where each string represents an individual (line 9). Here, M and N represent the maximum number of qubits and layers, respectively, predetermined hyperparameters for the optimization process. **Step 2.** Individuals are assessed using the fitness function defined in Figure 3.8 (lines 1-8). **Step 3.** Each individual is then decoded and transformed into a quantum circuit, as shown in Figure 3.4 (line 2). Figure 3.6 shows an example of this decoding process. In **Step 4**, the circuit is used to compute the kernel of a QSVM, which trains a classifier using the provided training dataset (lines 3-4). **Step 5.** The accuracy of the classifier is then calculated using the test set, as it is the metric to be maximized (line 5). **Step 6** involves computing the circuit's complexity, which is determined by a weighted sum of the number of local and entangling gates present in the circuit (line 6). The objective balance is calculated. The goal is to minimize this metric while maximizing the accuracy. **Step 7.** The non-pareto-dominated solutions are obtained, and it is checked if the stop conditions criteria have been reached. If they have been reached, the best solutions are obtained. However, if they have not been reached, the genetic operators are applied to obtain the individuals of the next generation for a new fitness evaluation (lines 12-20). **Step 8.** After the evaluation, the Pareto front is updated, keeping the solutions that improve at least one of the objectives without worsening the other (lines 21-29). **Step 9.** If the stop conditions are not met, iterate to **Step 3**. When these criteria are met, the best individuals are obtained (line 31).

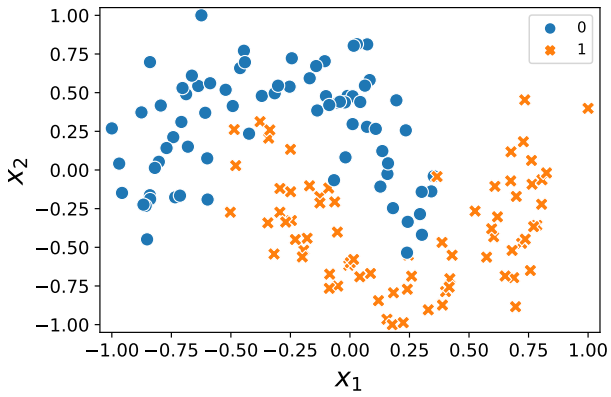
```
function (Fitness (ind))
    Decode ind into a quantum circuit
    Compute kernel of a QSVM using the circuit
    Train classifier with the training dataset
    Calculate classifier accuracy with the test set
    Calculate the effective size of the circuit (OB)
    return accuracy_test and OB
end function



Initialize population P of individuals ind with M × N × 5 bits
for ind ∈ population do
    Fitness function (ind)
end for
Pareto front ← Non-dominated individuals from P
while not stop condition do
    Select parents from P based on fitness
    Apply crossover and mutation operators to generate offspring
    population ← offspring
    for ind ∈ population do
        Fitness function (ind)
    end for
    for ind_new in P_new do
        for ind_Pareto in P_Pareto do
            if OB improves or a_test improves and no value worsens then
                Add ind to Pareto front
            else if ind_Pareto dominates ind_new then
                Discard ind_new
            end if
        end for
    end for
end while
return Pareto front composed of the best individuals
```

Figure 3.8: Genetic algorithm for quantum circuit generation and optimization for QML

3.3 Applications

Our algorithm has been benchmarked using both artificial and real-world datasets from the machine-learning world. The synthetic dataset is the Moon data from the `Scikit-learn` library (Pedregosa et al., 2011) generated with two classes as shown in Figure 3.9a. This dataset consists of 150 data points that are scaled between [-1,+1] as a preprocessing step and randomly divided into training (70%) and test (30%) sets.

a)



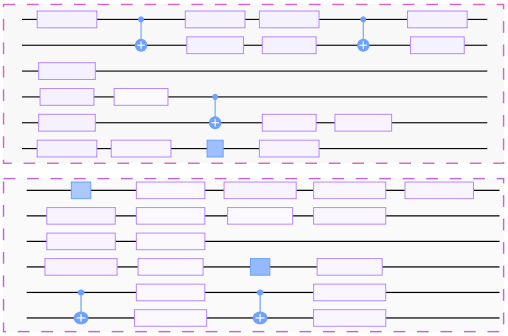
b)

Variables	Value
Generations	5000
Population (μ)	100
Offspring (λ)	15
Qubits	6
Max. Layers	6
Crossover Prob.	0.3
Mutation Prob.	0.7
Mutation Prob. Ind.	0.2

Figure 3.9: a) The dataset is composed of 150 points with a non-linear pattern and a binary target. b) Hyperparameters used to optimize the QSVM circuit with our genetic algorithm.

For circuit generation, we predefine an initial maximum number of qubits (M) and the maximum number of layers (N) are both set to six (see Figure 3.4). As discussed in the previous section 3.2.1, these parameters are user-modifiable depending on the number of variables that make up the data set. In this case, the data set is composed of two features, making a 6x6 matrix excessive. However, this allows us to understand how the complexity reduction works in the circuit generation by comparing with the solutions of the first generations that have an available number of gates of 36, as can be seen in Figure 3.10a.

a)



b)

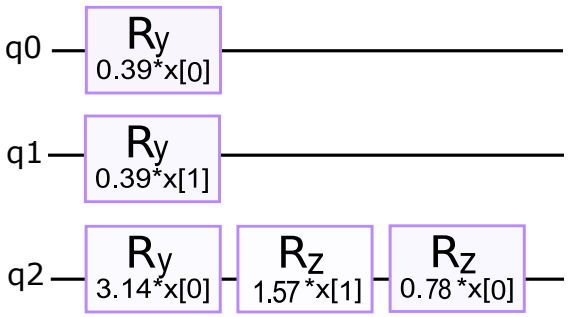


Figure 3.10: a) Structure of circuits created in the first generations of the genetic algorithm. b) Best quantum circuit resultant in the evolution with 1.0 accuracy.

Our genetic algorithm evolves over 5000 generations using a population of 100 individuals as shown in Figure 3.9b. Due to the stochastic nature of genetic algorithms, it is necessary to test them several times with different genetic hyperparameters of crossover and mutation probabilities to ensure that they are optimal and thus reach a good compromise between speed of convergence and optimal classification. In this case, the optimal hyperparameters are 30% as crossover probability and 70% for mutation, with a 20% probability that the bits mutate. This balance allows exploring drastic changes in the population through crossover while maintaining a high rate of small changes through mutation. Although this parameterization is very aggressive, the random component is kept in check by the high elitism of the competition between children and parents in the $\mu + \lambda$ strategy.

As the outcome of the algorithm, we obtain the best individual from the Pareto front, minimizing complexity while maintaining high classification accuracy. As shown in Figure 3.10b, the penalty associated with CNOT gates in Equation 3.12 leads to a decrease in the number of entanglement operators compared to other circuits in the literature. More interestingly, the *Pareto front* combined with the elitist strategies obtain models that do not require any entanglement to fit this nonlinear pattern, producing a simple uncorrelated circuit that fits the test set with maximum accuracy (see Figure 3.10).

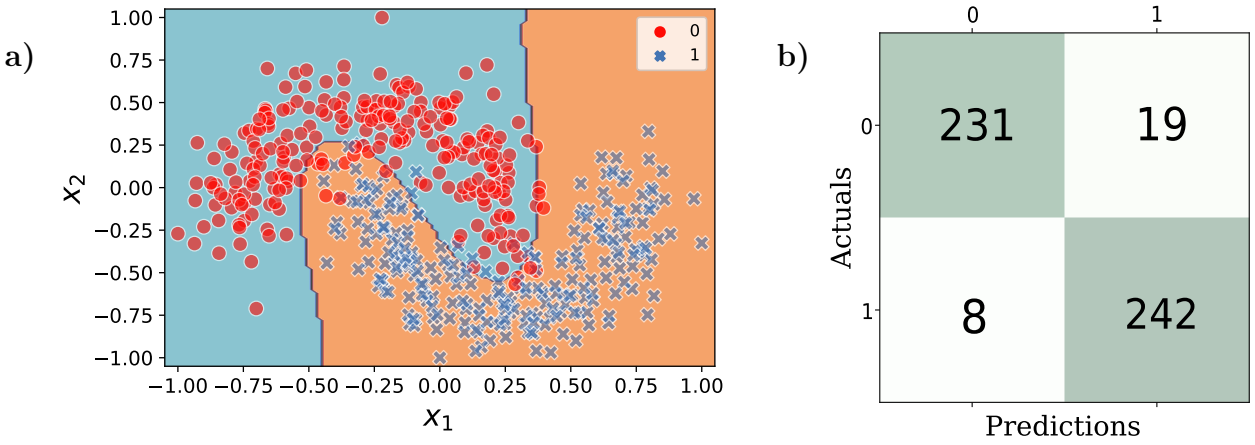


Figure 3.11: a) Validation dataset, together with the predictions and decision boundary from the generated model. b) Confusion matrix produced by the application of the QSVM model onto the validation dataset.

The model generated with maximum accuracy may be useless when it fails to generalize to new data of the same distribution. To assess the model's effectiveness, a validation set consisting of 500 points is generated using the same synthetic algorithm. This dataset is preprocessed by [-1, +1] scaling, similar to the training set. Figure 3.11a shows the predictions made by the optimized quantum support vector machine, with the decision boundary defined by the generated model. In addition, Figure 3.11b presents the confusion matrix of the validation process, which considers the actual and predicted labels. It shows that 473 of the 500 points in the dataset are correctly classified, performing an accuracy rate of 94.6% or an accuracy of 0.946, indicating that the automatically generated and optimized QSVM successfully extrapolates to unseen data with the same non-linear pattern.

3.3.1 Quantum-Inspired Models Interpretability

The discovered model's simple structure of Figure 3.10b allows us to analyze how each part of the circuit contributes to the prediction. This helps identify the key components and understand how they work together, making it easier to explain the models' outcomes. This interpretability approach can be applied to other complex models, improving our understanding and confidence in their predictions by creating a rule-fit explanation.

Since the model generated in the last section consists of uncorrelated qubits, the resulting circuit can be decomposed into separate unitaries. The quantum kernel becomes a scalar

product of individual kernels represented as $K(\mathbf{x}, \mathbf{x}') = \prod_{i=1}^{m} K_i(\mathbf{x}, \mathbf{x}')$. Note that each kernel $K_i(\mathbf{x}, \mathbf{x}')$ may actually depend only on a subset of the features or a combination thereof.

The classification induced by each K_i kernel independently and their combination is analyzed as a strategy for interpreting the rules generated by the evolutionary approach. Figure 3.12a illustrates the complete decision boundary of the kernel, achieving 100% accuracy. On the other hand, Figures 3.12b-d, illustrate the decision boundaries drawn by applying each kernel separately. The first and second qubits generate linear hyperplanes, while the third qubit introduces a level of non-linearity through the implementation of three rotations. As shown in Figure 3.12, the accuracy obtained by the individual qubits is low in all cases, with the first qubit being 0.57, the second 0.68, and finally, the third 0.53. However, when all these qubits are combined in the final kernel, they provide accurate distribution predictions, producing the desired nonlinear pattern.

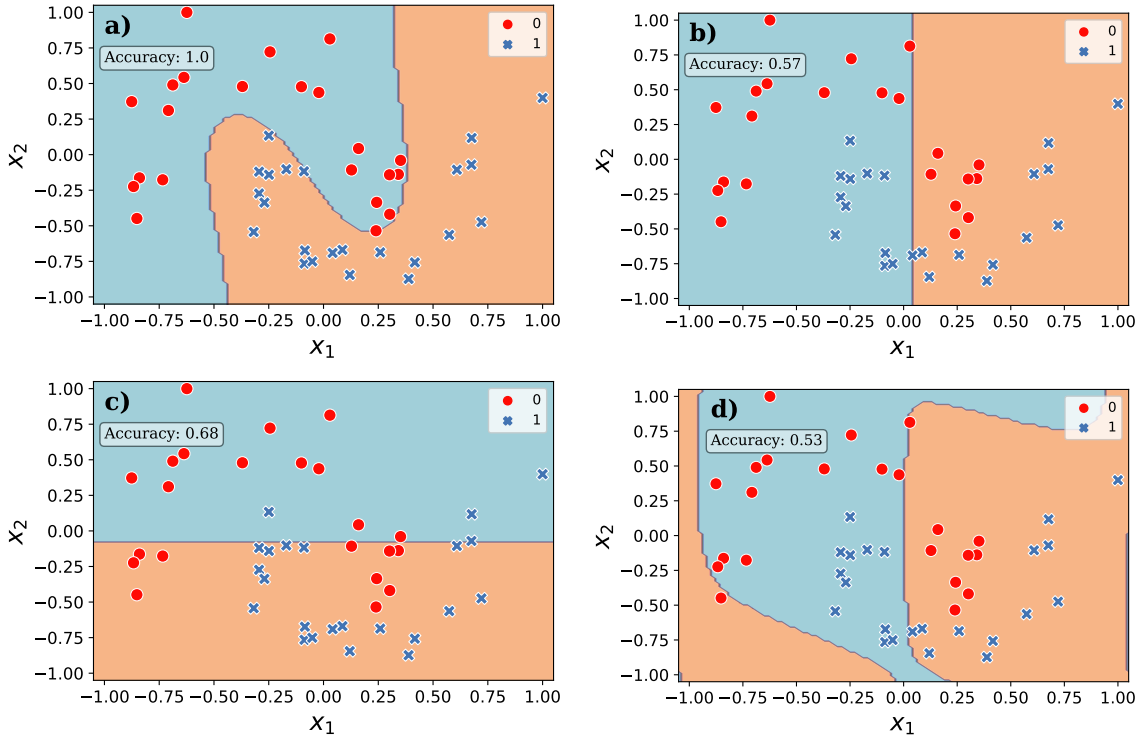


Figure 3.12: a) Data points and prediction boundaries from the full quantum kernel SVM. (b), (c) and (d) are the decision boundaries the circuits provide on the first, second, and third qubits, respectively.

Hence, the genetic algorithm searches for the optimal combination of the separate kernels such that the product of all of them produces the desired distribution, regardless of the individual kernels.

3.3.2 Real-world Datasets

The methodology has been demonstrated to be very effective in a nonlinear binary classification task. Now, it is applied to three realistic datasets to analyze its performance on multiclass

classification and on tabular data with higher dimensionality, with the objective of verifying its versatility and applicability in a variety of scenarios.

The first dataset ("Irrigation dataset", 2020) related to *irrigation* includes as features the humidity and temperature factors. The objective of this model is to determine when to open or close an irrigation valve. After the optimization process, it is obtained the best circuit shown in Figure 3.13, which involves only two qubits and no entanglement operators for this binary classification, producing a 1.0 in accuracy. Interestingly, as seen in Table 3.1, an initial matrix is defined for circuit building that allows the use of 25 gates; however, the best circuit needs only 3 to produce a 100% success rate.

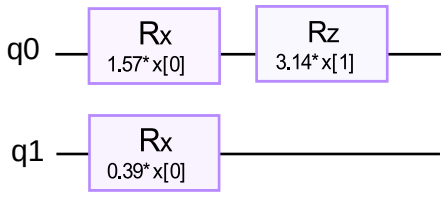


Figure 3.13: Best individual provided automatically from this technique for each supervised learning problem in Table 3.1. IoT irrigation ("Irrigation dataset", 2020).

Our algorithm is applied to a second dataset whose target variable has more than two classes (Tripathi, 2020). The goal of this dataset is to classify among five *drugs types* based on factors such as age, gender, and medical factors such as blood pressure, cholesterol levels, and sodium-potassium ratio, as shown in Table 3.1. Figure 3.14 shows the best circuit, which provides 1.0 accuracy using only one entanglement operator.

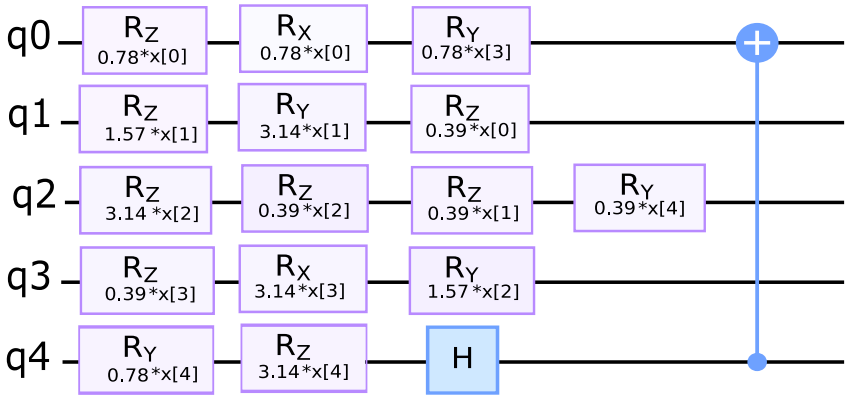


Figure 3.14: Best individual for Drug classification (Tripathi, 2020).

Finally, our methodology is applied to a high-dimensional tabular dataset related to Parkinson's disease, consisting of twenty-two features (Little et al., 2007). The objective is to verify that the technique is still effective, reducing the circuit size without losing accuracy compared to templates where each variable uses a qubit. The discovered model shows that using features multiple times is more beneficial than simply inserting them once and applying a classifier as previously explored (Pérez-Salinas et al., 2020). The quantum circuit depicted in Figure 3.15 exhibits twenty-two variables distributed across only eight qubits, resulting in a

circuit with high expressive power. Furthermore, the lack of correlations between qubits makes it similar to a machine learning algorithm inspired by quantum components (Altares-López et al., 2022).

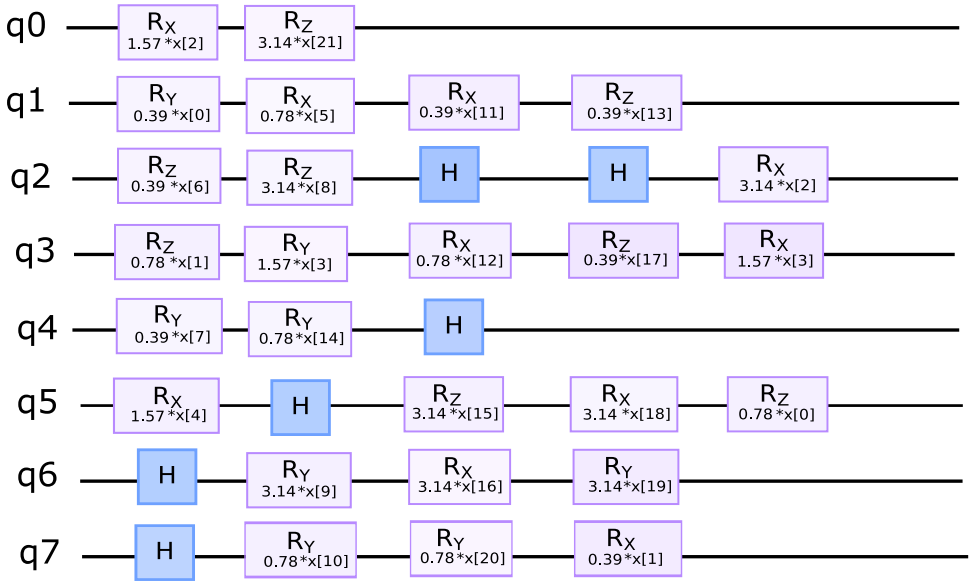


Figure 3.15: Best individual provided automatically for Parkinson problem (Little et al., 2007).

Table 3.1: Results from applying the genetic engineering of QSVM to other model problems in supervised machine learning. The accuracy of other classical methods is also provided: a k-NN, a linear support vector machine, and an SVM with a polynomial kernel of degree 2 (poly).

	Parkinson	IoT irrigation	Drug classification
Accuracy QSVM (*test*)	**1.0**	**1.0**	**1.0**
Num. CNOT	**0**	**0**	**1**
Generations	5000	1000	500
# attributes	22	2	5
# classes	2	2	5
Max qubits	8	5	5
Max depth	15	5	5
Mutation probability (p_{mut})	0.7	0.7	0.7
Mutation ind. prob. (p_{ind})	0.2	0.2	0.2
Crossover prob. (p_{cross})	0.3	0.3	0.3
k-NN accuracy	0.82	1.0	0.70
SVM (linear) accuracy	0.89	1.0	0.87
SVM (poly - 2) accuracy	0.89	1.0	0.65

Table 3.1 presents the three problems analyzed, detailing the number of dataset features, the hyperparameters of the genetic algorithm, the number of entangling operators, and the accuracy obtained on the test set. As can be seen, the technique demonstrates good

performance in all cases. Comparison with classical methods, including kernel methods, also highlights an advantage of the QSVM technique, especially in multiclass classification.

3.4 Conclusions

This chapter describes the design of a global optimization system using multiobjective genetic algorithms (NSGA-II) to automatically generate quantum circuits for machine learning applications. This methodology allows the creation of quantum circuits from binary strings, taking into account crucial aspects such as topology, operators, and parameters, as well as the encoding of classical variables into quantum states.

The optimization process not only focuses on finding the most accurate quantum circuit but also on minimizing its complexity. This duality in the objectives ensures that the generated circuits are highly effective and computationally tractable. The methodology allows the generation of solutions that balance accuracy and simplicity, which is fundamental for their practical application in machine learning. As a result, the generated quantum models, which use few or no entanglement gates, represent a *quantum-inspired machine-learning technique*.

By analyzing the resulting qubits individually and combining their contributions, it is possible to determine how each component contributes to the overall model kernel. Combined qubits in the model capture perfect nonlinear patterns, while solo qubits cannot.

Building on the application of this technique to tabular data, the next chapter will explore how to adapt and extend this methodology to handle higher-complexity data, such as images. We will explore strategies for encoding high-dimensional data in quantum circuits, addressing the challenges of high complexity and volume of information.

Chapter 4

Quantum Circuits for Single-channel Image Classification

In the previous chapter, we have presented a methodology for automatically generating quantum circuits for tabular data. This method offers an efficient solution, but its applicability is limited when dealing with high-dimensional data. Images, in particular, pose a challenge due to their large number of pixels and complex structure, making direct representation or processing in a quantum circuit challenging.

In this chapter, we will describe a specialization of the previously described methodology, which focuses on the application of quantum circuits in image classification. Given the challenge of working with the high dimensionality data, we explore different strategies that include the use of dimensionality reduction techniques such as Principal Component Analysis (PCA) and transfer learning methods. These methods seek to extract as much relevant information as possible from the images, allowing a more manageable and optimized representation for quantum models. In addition, we optimize the principal component analysis method within the genetic process allowing them to automatically adjust to each individual and thus achieve the maximum precision.

The first section presents improvements in individuals' genetic codes to make universal sets of gates available and extends the number of angles to increase accuracy. It also introduces the generic training algorithm that will be specialized for each dimensionality reduction method in section 4.2. Section 4.3 applies our algorithms to several medical imaging datasets, such as brain tumors and COVID-19, and analyzes the obtained results regarding the accuracy and complexity of the obtained models. Finally, it presents the chapter's main conclusions.

4.1 Quantum Single-Channel Image Classification

Image processing for classification tasks in quantum machine learning presents several challenges due to its high dimensionality, which hinders classification efficiency, as it is not possible to embed the entire image directly into a quantum circuit. Dimensionality reduction techniques simplify the data while preserving the most important features for efficient classification,

making them a suitable choice for use in quantum machine learning. Therefore, this section presents a specialization of the method described in Chapter 3 for single-channel images, combining automatic circuit generation and dimensionality reduction techniques. In section 4.1.2, the general algorithm and data preprocessing required for principal component analysis are explained, while section 4.2 describes the dimensionality reduction methods and their application. In addition, to obtain a higher performance of the technique, the previous genetic code is improved by adding a larger number of gates and angles, forming a universal set of gates (section 2.2.1) that enhances the search for solutions during the genetic process.

4.1.1 Universal Quantum Circuits DNA

In the previous chapter, a strategy was introduced to represent quantum circuits as binary strings, effective for tabular data but limited in operator range, not achieving a universal gate set as Section 2.2 explains. The method uses two bits for encoding θ angles, allowing only 2^2 combinations, which limits the system's flexibility in exploring the Hilbert space. Considering the complexity of the data addressed in this chapter, the coding of the circuit has been enhanced to include a universal gate set, along with a wider array of angles.

The binary coding strategy is highly scalable, as the potential states multiply exponentially with the number of bits added in the strings; two bits can give rise to four 2^2 different combinations, four bits can provide sixteen 2^4 combinations, and so on. Therefore, to have available a more significant number of operators or to introduce additional angles, more bits are added to the binary string that encodes the circuit; in this case, the number of bits needed to encode a gate increases to seven: $s_0, s_1, s_2, s_3, s_4, s_5$ and s_6. This slight modification makes it possible to code eight different types of gates and sixteen different angles. The first three bits, s_0, s_1, s_2, correspond to the gate type, while the remaining four bits, s_3, s_4, s_5, s_6, correspond to the parameter θ, as shown in the genetic code in Figure 4.1.

Gate	Code
$R_x(\theta; x)$	000
$R_y(\theta; x)$	001
$R_z(\theta; x)$	011
CNOT	101
Identity	100
$R_x(\theta)$	110
$R_y(\theta)$	111
$R_z(\theta)$	010

Angle	Code
$\pi/8$	0000
$\pi/4$	0001
$3\pi/8$	0010
$\pi/2$	0011
$5\pi/8$	0100
$3\pi/4$	0101
$7\pi/8$	0110
π	0111
$9\pi/8$	1000
$5\pi/4$	1001
$11\pi/8$	1010
$3\pi/2$	1011
$13\pi/8$	1100
$7\pi/4$	1101
$15\pi/8$	1110
2π	1111

Figure 4.1: Genetic Code for universal gates set.

Local quantum gates on the three axes R_x, R_y, R_z are encoded, together with entanglement operators (CNOT), to correlate consecutive qubits and identity operators that allow the quantum classifier's complexity to be reduced during evolution, as shown in Figure 4.2c. Parameterized rotations are represented by operators on the three axes $R_x(\theta; x)$, $R_y(\theta; x)$, $R_z(\theta; x)$, where θ is the angle and x is an input datapoint from the dataset X. The expression $R_\beta(\theta_i; x_k) = \exp(-i\theta_i x_k \sigma^\beta)$ describes the general rotation around a β axis with angle θ_i, where σ^β is the Pauli matrix describing the rotation direction. This extension also encodes fixed-angle rotations on the three axes as $R_x(\theta)$, $R_y(\theta)$, and $R_z(\theta)$, with $R_\beta(\theta_i) = \exp(-i\theta_i \sigma^\beta)$ as a general expression, where they do not depend on vector X but only on the parameters θ_i. These gates, together with the two-qubit gate, made the gates set to be universal and capable of exploring the entire Hilbert space as described in section 2.2.2. When considering the parameters θ, a uniform sample is established within the interval $[\pi/8, 2\pi]$ based on the $n\pi/8$ rule, where n ranges from one to sixteen due to the use of 2^4 in encoding individuals.

The total chain length of the individual is calculated as $MxNx7$ based on this genetic code (Figure 4.2). Here, M represents the maximum number of qubits, and N represents the maximum number of layers. This encoding forms the basis of the circuit. The next sections of this chapter will explain how to include dimensionality reduction methods needed for image processing in individuals by increasing the chain length.

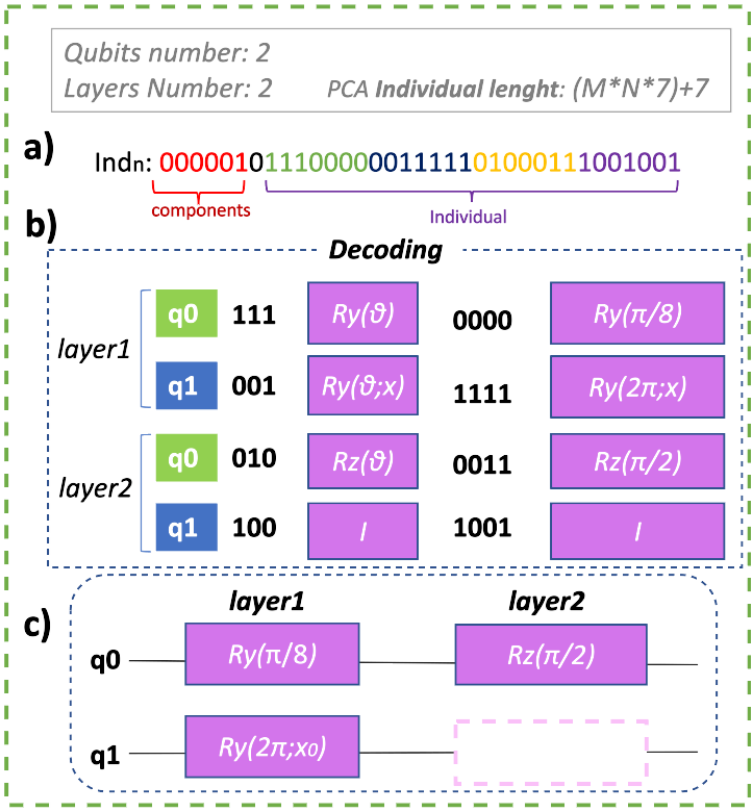


Figure 4.2: Quantum circuit encoding: PCA approach. a) The individual is created based on the number of qubits and layers. b) Coding phase. c) Quantum circuit assembly.

4.1.2 Training Algorithm

The algorithm is divided into two phases: the first is preparing the data to avoid bias, and the second is the fitness function. Data are divided into train and test first to avoid data leakage effects. This phenomenon occurs when information from the test set affects the training process, biasing the model evaluation. The splitting is stratified according to the target variable to balance the distribution of classes in both sets. Data are standardized in the interval [-1, +1], a necessary process when applying some dimensionality reduction methods (section 4.2), since significant differences in the ranges of the variables produce biases in the results towards variables with higher values. Following data normalization, a dimensionality reduction method can be applied, as will be explained in section 4.2. This dimensionality reduction process is fitted with the training data, and the transformation is applied to the test set, as depicted in Figure 4.3b, obtaining data that can be used directly in training and testing individuals.

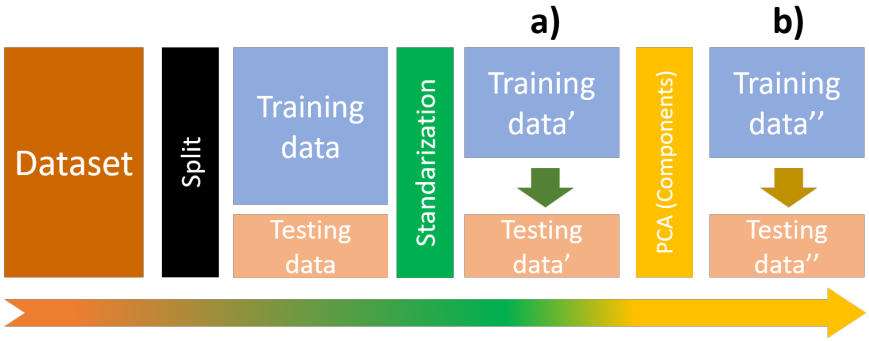


Figure 4.3: Data preprocessing steps. a) Both sets are standardized. b) The standardized data fit the PCA in the training set, applying the transformation to the test data. Train data in blue, and test data in orange. The processing data are arranged vertically, differentiating each by colors black, green, and yellow. The standardization and PCA are set with the train data, and the transformation to test is applied as indicated by the arrows.

For the fitness function, the binary strings are randomly generated considering the maximum number of qubits (M) and layers (N) predefined by the user, as shown in Figure 4.4a. The algorithm starts by decoding each binary string into a quantum circuit, as illustrated in Figure 4.4; the first step is to perform seven-by-seven splits across the string to separate each gate in the circuit. The first three bits of each split are used to decode the gate type (green in Figure 4.4b-c), while the last four bits decode the θ angle if needed (blue in Figure 4.4b-c).

The generated quantum circuit embeds the input data into the quantum rotation gates and then trains the model. The model is fitted with the training data only, as shown in Figure 4.5, leaving the test set out of the training process used as regularization within the system. The fitness objectives are the accuracy on the test set derived from the fitted model and the complexity of the circuit, balancing these two objectives in Equation 3.12.

Once the optimization process is completed, the best individuals are stored in the Pareto front. The goal is to obtain the optimal classifier with the minimum size while maintaining

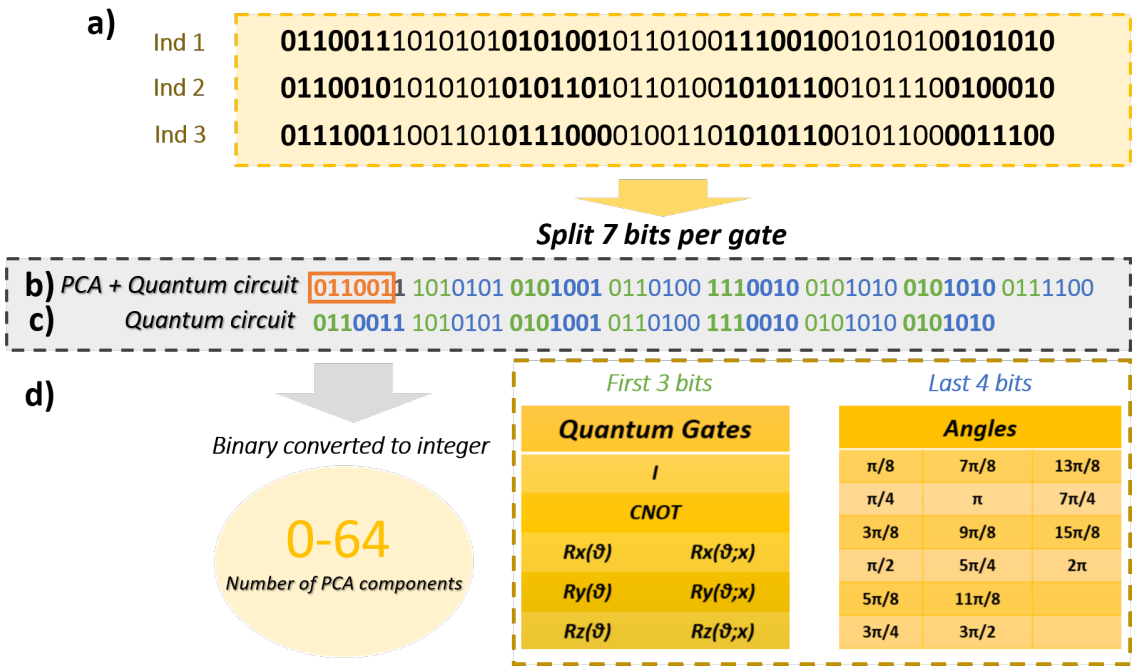


Figure 4.4: a) Population initialized. b) Encoding strategy embeds dimensionality reduction method within individuals. c) Explores transfer learning methods. d) In approach b), the first 6 bits of individual represent PCA components.

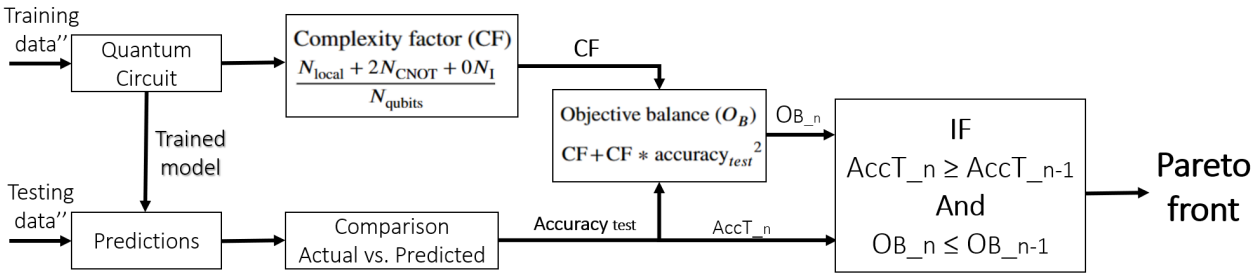


Figure 4.5: Block diagram of the evaluation process.

predictive performance by prioritizing the accuracy metric (Altares-López et al., 2023a, 2024a). The solutions that have belonged to the Pareto front throughout the evolution are visualized in a curve as shown in Figure 4.6a, where the two targets face each other. The red dots are the individuals that have provided the highest accuracy for the classification task with the lowest complexity in each case. The individuals with the highest precision also tend to have the highest complexity, as shown in Figure 4.6b. The green dot in Figure 4.6b represents the best-selected individual from the Pareto front (Van Veldhuizen, Lamont, et al., 1998), which exhibits superior accuracy without significantly higher complexity compared to less accurate individuals from earlier generations.

Interestingly, it is observed that the later generations tend to produce individuals with higher precision and lower complexity as evolution progresses, as shown in Figure 4.6a. Compared to the earlier generations' less accurate individuals, there is a significant difference in complexity,

going from 16 to as low as 11 in the last generations (Figure 4.6). Therefore, the optimal evolutionary individual is defined as the circuit with the highest accuracy. Among those with the same accuracy, preference is given to the one with the lowest complexity.

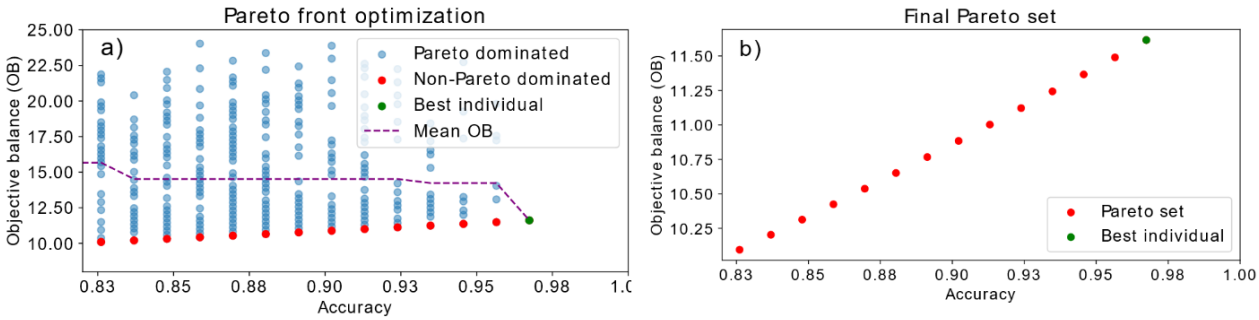


Figure 4.6: Pareto front solutions. a) Optimization of the Pareto front along the evolution. b) Final Pareto front and best individual based on accuracy and complexity.

4.2 Dimensionality for Quantum Image Processing

Images are too large for handling in near-term quantum circuits. Fortunately, relevant features from images can be extracted using dimensionality reduction tools. These tools convert the high-resolution single-channel picture into a small set of numbers that a quantum algorithm can easily process. The following sections describe two reduction techniques used with the genetic algorithm from section 4.1.

4.2.1 Principal Component Analysis

Principal Component Analysis is a widely used method to reduce the dimensionality of a data set and extract the essential features while keeping as much information as possible (Pearson, 1901; Wold et al., 1987). This method transforms a dataset with correlated variables into a new set of uncorrelated variables called *principal components*. This transformation simplifies the complexity of the dataset while preserving the essential information present in the original variables.

Given an L-dimensional dataset, this method reduces the data to a smaller number of variables, denoted as Y (where $|Y| \ll L$), which captures a significant portion of the original variance. These variables, known as *components*, represent a compressed version of the data in a lower-dimensional space. It uses the covariance matrix to identify the redundant information of the possible correlated variables in the L dimensional dataset. Normally, a dataset can be considered a matrix X consisting of n observations and p variables. The matrix X is organized as an $n \times p$ matrix, where each element x_{ij} represents the value of the i-th observation for the j-th variable,

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix} \tag{4.1}$$

When applying principal component analysis, it is essential to standardize the data to ensure that all variables contribute equally to the analysis. The standardized data Z results from subtracting the mean vector ($\bar{X}$) from each feature vector and normalizing it by the standard deviation vector (σ),

$$Z = \frac{(X - \bar{X})}{\sigma}. \tag{4.2}$$

Next, the covariance matrix of the standardized dataset C,

$$C = \frac{1}{n} Z^T Z. \tag{4.3}$$

The eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_p)$ and eigenvectors $(v_1, v_2, \ldots, v_p)$ of the covariance matrix C will be used to inform the dimensionality reduction. These eigenvalues and eigenvectors provide important information about the variability and directions of the principal components, $PC1 = Zv_1, \quad PC2 = Zv_2, \quad \ldots, \quad PC_p = Zv_p$. The results are linear combinations from the original variables with associated weights, capturing different proportions of the total variance: the *principal components*. For dimensionality reduction, selecting the first k principal components and obtaining a new X' matrix with these k columns is possible. This reduced data set retains the most important information while discarding the less significant dimensions. Inside this PCA method, the Singular Value Decomposition (SVD) solver (Halko et al., 2010; Martinsson et al., 2011; Sadek, 2012) is a randomized algorithm due to its high computational efficiency and its ability to handle sparse matrices.

The strategy of PCA is incorporated into the global optimization method described in chapter 3, optimizing the quantum circuits and the principal components from the images. This is enabled by extending the length of the individuals to integrate the number of components, optimizing both simultaneously: quantum circuit and dimensionality reduction. Seven more bits are added to the standard size of the individual $M \times N \times 7$ to encode the maximum number of principal components of the PCA method, as shown in Figure 4.7a. Thus, the new individual length is calculated as *(MxNx7)+7*, where the first six bits are taken and used to determine the number of components used in the PCA. Considering the size limitations of these circuits, adding six bits to encode the reduction method leads to a maximum of 64 components, which is considerable. However, the technique is scalable, and this value can be enhanced by augmenting the number of bits used to encode the reduction method part. Since the number of components that appear in the individuals are random values, those with zero or one component are applied a *death penalty* in the fitness function because it must be at least two.

Next, the optimization process is presented in steps, followed by pseudocode for better understanding and reproducibility. **Step 1.** The algorithm begins by defining the maximum circuit size as $M \times N \times 7 + 7$ (see Figure 4.7a). In **Step 1.1** and generating the initial population P consisting of random binary chains of that length in **Step 1.2**. In **Step 1.3**, the dataset X, y is split into training and testing sets, and in **Step 1.4**, the training set is standardized, and the transformation is applied to the test set. In **Step 2**, the initial population is evaluated: for each individual *ind*, in **Step 2.1**, the number of components represented in $ind[0:6]$ are extracted and converted into an integer value (see Figure 4.7a); in

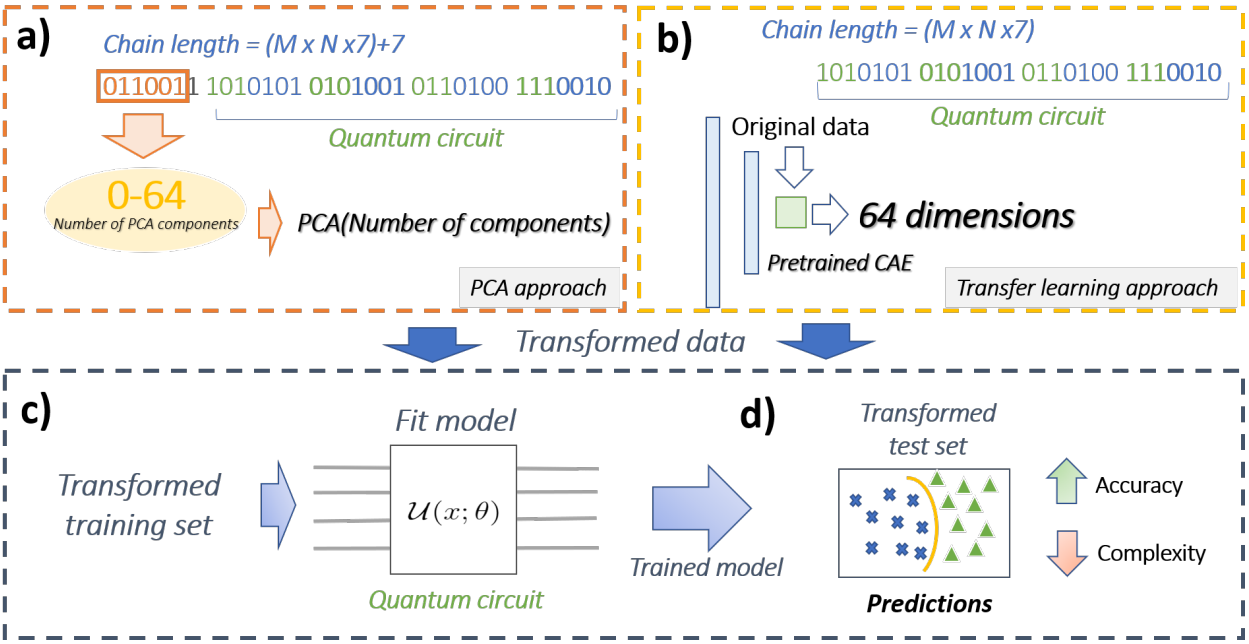


Figure 4.7: a) PCA approach. b) A pretrained CAE model with a fixed 64-dimensional output is applied. c) The training procedure is the same for both cases.

Step 2.2, the quantum circuit C is extracted from $ind[7 : (M \times N \times 7)]$. In **Step 2.3**, if the number of components is 0 or 1, a penalty is applied to the individual, but otherwise, in **Step 2.4**, a PCA model is fitted with the selected number of components using the training data X_{train}, and the test set X_{test} is transformed in **Step 2.5**. In **Step 2.6**, a quantum feature map f_I is constructed based on the circuit C and X_{train} (see Figure 4.2), and in **Step 2.7**, a quantum support vector machine model M_I is trained using f_I and the labels y_{train}. Then, in **Step 2.8**, the model M_I is used to predict labels $\hat{y}$ for X_{test}, and in **Step 2.9**, the accuracy a_{test} is calculated. In **Step 2.10**, the complexity is computed, and in **Step 2.11**, we calculate the objective balance (OB) by combining the complexity with the square of the accuracy. After evaluating the initial population, in **Step 3**, the Pareto front is identified as the set of non-dominated solutions (line 26 in Figure 4.8). The algorithm then enters an iterative loop, starting in **Step 4**, where until the stopping conditions are met, in **Step 4.1**, parents are selected from the population (line 28 in Figure 4.8), in **Step 4.2**, genetic operators such as crossover and mutation are applied to generate offspring (line 29 in Figure 4.8), and in **Step 4.3**, the fitness of the new individuals is evaluated following the same steps in the fitness function (lines 1-18 in Figure 4.8). If an individual improves the objective balance or accuracy without worsening other metrics, it is added to the Pareto front (lines 30-35 in Figure 4.8). **Step 5**. if the stop conditions are reached, the algorithm returns the Pareto front with the optimal solutions. Otherwise, the loop is repeated from **step 4**.

function EVALUATEFITNESS($ind, X_{train}, X_{test}, y_{train}, y_{test}$)
    PCA components $\leftarrow ind[0:6]$ (binary)
    N components $\leftarrow$ Convert from binary to integer
    Extract quantum circuit C from $ind[7:(M \times N \times 7)]$
    if number of PCA components is 0 or 1 **then**
        Apply death penalty to ind
    else
        Fit PCA(N components) with X_{train}
        Apply transformation to X_{test}
    end if
    Construct QFMap f_I from C and X_{train}
    Train QSVM model M_I on f_I and y_{train}
    Use M_I to predict labels $\hat{y}$ for X_{test}
    Compute accuracy a_{test}
    Compute complexity factor (CF) as $CF \leftarrow N_l + 2N_{CNOT} + 0N_{id}/M$
    Compute objective balance (OB) as $OB \leftarrow CF + CF \cdot a_{test}^2$
    return a_{test} and OB
end function

Require: Dataset X, labels y, number of qubits M, number of layers N, stop conditions SC
Ensure: Pareto front with optimal individuals.
Define maximum circuit *size* $M \times N \times 7 + 7$
Generate initial population P with random binary chains within the maximum size: $P \leftarrow ind_1, ind_2, ..., ind_{popsize}$, where ind_i is a binary chain of length *size*
$X_{train}, X_{test}, y_{train}, y_{test} \leftarrow$ split X and y into training and test sets
Standardize using the training set and apply it to the test set.
for each ind in P **do**
    EvaluateFitness(ind)
end for
Pareto front $\leftarrow$ non-dominated solutions
while not SC met **do**
    Select parents from P
    Apply genetic operators to generate offspring O
    for each individual ind in P **do**
        $\{a_I, OB\} \leftarrow$ **EvaluateFitness**($ind, X_{train}, X_{test}, y_{train}, y_{test}$)
    end for
    if OB improves or a_{test} improves and no value worsens **then**
        Add ind to Pareto front
    end if
end while
return Pareto front

Figure 4.8: Evolutionary Quantum Inspired ML: PCA approach

4.2.2 Convolutional Autoencoder

This section presents a second alternative using pre-trained models, which allows the application of *transfer learning* to leverage knowledge from other datasets. In this case, a widely used model is selected: the Convolutional Autoencoder (CAE) (LeCun et al., 2015). Autoencoders are a type of neural network designed to accurately reconstruct inputs into outputs while compressing the data into a latent space of reduced dimensions, preserving as much of the original information as possible. This architecture, as shown in Figure 4.9, is composed of three distinct elements:

- **Encoder**: In this stage, the dimensionality is reduced by fully connected layers until the latent space, composed of a defined number of variables, is reached.
- **Latent space**: It is the intermediate point in Figure 4.9, where the lower dimensionality representations are stored.
- **Decoder**: It attempts to reproduce the original images as similarly as possible from the lower dimensionality vector constructed in the latent space.

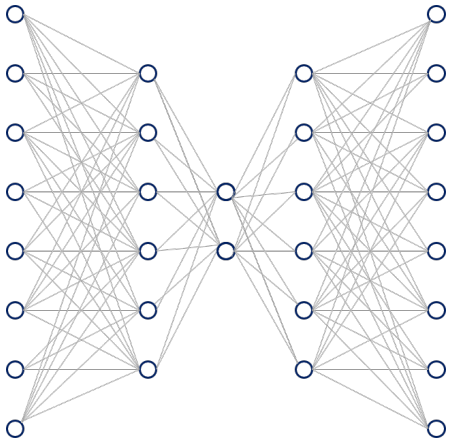


Figure 4.9: Autoencoder architecture

The Equation 4.4 represents the fundamental basis of an autoencoder, where f_{enc} refers to the set of layers that transform the input X into a compressed representation h and f_{dec} refers to the decoding layers that reverse the process by transforming the compressed representation h into the reconstructed output X'. These networks are trained to minimize the cost function L as the similarity between the input and output generated in the latent space increases. Thus, when the cost function approaches zero, the output images are similar to the input images, meaning that the low dimensionality vector has captured the pattern to produce the images.

$$h = f_{\text{enc}}(X); \quad X' = f_{\text{dec}}(h). \tag{4.4}$$

Since data in this chapter are images, simple convolutional layers are used as encoders and decoders in the autoencoder to reduce dimensionality (Y. Zhang, 2018). Note that, unlike the case of PCA, the implementation of the dimensionality reduction is done externally to the

genetic algorithm as shown in Figure 4.7, so the entire individual *MxNx7* is used to encode the quantum circuit, and no further bits need to be added.

The method is presented in a series of ten steps for better understanding. Pseudocode is also included to enhance reproducibility. **Step 1:** The pre-trained Convolutional Autoencoder (CAE) network is implemented to extract information, using a latent space with 64 dimensions as described in Section 4.2.2. **Step 2.** The circuit size is calculated as $M \times N \times 7$, where M is the number of qubits and N is the number of layers. **Step 3.** The dataset X and labels y are split into training and test sets, and both sets are standardized using the training set transformation. **Step 4.** The pre-trained model is applied to the images, obtaining 64 fixed dimensions from the original data, as described in Section 4.2.2. The new features X'_{train} and X'_{test} are obtained after applying the model to the original data X_{train} and X_{test}. **Step 5.** The initial population is generated with random binary chains of length, where each individual in the population represents a solution. **Step 6:** The function *EvaluateFitness* is called for each individual in the population P to evaluate its performance based on accuracy and quantum circuit complexity. **Step 7.** The Pareto front is initialized with the non-dominated solutions from the initial population. **Step 8.** The algorithm enters a loop where, until the stopping conditions are met, the following steps are executed: **Step 8.1:** Parents are selected from the population based on their fitness. **Step 8.2:** Genetic operators, such as crossover and mutation, are applied to generate offspring. **Step 8.3:** The fitness of each individual in the new population is evaluated using the function *EvaluateFitness*, where the accuracy on the test set and objective balance are computed. **Step 8.4:** If the objective balance improves or improves accuracy without worsening other metrics, the individual is added to the Pareto front. **Step 9:** The loop continues from **step 8** until the stopping conditions are met. **Step 10:** The algorithm returns the Pareto front with the optimal individuals.

function EVALUATEFITNESS($ind, X'_{train}, X'_{test}, y_{train}, y_{test}$)
    Extract quantum circuit C from ind
    Construct QFMap f_I from C and X'_{train}
    Train QSVM model M_I on f_I and y_{train}
    Use M_I to predict labels $\hat{y}$ for X'_{test}
    Compute accuracy a_{test} as the fraction of correct predictions
    Compute complexity factor as $CF \leftarrow N_l + 2N_{CNOT} + 0N_{id}/M$
    Compute objective balance (OB) as $OB \leftarrow CF + CF \cdot a_{test}^2$
    return a_{test} and OB
end function

Require: Pre-trained network CAE, labels y, number of qubits M, number of layers N, stop conditions SC
Ensure: Pareto front with optimal individuals.
Implement CAE neural network to extract information, take encoding part with 64 dimensions, (See Section 4.2.2).
Calculate circuit $size \leftarrow M \times N \times 7$
$X_{train}, X_{test}, y_{train}, y_{test} \leftarrow$ split X and y into training and test sets
Standardize both using training set and apply to test set.
Apply a pre-trained model of the encoding to images, obtaining 64 fixed dimensions.
The new features (X'_{train}, X'_{test}) are obtained after applying the model (See Section 4.2.2) to the original data (X_{train}, X_{test}).
Generate initial population P with random binary chains within the maximum size: $P \leftarrow ind_1, ind_2, ..., ind_{popsize}$, where ind_i is a binary chain of length $size$
for each ind in P **do**
    EvaluateFitness(ind)
end for
Pareto front $\leftarrow$ non-dominated solutions
while not SC met **do**
    Select parents from P
    Apply genetic operators to generate offspring O
    for each ind in P **do**
        $\{a_{test}, OB\} \leftarrow$ **EvaluateFitness**($ind, X'_{train}, X'_{test}, y_{train}, y_{test}$)
    end for
    if OB improves or a_{test} improves and no value worsens **then**
        Add ind to Pareto front
    end if
end while
return Pareto front

Figure 4.10: Evolutionary Quantum Inspired ML: Transfer learning approach

4.3 Real-world Applications

We extensively benchmarked our algorithm using various medical imaging datasets to validate the performance of its PCA and autoencoder variants. Below, we describe the datasets and the quantum and classical models used. Additionally, we provide detailed information on the genetic process and its hyperparameters, including mutation probabilities, crossover methods, and the number of generations.

4.3.1 Data and Model Setup

The benchmark uses two datasets from the medical domain. The first dataset consists of X-ray images of lungs from SARS-CoV-2 affected patients (COVID-19) and healthy lungs (Raikote, 2020). Two categories are extracted from the entire dataset, building a binary dataset consisting of 227 images as seen in Figure 4.11a-b. The second medical dataset consists of 253 magnetic resonance imaging (MRI) images with two classes: brain tumors and healthy brain structures (Chakrabarty, 2019) as shown in Figure 4.11c-d, where the objective is to identify abnormalities in brain structure.

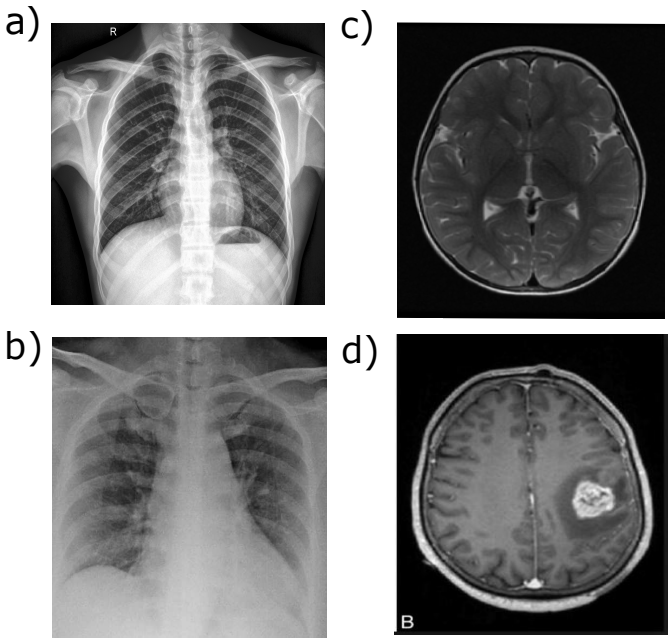


Figure 4.11: Samples from the datasets used in the experiments, displaying pneumonia and brain tumor.

Before utilizing the methods described in section 4.2, it is essential to standardize the images to ensure consistency and reliability in the model's applications. For Principal Component Analysis (PCA), images are resized to 250x250 pixels. In contrast, for the Convolutional Autoencoder (CAE) model, images are substantially reduced to 28x28 pixels. Although this reduction may seem drastic, it is strategically chosen to avoid the need for a large, resource-intensive deep learning model and to examine the effectiveness of this CAE+QSVM

system when used with low-resolution images. This approach is particularly useful in scenarios where computational resources are limited or when the aim is to understand the robustness and limitations of the model in handling low-resolution data. In this chapter, we use PCA and CAE techniques for data dimensionality reduction and feature learning. PCA, as explained in section 4.2.1, requires an explicit number of components definition from each individual, while convolutional autoencoder follows a fixed model approach (section 4.2.2). The CAE model processes 28x28 images, utilizing four convolutional layers with (3,3) kernels, non-linear activation functions such as *ReLu* to find non-linear patterns and a 0.5 dropout rate (Xu, 2015). A (2,2) max-pooling step follows each layer. The model undergoes 250 training epochs using the Adam optimizer and binary cross-entropy loss (Z. Zhang, 2018). This CAE model provides a 64-dimensional latent space vector, which is the input of the quantum circuits (Figure 4.7b).

As described in section 4.1.1, we build the quantum circuits based on an MxN matrix. In this case, a maximum of six qubits (M) and eleven layers (N) are predefined, allowing all data to be embedded in the circuit with 66 gates since the dimensionality reduction methods have been limited to 64 dimensions (See section 4.1.1). In addition, the performance of quantum models is compared with that of a classical neural network: a Multilayer Perceptron (MLP) (LeCun et al., 2015). This model is trained with data compressed to 64 dimensions using PCA from the original images, ensuring a fair comparison with respect to the quantum approaches, thus also ensuring that the dimensionality reduction method does not do all the classification tasks. This classical network consists of a hidden layer with six neurons (equivalent to the predefined number of qubits in quantum models) and a two-neuron output with softmax activation. We adjust the hyperparameters and employ *early stopping* to determine the optimal learning rate to prevent overfitting. The cost function used is binary cross-entropy (Bishop, 2006). For optimization, the model employs the *Adam* optimizer (Kingma and Ba, 2017), which initializes weights randomly, introducing a stochastic element in the models.

Our algorithm also requires a precise adjustment of the hyperparameters of the genetic algorithms since their efficiency, speed of convergence, and quality of the solutions depend to a large extent on this, as explained in Chapter 3. Thus, we test several hyperparameter values over ten generations to determine the optimal. The values that achieve the highest accuracy in these generations are the ones taken to run the genetic algorithm. For the COVID-19 dataset, we found that the optimal values are 0.4 for p_{mut} and 0.6 for p_{cross}. For the brain tumor dataset, the optimal values are 0.3 for p_{mut} and 0.7 for p_{cross}. Once the optimal values are defined, the genetic algorithms proceed to a more extensive and thorough iterative phase. In this subsequent stage, a significantly higher number of iterations are performed: 2000 generations. This prolonged iteration process is crucial for thoroughly exploring the solution space, thus identifying the most effective models for handling the tasks specific to the COVID-19 and brain tumor datasets. Due to the stochastic nature of genetic algorithms, each experiment is repeated ten times to guarantee result stability and achieve the most accurate solution approximation.

4.3.2 Quantum Models Performance

The PCA+QSVM system is applied to the COVID-19 dataset, generating the circuit and fine-tuning the number of components simultaneously, finding that the optimal number of features is two. Figure 4.12a illustrates the best quantum circuit, highlighting that within the model's gates, only the variables x_0 and x_1 are present, directly corresponding to the principal components from PCA. This best individual, with an accuracy of 0.967, appears in generation 1763 as shown in the lower part of Figure 4.12a and corresponds to the one selected on the Pareto front of Figure 4.6a,b. This quantum circuit, created in the last generations, presents high precision and low complexity compared to the initially defined maximum of 6x11 (MxN); thus, complexities decrease while precision is maximized during the evolution. The application of the CAE+QSVM model to the COVID-19 dataset uses only single-qubit gates, as in PCA+QSVM, as shown in Figure 4.12b, implying that the model behaves as a single unitary function with no correlations between qubits, making these models simulatable on classical machines. Although the pre-trained model provides 64 features, the optimal circuit uses only 18 variables selected by the evolutionary system, obtaining the best accuracy of 0.919 for this dataset. As commented in section 4.3.1, a multilayer perceptron is also applied as a prediction baseline, achieving an accuracy of 0.945.

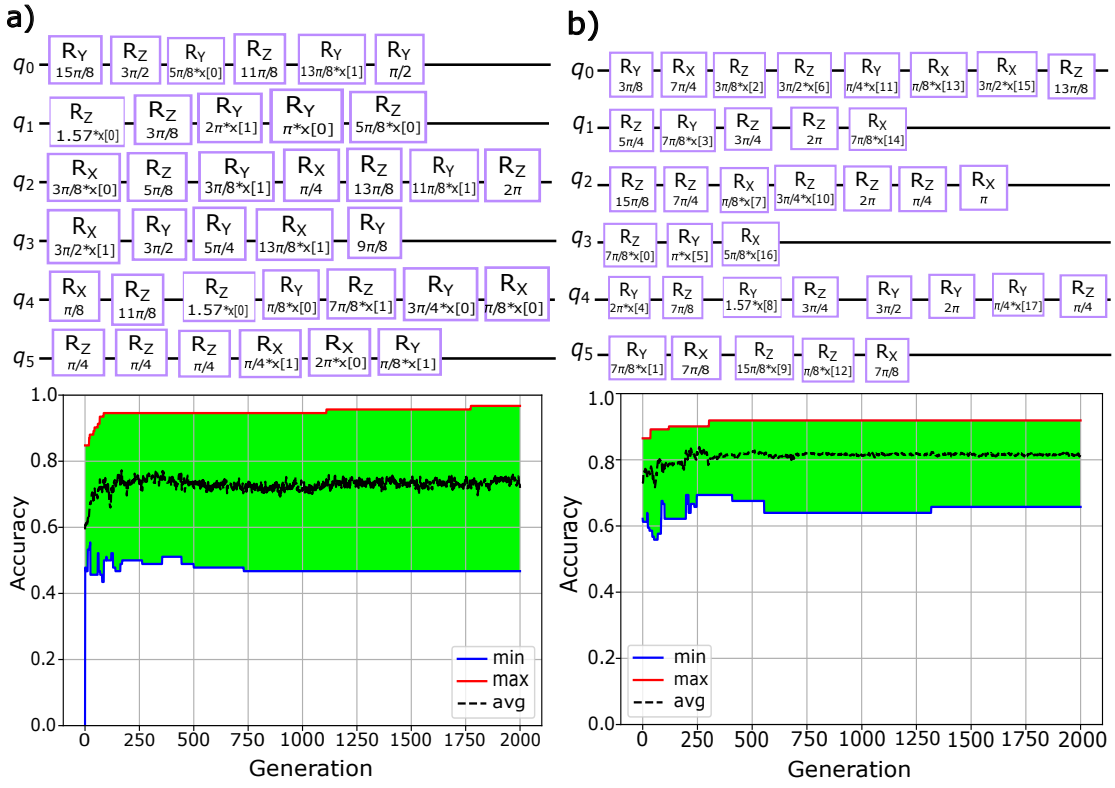


Figure 4.12: Best circuits for COVID-19 dataset. a) PCA+QSVM with 2 components. b) CAE+QSVM with a 64-dimensional vector input from a convolutional autoencoder network. The lower part shows the evolution of the individuals along the evolution. In green, the whole space of solutions is tested for accuracy. The maximum values found in each generation are in red.

Our algorithms are also applied to the brain tumors dataset. When applying the PCA+QSVM technique, the best circuit obtained is the one in Figure 4.13a. In the PCA+QSVM system, the decoding process for the most optimal individual reveals that the best number of components is 45. However, in the circuit depicted in Figure 4.13a, only 15 components are used, replacing the remaining ones with fixed rotations, resulting in a maximum accuracy of 0.859. Our algorithm is able to discover that using the first 15 components, which have the highest eigenvalues and explain most of the variance in the data (see section 4.2.1), is enough and that increasing this number does not improve model performance. This double-dimensionality reduction allows for improved computational efficiency, faster training, and reduced risk of overfitting. However, even if only 15 components are used in the final model, the original 45 features capture a complete picture of the variance present in the data. They are, therefore, all necessary for the correct reconstruction of the input data. The CAE system uses only 27 dimensions, resulting in a selection of variables over the latent space. The best circuit, shown in Figure 4.13b, achieves an accuracy of 0.846. In the same way, as with the COVID-19 case, the MLP is used as a benchmark, achieving an accuracy of 0.684 on the test.

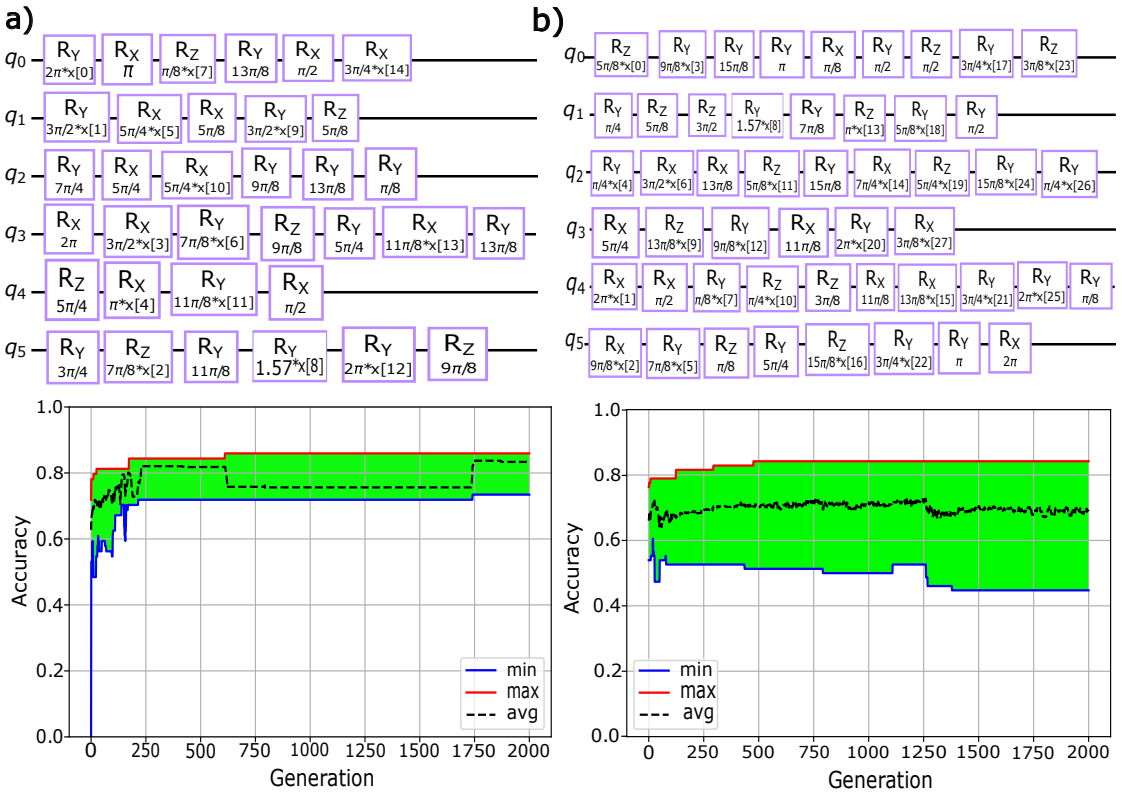


Figure 4.13: Best circuits for the brain tumor dataset. a) PCA+QSVM has 45 components with no entanglements. b) CAE+QSVM with a 64-dimensional vector input from a convolutional autoencoder network with no correlations among qubits. At the bottom, the individuals with the highest values found per generation are on the red line.

Table 4.1 compares the best accuracy results and a statistical analysis in terms of mean and variance to evaluate the stability of the results in the test set for each use case and method.

Numerically, the best quantum-inspired model obtained by applying the PCA+QSVM system shows better results in both use cases in terms of the mean. All generated circuits are entanglement-free; thus, they can be considered as *quantum-inspired image classification models*. In classical ML models, the complexity is determined by the number of trainable parameters, while in quantum circuits, it corresponds to the angles of each gate since each rotational gate is associated with a specific angle θ. In the context of COVID-19, both approaches show similar complexities with 36 parameters. However, the PCA+QSVM system shows superior performance with lower complexity, differing by 14 parameters. This can be attributed to the automatic optimization of the dimensionality reduction method, which incorporates the number of components within the individual when developing the circuit. In contrast, the number of dimensions remains permanently fixed in a pre-trained system. The quantum-inspired model achieves better results with fewer parameters than similar models in classical machine learning.

Table 4.1: Results of the accuracy on the test set and models complexities.

Use Case	Models	Best	Mean	Variance	Parameters
COVID-19	**PCA(2) + QSVM**	**0.967**	**0.959**	7.8 10^{-5}	**36/66**
	CAE(64) + QSVM	0.919	0.898	0.000187	**36/66**
	PCA(64) + MLP	0.945	-	-	404
Brain tumor	**PCA(45) + QSVM**	**0.859**	**0.841**	0.000243	**34/66**
	CAE(64) + QSVM	0.846	0.814	0.000972	50/66
	PCA(64) + MLP	0.684	-	-	404

4.4 Conclusions

Quantum circuits present limitations in their size, directly impacting the models' efficiency. In the previous chapter, we developed a methodology allowing us to generate quantum circuits for tabular data automatically. However, this methodology is not suitable for high-dimensional data, since it is not possible to embed a complete image in a quantum circuit. For this reason, this chapter has described a specialization that enables the application of quantum circuits to images. By using preprocessing techniques, such as dimensionality reduction methods, we can reduce the number of features and, thus, embed the data in the circuits to perform a classification. Specifically, the PCA method and a transfer learning approach based on a convolutional autoencoder network, which extracts the vector in the latent space, have been used. In the case of PCA, a procedure has been described to simultaneously optimize the number of components within the genetic procedure, reaching optimal component values that maintain maximum variance. Both approaches achieve competitive results using fewer parameters than traditional methods, obtaining circuits without correlations between qubits and, therefore, classically simulatable (*quantum-inspired models*).

The promising results open several directions for further research. One approach, which will be explored in Chapter 5, is to increase the number of channels, such as incorporating color or multispectral data, which is particularly relevant in applications like satellite imaging. Additionally, instead of relying on dimensionality reduction, future models could leverage

local information present in the original high-dimensional space, enhancing the representation of complex features, as discussed in Chapter 2. Finally, expanding the analysis to sequences of images rather than individual ones allows for temporal and contextual insights. These extensions will be explored in more detail in the remaining of this thesis, where we discuss advanced strategies for model enhancement and practical applications in various domains.

Chapter 5

Quantum Pixel Segmentation in Multiband Images

In the previous chapter, we discussed the processing of single-channel images, which, although they present dimensionality issues, are composed of a single matrix. In contrast, multiband images present significant challenges due to their high dimensionality and structure of several correlated matrices, as illustrated in Figure 5.1. Processing this data type from a classical machine learning perspective often requires sophisticated algorithms, such as convolutional neural networks and vision transformers (LeCun et al., 2015; Z. Liu et al., 2021).

When quantum circuits are used for processing such images, the complexity increases considerably due to the high number of pixels and bands, requiring structures with many qubits, higher resource demands, complex training needs, and possible compromises in accuracy. However, the challenges posed by multiband data processing can be addressed more effectively by focusing on local problems, which affect specific sections of the image rather than the image as a whole. This localized approach allows for more efficient management of the inherent complexity, optimizing the resources available to quantum models.

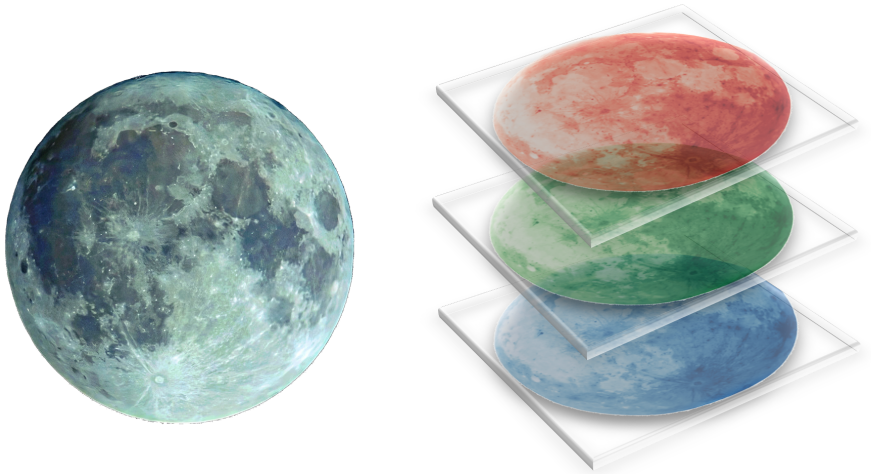


Figure 5.1: RGB image and its decomposition in channels.

This chapter focuses on image segmentation, a locally tractable problem of grouping regions of images with similar characteristics, such as color, texture, or shape (Haralick and Shapiro, 1985). From the classical point of view, several methods have been developed to address this challenge, ranging from traditional techniques to more sophisticated approaches that take advantage of deep learning, such as convolutional neural networks, U-Net (Ronneberger et al., 2015), Mask R-CNN (He et al., 2017) and Transformers (Vaswani et al., 2017). These methods have diverse applications, including medical imaging to identify and delineate anatomical structures such as tumors or organs (Pham et al., 2000), agricultural applications to monitor crop growth and detect weeds (Guijarro et al., 2011), and autonomous vehicles for recognition and classification of objects in their environment (Sun et al., 2006), among other uses. However, this problem has not been sufficiently investigated using quantum methods due to technical and computational constraints, which has limited these models' development and practical application in real scenarios.

This chapter in Section 5.1 details strategies for capturing image information so that quantum models can process the information, describing how these strategies can be integrated seamlessly into the methodology described in Chapter 3. Section 5.2 delves into the practical application of these methods for segmenting real-world images, such as vegetation cover and solar panels. Finally, the chapter summarizes the main results and conclusions from the previous sections.

5.1 Efficient Quantum Pixel-level Image Segmentation

This image segmentation method relies on a supervised approach, meaning it requires a dataset consisting of RGB input images and their corresponding segmented images, which act as ground-truth labels. We use a pixel-based method to extract relevant information, where the bands or channels of each pixel are captured and assigned binary labels (1 or 0) based on their characteristics. Using the methodology described in Section 5.1.1, each pixel in the input image is classified into one of two categories, resulting in a fully segmented image. Pixels representing areas of interest are labeled as 1, while those outside the region of interest are labeled as 0. This binary classification allows for precise identification of the target regions within the image.

5.1.1 Multiband Pixel Dataset Assembly

Next, we describe a methodology for generating an image pixel dataset labeled with a binary classification label, determining whether or not it belongs to a segmented region. For this procedure, we need two datasets: the first contains the original images, and the second consists of the segmented images that will serve as training targets where the areas to be identified are highlighted. From these image sets, pixel-level datasets composed of pixels are constructed. These sets are constructed by randomly sampling pixels from the image set, taking all the bands of each selected pixel, for example, red (R), green (G), and blue (B) (line 5 in Figure 5.2). It is crucial to account for all images in the set to capture color, lighting, and shading variations. This ensures the model is trained on the necessary variability to generalize effectively across different scenarios. Pixels are selected within the image dimensions

(resolution $Q \times S$), ensuring that the coordinates (x_i, y_j) of each pixel satisfy $x_i \leq Q$ and $y_j \leq S$, generating a set of pixels with D consisting of all selected pixels and bands as features.

Require: Original dataset RGB, Segmented images dataset seg, Number of pixels (N)

1: Initialize empty dataset $D \leftarrow \emptyset$
2: **while** $|D| < N$ **do**
3: **for** each image I in RGB **do**
4: Randomly select a pixel position (x_i, y_j)
5: $Pixel_{RGB} \leftarrow (\mathrm{R}_{\mathrm{RGB}_I}[x_i][y_j], \mathrm{G}_{\mathrm{RGB}_I}[x_i][y_j], \mathrm{B}_{\mathrm{RGB}_I}[x_i][y_j])$
6: $Pixel_{seg} \leftarrow \mathrm{seg}_I[x_i][y_j]$
7: $D \leftarrow Pixel_{seg}, Pixel_{RGB}$
8: **end for**
9: **end while**
10: $n = \min(|D_{C_1}|, |D_{C_2}|)$
11: $D_{balanced} \leftarrow \{$Randomly select n pixels from C_1 and n pixels from $C_2\}$
12: **return** $D_{balanced}$

Figure 5.2: Pixel segmentation strategy for multiband images in quantum model processing. C_1 represents areas of interest (labeled as 1), and C_2 represents the background or non-interest areas (labeled as 0).

Subsequently, we take the labels from the segmented images of the initial set to build the target part of the D set (see Figure 5.2 in line 6). To correctly relate the RGB pixels with the binary ones, the pixels from the same positions x_i, y_j in the pre-segmented images are extracted. This forms a dataset D, where each data point represents a pixel. The image bands serve as features (inputs), and the binary value of each pixel is the target. The process is algorithmically detailed in the Figure 5.2.

Due to the nature of the segmentation task, which seeks to distinguish specific pixels in whole images, it is common to find an imbalance in these classes. This means that there are more pixels from one category (e.g., the background) than another (e.g., the object of interest), which can unbalance the dataset and make it difficult to learn the model. To avoid this, we do a *stratified subsampling* to the D pixel data set based on the labels to ensure a balanced distribution of both classes in the pixel set. With this stratified subsampling, we reduce the number of samples of the majority class to achieve a balanced distribution of the classes (see lines 12 and 13 in code shown in Figure 5.2). This new pixel-balanced data set is also stratified by the target into training and test sets. Figure 5.3a-b depicts the extraction of image characteristics to form the pixel set and the genetic algorithm for evolving the models to obtain the best segmenter. The details of this genetic procedure and models' training are described in next sections.

5.1.2 Quantum Pixel Segmentation Models Training

The automatic generation of quantum models operates on the balanced training and test sets of pixels, generating a classifier that implements the segmentation task. In this process, the pixel components are treated as tabular data during training,g and the coding of individuals follows the genetic code described in chapter 4. The fitness metrics remain unchanged, considering both the accuracy of the binary pixel classification and the complexity of the quantum circuit, with penalties applied to the CNOT gates as defined in Equation 3.12. Figure 5.3b, shows the genetic procedure for training the models and obtaining the best quantum segmented (Altares-López et al., 2024c).

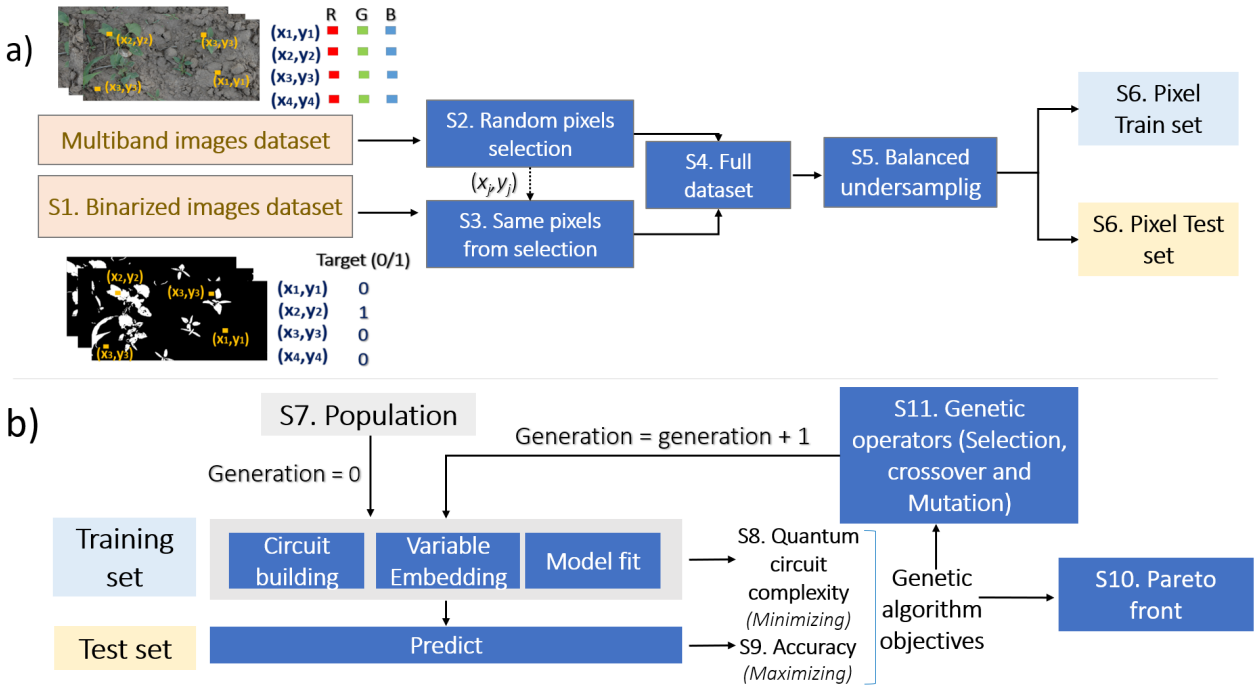


Figure 5.3: General scheme for multiband image processing and handling in quantum circuits for pixel segmentation.

This process is presented in steps, from the information extraction of the images to the genetic process. **Step 1**. Binary images are created manually with tools, such as magic wand from `Photoshop`, so they can be used as a target for training. **Step 2**. The entire image set is traversed, and pixels are randomly selected. The color properties corresponding to the RGB channels are extracted for each pixel. **Step 3**. From the same images and pixel positions, we extract the binary values from the binarized set. **Step 4**. A dataset is obtained, consisting of a vector with three features (RGB) and a binary target for each pixel. **Step 5**: Once the pixel selection is complete, we perform a stratified undersampling. Randomly, we select n pixels from the complete pixel dataset for each class $C1$ and $C2$ to obtain a balanced dataset. **Step 6**: The data is split into train and test sets in a stratified manner based on the target. **Step 7**. A population is generated randomly (see Chapter 3). The coding of the variables is carried out as explained in section 4.1.1. **Steps 8-9**. The individuals are evaluated, and the complexity metrics are obtained and calculated using equations 3.11 and 3.12, along with

the accuracy on the test set. **Step 10**. If at least one of the objectives improves compared to the previous generation without the other deteriorating, the individual is saved in the Pareto front. The stopping conditions are checked. If they are met, the best individuals are returned; otherwise, an iterative loop is entered. **Step 11**. Genetic operators are applied to generate the next generation, the offspring. This offspring is evaluated, and the Pareto front is updated with the best solutions. If the stopping conditions are met, the solutions are returned; otherwise, the loop repeats until the conditions are satisfied.

5.2 Applications

The segmentation algorithm has been benchmarked with two sets of images, one of the vegetation canopies and another of solar panels from aerial images (Benhamadi, 2021; Jiang et al., 2021). The following text describes the datasets, modeling, and performance of the automatically generated quantum-inspired models on these problems.

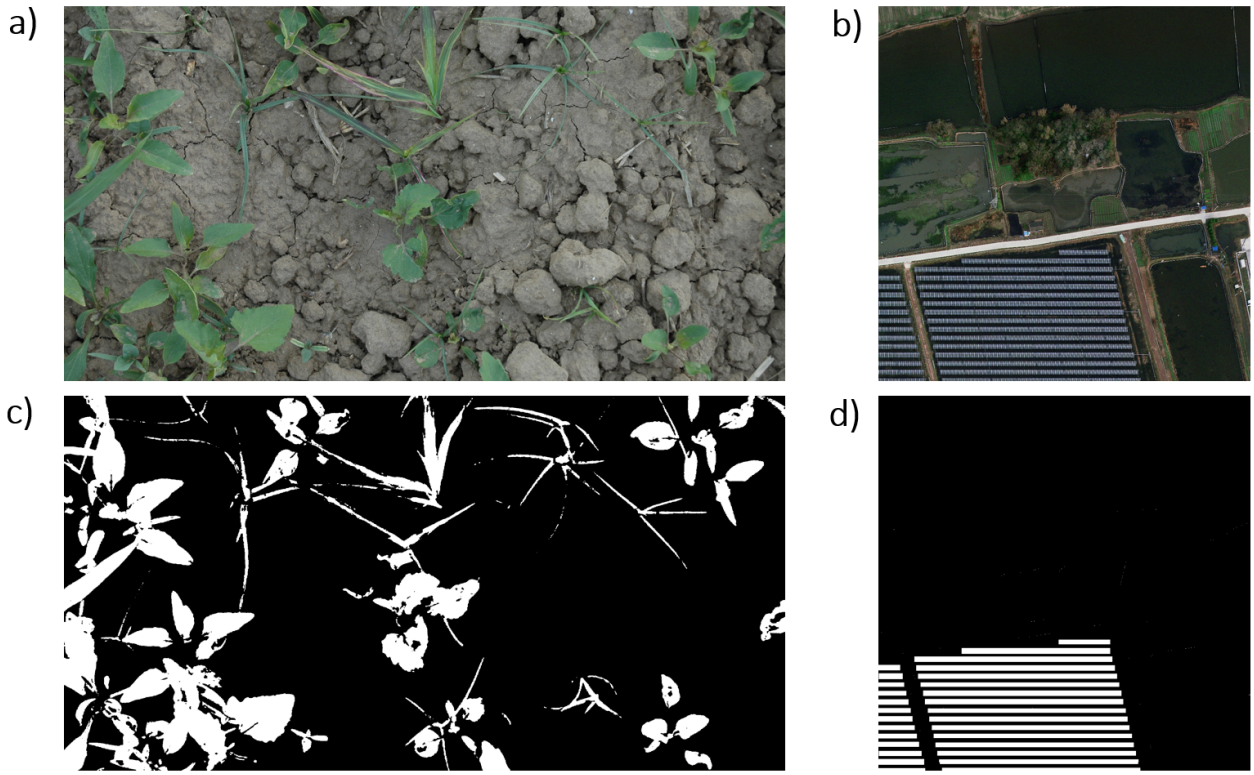


Figure 5.4: Images from the training dataset of both datasets. a) and c) depict the RGB and its binarized image cover vegetation dataset, while b) and d) represent the dataset of solar panels, both RGB and binarized.

5.2.1 Data Setup

In agriculture, canopy segmentation is widely used to address several complex challenges (Burgos-Artizzu et al., 2011; Guijarro et al., 2011), such as fertilizer application management or

Figure 5.5: Solar panels dataset (Benhamadi, 2021; Jiang et al., 2021). a) Real aerial images b) Approximate ground truth provided in the dataset. c) Ground truth more precise built for this study.

crop yield assessment (Payne et al., 2013; Sartin et al., 2014) among others. Considering these interests, this case has been chosen to segment the vegetation cover. Two sets of images are used: real RGB images and their segmented images, consisting of thirty-five RGB images with a resolution of 1300×2500. RGB images show vegetation cover and bare ground features with no lighting control (Figure 5.4a). In the segmented images (see Figure 5.4c), the vegetation cover consists of white pixels —class 1 —, and the background cover or bare soil is black —class 0 —. Typically, these processed images are constructed by an expert with the help of a computer application that allows for manual selection and removal of pixels from the image, which is a time-consuming and tedious task.

A dataset of aerial images with solar panels is also used to demonstrate the generalizability and universality of the proposed technique. In the original dataset, the images present masks of the total area covered by solar panels (Benhamadi, 2021; Jiang et al., 2021). This ground truth does not distinguish between the solar panels and the surrounding vegetation, which is a limitation for a solid identification of the panels and other tasks, such as determining the actual area covered by those panels by color feature. To overcome this problem, we have developed our ground truth, manually labeling the images to include only the solar panels, as depicted in Figure 5.5b-c. The complete process results in 21 aerial images with a resolution of 1024×1024 pixels, with their corresponding binary masks at the same resolution.

This last use case presents significant complexity due to several factors: (i) Unlike the cover vegetation use case, where the background is uniform, the backgrounds in the images are highly diverse, as can be seen in Figure 5.7d, which requires the model to be able to learn and generalize appropriately in a variety of different scenarios. (ii) The color of the solar panels changes with height since the illumination varies, as shown in Figure 5.7d. These lighting variations force the classifier to develop a more complex color filter within the same class of objects. (iii) The vegetation cover images are taken closer to the plants and show more significant color differences with the background than the aerial images of the solar panels.

5.2.2 Model Setup

The quantum-inspired segmentation model is developed using a training and test set. For the vegetation cover dataset, we take 15,000 pixels to build the global D pixel dataset. Next, we apply a *random stratified undersampling*, a data preprocessing technique that reduces the number of examples in majority classes randomly while maintaining the proportion of classes in the data set, to address the problem of classification imbalance. This undersampling allows us to maintain the fitness objectives used in previous chapters. We take 1,500 pixels from D for class 0 and another 1,500 pixels with a target value of 1, indicating pixels with properties to be identified.

In the case of solar panels, as there is a greater diversity of illumination and backgrounds in the images, selecting as many pixels as possible is important to have more information so the models can be trained and generalized. Therefore, we take 25,000 pixels to construct D and apply subsampling to have 5,000 pixels for each class. The proper splitting of training and testing sets is also important to avoid data leakage effects in the models, where the inadvertent influence of test set information during training can lead to overly optimistic estimates of model performance.

The genetic procedure depicted in Figure 5.3b is applied to both use cases. Similar to previous chapters, ten previous iterations are performed to define the best hyperparameters for the genetic algorithm. For the vegetation cover, the best values are a mutation probability of 0.2 and a crossover probability of 0.7, while the optimal mutation and crossover probabilities in the solar panel data set are 0.1 and 0.8, respectively. Considering these optimal hyperparameters, we run a genetic algorithm with an initial population of fifty individuals that evolves over two thousand generations. The *MxN* matrix that builds the quantum circuit (Chapter 4) has been set to 6x6, allowing for a maximum of thirty-six gates in both cases.

The quality of the obtained quantum segmenters is compared against classical kernel methods such as SVM. For a fair comparison between the quantum and classical models, we also performed hyperparameter optimizations of the classical models using two techniques commonly used for these tasks, *gridsearch* and *genetic algorithms* (Samadzadegan et al., 2010; Syarif et al., 2016). Three hyperparameters are considered for optimizing these models: (i) the kernel type, (ii) the C parameter, and (iii) γ. The kernel type (polynomial, linear, radial, or sigmoid) determines how the data is transformed for classification. The C parameter adjusts the trade-off between maximizing the margin that separates classes and minimizing classification errors, with a high value prioritizing accuracy on the training data. On the other hand, γ controls the influence of training examples; a low γ creates a smoother model, while a high γ generates a more localized fit, which can lead to overfitting.

The genetic algorithm used to optimize the classical model aims to maximize accuracy. The crossover and mutation probabilities are set at 0.8 and 0.2 across two thousand generations, respectively. This process is iterated twenty times using a $\mu + \lambda$ strategy. The $\mu + \lambda$ algorithm is an evolutionary strategy for optimization that combines a population of λ new candidate solutions with μ existing solutions. It selects the best individuals from this combined group to create the next generation, balancing exploring the solution space with exploiting promising solutions (Suárez and Galán, 2021).

In the gridsearch process for the vegetation cover, the optimized parameter obtained for C is 9, and for γ the parameter is 0.01, the best kernel being *sigmoid*. The genetic algorithms provide that the best value for C is 2, and the best kernel is also *sigmoid*, with 0.0504 being the value for γ, the most optimal for cover vegetation. The same techniques are applied to the solar panel use case, obtaining from grid search optimization that the best value for C is 9 with a radial basis function kernel and a γ value of 0.1. The genetic optimization shows that the best value C is 89 with a kernel also, radial basis function, and a γ value of 0.995.

5.2.3 Segmentation Performance Analysis

The optimal individuals for each scenario depicted in Figure 5.6a-b are obtained through evolutions. The quantum circuits involve three variables: x_0, x_1, and x_2, representing a pixel's RGB components. The pixel classification, analyzed by the QSVM model (Chapter 2), determines whether the pixel belongs to a segmented region. In both cases, the final quantum circuits have fewer operators than the initial *MxN* matrix set to thirty-six gates, demonstrating the technique's complexity reduction and adaptive nature. Interestingly, both models present low or no correlations between qubits due to the weighting of the complexity factor calculated within the genetic algorithm's fitness function (Equation 3.11). Therefore, these circuits can be considered a single unitary, formed by multiplying these 2x2 matrices, making them classically simulatable models. As a result, the models produced can be seen as quantum-inspired image segmenters.

Once the models are trained, we apply them to new full images to verify the generalizability and performance of the strategy. The size of the validation images for the vegetation canopy images is 150 $\times$ 300 while the solar panel images are 400 $\times$ 400, as shown in Figure 5.7a,b,d,e. The circuits process 45,000 pixels in each image of the vegetation cover and 160,000 pixels in the solar panel case. The quantum model runs over the images, pixel to pixel, using the three components RGB from each pixel as input variables in the quantum-SVM and obtaining a binary classification as output. As shown in Figures 5.7c-f, the circuits can correctly identify the vegetation cover and the solar panels, obtaining accuracy of 96.5 and 94.5, respectively.

Table 5.1 shows the main metrics on the validation images after applying the models. When the training and validation sets present different scenarios, such as one being balanced and the other not, we need to use alternative metrics to assess performance accurately. In the training set, accuracy is possible as a fitness objective because the two classes have similar support. However, full images in segmentation problems, by identifying only the parts of interest, typically contain more pixels from one class than the other. Therefore, we use the metrics *Recall* or *F1-Score* to evaluate the validation results (Bishop, 2006). Although the quantum models discovered by the genetic algorithm do not improve upon the metrics obtained by the classical models in all cases (Table 5.1), the quantum models provide results comparable to those obtained with classical methods, making these circuits a viable alternative. Furthermore, as shown in Table 5.1, quantum models exhibit smaller deviations in the results, making them more stable and robust in some cases than classical models.

Statistical tests are performed to assess whether there are significant differences between the classical and quantum methods. Due to class imbalance in the images, we analyze the results

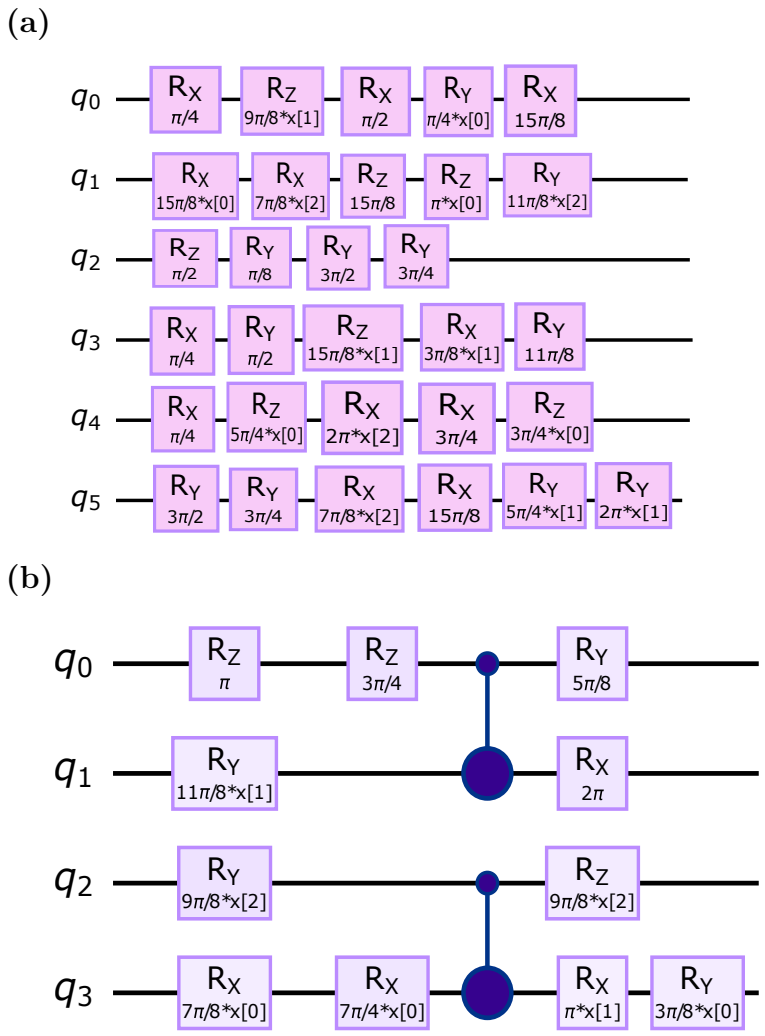


Figure 5.6: The best quantum feature maps produced by genetic algorithm. (a) Quantum-inspired segmenter for cover vegetation segmentation task. (b) Quantum model for solar panel identification. In both cases, the input variables are R, G, and B, corresponding to (x_0, x_1, x_2), respectively.

using a nonparametric test: Friedman's method (Friedman, 1937). The null hypothesis, H_0 states that quantum-inspired models show behaviors similar to those of optimized classical models. Therefore, the alternative hypothesis H_1 implies that significant differences do exist. We compare the models QSVM, SVM - GA, and SVM - Gridsearch in these tests.

The results show no statistically significant differences among the models when considering the three metrics from Table 5.1; hence, the null hypothesis H_0 is accepted: the quantum models have similar behaviors to that achieved by the classical models. Since there is no substantial gap between the optimized classical and quantum kernel models, the models presented in Figure 5.6 can be considered quantum-inspired image segmenters for direct multiband image processing of cover vegetation and solar panels.

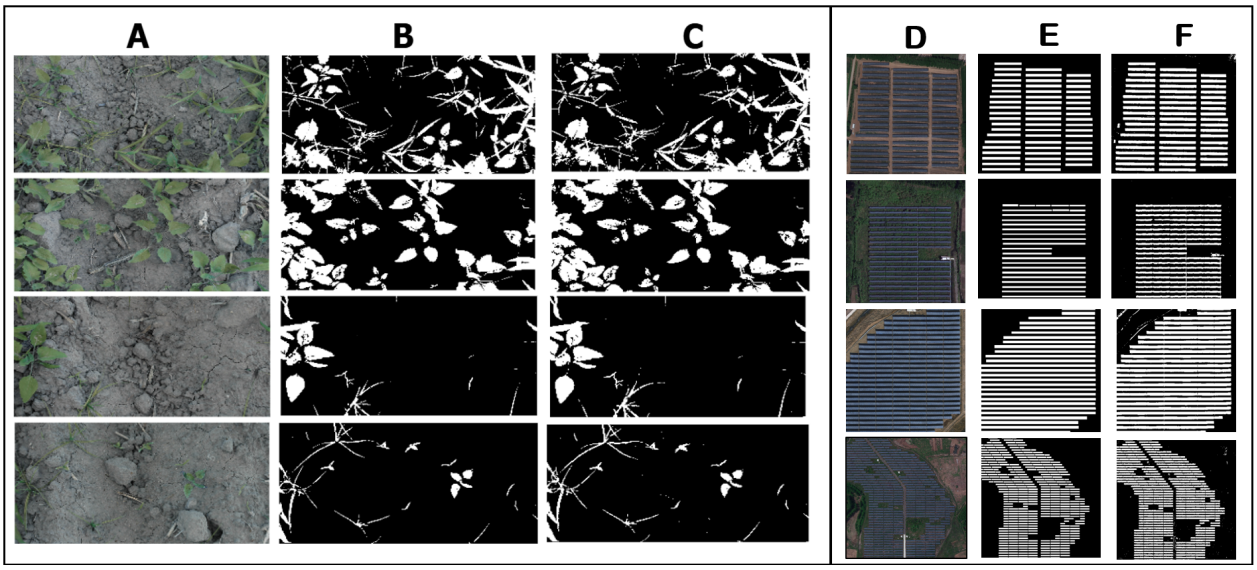


Figure 5.7: Segmentation results by using the best quantum circuits in Figure 5.6a,b. a-d) RGB images to segment the vegetation cover and solar panels. b-e) Binary images; thus, the ground truth for comparison is shown. c-f) The images produced by the quantum-inspired model.

Table 5.1: The main metrics with their respective standard deviations (σ) for the validation images are shown.

Use case	Models	Precision	Recall	F1-score
Cover vegetation	Quantum SVM	**0.965 ± 0.025**	**0.983 ± 0.013**	**0.973 ±0.006**
	SVM - GA	**0.965 ± 0.010**	0.958± 0.031	0.963 ± 0.014
	SVM - Gridsearch	**0.965 ± 0.010**	0.957± 0.031	0.963 ± 0.014
Solar panels	Quantum SVM	0.945 ± 0.008	0.944 ± 0.009	0.944 ±0.009
	SVM - GA	**0.946 ± 0.011**	**0.945± 0.015**	**0.945± 0.015**
	SVM - Gridsearch	0.945 ± 0.010	**0.945± 0.015**	0.944± 0.015

5.3 Conclusions

Quantum models face significant challenges when embedding images, especially in multiband image processing. The number of data points per pixel increases with each additional channel, significantly amplifying the complexity of the information that needs to be processed. Unlike classical models, quantum circuits have difficulty efficiently handling this vast amount of data, making image processing a demanding task.

This chapter presents strategies for multiband image processing using ad-hoc quantum circuits in response to this challenge. It focuses on multiband image segmentation, a key challenge in computer vision. The models are trained at the pixel level, which allows them to accurately capture the relationships between colors, intensities, and illumination conditions.

The obtained results are promising, with complete segmentations reaching accuracies above 94% in all cases evaluated. A remarkable aspect of the proposed circuits is their structural

simplicity, characterized by low or no quantum entanglement involvement. This underlines that the models do not rely on deeply quantum properties, such as entanglement, suggesting that they can be considered segmenters inspired by quantum computation, with a high potential to be efficiently implemented in classical architectures.

In the upcoming chapter, we will build upon the strategies discussed in this chapter for image processing. The focus will shift to addressing image sequences with temporal characteristics, extending the complexity. We will examine the challenges associated with capturing spatial relationships between pixels and the temporal variations between consecutive images.

Chapter 6

Hybrid Neural Networks for Processing Sequential Satellite Images

Understanding and predicting temporal patterns is a significant challenge, especially when working with high-resolution satellite imagery. Each image comprises multiple spectral bands, providing a detailed snapshot of a specific location at a specific time. When these images are captured over time, they allow for the analysis of sequences that facilitate observing, understanding, and predicting dynamic changes, such as urban development or vegetation growth. One of the main challenges in satellite image analysis is to create models capable of detecting patterns in spatial and temporal dimensions so that they can be applied to make predictions of their future evolution with the highest possible accuracy. From a classical machine learning point of view, models such as recurrent neural networks and Long-Short Term Memory (LSTM) networks are typically used to model sequential data (Hochreiter and Schmidhuber, 1997), such as time series, capturing temporal dependencies. Convolutional neural networks, on the other hand, typically process spatial data as images (LeCun et al., 2015). By combining CNNs with LSTMs, models can simultaneously capture spatial structures and track their evolution over time, enabling better pattern recognition in satellite image sequences (Shi et al., 2015). These models have proven efficient but are still an evolving field of study.

The previous chapter 5 discussed strategies for processing multispectral images, considering local features. The satellite images discussed in this chapter also present the multiband complexity, amplified by temporal dependencies that must be modeled together with pixel features. This is crucial to comprehensively capture the changes that occur over time. Applying quantum machine learning models to the multidimensional nature of satellite imagery, which encompasses both spatial and temporal relationships, poses several challenges. These include the need to handle large volumes of data and the difficulty of modeling non-linear and complex relationships between spectral bands and time. This chapter explores how hybrid models combine the strengths of classical machine learning techniques with the potential advantages of quantum computing, enabling the capture of temporal patterns in satellite image sequences for crop yield prediction.

The chapter begins in section 6.1 with an explanation of hybrid models, which combine quantum neural networks and perceptrons. Subsequently, section 6.2 describes the process of generating and optimizing these hybrid models based on the methodology of the previous chapters. Here, we explain a second optimization process in which the parameters are refined after the evolutionary process to improve the predictions. In the following sections, we apply these techniques to a real use case to make crop yield predictions in a vineyard area in Spain. To validate the generated models, we apply the models discovered by the optimization system to several random dates to generate complete images and evaluate their performance. Finally, the main conclusions are presented.

6.1 Hybrid Predictors for Sequential Images

Time series have been a central object of study in many disciplines due to their ability to model and forecast time-varying data. The classical approach to analyzing time series is based on statistical techniques, where data are assumed to follow certain properties such as stationarity or time dependence. Among the best-known traditional methods are autoregressive (AR), moving average (MA) models and their combinations (ARMA, ARIMA). These approaches require a deep knowledge of the statistical characteristics of the series, such as autocorrelation, trend, and stationarity, which often complicates their implementation (Hyndman and Athanasopoulos, 2018). In these methods, it is crucial to carefully select models and often manually adjust parameters like autoregression order or differencing to ensure stationarity. Moreover, these models frequently assume linear relationships between time variables, which may limit their ability to capture more complex patterns.

In recent years, the rise of machine learning has expanded the tools available for time series analysis (Masini et al., 2023). Models such as decision trees, neural networks, and, more recently, models based on recurrent network architectures and transformers have overcome some of the limitations of the classical approach by learning from large volumes of data without making rigid assumptions about the structure of the series (Lim and Zohren, 2021). While machine learning models offer great flexibility, they also face challenges such as the need for large data volumes, high computational requirements, and the risk of overfitting.

Hybrid models, which combine classical and quantum techniques, take advantage of the best of both worlds: the ability of classical methods to handle complex data and the efficiency of quantum algorithms for certain tasks. This section will describe a methodology to automatically generate hybrid quantum-classical neural networks adapted to this data type, optimizing the performance.

6.1.1 Hybrid Neural Networks for Continuous Values

The fundamentals of quantum neural networks have already been described in section 2.4.2 of this book. Several studies have explored integrating classical and quantum techniques to create hybrid models that combine the advantages of both paradigms (Killoran et al., 2019). In this chapter, hybrid neural networks, which include quantum and classical components, are employed to develop a base architecture, depicted in Figure 6.1b. However, it is important to

note that the time series predictor, based on this hybrid model, has a recurrent structure. This means that, unlike the static model, the predictor explicitly labels the inputs with different *times*, associating each input to a specific time instant and preserving the temporal dependence between them. The recurrent nature of the model, similar to recurrent neural networks (RNNs), allows it to capture sequential and temporal relationships in the data so that predictions at one point in time are based on information from previous steps. The output is also labeled with time stamps, indicating its direct relationship to time and facilitating accurate series prediction.

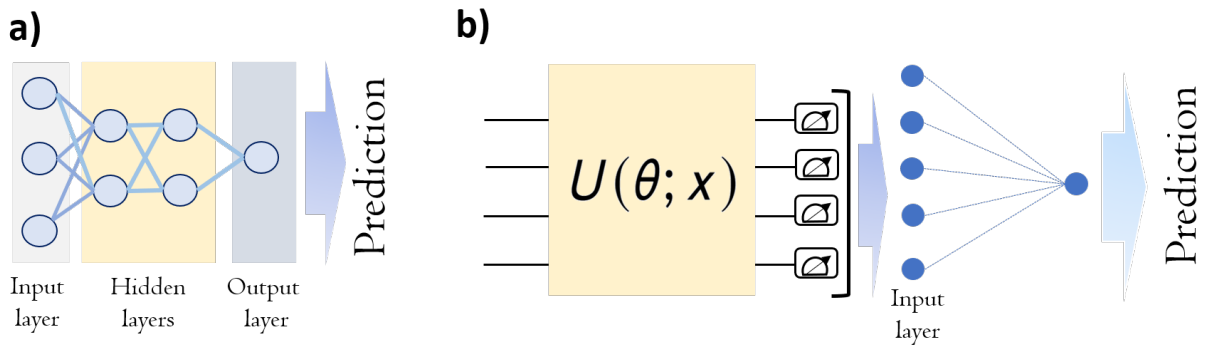


Figure 6.1: Classical and quantum neural networks. a) Classical multilayer perceptron (MLP) consisting of an input layer, two hidden layers, and an output layer. b) Hybrid-quantum neural networks combining a quantum circuit, measurement layer, and a final linear perceptron.

The quantum component of our networks uses a variational quantum circuit (Section 2.4) that incorporates the data reuploading technique as the unitaries in previous chapters (Pérez-Salinas et al., 2020). In this way, the input variables appear at different locations within the quantum circuit, allowing a richer representation of the information being processed and enabling non-linear dependencies among the variables to be found. In these variational quantum circuits, qubits are manipulated by quantum gates, and measurements are made after the operations. These measurements are essential since they allow information to be extracted from quantum circuits, generating probabilities associated with each possible quantum state. Given the probabilistic nature of quantum systems, multiple measurements are necessary to establish a realistic distribution of state probabilities.

The vector of probabilities obtained from this measurement process is then used as the input vector for a perceptron. This perceptron is designed with a linear activation function, which linearly combines the probabilities generated by the quantum component. The classical part does a linear combination of the probabilities, while the quantum circuit captures the non-linear dependencies. This linear combination allows the perceptron to generate a result based on the weighted sum of the probabilities that result in the prediction. It is important to note that in this hybrid system, there are two types of trainable parameters: the quantum parameters, which are adjusted during the training of the quantum circuit, and the specific parameters of the linear perceptron, which are adjusted in the training phase of the classical component of the model. This duality allows both the quantum and classical components to

learn and optimize their performance depending on the input data and the specific prediction tasks.

6.1.2 Training Algorithm

The methodology follows that described in the previous chapters, where NSGA-II is used for model generation. First, as shown in Figure 6.2a, individuals are generated as strings of strings encoding a quantum circuit in the same way as the original framework. Note that the genetic algorithm constructs only the quantum component of the hybrid system. At the same time, the classical architecture remains fixed as a linear perceptron with the goal of producing continuous values from the linear combinations. Models are trained with one set of data and tested with another set. The network output and targets of both sets must be in the same range for the cost function to be calculated correctly, so it is important to normalize the data to the range of values the perceptron activation function can provide.

Since the target variable is continuous, precision cannot be used as a metric in the fitness function. Instead, we use two common error metrics in regression and prediction models, the *mean squared error* (MSE) and the *mean absolute percentage error* (MAPE) as expressed in Equation 6.1 (Makridakis and Hibon, 1979),

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2; \quad \text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\%. \tag{6.1}$$

Although both error metrics are equally suitable, MSE is chosen as the metric for the fitness function.

The second target in the NSGA-II optimization is calculating the complexity of the quantum circuit based on the number of gates and, consequently, the number of parameters. In contrast to the fitness functions shown in previous chapters, the presence of CNOT gates is not penalized in the complexity Equation 3.11. Instead, equal importance is given to CNOT and local gates to achieve an optimal model without constraints due to the complexity of the modeling. Individuals with lower errors and complexities are kept in the Pareto front (Figure 6.2d). In this manner, in optimization terms, the problem is transformed from a Max-Min in previous chapters to a Min-Min. From the Pareto front, the circuit with the lowest error and the lowest complexity is selected.

The optimal individual is still subject to a refinement phase that further tunes the model parameters. Initially, the gate parameters are uniformly distributed discrete values. In the final solution, these parameters come close to minimizing the cost function, although they may not be ideal. Therefore, a new step is added that adjusts the variational parameters to further reduce the error depicted in Figure 6.2f. The fine-tuning of the global minimum is implemented with a gradient-based method, preserving the optimal structure of the first optimization, as shown in Figure 6.2g. Note that the initial weights used in this second optimization step are those extracted from the genetic optimization since they are assumed to be close to the maximum performance (Altares-López et al., 2024b).

The strategy of implementing a global, gradient-free optimization, followed by a fine-tuned, gradient-based optimization, effectively avoids the issues commonly associated with *barren*

plateaus. This approach ensures that the initial structure is established through the first optimization phase, allowing the parameters to be refined close to their initial values during the subsequent phase, thereby achieving highly accurate models and avoiding the detrimental effects of *barren plateaus*.

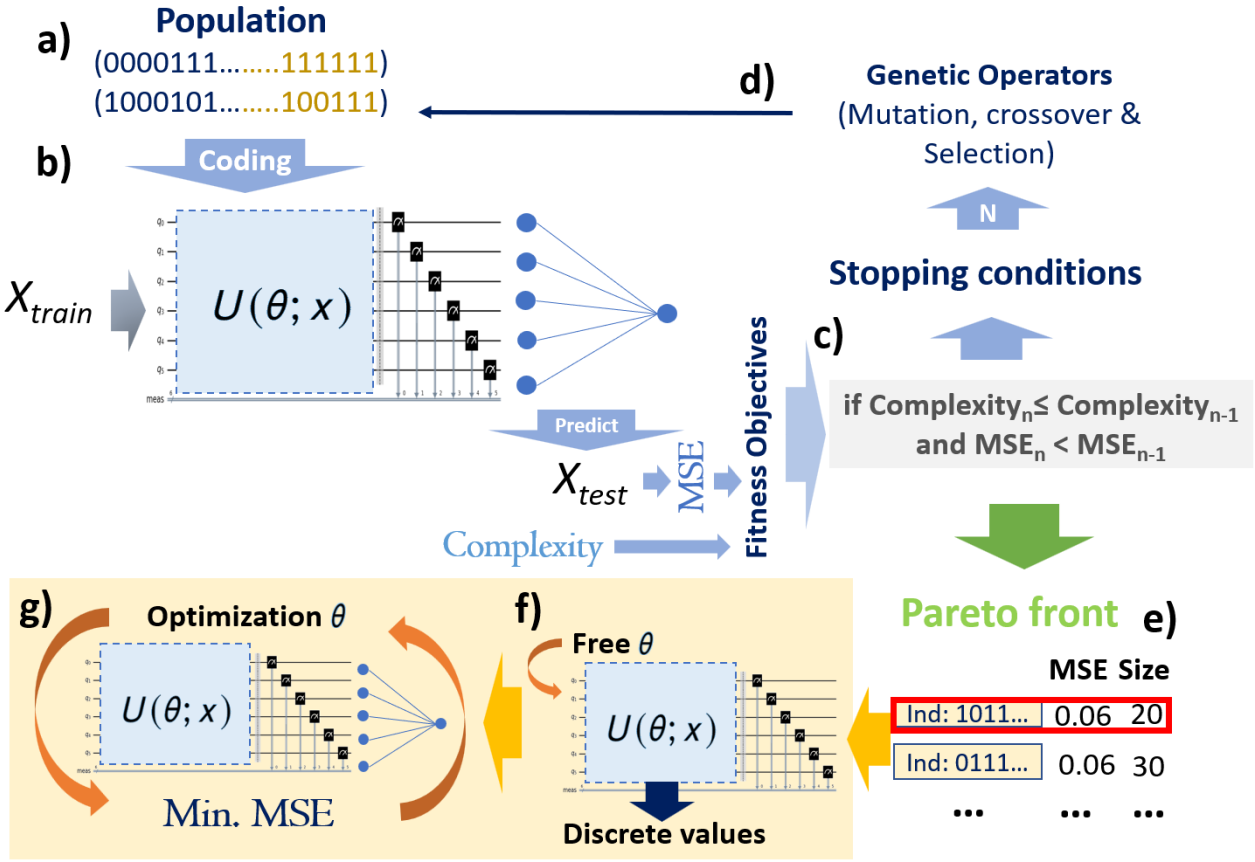


Figure 6.2: General scheme for generating optimized hybrid models through genetic algorithms. a) A random set of parametric circuits encoded into binary strings is created for evaluation. The binary string is decoded into circuits, and data from the training set, X_{train}, is input to the unitary operation $\mathcal{U}(\theta, x)$. The measurements act as the linear perceptron's input, generating continuous values. Once the hybrid model is established, X_{test} values are processed through the unit operation and the same previously tuned perceptron. This produces continuous values that are compared to the actual values, resulting in an error metric that is sought to be minimized. b) The fitness function evaluates the quality or suitability of each solution within the population. c) The conditions for entry into the Pareto front e). d) The genetic operators, such as crossover and mutation, introduce variation and facilitate the solution space exploration. f) The parameters of the quantum gates are released to allow their adjustment. g) Gradient-based optimization methods adjust these parameters, starting from the initial values established for the gates.

6.2 Quantum-Classical Neural Networks for Yield Crop Prediction

Predicting agricultural yields is a critical component of modern agriculture, especially in *precision agriculture*, where the goal is to maximize crop production in a sustainable manner. Satellite imagery has played a transformative role in providing extensive data that helps farmers and researchers analyze soil conditions and identify key factors influencing crop growth. This chapter explores the use of quantum-classical neural networks to develop a model capable of predicting crop yields by using satellite imagery from Sentinel-II (Agency, 2016) following the scheme depicted in Figure 6.3.

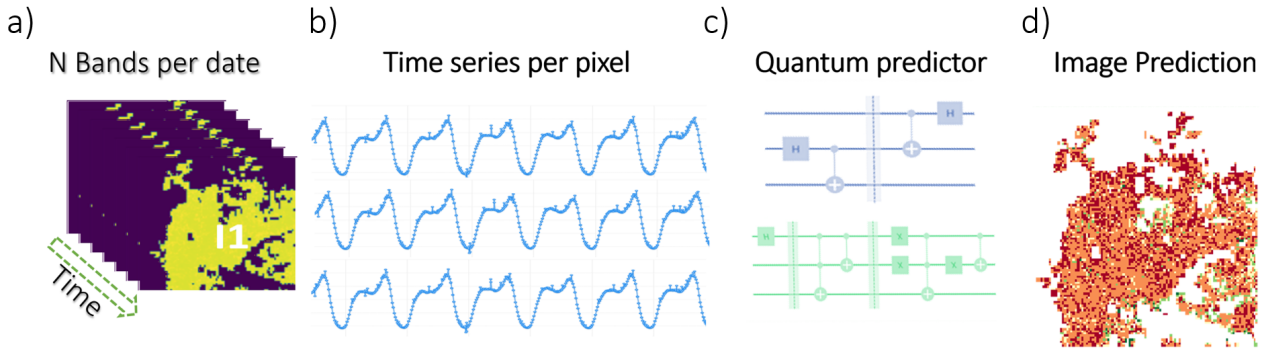


Figure 6.3: a) Geolocated images of the same areas over time composed by several bands. b) Each pixel of the image has its own time series. c) A quantum predictor must be able to capture the trends of these time series. d) The objective is to obtain predicted future images based on previous images.

A fundamental aspect of agricultural yield prediction is understanding the vegetation's health and density. To this end, several vegetation indices are used to assess crop conditions. In this chapter, we employ the most common index: the Normalized Difference Vegetation Index (NDVI) (Rouse et al., 1974; Tucker, 1979). This remotely sensed index is calculated using two spectral bands: the near-infrared (NIR) band, which captures reflectance in the near-infrared range, and the red band (RED), which reflects the amount of visible red light. NDVI is calculated by applying the Equation 6.2,

$$\text{NDVI} = \frac{\text{NIR} - \text{RED}}{\text{NIR} + \text{RED}} = \frac{\text{B8} - \text{B4}}{\text{B8} + \text{B4}}. \tag{6.2}$$

This equation provides a value for each pixel of the satellite image, ranging from -1 to +1. Values closer to +1 indicate healthier and denser vegetation, while values closer to -1 correspond to areas with little or no vegetation. As indicated in previous studies, higher NDVI values correlate with better crop health and, consequently, higher yield potential (Wang et al., 2003). Conversely, negative values suggest the absence of vegetation, which should be considered when selecting data for model training (Rhew et al., 2011).

6.2.1 Satellite Data Extraction

Satellite imagery poses a considerable challenge due to its high resolution and the composition of multiple spectral bands. This complexity is amplified when image sequences are processed to identify patterns over time, as information from previous images must be integrated into the models. This section describes the satellite image dataset and details a strategy to extract relevant information from the images for training the prediction models.

Data Setup

The image set is composed of historical data from January 2017 to May 2022 from the Ribera del Duero D.O. area in Burgos, Spain. The specific location is within the Sentinel-2 30TVM frame, which covers an extensive area of 203,885 hectares, as observed in Figure 6.4a. This study focuses on vineyard crops, declared in the Spanish SigPAC 2022 database (MAPA, 2022). In this area, the total area covered by vineyards amounts to 19,520 hectares, as shown in Figure 6.4b. Since the focus is on the NDVI vegetation index and this, as explained above, can be calculated using the B8 and B4 bands that are systematically available, the band produced by the formula described in the equation 6.2 is used as the actual target of the models.

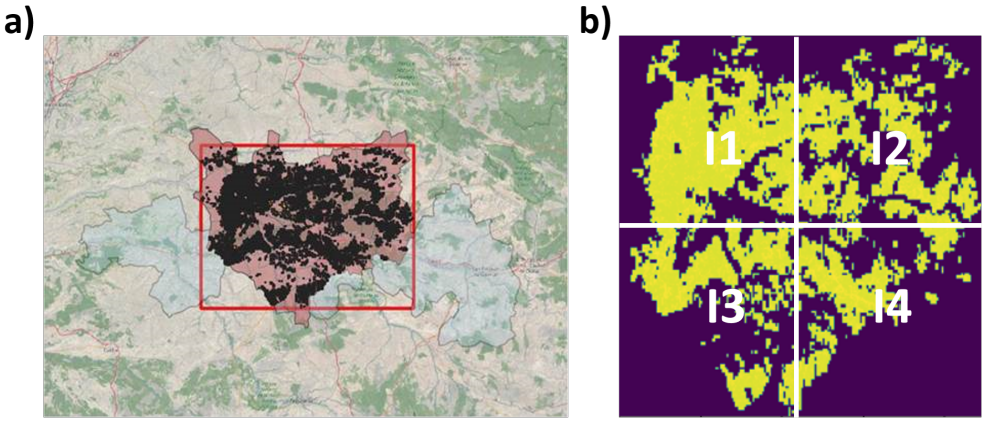


Figure 6.4: a) The area in the Province of Burgos belongs to the Ribera del Duero designation of origin (D.O). b) Expansion of the vineyard study area. The satellite image was reduced to a resolution of 224x224 pixels and proposed intervals.

The dataset provides both temporal and spatial information of the vineyard area, however, it is important to have all the images available in order to have stability in the models. The lack of some dates produces inconsistency in the series, which makes it difficult to establish a consistent time interval to fit the model. Our dataset has gaps in random dates, making it impossible to use it in a time series since there is no consistent time interval. There are several approaches to solving this problem. In this case, it is applied a *cubic smoothing interpolation*. This method generates an intermediate image by calculating smoothed values between the original images. It uses cubic polynomials to interpolate the corresponding pixels, ensuring smooth and natural transitions between the images (Ibánez et al., 2004). In this way, the missing images can be obtained within the desired time interval. For convenience in this

chapter, the images are obtained with a one-day interval (Kandasamy et al., 2013; Lepot et al., 2017). In this way, all dates for the period 2017-2021 are available, allowing the evaluation of different time frames of the image series.

Data Extraction

Once the data is completed, a random selection of balanced pixels is made from all the images in the series, considering the study area as illustrated in Figure 6.4. This pixel selection is similar to the strategy defined in chapter 5 with some refinements. For this selection, it is important to consider the size of the study area since, being a large area, it is likely that there are different climatic and orographic conditions that influence differently the growth phases of the crops. Each image has a resolution of 224x224 pixels; when divided into four parts, each part will have a resolution of 112x112 pixels.

A balanced number of pixels is selected in each interval to ensure a complete representation of the image in the model. Within this selection, to avoid overfitting problems at some NDVI values and ensure an equal representation across the entire range of index data in the training set, an equal number N of pixels (x_i, y_j) is taken in each data range. Since the aim is to capture how NDVI changes over time at each pixel, it takes the values of those same pixels in previous satellite images and creates a time series that shows the evolution of NDVI at each specific point.

It is important to consider in this data selection that the objective is to evaluate the crop's vegetation. As previously commented, negative values of this index indicate the absence of vegetation cover; thus, only positive values of the index are used for the study.

Typically, the models used for these image prediction tasks consider the entire images and bands requiring sophisticated algorithms with many parameters such as convolutional neural networks, recurrent networks, or autoregressive models (Hyndman and Athanasopoulos, 2018). Some models even incorporate weather data to assist in the prediction (Fernandez-Beltran et al., 2021; Saeed et al., 2017). In this approach, considering the challenges presented by quantum circuits in handling high-dimensional data and the inherent size limitations of these models, it uses the six previous data points to capture the trend pattern and predict the next value in the series. Thus, the model captures the cyclical nature of the year's seasons, allowing it to establish connections between specific patterns as this vegetation index varies according to the crop cycle: *phenological status*. In the specific case of vineyards, sprouting, flowering, and fruit sets usually occur in spring. During the summer, the grapes grow and begin to change color due to the accumulation of sugars. Finally, from mid-summer and early autumn, the grapes are usually harvested (Kurtural and Fidelibus, 2021; Tomasi et al., 2011). It is important to estimate the yield at this time to organize the logistics.

Temporal forecasting models use a parameter called *lag*, which defines the time interval between data points considered for the prediction. It is important to accurately set this parameter as an excessively short period may not capture the temporal pattern. At the same time, an excessively long one could result in significant and unpredictable fluctuations between stages (Lukoseviciute and Ragulskis, 2010). For this use case, 20 days has been defined numerically as the optimal time window for this use case. Thus, given the 20-day time

frame, it uses the previous six NDVI data points: NDVI_{t-100}, NDVI_{t-80}, NDVI_{t-60}, NDVI_{t-40}, NDVI_{t-20}, NDVI_{t} to determine the future value NDVI_{t+20} , as shown in Figure 6.5.

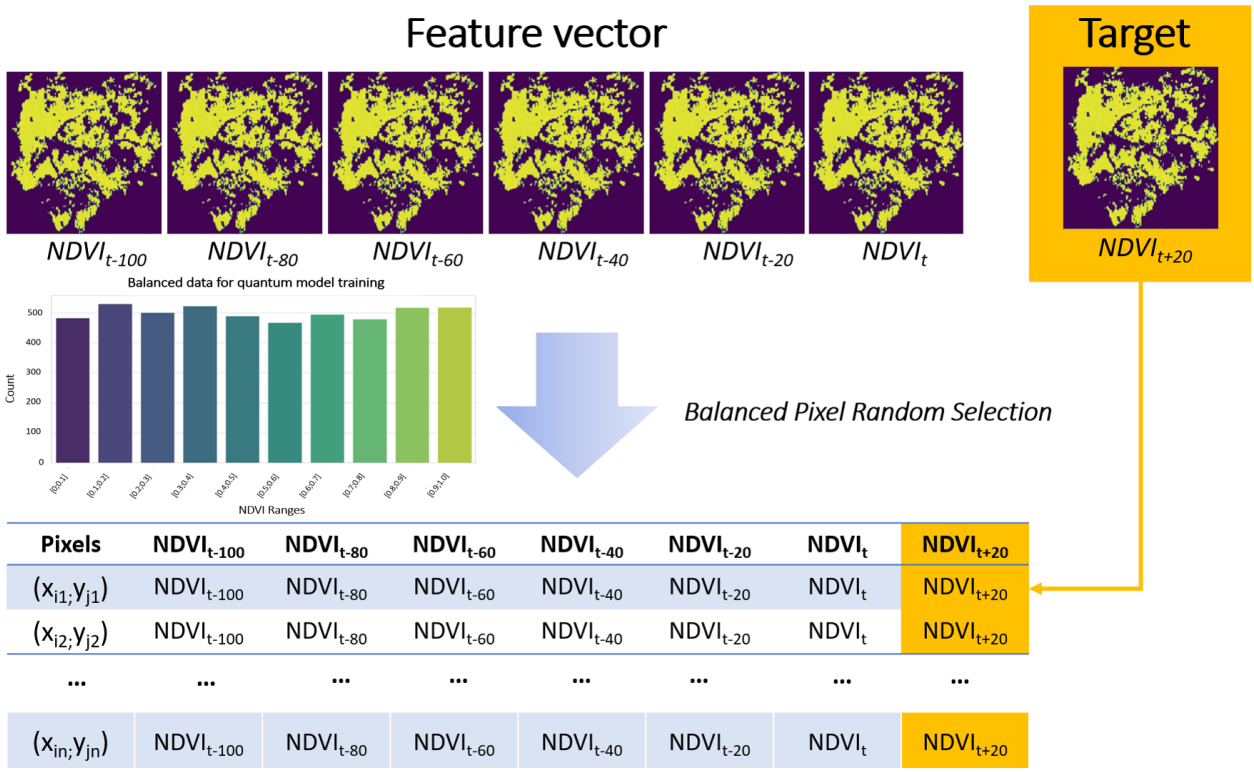


Figure 6.5: Processing of NDVI satellite images and construction of the balanced dataset for the temporal series.

During the training phase, the actual values at $t+20$ are used as ground truth in the fitness function. As the NDVI index is a continuous and non-categorical value, performing a prior stratification to balance the training and test sets is impossible. To address this, NDVI values are categorized, which allows for the creation of classes and ensures adequate representation of each range of data. As shown in Figure 6.5, the intervals are set in 100-unit slices, allowing the training set to have a widely representative distribution. 500 data points are selected from each interval, resulting in a total of 5000 data points obtained from random dates and pixels, also considering the image intervals. Removing the categorical variable NDVI before training and predictions is important to avoid influencing the results.

6.2.2 Hybrid Model Generation for NDVI Prediction

In this methodology, the neural network is trained to minimize the mean squared error (MSE) and the circuit's size, using the probabilities generated by the quantum circuit states, as described in section 6.1.1. It is designed to capture the trend of an image sequence, particularly the NDVI index band.

Regarding the evolution parameters, 2000 generations are repeated ten times to find the best approximation to the optimal solution. The crossover probability used is 0.6, while the

mutation probability is 0.2 with an initial population of 30 individuals. For all experiments, a maximum number of eight qubits and a maximum number of eight layers are defined so that all variables can appear in the model and be repeated if necessary. The training set comprises 75% of the data, while the test set comprises 25%. In addition to the above set of NDVI values per pixel, it included information on the week of the year at time t in the predictor variables to provide NDVI sequence data and temporal information throughout the year. Note that the week number is also normalized to [0,1] by dividing by the total number of weeks in the year: fifty-four.

The resulting circuit in Figure 6.6 shows an increase in correlations after applying this methodology compared to previous chapters. This can be attributed to eliminating penalties in the complexity calculation applied (see Chapter 3). This hybrid system generated automatically and, after having refined the parameters with the second optimization step, demonstrates predictions that closely align with the actual values of $t + 20$, resulting in an R^2 value of 0.90 as shown in Figure 6.7a. Looking at the errors calculated as the difference between the actual and predicted values, it is observed that the variances exhibit a mean close to zero, indicating stability in the generated model, as shown in Figure 6.7b. Concerning the other fitness metric, the model's size, the classical neural network has 257 trainable parameters. In comparison, the quantum circuit obtained is composed of 32 parameters, as shown in Figure 6.6, since each rotational gate of the circuit has its associated parameter, resulting in a total of 289 trainable parameters, being notably lower compared to classical models commonly used in tasks such as CNN or RNN.

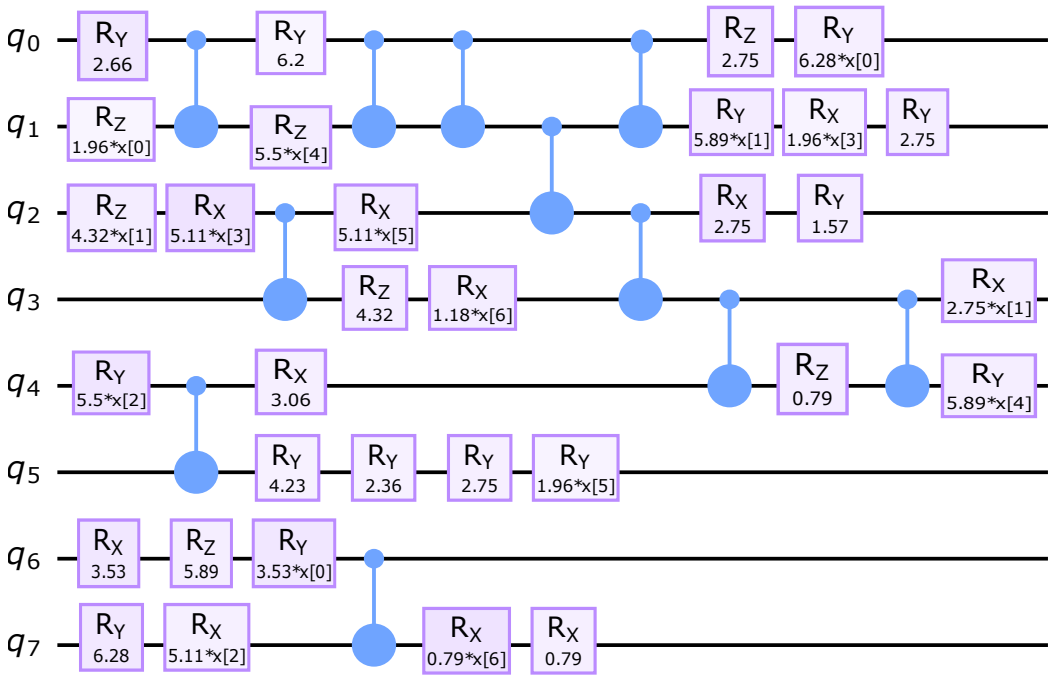


Figure 6.6: a) The best quantum circuit for NDVI prediction obtained by using multiobjective genetic algorithms for crop yield prediction in $t + 20$.

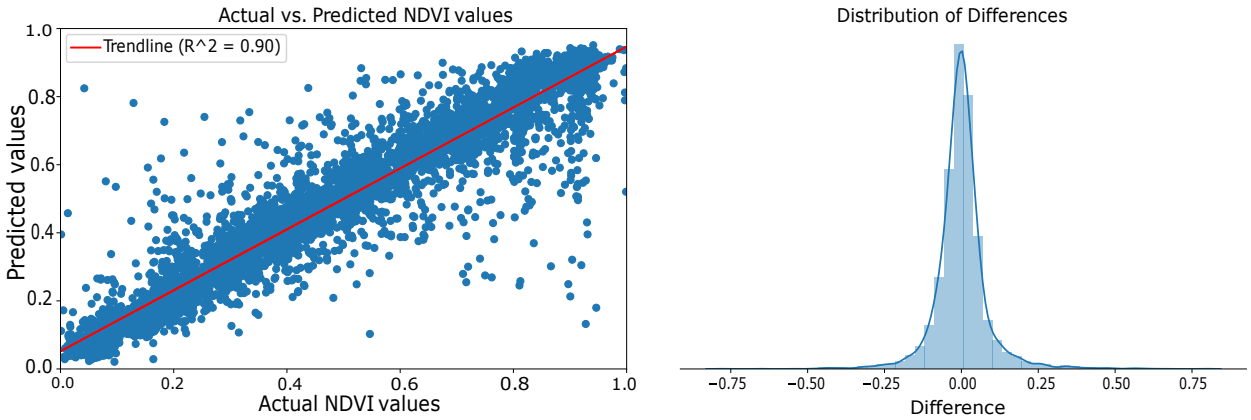


Figure 6.7: a) Regression line between actual and predicted values of NDVI_{t+20} on test data. b) Histogram of the differences in the predictions.

6.2.3 Robustness Evaluation

To perform this validation, a new set of random validation data is prepared to evaluate the robustness of the hybrid model. The data collection process for the image time series is performed in the same way as in the training set explained in section 6.2.1. In this way, balanced data is collected, considering the intervals of Figure 6.4 to maintain a balance among the different zones. With these data, a more detailed study is carried out, analyzing the ranges of the NDVI index to better understand the model's performance. Thus, the methodological procedure begins with establishing the NDVI ranges, which implies dividing the entire NDVI spectrum into different categories to better understand the error (Yohe and CENTER, 1976).

For the prediction study, five homogeneous ranges are created and evenly distributed according to significance, as shown in Table 6.1. When the NDVI index is between [0; 0.2], the areas have no or sparse vegetation, indicating bare soils, water bodies, or urban areas. In the range [0.2; 0.4], vegetation cover may be sparse or unhealthy, indicating areas with sparse vegetation, water-stressed, or suboptimal. An index value of [0.4; 0.6] suggests moderately dense and healthy vegetation, such as grasslands, light forests, or early crops. The pixel value in the interval [0.6; 0.8] indicates that the vegetation is more dense and healthy and can be associated with lush forests, mature crops, and robust vegetation. Finally, in the interval [0.8; 1.0], the vegetation is extremely dense and healthy and indicates areas with abundant and fully developed vegetation, such as rainforests or crops at maximum productivity. These intervals represent different levels of vegetation, ranging from very low vegetation green to very dense vegetation green. Each range described is studied separately to analyze the distributions and errors of the actual data and model estimates. A scatter plot is also included to better visualize each range's relationship between these values.

Errors are calculated as the difference between actual and estimated data for each point. As shown in Figure 6.8, the actual and predicted data distributions align closely within the defined intervals, centered consistently close to zero in the differences in all ranges, as seen in Table 6.1. On the other hand, it is observed in the difference column of Figure 6.8 that the greater the observed NDVI range, the greater the variance exhibited in the data, as analyzed

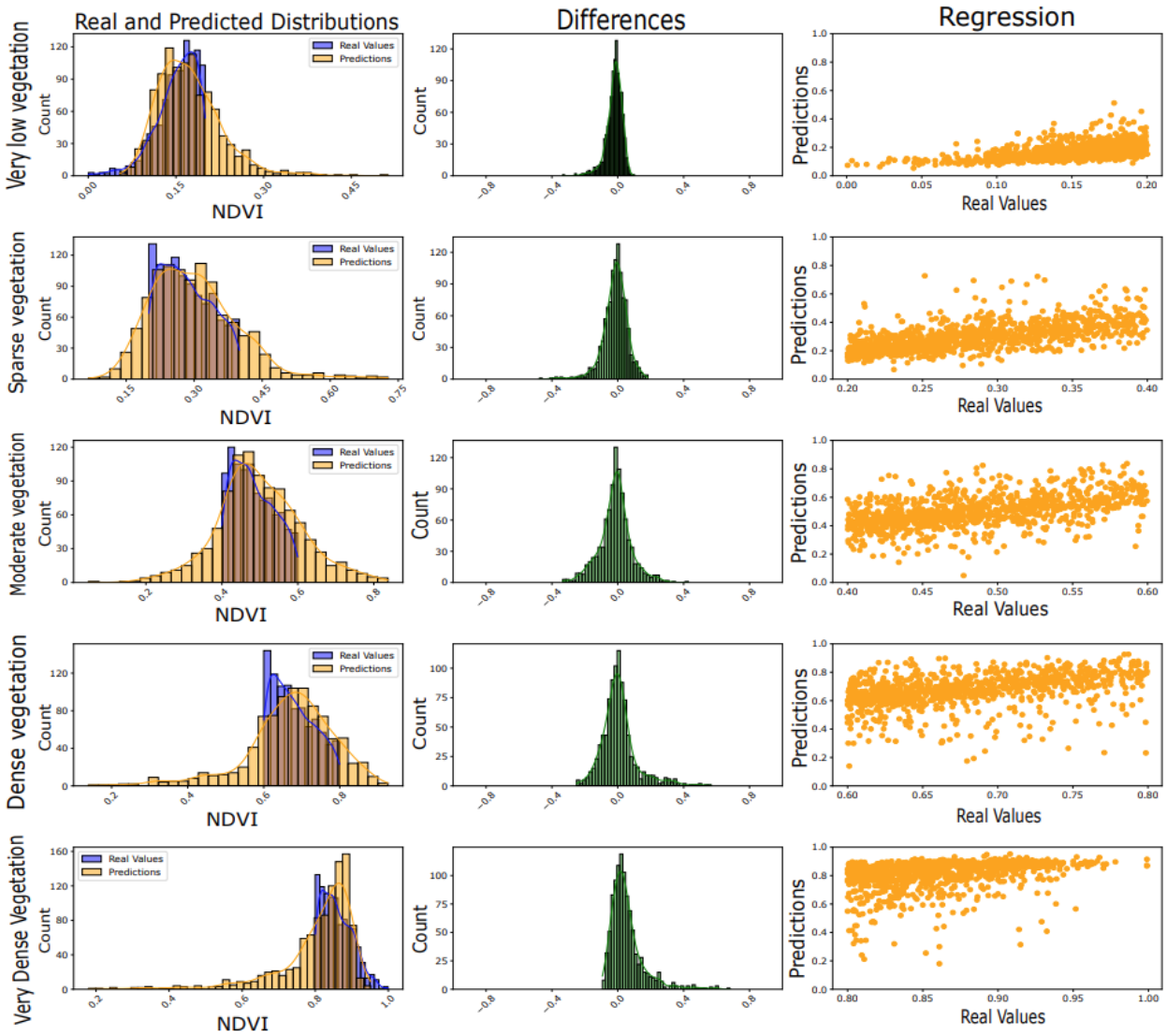


Figure 6.8: In each NDVI range, both the actual data and the predictions made by the quantum predictor for t+20.

in Table 6.1. Examining the scatter plots between the actual and predicted values, it can be seen that the predicted values fall mostly on the regression line of the same interval. It can be seen that in the interval [0.8-1.0], the model slightly underestimates these values. Still, overall, it can be concluded that the hybrid model obtains promising results on randomly chosen validation data with only 289 parameters. This suggests that the optimized hybrid model has generalizability.

Table 6.1: Ranges of NDVI and their interpretation. Average errors and σ on validation data. Errors are calculated as the differences between actual and predicted values.

NDVI Interval	Interpretation	Mean error	Mean σ of errors
0.0 - 0.2	Very low vegetation	-0.020	0.056
0.2 - 0.4	Sparse vegetation	-0.022	0.073
0.4 - 0.6	Moderate dense vegetation	-0.010	0.093
0.6 - 0.8	Dense vegetation	0.013	0.093
0.8 - 1.0	Very dense vegetation	0.054	0.110

6.2.4 Predicting Future Crop Status

The study aims to assess the model's performance in a real-world setting. To achieve this, working with complete images is crucial, simulating real field analysis. Additionally, selecting specific date ranges close to harvest time allows farmers to effectively utilize the model's recommendations for planning subsequent activities. To limit the size of the validation set, two random dates are chosen from two consecutive years: 2020 and 2021.

Final validation is performed by selecting the pre-harvest images taken at 20-day intervals *(t, t-20, t-40, t-60, t-80, t-100)*. This approach allows for establishing a temporal sequence of pixels, where each pixel represents the state of the crops at specific times before harvest. This organization of the data set facilitates the feeding of information to the hybrid model, allowing it to process each pixel of the input images sequentially. From this time series, the model can generate a new image that represents a prediction of crop conditions at a future point in time, specifically 20 days after the last capture point, i.e., at time *t+20*.

Figure 6.9 presents the actual images of the bands on those dates provided by Sentinel-2, as well as an image focused on the most productive NDVI index, which is greater than 0.4 — from moderate to very dense since it indicates a higher crop performance (Table 6.1). Despite the constraints imposed by the quantum modeling on input data and the limited number of trainable parameters, the predictions strongly correlate with the actual images, yielding an average MAPE of 17%. The results indicate that, on average, the model achieves an accuracy rate of 97% when evaluating all pixels within the images of productive areas compared to the actual NDVI. In addition, it exhibits a sensitivity rate of 82% for pixels possessing NDVI values greater than 0.4, as visually depicted in Figure 6.9a-b. In the 2020 image, a total of 759 pixels are detected with a productive capacity of 918 in the real band, while in the 2021 image, 841 pixels are identified out of a total of 1029. Therefore, it can be deduced that the optimized hybrid model can effectively forecast crops' productive capacity, considering the time series of individual pixels of the NDVI satellite bands.

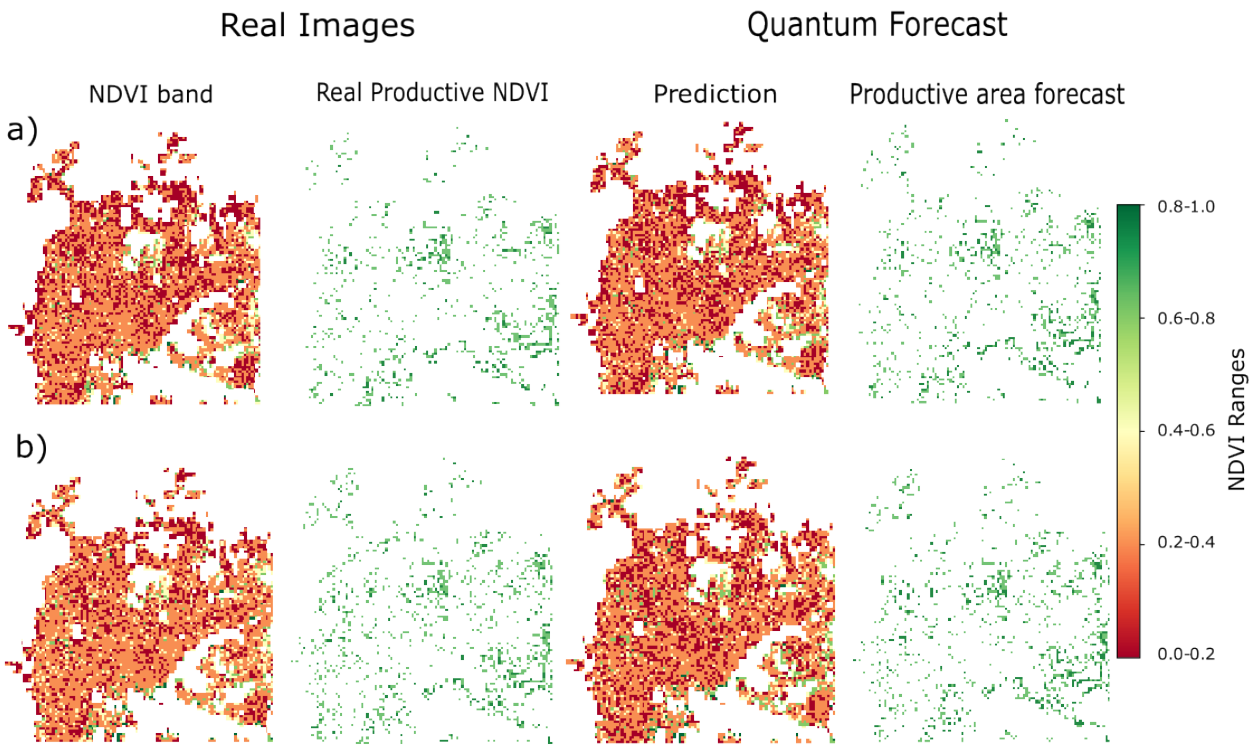


Figure 6.9: a) Real image of a random date 2020-08-30, and the predicted image by the hybrid model for the same date. Both scenarios emphasize areas with a moderate-high NDVI value and higher agricultural productivity. b) The actual depiction of a randomly selected date, specifically July 28, 2021, and the anticipated representation produced by the hybrid model for the identical date are compared. The focus is on regions exhibiting a moderate to high Normalized Difference Vegetation Index (NDVI) value, indicating elevated agricultural productivity in both situations.

6.3 Conclusions

This chapter describes a methodology for treating time series using hybrid models. These time series are composed of images and analyzed using an approach that considers each pixel as an independent time series. It is possible to generate models capable of capturing complex temporal patterns for accurate predictions through the described feature selection.

For the generation and training process, a genetic algorithm creates an initial model, which explores various network configurations and selects the most promising ones based on their performance. This chapter presents how to refine the model parameters using gradient-based optimization methods. This stage allows more precise tuning of the parameters, improving the processing circuit configuration and optimizing the accuracy of the predictions while avoiding *barren plateaus* effects.

The presented methodology allows the automatic generation of hybrid neural networks that integrate classical and quantum techniques. For the generation and training process, a genetic algorithm is used to create an initial model, which explores various network configurations and selects the most promising ones based on their performance. This chapter presents a second optimization step to refine the model parameters using gradient-based optimization methods. This stage allows more precise tuning of the parameters, improving the processing circuit configuration and optimizing the accuracy of the predictions. In this way, combining genetic algorithms followed by gradient methods to refine the parameters improves the efficiency of the analysis and enhances the prediction capability of hybrid neural networks.

The results obtained show a high accuracy in predictions with a reduced number of parameters compared to more advanced classical methods. Furthermore, it has been demonstrated that the models developed with the proposed methodologies are applicable in real situations, using real data, which validates their practical utility. This effectiveness suggests that genetically designed and subsequently refined hybrid networks have great potential for prediction of multispectral image time series.

Chapter 7

Conclusions

This thesis has addressed the design of quantum circuits for machine learning, focusing on the relationship between expressivity and model complexity, which directly affects the trainability and performance of the model. The central contribution in this thesis is the design of a *global optimization method for the automatic generation of ad-hoc quantum circuits* using quantum support vector machine and quantum neural networks for classification, segmentation, and prediction tasks (Abbas et al., 2021; Rebentrost et al., 2013) (Chapter 2). This method, based on multiobjective genetic algorithms, balances expressiveness and complexity, improving the trainability and allowing the adaptation of the quantum circuit to each use case with total flexibility in structure, gate types, and parameters. The genetic operators used in this procedure, such as mutation and crossover, allow the introduction of random variations in quantum circuits, facilitating the exploration of more topologies, overcoming local minima, and expanding the possibilities of discovering optimal quantum circuit configurations (Chapter 3), unlike the generic circuits typically used in state of the art with fixed structures and gradient-based parameters optimization (Havlíček et al., 2019).

This work has also presented an innovative approach to encoding classical data into a quantum state, introducing customized encoding strategies that dynamically adjust to the dataset variables, allowing for more precise adaptation to different data types and problem domains, such as tabular data (Chapter 3) image classification (Chapter 4), segmentation of areas of interest in images (Chapter 5), and sequential image predictions (Chapter 6). This contribution represents a significant advancement over the state-of-the-art, traditionally relying on static and generic encodings (Bergholm et al., 2018; Qiskit contributors, 2023).

In parallel to this work, various techniques have also been proposed to optimize quantum algorithms, such as adaptive pruning (Sim et al., 2021b), random features for distribution approximation (González et al., 2022), or reinforcement learning (Ostaszewski et al., 2021). However, these solutions only partially address the challenges associated with circuit optimization.

In this thesis we have successfully used the data reuploading technique described in previous research, in which variables are applied multiple times throughout the circuit (Pérez-Salinas et al., 2020), unlike standard coding that assigns one variable per qubit, which exponentially

increases both the number of entanglement gates and the number of qubits as the number of variables in the dataset grows, producing the *barren plateaus* effect (Havlíček et al., 2019; Holmes et al., 2022; Schuld et al., 2021b). The flexibility of the proposed method to represent information more efficiently facilitates the exploration of different configurations. It improves the circuits' ability to execute complex tasks by identifying nonlinear dependencies between variables.

The genetic optimization can reduce a quantum circuit's complexity, measured in the number of entangling gates, while preserving the expressivity and accuracy. This facilitates the training of quantum models and opens the door to their implementation in classical hardware, leading to machine learning models inspired by quantum computing: *quantum-inspired machine learning*. This possibility marks a fundamental distinction from the dominant trend in the literature, where high levels of entanglement are considered necessary to improve model representativeness (Sim et al., 2019). This contribution is not trivial since these gates are more costly, have lower fidelities, and are typically slower. Moreover, reducing the circuit's complexity facilitates the creation of model interpretation methods, such as rule-based explanations, contributing to the growing demand for transparency in machine learning models.

This work has also demonstrated how quantum machine learning can work in large datasets of images using two different approaches. The first approach combined dimensionality reduction techniques with the expressive power of quantum circuits (Chapter 4). The second technique focused on a local inspection of multiband images to address problems such as segmentation or feature resolution (Chapter 5). These local inspection techniques have been applied to image sequences to identify temporal trends, allowing predictions to be made (Chapter 6). The techniques proposed have allowed the optimization of performance in complex tasks using quantum circuits and have demonstrated their feasibility and effectiveness in real-world applications, opening new opportunities for research and practical application of quantum machine learning methods (Cerezo et al., 2022).

The impact of this thesis goes beyond quantum circuit optimization. It introduces a method that balances model complexity with computational efficiency, improving training processes and extending the use of quantum models on traditional hardware. This hybrid approach allows the implementation of simplified quantum models in classical systems, which opens up the possibility of their use in industry and academia. The methods proposed in this thesis have inspired other researchers to continue studying the relationship between complexity and accuracy in quantum machine learning, using the methodology presented as a reference (Ardila-García et al., 2024; Incudini et al., 2022; Pellow-Jarman et al., 2024). Some works have modified aspects of our fitness function described in Chapter 3, such as the size metric or the kernel evaluation, while preserving the fundamental principles of evolutionary optimization proposed in this thesis (Chen et al., 2022; Tjandra and Sugiarto, 2023).

The work derived from this thesis opens new lines of research in several directions. The possibility of optimizing quantum circuits with low entanglement gives rise to a new field of study on the relationship between entanglement complexity and model representability. Combining evolutionary techniques with gradient optimization methods is another promising framework for improving the efficiency and accuracy of quantum machine learning models,

avoiding *barren plateaus*. The proposed methodology opens up new ways of encoding classical data into quantum states, which expands both circuit design options and ways of representing classical information. Finally, the ability to interpret quantum models using rule-based explanations, as explained in Chapter 3, contributes to the growing interest in the transparency of machine learning models, which could significantly impact fields such as medicine, justice, and finance, where interpretability is crucial.

References

Abbas, A., Sutter, D., Zoufal, C., Lucchi, A., Figalli, A., & Woerner, S. (2021). The power of quantum neural networks. *Nature Computational Science*, *1*(6), 403–409.

Agency, E. S. (2016). Sentinel-2 user handbook [Accessed: 2024-10-08].

Altares-López, S., García-Ripoll, J. J., & Ribeiro, A. (2022). Automatic design of quantum feature maps: Auto-generated quantum-inspired kernels by using multi-objective genetic algorithms (aquik).

Altares-López, S., García-Ripoll, J. J., & Ribeiro, A. (2023a). Autoqml: Quantum inspired kernels by using genetic algorithms for grayscale images. https://doi.org/10.24433/CO.3955535.V1

Altares-López, S., García-Ripoll, J. J., & Ribeiro, A. (2023b). Robótica cuántica: Elementos principales. *Jornadas Nacionales de Robótica y Bioingeniería*, 59–64. https://doi.org/10.20868/UPM.book.74896

Altares-López, S., García-Ripoll, J. J., & Ribeiro, A. (2024a). Autoqml: Automatic generation and training of robust quantum-inspired classifiers by using evolutionary algorithms on grayscale images. *Expert Systems with Applications*, *244*, 122984. https://doi.org/10.1016/j.eswa.2023.122984

Altares-López, S., García-Ripoll, J. J., & Ribeiro, A. (2024b). Efficient hybrid quantum-classical neural networks generation for crop yield prediction using satellite images. *Under review*.

Altares-López, S., García-Ripoll, J. J., & Ribeiro, A. (2024c). Optimal quantum circuit generation for pixel segmentation in multiband images. *Applied Soft Computing*, 112175. https://doi.org/10.1016/j.asoc.2024.112175

Altares-López, S., Ribeiro, A., & García-Ripoll, J. J. (2021). Automatic design of quantum feature maps. *Quantum Science and Technology*, *6*(4), 045015. https://doi.org/10.1088/2058-9565/ac1ab1

Amari, S. (1967). A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, (3), 299–307.

Amari, S.-i. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, *5*(4-5), 185–196.

Ardila-García, J. E., Vargas-Calderón, V., González, F. A., Useche, D. H., & Vinck-Posada, H. (2024). Memo-qcd: Quantum density estimation through memetic optimisation for quantum circuit design. *arXiv preprint arXiv:2406.08591*.

Arrasmith, A., Cerezo, M., Czarnik, P., Cincio, L., & Coles, P. J. (2021). Effect of barren plateaus on gradient-free optimization. *Quantum*, *5*, 558.

Baker, J. E., et al. (1987). Reducing bias and inefficiency in the selection algorithm. *Proceedings of the second international conference on genetic algorithms*, *206*, 14–21.

Ballentine, L. E. (2014). *Quantum mechanics: A modern development*. World Scientific Publishing Company.

Benedetti, M., Lloyd, E., Sack, S., & Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, *4*(4), 043001. https://doi.org/10.1088/2058-9565/ab4eb5

Benhamadi, S. (2021). Dataset for solar panel detection and identification [Dataset for solar panel detection and identification].

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

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). Quantum machine learning. *Nature*, *549*(7671), 195–202. https://doi.org/10.1038/nature23474

Bishop, C. M. (2006). Pattern recognition and machine learning. *Springer google schola*, *2*, 1122–1128.

Bloch, F. (1946). Nuclear induction. *Physical review*, *70*(7-8), 460.

Bravyi, S., & Kitaev, A. (2005). Universal quantum computation with ideal clifford gates and noisy ancillas. *Physical Review A—Atomic, Molecular, and Optical Physics*, *71*(2), 022316.

Burgos-Artizzu, X. P., Ribeiro, A., Guijarro, M., & Pajares, G. (2011). Real-time image processing for crop/weed discrimination in maize fields. *Computers and Electronics in Agriculture*, *75*(2), 337–346.

Cerezo, M., Sone, A., Volkoff, T., Cincio, L., & Coles, P. J. (2021). Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, *12*(1), 1791.

Cerezo, M., Verdon, G., Huang, H.-Y., Cincio, L., & Coles, P. J. (2022). Challenges and opportunities in quantum machine learning. *Nature Computational Science*, *2*(9), 567–576.

Chakrabarty, N. (2019). Brain dataset.

Chen, C., He, Z., Zheng, S., Zhou, Y., & Situ, H. (2022). Generating the optimal structures for parameterized quantum circuits by a meta-trained graph variational autoencoder.

Clymer, J. R. (1999). Optimization of simulated system effectiveness using evolutionary algorithms. *Simulation*, *73*(6), 334–340.

Cong, I., Choi, S., & Lukin, M. D. (2019). Quantum convolutional neural networks. *Nature Physics*, *15*(12), 1273–1278.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273–297.

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems*. University of Michigan.

Deb, K., Agrawal, S., Pratap, A., & Meyarivan, T. (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*, 849–858.

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, *6*(2), 182–197.

Deutsch, D. (1985). Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, *400*(1818), 97–117.

Tripathi, P. (2020).

Fernandez-Beltran, R., Baidar, T., Kang, J., & Pla, F. (2021). Rice-yield prediction with multi-temporal sentinel-2 data and 3d cnn: A case study in nepal. *Remote Sensing*, *13*(7), 1391.

Feynman, R. P. (1982). Simulating physics with computers. In *Feynman and computation* (pp. 133–153). CRC Press.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*, 2171–2175.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, *32*(200), 675–701.

Géron, A. (2022). *Hands-on machine learning with scikit-learn, keras, and tensorflow*. " O'Reilly Media, Inc."

Goldberg, D. E. (2013). *Genetic algorithms*. pearson education India.

Goldberg, D. E., Deb, K., & Clark, J. H. (1991). Genetic algorithms, noise, and the sizing of populations. *Complex systems*, *6*, 333–362.

González, F. A., Gallego, A., Toledo-Cortés, S., & Vargas-Calderón, V. (2021). Learning with density matrices and random features. *arXiv preprint arXiv:2102.04394*.

González, F. A., Gallego, A., Toledo-Cortés, S., & Vargas-Calderón, V. (2022). Learning with density matrices and random features. *Quantum Machine Intelligence*, *4*(2), 23.

Grant, E., Wossnig, L., Ostaszewski, M., & Benedetti, M. (2019). An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, *3*, 214. https://doi.org/10.22331/q-2019-12-09-214

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 212–219.

Guijarro, M., Pajares, G., Riomoros, I., Herrera, P., Burgos-Artizzu, X., & Ribeiro, A. (2011). Automatic segmentation of relevant textures in agricultural images. *Computers and Electronics in Agriculture*, *75*(1), 75–83.

Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2020). A hybrid genetic algorithm for the traveling salesman problem with drone. *Journal of Heuristics*, *26*, 219–247.

Halko, N., Martinsson, P.-G., & Tropp, J. A. (2010). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.

Haralick, R. M., & Shapiro, L. G. (1985). Image segmentation techniques. *Computer vision, graphics, and image processing*, *29*(1), 100–132.

Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical review letters*, *103*(15), 150502.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, *567*(7747), 209–212. https://doi.org/10.1038/s41586-019-0980-2

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Holland, J. H. (1975). Adaptation in natural and artificial systems. *Ann Arbor: The University of Michigan Press*, 32.

Holland, J. H. (1992). Genetic algorithms. *Scientific american*, *267*(1), 66–73.

Holmes, Z., Sharma, K., Cerezo, M., & Coles, P. J. (2022). Connecting ansatz expressibility to gradient magnitudes and barren plateaus. *PRX Quantum*, *3*(1), 010313.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.

Ibánez, J., Santamaria, I., Pantaleón, C., & Vielva, L. (2004). Parametric smoothing of spline interpolation. *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, *2*, ii–597.

Incudini, M., Martini, F., & Di Pierro, A. (2022). Structure learning of quantum embeddings. *arXiv preprint arXiv:2209.11144*.

Irrigation dataset. (2020).

Jiang, H., Yao, L., Lu, N., Qin, J., Liu, T., Liu, Y., & Zhou, C. (2021). Multi-resolution dataset for photovoltaic panel segmentation from satellite and aerial imagery. *Earth System Science Data Discussions*, *2021*, 1–17.

Kandasamy, S., Baret, F., Verger, A., Neveux, P., & Weiss, M. (2013). A comparison of methods for smoothing and gap filling time series of remote sensing observations–application to modis lai products. *Biogeosciences*, *10*(6), 4055–4071.

Killoran, N., Bromley, T. R., Arrazola, J. M., Schuld, M., Quesada, N., & Lloyd, S. (2019). Continuous-variable quantum neural networks. *Physical Review Research*, *1*(3), 033063.

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization.

Kurtural, S. K., & Fidelibus, M. W. (2021). Mechanization of pruning, canopy management, and harvest in winegrape vineyards. *American Journal of Enology and Viticulture*, *5*(Suppl 1), 29–44.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Touresky, D., Hinton, G., & Sejnowski, T. (1988). A theoretical framework for back-propagation. *Proceedings of the 1988 connectionist models summer school*, *1*, 21–28.

Lepot, M., Aubin, J.-B., & Clemens, F. H. (2017). Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. *Water*, *9*(10), 796.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200209.

Little, M., McSharry, P., Roberts, S., Costello, D., & Moroz, I. (2007). Exploiting nonlinear recurrence and fractal scaling properties for voice disorder detection. *BioMedical Engineering OnLine 2007*, 6–23.

Liu, J., Lim, K. H., Wood, K. L., Huang, W., Guo, C., & Huang, H.-L. (2021). Hybrid quantum-classical convolutional neural networks. *Science China Physics, Mechanics & Astronomy*, *64*(9), 1–8.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. *Proceedings of the IEEE/CVF international conference on computer vision*, 10012–10022.

Lukoseviciute, K., & Ragulskis, M. (2010). Evolutionary algorithms for the selection of time lags for time series forecasting by fuzzy inference systems. *Neurocomputing*, *73*(10-12), 2077–2088.

Makridakis, S., & Hibon, M. (1979). Accuracy of forecasting: An empirical investigation. *Journal of the Royal Statistical Society: Series A (General)*, *142*(2), 97–125.

MAPA. (2022). Sistema de información geográfica de parcelas agrícolas. ministerio de agricultura, pesca y alimentación de españa.

Mari, A., Bromley, T. R., Izaac, J., Schuld, M., & Killoran, N. (2020). Transfer learning in hybrid classical-quantum neural networks. *Quantum*, *4*, 340. https://doi.org/10.22331/q-2020-10-09-340

Martinsson, P.-G., Rokhlin, V., & Tygert, M. (2011). A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, *30*(1), 47–68.

Masini, R. P., Medeiros, M. C., & Mendes, E. F. (2023). Machine learning advances for time series forecasting. *Journal of economic surveys*, *37*(1), 76–111.

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

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, *5*, 115–133.

Mercer, J. (1909). Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, *209*(441-458), 415–446.

Nielsen, M. A., & Chuang, I. (2002). *Quantum computation and quantum information*. American Association of Physics Teachers.

Ostaszewski, M., Trenkwalder, L. M., Masarczyk, W., Scerri, E., & Dunjko, V. (2021). Reinforcement learning for optimization of variational quantum circuit architectures. *Advances in Neural Information Processing Systems*, *34*, 18182–18194.

Pauli, W. (1933). *Die allgemeinen prinzipien der wellenmechanik*. Springer.

Payne, A. B., Walsh, K. B., Subedi, P., & Jarvis, D. (2013). Estimation of mango crop yield using image analysis–segmentation method. *Computers and electronics in agriculture*, *91*, 57–64.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, *2*(11), 559–572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pellow-Jarman, R., Pillay, A., Sinayskiy, I., & Petruccione, F. (2024). Hybrid genetic optimization for quantum feature map design. *Quantum Machine Intelligence*, *6*(2), 45.

Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., & Latorre, J. I. (2020). Data re-uploading for a universal quantum classifier. *Quantum*, *4*, 226. https://doi.org/10.22331/q-2020-02-06-226

Pham, D. L., Xu, C., & Prince, J. L. (2000). Current methods in medical image segmentation. *Annual review of biomedical engineering*, *2*(1), 315–337.

Preskill, J. (2018). Quantum computing in the nisq era and beyond. *Quantum*, *2*, 79.

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

Raikote, P. (2020). Covid-19 dataset.

Rebentrost, P., Mohseni, M., & Lloyd, S. (2013). Quantum support vector machine for big feature and big data classification. *arXiv preprint arXiv:1307.0471*, *1*.

Rhew, I. C., Vander Stoep, A., Kearney, A., Smith, N. L., & Dunbar, M. D. (2011). Validation of the normalized difference vegetation index as a measure of neighborhood greenness. *Annals of epidemiology*, *21*(12), 946–952.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, 234–241.

Rosenblatt, F. (1961). Principles of neurodynamics: Perceptrons and the theory of brain mechanisms.

Rouse, J. W., Haas, R. H., Schell, J. A., Deering, D. W., et al. (1974). Monitoring vegetation systems in the great plains with erts. *NASA Spec. Publ*, *351*(1), 309.

Sadek, R. A. (2012). Svd based image processing applications: State of the art, contributions and research challenges. *arXiv preprint arXiv:1211.7102*.

Saeed, U., Dempewolf, J., Becker-Reshef, I., Khan, A., Ahmad, A., & Wajid, S. A. (2017). Forecasting wheat yield from weather data and modis ndvi using random forests for punjab province, pakistan. *International journal of remote sensing*, *38*(17), 4831–4854.

Samadzadegan, F., Soleymani, A., & Abbaspour, R. A. (2010). Evaluation of genetic algorithms for tuning svm parameters in multi-class problems. *2010 11th International Symposium on Computational Intelligence and Informatics (CINTI)*, 323–328.

Sartin, M. A., Da Silva, A. C., & Kappes, C. (2014). Image segmentation with artificial neural network for nutrient deficiency in cotton crop. *Journal of Computer Science*, 1084–1093.

Schaffer, J. D., Grefenstette, J. J., et al. (1985). Multi-objective learning via genetic algorithms. *Ijcai*, *85*, 593–595.

Schuld, M. (2021). Quantum machine learning models are kernel methods.

Schuld, M., Fingerhuth, M., & Petruccione, F. (2017). Implementing a distance-based classifier with a quantum interference circuit. *EPL (Europhysics Letters)*, *119*(6), 60002.

Schuld, M., & Killoran, N. (2019). Quantum machine learning in feature hilbert spaces. *Physical review letters*, *122*(4), 040504.

Schuld, M., Sweke, R., & Meyer, J. J. (2021a). Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, *103*(3), 032430.

Schuld, M., Sweke, R., & Meyer, J. J. (2021b). Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, *103*(3). https://doi.org/10.1103/physreva.103.032430

Schumacher, B. (1995). Quantum coding. *Physical Review A*, *51*(4), 2738.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, *27*(3), 379–423.

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional lstm network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, *28*.

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factoring. *Proceedings 35th annual symposium on foundations of computer science*, 124–134.

Sim, S., Johnson, P. D., & Aspuru-Guzik, A. (2019). Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, *2*(12), 1900070. https://doi.org/10.1002/qute.201900070

Sim, S., Romero, J., Gonthier, J. F., & Kunitsa, A. A. (2021a). Adaptive pruning-based optimization of parameterized quantum circuits. *Quantum Science and Technology*, *6*(2), 025019. https://doi.org/10.1088/2058-9565/abe107

Sim, S., Romero, J., Gonthier, J. F., & Kunitsa, A. A. (2021b). Adaptive pruning-based optimization of parameterized quantum circuits. *Quantum Science and Technology*, *6*(2), 025019.

Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. *computer*, *27*(6), 17–26.

Srinivas, N., & Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, *2*(3), 221–248.

Stitson, M., Weston, J., Gammerman, A., Vovk, V., & Vapnik, V. (1996). Theory of support vector machines. *University of London*, *117*(827), 188–191.

Suárez, E. J. C., & Galán, S. F. (2021). *Fundamentos de computación evolutiva*. Marcombo.

Sun, Z., Bebis, G., & Miller, R. (2006). On-road vehicle detection: A review. *IEEE transactions on pattern analysis and machine intelligence*, *28*(5), 694–711.

Syarif, I., Prugel-Bennett, A., & Wills, G. (2016). Svm parameter optimization using grid search and genetic algorithm to improve classification performance. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, *14*(4), 1502–1509.

Syswerda, G., et al. (1989). Uniform crossover in genetic algorithms. *ICGA*, *3*, 2–9.

Syswerda, G. (1991). A study of reproduction in generational and steady-state genetic algorithms. In *Foundations of genetic algorithms* (pp. 94–101). Elsevier.

Tjandra, Y., & Sugiarto, H. S. (2023). An evolutionary algorithm design for pauli-based quantum kernel classification. *VLDB Workshops*.

Tomasi, D., Jones, G. V., Giust, M., Lovat, L., & Gaiotti, F. (2011). Grapevine phenology and climate change: Relationships and trends in the veneto region of italy for 1964–2009. *American journal of Enology and Viticulture*, *62*(3), 329–339.

Tucker, C. J. (1979). Red and photographic infrared linear combinations for monitoring vegetation. *Remote sensing of Environment*, *8*(2), 127–150.

Turing, A. (1936). Turing machine. *Proc London Math Soc*, *242*, 230–265.

Van Veldhuizen, D. A., Lamont, G. B., et al. (1998). Evolutionary computation and convergence to a pareto front. *Late breaking papers at the genetic programming 1998 conference*, 221–228.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Wang, J., Rich, P. M., & Price, K. P. (2003). Temporal responses of ndvi to precipitation and temperature in the central great plains, usa. *International journal of remote sensing*, *24*(11), 2345–2364.

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, *2*(1-3), 37–52.

Xu, B. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Yohe, J. M., & CENTER, W. U. M. M. R. (1976). *Application of interval analysis to error control*. Mathematics Research Center, University of Wisconsin.

Zhang, Y. (2018). A better autoencoder for image: Convolutional autoencoder. *ICONIP17-DCEC*.

Zhang, Z. (2018). Improved adam optimizer for deep neural networks. *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 1–2.

Zhao, N., Wu, Z., Zhao, Y., & Quan, T. (2010a). Ant colony optimization algorithm with mutation mechanism and its applications. *Expert Systems with Applications*, *37*(7), 4805–4810. https://doi.org/10.1016/j.eswa.2009.12.035

Zhao, N., Wu, Z., Zhao, Y., & Quan, T. (2010b). A population declining mutated ant colony optimization multiuser detector for mc-cdma. *IEEE Communications Letters*, *14*(6), 497–499.