

# Hybrid Genetic Optimisation for Quantum Feature Map Design

by

Rowan Pellow-Jarman

Submitted to the School of Mathematics, Statistics and Computer  
Science

in fulfillment of the requirements for the degree of

Master of Science (Computer Science)

at the

UNIVERSITY OF KWAZULU-NATAL

February 2024

© University of KwaZulu-Natal 2024. All rights reserved.

Author .....  
School of Mathematics, Statistics and Computer Science  
February, 2024

Certified by .....  
Mr. Anban Pillay  
Thesis Supervisor

Certified by .....  
Prof. Ilya Sinayskiy  
Thesis Supervisor

Certified by .....  
Prof. Francesco Petruccione  
Thesis Supervisor

## Abstract

Good feature maps are crucial for machine learning kernel methods for effective mapping of non-linearly separable input data into a higher dimension feature space, thus allowing the data to be linearly separable in feature space. Recent works have proposed automating the task of quantum feature map circuit design with methods such as variational ansatz parameter optimization and genetic algorithms. A problem commonly faced by genetic algorithm methods is the high cost of computing the genetic cost function. To mitigate this, this work investigates the suitability of two metrics as alternatives to test set classification accuracy. Accuracy has been applied successfully as a genetic algorithm cost function for quantum feature map design in previous work. The first metric is kernel-target alignment, which has previously been used as a training metric in quantum feature map design by variational ansatz training. Kernel-target alignment is a faster metric to evaluate than test set accuracy and does not require any data points to be reserved from the training set for its evaluation. The second metric is an estimation of kernel-target alignment which further accelerates the genetic fitness evaluation by an adjustable constant factor. The second aim of this work is to address the issue of the limited gate parameter choice available to the genetic algorithm. This is done by training the parameters of the quantum feature map circuits output in the final generation of the genetic algorithm using COBYLA to improve either kernel-target alignment or root mean squared error. This hybrid approach is intended to complement the genetic algorithm structure optimization approach by improving the feature maps without increasing their size. Eight new approaches are compared to the accuracy optimization approach across nine varied binary classification problems from the UCI machine learning repository, demonstrating that kernel-target alignment and its approximation produce feature map circuits enabling comparable accuracy to the original approach, with larger margins on training data that improve further with variational training.

## Acknowledgments

I would to thank the National Institute for Theoretical and Computational Sciences (NITheCS) and the Center for Artificial Intelligence Research (CAIR) for funding provided to carry out this research. I would also like to thank my supervisors for providing me their time, guidance, and experienced advice throughout the course of my studies, as well as my friends and family for their support.

# Preface

The research contained in this dissertation was completed by the candidate while based in the Discipline of Computer Science, School of Mathematics, Statistics and Computer Science of the College of Agriculture, Engineering and Science, University of KwaZulu-Natal, Westville, South Africa. The research was financially supported by the National Institute for Theoretical and Computational Sciences (NITheCS) and the Center for Artificial Intelligence Research (CAIR).

The contents of this work have not been submitted in any form to another university and, except where the work of others is acknowledged in the text, the results reported are due to investigations by the candidate.

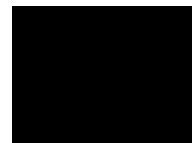
# Declaration: Plagiarism

I, Rowan Pellow-Jarman, declare that:

- (i) the research reported in this dissertation, except where otherwise indicated or acknowledged, is my original work;
- (ii) this dissertation has not been submitted in full or in part for any degree or examination to any other university;
- (iii) this dissertation does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons;
- (iv) this dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
  - a) their words have been re-written but the general information attributed to them has been referenced;
  - b) where their exact words have been used, their writing has been placed inside quotation marks, and referenced;
- (v) where I have used material for which publications followed, I have indicated in detail my role in the work;
- (vi) this dissertation is primarily a collection of material, prepared by myself, published as journal articles or presented as a poster and oral presentations at conferences. In some cases, additional material has been included;



- (vii) this dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.



# Contents

<b>1</b>	<b>Introduction</b>	<b>9</b>
1.1	Introduction to Quantum Computing . . . . .	10
1.1.1	Quantum Information and Qubits . . . . .	11
1.1.2	Single Qubits . . . . .	12
1.1.3	Multiple Qubits . . . . .	17
1.1.4	Quantum Circuit Diagrams and Bell States . . . . .	24
1.1.5	The Deutsch-Jozsa Algorithm . . . . .	26
1.1.6	Real Quantum Hardware . . . . .	32
1.2	Machine Learning . . . . .	34
1.2.1	Classification Tasks and Generalization . . . . .	35
1.2.2	Support Vector Machine . . . . .	37
1.2.3	Quantum Support Vector Machine . . . . .	38
1.3	Ancillary Optimization Methods . . . . .	39
1.3.1	Parameterized Quantum Circuit Optimization . . . . .	39
1.3.2	Genetic Algorithms . . . . .	41
1.4	Prior Works . . . . .	43
1.4.1	Automatic Design of Quantum Feature Maps . . . . .	43
1.4.2	Training Quantum Embedding Kernels on Near-Term Quantum Computers . . . . .	45
1.5	Methods . . . . .	46
1.6	Contributions . . . . .	48
1.7	Research Outputs . . . . .	49

1.8	Thesis Outline . . . . .	49
<b>2</b>	<b>Hybrid Genetic Optimisation for Quantum Feature Map Design</b>	<b>51</b>
2.1	Introduction . . . . .	53
2.2	Methods . . . . .	59
2.2.1	Binary classifiers using quantum kernels . . . . .	59
2.2.2	Kernel quality metrics . . . . .	61
2.2.3	Overview of genetic algorithms . . . . .	63
2.2.4	Experiments . . . . .	64
2.3	Results . . . . .	67
2.4	Conclusion . . . . .	72
2.A	Appendix . . . . .	77
<b>3</b>	<b>Conclusion</b>	<b>95</b>
3.1	Summary of the Work and Key Findings . . . . .	95
3.2	Implications and Significance . . . . .	97
3.3	Future Work . . . . .	97
3.4	Closing Thoughts . . . . .	99



# Chapter 1

## Introduction

The work reported in this thesis aimed to advance methods for designing quantum feature map circuits for use with the Quantum Support Vector Machine (QSVM) classification algorithm. The focus of the investigation was automated feature map design, since manual feature map design may be disadvantaged by relying on humans dealing with the unintuitive nature of quantum physics. Automated feature map design may be more able to make effective use of quantum resources while staying within the limitations of presently available quantum hardware.

The primary content of this work is a paper that was submitted to the Springer Nature journal Quantum Machine Intelligence (<https://www.springer.com/journal/42484>). This chapter provides the context and background to that paper. This chapter thus begins with an overview of quantum computing aimed at computer scientists who may not be familiar with the field. It then provides a brief overview of the machine learning techniques that are studied in this work. A third section discusses the important classical machine learning techniques central to the thesis. The remaining sections describe the prior art, the methods used, and the main contributions of this thesis.

## 1.1 Introduction to Quantum Computing

Quantum computing is a young discipline at the intersection of computer science and physics, which uses principles of quantum mechanics to aim to achieve computational power that is currently infeasible using classical techniques. It is broadly divided into two subfields: the development of quantum hardware and the development of quantum algorithms.

Unlike classical computers, which use bits to store and manipulate information, quantum computers make use of quantum bits (qubits), which can exist in a superposition of the states 0 and 1 simultaneously. Quantum computers can also make use of quantum entanglement. Briefly, superposition refers to the ability of a quantum state to be in a combination of multiple classical states at a single point in time, until it is measured. Entanglement between two quantum states refers to the phenomenon where measurements performed on one of the states will affect the outcome of future measurements performed on the other.

Although superposition can be efficiently simulated by a classical computer using pseudo-random number generation for any number of independent qubits, the combination of superposition and entanglement cannot be simulated classically for large numbers of qubits. These properties, in combination, enable quantum computers to operate on vast amounts of information at a time and perform certain types of computations significantly faster than classical computers. With enough qubits, these machines are, in theory, also able to perform computations that no classical computer that could feasibly be constructed could perform. Some well-known examples of such computations include Shor’s algorithm for factorization [29] and Grover’s search algorithm [13]. They have been shown theoretically to outperform all known classical algorithms for solving the same tasks [13, 29].

Quantum computing has the potential to revolutionize various industries by solving problems that are currently intractable for classical computers or require vast computing resources. These include:

1. Drug Discovery: modeling complex molecules and biochemical reactions, help-

- ing in the discovery and testing of new drugs.
2. Climate Modeling: modeling complex environmental systems, provide more accuracy climate predictions, and help address issues like global warming.
  3. Cryptography: breaking existing encryption methods based on the assumed difficulty of factorizing large numbers, such as RSA encryption.
  4. Supply Chain and Logistics: optimizing routes and solving complex logistics problems more efficiently.
  5. Financial Modeling: enhancing risk modeling, and performing complex financial simulations more accurately.
  6. Artificial Intelligence: potentially speeding up machine learning algorithms, including any new quantum machine learning algorithms that would be inefficient to run on a classical computer.
  7. Material Science: helping in the discovery of new materials by simulating their properties.

Work is ongoing in exploring and evaluating using quantum computing to solve some of these problems. However, realizing the full potential of quantum computers in solving these problems depends on overcoming significant technical challenges to build practical, large-scale quantum computers.

The rest of this section will explain the fundamental principles of quantum computing and quantum algorithms. Some familiarity with basic linear algebra operations such as matrix multiplication and transposition is assumed.

### **1.1.1 Quantum Information and Qubits**

The qubit is the basic unit of information manipulated by quantum algorithms. We focus on the concept of a qubit as a mathematical formalism. Quantum algorithms are often defined in terms of operations performed on an abstraction of a qubit called a logical qubit. A physical quantum hardware device is composed of multiple physical

qubits. These physical qubits are subject to noise due to unwanted interactions with their surroundings. A logical qubit is a mathematical model of an ideal physical qubit (one not affected by noise). Henceforth we will concentrate our attention on logical qubits as they are used throughout this work.

## **Representation of State in Classical and Quantum Computing**

In classical computing, the minimum unit of information is a bit (binary digit). Bits serve as the basic building block for classical information storage and processing. A single bit can be in one of two states, commonly named 0 and 1. The state of a sequence of bits can be represented as the concatenation of the state of each individual bit in the sequence. For the case of two bits, there are 4 possible states the sequence can be in, namely 00, 01, 10, and 11. In general, a sequence of  $n$  bits can be in any one of a total of  $2^n$  possible states, as each bit in the sequence can independently be in one of two states. Thus, an  $n$  bit sequence represents  $2^n$  distinct information states.

Much like classical bits, the state of a qubit or group of qubits can also be precisely described. While a bit can be in state 0 or 1, the state of a single qubit is more complex in that a qubit can exist in a combination of multiple states simultaneously, thanks to quantum superposition. This means it requires more information to fully describe the state of a single qubit than it does a single bit. At the time of measurement, a qubit in superposition collapses into a simple classical bit that is either in state 0 or state 1.

### **1.1.2 Single Qubits**

A single qubit state is described mathematically using a 2-element column vector of complex numbers called probability amplitudes (not to be confused with probabilities, which are real-valued). This vector of probability amplitudes is commonly called a state vector. The two probability amplitudes correspond to the 0 and 1 state of the qubit, respectively, describing numerically the nature of the superposition between

these states. We can relate the values of the probability amplitudes describing a qubit state to the probabilities of that qubit collapsing to a 0 or a 1 state respectively when measured. For example, given a single qubit described by the pair of probability amplitudes  $(\alpha, \beta)$ , we have:

$$P(0) = |\alpha|^2, \text{ and } P(1) = |\beta|^2,$$

where  $P(0)$  and  $P(1)$  are the probabilities of the qubit collapsing into the respective state. A measured qubit is much like a bit and collapses to a single state at a time. This behavior naturally implies a normalization condition on the probability amplitudes (by the nature of complementary probabilities) as follows:

$$P(0) + P(1) = 1 \implies |\alpha|^2 + |\beta|^2 = 1$$

Quantum states are most often denoted using Dirac notation, also known as *bra-ket* notation. The notation  $|\psi\rangle$  (pronounced “ket psi”) refers to a quantum state  $\psi$ , a column vector, while the notation  $\langle\psi|$  (pronounced “bra psi”) refers to the conjugate transpose of  $|\psi\rangle$ , a row vector. The notation  $\langle\phi|\psi\rangle$  refers to the inner product (or dot product) of the states  $\phi$  and  $\psi$ ; the operation is computing the single scalar resulting from the standard matrix product  $\langle\phi|\psi\rangle$ , but omitting one vertical bar and multiplication symbol for brevity.

For a single qubit state, it is conventional notation for the kets  $|0\rangle$  and  $|1\rangle$  to represent the quantum states  $(1, 0)^T$  and  $(0, 1)^T$  respectively. These two states can be seen as the quantum versions of the classical 0 and 1 states; they do not make use of superposition as only one probability amplitude is nonzero. Any arbitrary single qubit state  $|\psi\rangle = (\alpha, \beta)^T$  can be written in terms of these kets as  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ .

The state space of a single qubit can be represented graphically using a diagram known as a Bloch sphere. This is a three dimensional unit sphere where each point on the sphere surface represents a valid qubit state. An arbitrary qubit state  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$  can be mapped to the Bloch sphere by writing it in the form  $\cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle$ , where  $\theta = 2\arccos(|\alpha|)$  and  $\phi = \arg((\beta\alpha^*)/(|\alpha||\beta|))$ .  $\phi$  and  $\theta$  uniquely

define coordinates on the unit Bloch sphere as polar coordinates.

## Single Qubit Gates

We have so far discussed how the quantum information of a qubit is represented using quantum mechanical concepts. Here we describe how to perform computations with that information by manipulating a qubit state. Quantum states can be manipulated by applying quantum gates. A quantum gate is a matrix that a state vector can be multiplied by to produce a new valid state vector. Due to the normalization condition on state vector entries (the sum of the square magnitudes of the elements must equal 1), valid quantum gate matrices are restricted to unitary matrices, as multiplying a state vector by a unitary matrix produces a new state that obeys the state vector amplitudes normalization condition. A unitary matrix is a complex-valued square matrix that has an inverse equal to its conjugate transpose. The conjugate transpose is calculated by taking the element-wise conjugate of the transpose of a matrix, and is indicated with the dagger ( $\dagger$ ) notation. The inverse of a unitary matrix is therefore also unitary. An interesting effect of this property is that all quantum computations are fully reversible before measurement (in the absence of noise).

Quantum gates can broadly be divided into two types: single qubit gates and multi-qubit gates. Single qubit gates are used to change the state of a single qubit, while multi-qubit gates are necessary to create entanglement between different qubits. Below, we define four commonly used single qubit gates. We also demonstrate the effects of applying them on an arbitrary single qubit quantum state  $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ . For each of the following gates  $G$ , we can verify by inspection that  $G = G^\dagger$  due to their small sizes, although this is not true of quantum gates in general. In general, we know quantum gates to be unitary. A matrix  $M$  is defined to be unitary if and only if its inverse is its conjugate transpose, i.e.,  $MM^\dagger = M^\dagger M = I$ . From  $G = G^\dagger$  and  $GG^\dagger = I$ , we have  $GG = I$  and know that each of these 4 gates is its own inverse. Practically, this means that their effects on a qubit can be reversed by applying them a second time.

1. The Hadamard ( $H$ ) gate:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Application of the Hadamard gate to  $|\psi\rangle$  leads to:

$$H |\psi\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} \alpha + \beta \\ \alpha - \beta \end{pmatrix}.$$

An interesting effect occurs when  $|\psi\rangle$  is in the  $|0\rangle$  state or  $|1\rangle$  state. In these cases, we have  $H |0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$  and  $H |1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ .

These two states created by the action of the  $H$  gate have equal probabilities of collapsing to 0 and 1 at measurement, as the square magnitudes of the state vector probability amplitudes are the same. This can be verified by multiplying the external constant into the state vectors and taking the square magnitudes of the probability amplitudes to get probabilities. Due to this behavior, the  $H$  gate is often said to produce an equal superposition of  $|0\rangle$  and  $|1\rangle$  when acting on a  $|0\rangle$  or  $|1\rangle$  state as input. This is a useful operation in many algorithms, since most quantum programs are written with the assumption that the initial state of their  $N$  qubits will be  $|0\rangle^{\otimes N}$ .

2. The Pauli-X gate:

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

Application of the X gate to  $|\psi\rangle$  leads to:

$$X |\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}.$$

As seen above, this gate swaps the  $|0\rangle$  and  $|1\rangle$  components of the qubit's state vector. When the qubit is in a classical state ( $|0\rangle$  or  $|1\rangle$ ), this can be seen as a logical NOT gate operating on a single bit in classical computing. This gate is often also called a quantum NOT gate for this reason.

### 3. The Pauli-Y Gate:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}.$$

Application of the Y gate to  $|\psi\rangle$  leads to:

$$Y |\psi\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -\beta i \\ \alpha i \end{pmatrix}.$$

### 4. The Pauli-Z Gate:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Application of the Z gate to  $|\psi\rangle$  leads to:

$$Z |\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix}.$$

The quantum gates shown above are all constant; their matrices contain only constant entries. Gates defined to make use of unknowns are called parameterized quantum gates and can be used much like classical neural network parameters; they can be trained by an optimizer to minimize some cost value or used to input data into a circuit to operate on. Below, we show three commonly used parameterized gates which are each defined in terms of one free parameter. The gates are named the  $R_x$ ,  $R_y$ , and  $R_z$  gates, and they are so named because they are written in terms of the



Pauli  $X$ ,  $Y$ , and  $Z$  gates respectively.

$$R_x(\theta) = \cos(\theta/2)I - i\sin(\theta/2)X = \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

$$R_y(\theta) = \cos(\theta/2)I - i\sin(\theta/2)Y = \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

$$R_z(\theta) = \cos(\theta/2)I - i\sin(\theta/2)Z = \begin{pmatrix} \cos(\theta/2) - i\sin(\theta/2) & 0 \\ 0 & \cos(\theta/2) + i\sin(\theta/2) \end{pmatrix}$$

### 1.1.3 Multiple Qubits

This description of a qubit can be generalized to describe a sequence of qubits with some work. In contrast to classical bits, the state of an arbitrary system of  $n$  qubits cannot in general be fully described by the simple concatenation of the descriptions of the  $n$  individual qubits due to the phenomenon of quantum entanglement.

Where a single qubit can be in a superposition of 0 and 1 (the possible states of a bit), a sequence of qubits can be in a superposition of all  $2^n$  possible states of a sequence of classical bits *at the same time*. This superposition can be described by a column vector of  $2^n$  probability amplitudes, where each amplitude corresponds to one of the bit sequences that could be read off the qubit sequence after measuring each qubit. Like the single qubit case, the square magnitudes of the probability amplitudes represent the probabilities of measuring the corresponding bit values across the qubit sequence. The probability amplitudes still obey the same normalization condition in that their square magnitudes must sum to 1. An important observation of these definitions is that an arbitrary noise-free quantum state can be exactly described using classical information (although at a space cost exponential in the number of qubits in the state). The space of valid quantum states for a quantum system forms a complex vector space with an inner product operation, called a Hilbert space. The dimensionality of the Hilbert space is the number of probability amplitudes

When working with state vectors, it is convenient to use the convention that

the probability amplitudes are ordered such that their zero-based index written as a binary number is the state that their probability is associated with measuring. In other words, for a state vector  $\vec{x} = (x_0, x_1, x_2, x_3)^T$  describing 2 qubits, we have  $P(00) = |x_0|^2$ ,  $P(01) = |x_1|^2$ ,  $P(10) = |x_2|^2$ , and  $P(11) = |x_3|^2$ .

## The Tensor Product

In classical computing, two separate bit sequences can be combined into a larger classical state through concatenation. The analogous operation for combining two quantum states into a larger composite state is the tensor product operation (written  $\otimes$ ). It is also called the Kronecker product.

The tensor product is defined between two matrices of arbitrary dimensions,  $M_{p \times q}$  and  $N_{r \times s}$ , by the following equation:

$$M \otimes N = \begin{bmatrix} m_{11}N & \cdots & m_{1q}N \\ \vdots & \ddots & \vdots \\ m_{p1}N & \cdots & m_{pq}N \end{bmatrix}.$$

This operation can be visualized as placing a copy of the matrix  $N$  at each element in  $M$ , multiplying the elements of  $M$  into the copies of  $N$ , and then “unpacking” the nested matrix elements into a single, larger matrix. This larger matrix is of dimensions  $pr \times qs$ ; its dimensions are the products of the corresponding dimensions of the original matrices.

## Multi-Qubit States

For the purpose of combining quantum states, we view them as matrices composed of a single column. A straightforward application of the tensor product operation can combine quantum states of any number of qubits into a single composite state that describes the entire combined system. For example, applying the tensor product to

two arbitrary single-qubit states  $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$  and  $|\psi\rangle = \gamma|0\rangle + \rho|1\rangle$  resolves to

$$|\phi\rangle \otimes |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \gamma \\ \rho \end{pmatrix} = \begin{pmatrix} \alpha \begin{pmatrix} \gamma \\ \rho \end{pmatrix} \\ \beta \begin{pmatrix} \gamma \\ \rho \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \alpha\gamma \\ \alpha\rho \\ \beta\gamma \\ \beta\rho \end{pmatrix}$$

The resulting state is now described by four probability amplitudes instead of two. In general, the size of the resulting state is the product of the sizes of the operand states. The tensor product operation preserves the convention that the index of a probability amplitude in binary is also the sequence of bits it is associated with measuring. We verify this convention preservation for the provided example. The convention would state that in the above example, the probabilities of measuring 00, 01, 10, and 11 on the pair of qubits are given by

$$\begin{aligned} |\alpha\gamma|^2 &= |\alpha|^2|\gamma|^2 \text{(as probability amplitudes are complex numbers)} \\ |\alpha\rho|^2 &= |\alpha|^2|\rho|^2 \\ |\beta\gamma|^2 &= |\beta|^2|\gamma|^2 \\ |\beta\rho|^2 &= |\beta|^2|\rho|^2 \end{aligned}$$

respectively. It is important to note that the tensor product operation does not modify the information in the operand states, it only creates a combined state representation. This means that the measurement outcomes for each qubit are independent events;  $P(A \cap B) = P(A)P(B)$  where  $A$  and  $B$  are the measurement outcomes for the two qubits. We can easily observe the convention probabilities are correct for the example case by referring to the probability amplitudes of the original state vectors for the qubits, as they show that the expressions derived by the convention reflect the expected result for probabilities of independent events.

Additionally, the sum of these probabilities is given by

$$\begin{aligned}
|\alpha|^2|\gamma|^2 + |\alpha|^2|\rho|^2 + |\beta|^2|\gamma|^2 + |\beta|^2|\rho|^2 &= |\alpha|^2(|\gamma|^2 + |\rho|^2) + |\beta|^2(|\gamma|^2 + |\rho|^2) \\
&= |\alpha|^2(1) + |\beta|^2(1) \\
&= 1,
\end{aligned}$$

confirming that the normalization condition on the amplitudes is preserved in the example.

Although we only demonstrate combining single-qubit states, the tensor product operation can be applied to combine two quantum states of any number of qubits in general, while preserving state vector index numbering conventions as well as the normalization condition on state probability amplitudes.

### Tensor Product Notational Conventions

In bracket notation, a sequence of kets  $|\psi\rangle |\phi\rangle$  indicates an implicit tensor product of quantum states;  $|\psi\rangle |\phi\rangle = |\psi\rangle \otimes |\phi\rangle$ . Another common notational convention is that a state  $|\psi\rangle^{\otimes N}$  represents a tensor product of  $N$  repetitions of  $|\psi\rangle$ .

### Computational Basis States

By convention, kets with numeric names (for example  $|0\rangle$ ,  $|1\rangle$ ,  $|2\rangle$ ,  $|3\rangle$ ,  $|4\rangle$ , etc.) represent the states that a sequence of qubits can collapse to after measurement. They are essentially the quantum versions of classical states in that they do not represent a superposition. The binary form of the number inside the ket gives the classical bit values that each qubit takes when the qubit system is in this state. The exact number of qubits in a state with a numeric name is context dependent, in exactly the same way that the exact number of bits in a classical representation of a number in binary is context dependent. For example, the number 3 in a 2-bit binary representation is simply 11, while it can be represented with more bits by prepending 0's onto the 2 bit representation. In the same way, the state  $|3\rangle$  represents a set of at least two qubits that are in the one state, followed by some context-dependent number of

qubits in the zero state. These classical states are also called the computational basis states. They form a basis for the space of quantum states since any quantum state can be formed with a linear combination of computational basis states using complex coefficients.

This convention is also sometimes used when the ket name is numeric when interpreted as a binary number;  $|101\rangle$  and  $|5\rangle$  typically represent the same three-qubit state.

We show an example for decomposing an arbitrary two qubit state  $|\psi\rangle = (a, b, c, d)^T$ ,  $a, b, c, d \in \mathbb{C}$  into a weighted sum of computational basis states:

$$\begin{aligned} |\psi\rangle &= (a, b, c, d)^T \\ &= (a, 0, 0, 0)^T + (0, b, 0, 0)^T + (0, 0, c, 0)^T + (0, 0, 0, d)^T \\ &= a(1, 0, 0, 0)^T + b(0, 1, 0, 0)^T + c(0, 0, 1, 0)^T + d(0, 0, 0, 1)^T \\ &= a|0\rangle + b|1\rangle + c|2\rangle + d|3\rangle. \end{aligned}$$

In summary, each computational basis state corresponds to a unique index (and therefore probability amplitude) in the state vector, and any quantum state can be exactly described as a sum of computational basis states weighted by their corresponding amplitude in the state vector.

## Multi-Qubit Gates

The tensor product operation can also be used to construct gates that operate on multi-qubit states. To apply a single qubit gate to a qubit that is part of a multi-qubit state, we can transform it into a multi-qubit form using the tensor product operation.

Suppose we have two single qubit states,  $|\psi\rangle$  and  $|\phi\rangle$ , that form a composite two qubit state,  $|\psi\rangle \otimes |\phi\rangle$ . We can apply single qubit gates  $A$  and  $B$  to each qubit in the state independently by taking the tensor product of their  $2 \times 2$  matrices to form a  $4 \times 4$  matrix which acts equivalently on the 2 qubit state as the single qubit gates would on the independent single qubit gates. This fact is stated by the following

equation:

$$(A|\psi\rangle) \otimes (B|\phi\rangle) = (A \otimes B)(|\psi\rangle \otimes |\phi\rangle).$$

In general, the equation holds if  $A$  and  $B$  are valid gates for acting on quantum states  $|\psi\rangle$  and  $|\phi\rangle$  respectively, regardless of the number of qubits in states  $|\psi\rangle$  and  $|\phi\rangle$ . This principle serves to relate the application of single and multi-qubit gates on larger  $n$  qubit quantum states.

Multi-qubit gates may also be defined from scratch without needing to combine single qubit gates, so long as they are unitary matrices. One such gate of importance in quantum computing is the controlled-X gate, which operates on two qubits. This name is commonly abbreviated as CX or CNOT. It is a gate that can be used to create entanglement between two qubits and is defined as below:

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

We show the application of the CNOT gate to an arbitrary two-qubit state  $|\psi\rangle = a|0\rangle + b|1\rangle + c|2\rangle + d|3\rangle$  below.

$$\text{CNOT}|\psi\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = \begin{pmatrix} a \\ b \\ d \\ c \end{pmatrix}.$$

By inspection, it can be seen that the probability amplitudes of the  $|0\rangle$  and  $|1\rangle$  components of the state are unaffected. These probability amplitudes correspond to the probabilities of measuring 00 and 01 on the qubit pair respectively. The amplitudes for the  $|2\rangle$  and  $|3\rangle$  components, which correspond to measurements of 10 and 11, are swapped.

Note that the 0 and 1 measurement probabilities for the first qubit have not changed after the CNOT application: for each qubit, the probability of measuring it in the 0 or 1 state is just the probability of measuring the set of qubits to be in one of the basis states in which that qubit is a 0 or a 1, respectively. Therefore, for the first qubit, the probability of measuring a 0 is  $P(00) + P(01) = |a|^2 + |b|^2$ , both before and after applying the CNOT. The probability of measuring a 1 before the CNOT application is  $P_i(10) + P_i(11) = |c|^2 + |d|^2$  and  $P_f(10) + P_f(11) = |d|^2 + |c|^2$  afterwards, which are equal.

On the second qubit however, the probability of measuring a 0 before the application of the CNOT is  $P_i(00) + P_i(10) = |a|^2 + |c|^2$  and the probability of measuring a 0 after the application is  $P_f(00) + P_f(10) = |a|^2 + |d|^2$ , which is clearly not in general equal to the previous probability.

If we examine the cases where  $|\psi\rangle$  is a system where the first qubit is in a fully classical state ( $|0\rangle$  or  $|1\rangle$ ) and the second qubit is in an arbitrary state (here represented as  $\alpha|0\rangle + \beta|1\rangle$ ), we can demonstrate the reasoning behind the gates name. Mathematically, these states are computed as  $\text{CNOT}(|0\rangle \otimes (\alpha|0\rangle + \beta|1\rangle))$  and  $\text{CNOT}(|1\rangle \otimes (\alpha|0\rangle + \beta|1\rangle))$ , which we perform below:

Case 1, the first qubit is  $|0\rangle$ :

$$\begin{aligned}
\text{CNOT}(|0\rangle \otimes (\alpha|0\rangle + \beta|1\rangle)) &= \text{CNOT}([1, 0]^T \otimes [\alpha, \beta]^T) \\
&= \text{CNOT}[\alpha, \beta, 0, 0]^T \\
&= [\alpha, \beta, 0, 0]^T \\
&= [1, 0]^T \otimes [\alpha, \beta]^T \\
&= |0\rangle \otimes (\alpha|0\rangle + \beta|1\rangle).
\end{aligned}$$

Case 2, the first qubit is  $|1\rangle$ :

$$\begin{aligned}
\text{CNOT}(|1\rangle \otimes (\alpha |0\rangle + \beta |1\rangle)) &= \text{CNOT}([0, 1]^T \otimes [\alpha, \beta]^T) \\
&= \text{CNOT}[0, 0, \alpha, \beta]^T \\
&= [0, 0, \beta, \alpha]^T \\
&= [0, 1]^T \otimes [\beta, \alpha]^T \\
&= |1\rangle \otimes (\beta |0\rangle + \alpha |1\rangle).
\end{aligned}$$

In both cases, we resolve the CNOT gate application using the rule we developed just prior from applying CNOT to an arbitrary 2 qubit state. By analyzing the states resulting in each case, we can see that the CNOT gate has no effect if the first qubit is in the  $|0\rangle$  state, but has the effect of an X gate applied to the second qubit if the first qubit is in the  $|1\rangle$  state. It is for this reason that the gate is called a controlled-X gate; the state of the first qubit controls the application of an X gate to the second qubit. Additionally, the first qubit is called the control qubit and the second qubit is called the target qubit.

## Universal Gate Sets

According to the Solovay-Kitaev theorem [9], there exist sets of single qubit gates that can be used to approximate any single-qubit quantum gate efficiently. With the addition of the CNOT gate, any multi-qubit gate can be approximated. Such a set, including the CNOT gate, is called a universal gate set. An example of a universal gate set using only gates defined in our work is the set of single qubit parameterised gates  $R_x$ ,  $R_y$ ,  $R_z$ , grouped together with the CNOT gate. Proving the existence of efficient universal gate sets is an important theoretical result, as it implies that real hardware need only reliably implement some finite number of gates to access the full power of quantum computation.



### 1.1.4 Quantum Circuit Diagrams and Bell States

A quantum program is a sequence of single and multi-qubit gates that can be applied to a collection of qubits to manipulate their quantum state. Quantum programs are typically visualized using quantum circuit diagrams. A quantum circuit diagram is the quantum analogue of a logic gate circuit diagram in classical computing. A quantum circuit diagram describes a quantum program visually by specifying the initial states of the qubits, the ordered sequence of quantum gates that are applied to them, and finally the measurements that are performed on the resulting state.


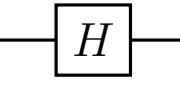
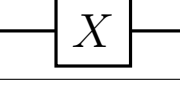
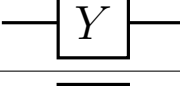
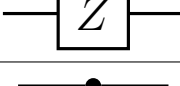
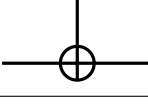

Component	Interpretation
	Empty wire, connects components
	Hadamard Gate
	X Gate
	Y Gate
	Z Gate
	CNOT gate
	Measurement operation

Figure 1-1: A table showing some common components of quantum circuit diagrams. An empty wire is mathematically equivalent to applying an identity gate. Note that the identity matrix  $I$  is unitary and a valid quantum operation. In the CNOT gate diagram, the control qubit is the wire with the small dot end and the target is the wire with the circled plus end ( $\oplus$ ).

Figure 1-1 shows some of the commonly used components of quantum circuit diagrams and explains their meanings. These components can be joined together with wires to describe full or partial quantum programs.

In Figure 1-2, we show an example of a quantum circuit diagram that creates a Bell

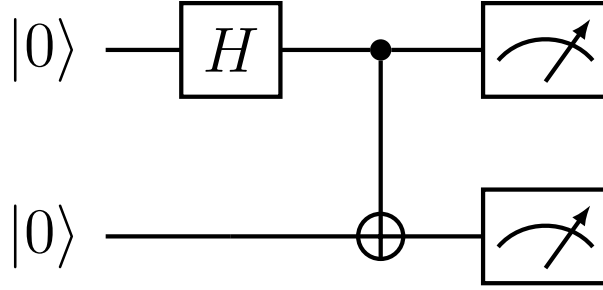


Figure 1-2: A quantum circuit diagram showing the construction of the Bell state  $\text{CNOT}(H|0\rangle \otimes |0\rangle) = 1/\sqrt{2}(|00\rangle + |11\rangle)$ , followed by measurement of both qubits. The initial state of each qubit in the system is specified by the kets on the left side of the diagram. The flow of time is represented by moving from left to right along the lines, which are called wires (gates are applied to qubits in left-to-right order). Quantum gate matrices are represented as matrix names in boxes, where the boxes are positioned on the wires of the qubits that they operate on. The order of application of the gates to the qubits is leftmost first.

state, which is a maximally entangled two-qubit state. This state demonstrates an ability not replicable on a classical computer. We can explain this ability by evaluating the circuit manually. According to the quantum circuit diagram, the first gate to be evaluated is the Hadamard gate on the first qubit. The state of the second qubit is not altered at this point in time. It leaves the pair of qubits in the state  $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle$ . Take note of the fact that matrix multiplication and the tensor products operation are both distributive. Therefore, applying the CNOT gate to this state gives us  $\frac{1}{\sqrt{2}}(\text{CNOT}(|0\rangle \otimes |0\rangle) + \text{CNOT}(|1\rangle \otimes |1\rangle))$ . In general, due to this same principle, the result of applying a quantum gate to a superposition of states is equivalent to forming a superposition of the results of applying the same gate to each state individually. This observation is known as quantum parallelism, but it is not as powerful an effect as it might at first seem due to the fact that quantum superposition states collapse into a single classical state on measurement. We can evaluate the CNOT gates by our previously established understanding of their behavior when processing classical inputs. This gives the state  $\frac{1}{\sqrt{2}}(|0\rangle \otimes |0\rangle + |1\rangle \otimes |1\rangle)$ .

When measurement is performed on either one of the qubits, the state of that

qubit collapses to a simple zero or one. However, due to the nature of the superposition the pair of qubits is in, it is impossible for the qubits to measure different values. This behavior demonstrates an unintuitive quantum principle that can not be reproduced classically. Especially surprising is when we consider that there is no explicit limit on the physical distance between the two qubits at the time of measurement; even if the qubits were separated by a great distance (for example, on the order of light years), measuring the value of one of them would cause the other to leave superposition and collapse to a known value instantaneously. As a result, this correlation of measurement outcomes cannot be explained by local physical theories.

### 1.1.5 The Deutsch-Jozsa Algorithm

We now have the tools to examine a quantum algorithm. In this section we will explain the Deutsch-Jozsa algorithm [11]. This is an early example of a simple quantum algorithm demonstrating a provable advantage over classical computing.

The problem solved by the Deutsch-Jozsa algorithm can be stated as follows: Suppose  $f$  is a black-box function which maps an  $n$ -bit binary string to a single bit, 0 or 1. We state this mathematically as  $f : \{0, 1\}^n \rightarrow \{0, 1\}$ . It is known that  $f$  is either a constant function (the output is always the same, regardless of the input) or a balanced function (the output is 0 for exactly half of the  $2^n$  possible input states and 1 for the other half). How do we determine whether the function is constant or balanced with as few evaluations as possible?

On a classical computer, we must perform at least 2 evaluations of the function in the best case to know the solution; if we evaluate the function for 2 different inputs and get different values, we know the function is not constant and must therefore be balanced. In the worst case, however, we must perform up to  $2^{n-1} + 1$  evaluations. This is because a balanced function may appear to be constant when evaluating up to half of the possible inputs. One additional evaluation is then enough to determine whether the function is constant or balanced.

On a quantum computer, we can use the Deutsch-Jozsa algorithm to determine whether the function is constant or balanced with a single evaluation of  $f$ . To execute

the algorithm, we require a quantum implementation of  $f$ , which we represent as a unitary matrix  $U_f$ .  $U_f$  must be implemented such that

$$U_f |x\rangle \otimes |y\rangle = |x\rangle \otimes |y \oplus f(x)\rangle,$$

where  $|x\rangle$  is an  $n$  qubit state encoding the input to  $f$ ,  $|y\rangle$  is an additional single qubit state, and  $\oplus$  is the addition modulo 2 operator. Note that the exact implementation of  $U_f$  depends on  $f$ , since  $f$  is a black-box function.

To implement the Deutsch-Jozsa algorithm for a function taking  $n$  bit strings as input, we first construct an initial  $n + 1$  qubit state  $|\psi_0\rangle = |0\rangle^{\otimes n} \otimes |1\rangle$ . On a real device, the initial state is typically  $|0\rangle^{\otimes n+1}$ , meaning we can achieve this by simply applying a single X gate on the last of our qubits. The second step is to apply a Hadamard gate to each qubit. This creates the state

$$|\psi_1\rangle = H^{\otimes n+1} |\psi_0\rangle = (H |0\rangle)^{\otimes n} \otimes H |1\rangle$$

Evaluating the Hadamard gate applications, we get

$$|\psi_1\rangle = \left( \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \right) \otimes \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle). \quad (1.1)$$

At this point, the first  $n$  qubits are in an equal superposition of all the possible values of  $n$  classical bits. Next we carry out the third step, which is to apply  $U_f$  to the  $n + 1$  qubits. This gives

$$\begin{aligned} |\psi_2\rangle &= \frac{1}{\sqrt{2^{n+1}}} U_f \sum_{x=0}^{2^n-1} (|x\rangle \otimes (|0\rangle - |1\rangle)) = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} U_f (|x\rangle \otimes (|0\rangle - |1\rangle)) \\ &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (U_f |x\rangle \otimes |0\rangle - U_f |x\rangle \otimes |1\rangle). \end{aligned}$$

By the definition of  $U_f$ , this gives us

$$\begin{aligned}
|\psi_2\rangle &= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (|x\rangle \otimes |f(x)\rangle - |x\rangle \otimes |1 \oplus f(x)\rangle) \\
&= \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (|x\rangle \otimes (|f(x)\rangle - |1 \oplus f(x)\rangle)).
\end{aligned}$$

For each particular value of  $x$ ,  $f(x)$  is either 0 or 1. In the case that  $f(x)$  is 0, we have

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (|x\rangle \otimes (|0\rangle - |1\rangle)).$$

In the case that  $f(x)$  is 1, we have

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (|x\rangle \otimes (|1\rangle - |0\rangle)).$$

From these two observations, we can conclude that

$$|\psi_2\rangle = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} ((-1)^{f(x)} |x\rangle \otimes (|0\rangle - |1\rangle)).$$

We can separate the state as

$$|\psi_2\rangle = |\psi_3\rangle \otimes |\psi_4\rangle,$$

where

$$|\psi_3\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle,$$

and

$$|\psi_4\rangle = \frac{1}{\sqrt{2}} (|0\rangle - |1\rangle).$$

Comparing these components to the state before applying  $U_f$  (refer to equation 1.1), we can see that total effect of  $U_f$  was adding a conditional negation to the

amplitudes of each basis state; each basis state  $|x\rangle$  for which  $f(x) = 0$  experiences no change in its corresponding amplitude, but each basis state for which  $f(x) = 1$  has its amplitude made negative. If  $f$  is constant, either all terms or no terms will have the negation applied. If  $f$  is balanced, exactly half the terms will have the negation applied and half will not.

The fourth step is to apply a layer of Hadamard gates to the first  $n$  qubits, which are in state  $|\psi_3\rangle$ . Before this step, it is useful to observe the effect of applying a Hadamard gate layer to a single qubit in the 0 or 1 state, and then to an arbitrary  $n$  qubit computational basis state  $|x\rangle$ .

When applying the Hadamard gate to a single qubit not in superposition, we get

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \text{ or } H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle),$$

depending on whether the qubit has a value of 0 or 1. In the case of the  $n$  qubit state  $|x\rangle$ , we have

$$H^{\otimes n}|x\rangle = \otimes_{i=0}^{n-1} H|x_i\rangle$$

where  $x_i$  is the bit on qubit  $i$ .

Looking at a 3 qubit example with  $x = 011$ , we get

$$\begin{aligned} H^{\otimes 3}|x\rangle &= H|0\rangle \otimes H|1\rangle \otimes H|1\rangle \\ &= \frac{1}{\sqrt{2^3}} ((|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle) \otimes (|0\rangle - |1\rangle)) \\ &= \frac{1}{\sqrt{2^3}} (|000\rangle - |001\rangle - |010\rangle + |011\rangle + |100\rangle - |101\rangle - |110\rangle + |111\rangle). \end{aligned}$$

We can see that in general (including for cases using more qubits and an arbitrary basis state for  $x$ ), the result is a summation over all  $n$  bit basis states. The number of negative coefficients in the components multiplied to produce the basis state  $|y\rangle$  in the summation above is just the number of corresponding bits that both equal 1 in  $x$  and  $y$ , since each such bit means that a negated  $|1\rangle$  was used in the construction of

$y$ . We can represent this total mathematically as the dot product of the bits of  $x$  and  $y$ , which we denote  $x \cdot y$ . This quantity is important because it allows us to express the effect of a layer of Hadamard gates on a computational basis state  $|x\rangle$  as

$$H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle.$$

We are now ready to carry out step 4 and apply a layer of Hadamard gates to  $|\psi_3\rangle$  to create a final state  $|\psi_5\rangle$  using the identity developed above:

$$\begin{aligned} |\psi_5\rangle &= H^{\otimes n} |\psi_3\rangle = H^{\otimes n} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} H^{\otimes n} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} \left( (-1)^{f(x)} \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} (-1)^{x \cdot y} |y\rangle \right) \\ &= \frac{1}{2^n} \sum_{y=0}^{2^n-1} \sum_{x=0}^{2^n-1} ((-1)^{f(x)} (-1)^{x \cdot y} |y\rangle) = \sum_{y=0}^{2^n-1} \left( \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{x \cdot y} \right) |y\rangle. \end{aligned}$$

In this form, we can see that the probability amplitude  $\alpha_y$  of any particular basis state  $|y\rangle$  in state  $|\psi_5\rangle$  is simply

$$\alpha_y = \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} (-1)^{x \cdot y}.$$

Consider the case where  $|y\rangle = |0\rangle$ . For this state, we can further simplify to

$$\alpha_0 = \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)}.$$

We know  $f$  is either constant or balanced. If  $f$  is constant, this simplifies to either

$$\alpha_0 = \frac{1}{2^n} \sum_{x=0}^{2^n-1} 1 = \left(\frac{1}{2^n}\right)(2^n)(1) = 1.$$

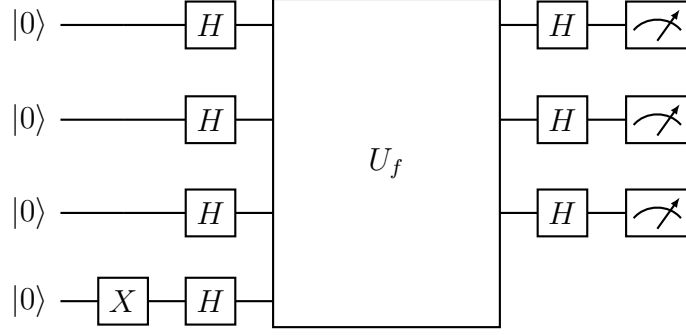


Figure 1-3: This circuit diagram shows the Deutsch-Jozsa algorithm being executed for a function  $f$  that takes 3-bit inputs. The X gate on the last qubit initializes it to the single-qubit  $|1\rangle$  state. It can be omitted if the last qubit starts in the  $|1\rangle$  state.

or

$$\alpha_0 = \frac{1}{2^n} \sum_{x=0}^{2^n-1} -1 = \left(\frac{1}{2^n}\right)(2^n)(-1) = -1.$$

depending on whether  $f(x) = 0$  or  $f(x) = 1$ . In both cases, the probability of measuring all qubits to be zero is then  $P(|0\rangle) = |\alpha_0|^2 = |1|^2$  or  $|-1|^2 = 1$ . This proves that if  $f$  is constant, we will definitely measure the 0 state on all qubits.

If  $f$  is balanced, then  $f(x) = 0$  for half of the values of  $x$  and  $f(x) = 1$  for the other half, and the expression instead simplifies to:

$$\alpha_0 = \frac{1}{2^n} \sum_{x=0}^{2^n-1} (-1)^{f(x)} = \frac{1}{2^n} ((2^{n-1})(-1) + (2^{n-1})(1)) = 0,$$

and the probability of measuring all qubits to be 0 is  $P(|0\rangle) = |\alpha_0|^2 = |0|^2 = 0$ . This proves that if  $f$  is balanced, we cannot measure 0 on all qubits.

If we prepare state  $|\psi_5\rangle$  and measure it to find all qubits are 0,  $f$  cannot be balanced and must therefore be constant. If we measure all qubits and find that some are not zero, we can then conclude that  $f$  cannot be constant and must therefore be balanced.

In summary, constructing the state  $|\psi_5\rangle$  requires only one evaluation of  $f$  through the  $U_f$  unitary to determine if  $f$  is balanced or constant, whereas the best classical



solution requires  $2^n/2 + 1$  evaluations of  $f$ , a value that scales exponentially with the problem size  $n$ . This proves that the quantum algorithm achieves an exponential speedup over the classical algorithm. Figure 1-3 shows the quantum circuit diagram of the entire algorithm for the case of a function operating on three bits.

### 1.1.6 Real Quantum Hardware

Unlike the logical qubit abstraction we have been working with so far, real physical qubits are subject to unwanted interactions with their surrounding environment, and applying quantum gates is not entirely precise due to engineering challenges. The umbrella term “*noise*” is used to refer to any effect on quantum hardware that causes an unwanted influence on a quantum state. The two major types of noise affecting quantum hardware are *dissipation* and *decoherence*. Dissipation refers to noise induced by the quantum system losing energy due to the surrounding environment. The energy loss to the environment leads to a change in the energy state of the qubits, directly changing the information encoded in the energy states. This can cause a qubit in the excited state ( $|1\rangle$ ) transitioning to the ground state ( $|0\rangle$ ), but not the reverse transition. Decoherence refers to the decay of a system’s quantum properties due to unintended interactions with the environment, which can happen without directly causing a loss of energy. Both types of noise have increasing cumulative influence on the quantum state with time. This limits the achievable length of useful programs. As a general rule, the higher the number of gates applied in a circuit, the more the computation will be affected by noise. This is especially true of multi-qubit gates like CNOT. The presence of noise in a hardware device often makes it unsuitable for running algorithms. For the Deutsch-Jozsa algorithm, even a small unintended modification to a single qubit after producing the final state could allow us to measure non-zero values and wrongly conclude that a constant function is balanced.

The current era of quantum devices is known as the Noisy Intermediate-Scale Quantum (NISQ) era [23]. These are devices with low qubit counts (up to a few thousand), limited entanglement connectivity for superconducting qubits (meaning multi-qubit gates cannot be directly applied between arbitrarily positioned qubits),

and relatively high error rates. Although ion trap qubits achieve better entanglement connectivity, they cannot apply more than one two-qubit gate in a single time step. Computing with future machines that have more qubits and lower error rates, combined with techniques to account for noise, is referred to as fault-tolerant quantum computing (FTQC).

There are two main proposals for dealing with noise in quantum hardware: Quantum error correction (QEC) [26], and quantum error mitigation [5]. Quantum error correction involves bringing additional qubits into the computation in such a way that errors can be detected and corrected without interrupting the computation. QEC will require the use of many additional qubits to achieve noise resistance, making it unsuitable for use on currently available devices. However, it could be suitable for future devices with greater qubit counts and lower gate error rates. Quantum error mitigation refers to methods that minimize the negative effects of noise rather than trying to correct for it. This can be done by modeling the effects of noise with a particular device and post-processing measurements to try to account for them. An example of error mitigation is zero-noise extrapolation [12], where computations are performed at multiple adjustable noise levels and the results without noise are extrapolated. These are notably less effective at dealing with noise than error correction, but are important for implementing practical algorithms on NISQ devices.

The other major area of our work is quantum machine learning. The following section gives an overview of pertinent concepts.

## 1.2 Machine Learning

Machine learning is a branch of artificial intelligence that studies algorithms designed to create models of functions from data. When successfully applied, this allows computers to be used to perform tasks that are difficult to explicitly program solutions for, such as understanding the nuances of human language in natural language processing [17] or making sense of image data in computer vision [14] tasks.

These algorithms can broadly be separated into three categories: supervised learn-

ing, unsupervised learning, and reinforcement learning. Supervised learning algorithms make use of prelabelled training data, while unsupervised algorithms attempt to uncover the underlying structure of a dataset without the explicit guidance of labels [1]. In reinforcement learning, agents learn behavior by interacting with an environment that provides rewards and punishments to guide them.

An industry trend in recent years is a rapidly increasing scale of AI models, particularly in the domain of language modeling [33]. These large models are termed Large Language Models (LLMs) due to their size being a defining characteristic. LLMs are expensive and slow to train and require vast computational resources to make training feasible. An interesting question to consider is whether or not the computational power of future quantum computers could be used to alleviate this problem.

The field of Quantum Machine Learning (QML) lies at the intersection of quantum computing and machine learning. The term “quantum machine learning” can refer to either the application of classical machine learning algorithms to quantum data (to be specific, classical data generated from a quantum source), or the application of quantum algorithms to machine learning tasks on either quantum or classical data. Quantum algorithms are expected to have a clear advantage over classical algorithms when operating on quantum data due to the fact that they can in principle operate directly on quantum states without collapsing them and losing much of the information they contain. However, it is not yet clear whether quantum algorithms can provide a significant practical advantage over classical algorithms when working with machine learning tasks on classical data. This question is a high priority research focus in the field of QML, but is difficult to answer.

Another interesting research question is whether a quantum algorithm can be designed that performs better on some useful machine learning task than any known classical algorithms. Although provably faster quantum machine learning algorithms have been discovered [18], they are not necessarily of practical relevance. Note that in order to provide a true quantum advantage over classical algorithms the quantum algorithm must be computationally hard to simulate on classical computers, or else it merely constitutes a new quantum-inspired classical algorithm.

### 1.2.1 Classification Tasks and Generalization

One class of supervised learning tasks is classification. In classification tasks, a model is required to learn a function that maps an input data point to a label from a set of class labels by determining a mapping based on labeled training data. An example of a classification problem is the task of determining whether or not an input image contains a human face in it. Mathematically, this can be formulated as the task of modeling a function that maps an image data array to one of two labels, corresponding to the classes “face” and “no face”. We know that every image either does or does not contain a human face by necessity, and therefore this function does exist in the mathematical sense. In this classification task, the goal of a model is to learn a computable implementation of an approximation of this abstract function through a training process. A supervised machine learning classifier model would be trained to perform this task using labeled example inputs that pair images with their known-correct labels. In mathematical terms, the labeled training data contains a subset of the domain of the function being modeled, as well as the image of that subset in the function’s range. A single sample in the training data is typically represented as a real-valued sequence of *feature values*. A feature value is a number that corresponds to some real-world descriptor that should be relevant to solving the problem. In the case of image classification, the numeric pixel intensity values for each color channel in an image are typically used as feature values. Note that a trivial model can often be defined that perfectly approximates a target function’s operation on training data; it must simply record the entire training set and look up correct labels when queried for an output. However, such a model would not be useful as it would be highly unlikely to correctly approximate the true function when queried for the labels of points not included in the training data.

This introduces a concept of great importance in supervised classification tasks: the concept of generalization. The term *generalization* refers to the ability of a model to maintain the accuracy of its approximation of the true labeling function when applied to data not seen during its training. A classifier is said to generalize well

when it accurately approximates the true labeling function on unseen data. For supervised classifiers, this can be quantified by comparing the accuracy of the model on the training data with the accuracy of the model on a reserved set of test data points with known labels that are not given to the model during training. In practice, good generalization is a requirement for a trained model to be useful.

A popular algorithm for supervised classification tasks is the Support Vector Machine (SVM) [4] due to its empirically good performance, theoretical relations to other learning models [16, 28] and theoretical results regarding its generalization ability [32, 31].

### 1.2.2 Support Vector Machine

The SVM algorithm is a supervised machine learning algorithm for solving classification tasks. It can be seen as a higher-dimensional generalization of the idea of drawing a straight line to separate two clusters of points on a Cartesian plane; it makes use of a hyperplane to linearly separate two clusters of points in an arbitrary-dimensioned space of feature values. The SVM training process partitions the coordinate space into two disjoint subsets; the goal of the SVM training process is to find an optimal vector  $w$  and optimal scalar  $b$  such that the sign of  $w \cdot x + b$  is the same for all points  $x$  that belong to the same class. This forms a simple classification rule for any point  $x$  in the space; the equation  $w \cdot x + b = 0$  defines a hyperplane in the space, and taking the sign of  $w \cdot x + b$  acts as a prediction rule assigning every point in the space to one of the two classes.

A hyperplane is defined by a simple linear equation, which has the downside that it is unable to learn nonlinear classification rules, which may be required to model many interesting functions. However, there is the benefit that the training algorithm can select an optimal hyperplane in the cases where the training data is linearly separable. This optimality is with respect to a *margin* metric. The margin of a data point is defined as its distance from the hyperplane in coordinate space, where greater values for the margin intuitively correspond to higher confidence in the prediction being made. The margin metric is also defined for a choice of hyperplane

in the context of some dataset; the margin of a candidate hyperplane is the minimum of the margins of the points in that dataset. The SVM training algorithm achieves an optimal margin for linearly separable data in that it picks the choice of hyperplane that maximizes the minimum of the margins of the points in the training data. Larger margin sizes have been shown theoretically to correlate with better generalization ability [32, 31].

The downside of using a linear classification rule is overcome in practice by first applying a non-linear transformation function to put the data points into a higher dimensional space before applying the hyperplane separation. This transformation function is commonly called a *feature map*. Adding non-linearity and increasing the dimensionality of the data points makes them easier to linearly separate in the higher dimensional space as shown in Figure 1 of chapter 2, greatly improving the class of functions the SVM algorithm is capable of modeling while maintaining the margin optimality benefit.

Another strength of the SVM algorithm is the fact that the hyperplane optimization problem has a dual formulation which casts the problem in terms of the inner products between points rather than their distances from the hyperplane [4]. This allows for the use of complex feature map functions that are hard to explicitly compute, so long as the inner products of points in the higher dimensional mapped space can be efficiently computed. For a feature map function  $\phi(x)$  performing some non-linear mapping of a data point to a higher dimensional space, the function

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

can be defined to map to the inner product of two data points in the higher dimensional space. Such a function  $\kappa$  is called a *kernel* function. This removal of the explicit need for a feature map function is known as the “*kernel trick*”, and it allows the SVM algorithm to access complex feature maps indirectly through their corresponding kernel functions for greater expressivity.

### 1.2.3 Quantum Support Vector Machine

A parameterized quantum circuit (a quantum circuit making use of parameterized gates) can be made to operate as a feature map function. This is achieved by substituting the feature values of data points into the parameters of the parameterized gates. Executing the resulting circuit maps the input data point to a quantum state in a high dimensional Hilbert space. The quantum circuit used to perform the mapping is called a feature map circuit. This feature map cannot be directly used with the SVM algorithm since the mapped data is still in the form of a quantum state. However, any feature map circuit can easily have a kernel circuit defined by construction, which instead operates on two data points and allows computing the inner product of the mapped quantum states on the quantum device. This kernel circuit can be used as a kernel function in the SVM algorithm, thanks to the kernel trick. In the quantum machine learning literature, kernel circuits are also referred to as Quantum Embedding Kernels (QEKs) and the practice of using them to evaluate kernels is called Quantum Kernel Estimation (QKE). When using a quantum kernel with the SVM algorithm, the resulting algorithm constitutes a hybrid quantum-classical machine learning algorithm known as a Quantum-Enhanced Support Vector Machine (QSVM) [25, 27]. The choice of feature map circuit is both important and data-dependent; a good choice of feature map must make data linearly separable in the higher dimensional space.

Note that the potential advantage of the QSVM algorithm is in using quantum hardware to access kernels that cannot be efficiently estimated classically, since if some well-performing quantum kernel can be efficiently estimated classically, there is no requirement to use a quantum device.

The research in this work deals with improving existing methods for the automated design and optimization of feature map circuits for use with the QSVM algorithm.

## 1.3 Ancillary Optimization Methods

Understanding the experiments performed in this work requires understanding some classical techniques used in quantum circuit design and optimization, which are detailed below. While they are not the focus of the work’s contributions, they are used in the experimental setup.

### 1.3.1 Parameterized Quantum Circuit Optimization

A common paradigm in NISQ algorithm development is the optimization of the parameters of parameterized quantum circuits (PQCs) to be fit for some purpose. The structure and placement of parameters in the PQC is generally dependent on the task that it will be trained to perform. For this reason, PQCs are often also called *ansätze* (the plural form of *ansatz*). “*Ansatz*” is a German word which in scientific fields refers to an assumption made about how to find the solution to a problem, often in the context of making an approximation, model, or theory.

While circuits running on real devices can only directly convey outputs through measurements, these measurements can be used as a basis upon which to develop more complex outputs. The form of the complex outputs can often be problem-dependent. For example, repeatedly executing a circuit and measuring the produced state can provide an empirical estimate of the probabilities that the state collapses to any of its possible basis states. In a machine learning classification task, these probabilities could be interpreted as membership probability predictions for classes.

Outputs produced by PQCs can be optimized through the use of classical optimization algorithms in a hybrid quantum-classical optimization loop as follows:

1. A classical device sets some initial parameters for a PQC.
2. The quantum device executes the PQC and returns information about the measurements made on the produced state to the classical device.
3. The classical device interprets these measurements to produce a problem-dependent metric value, which is given to a classical optimization algorithm. If the opti-



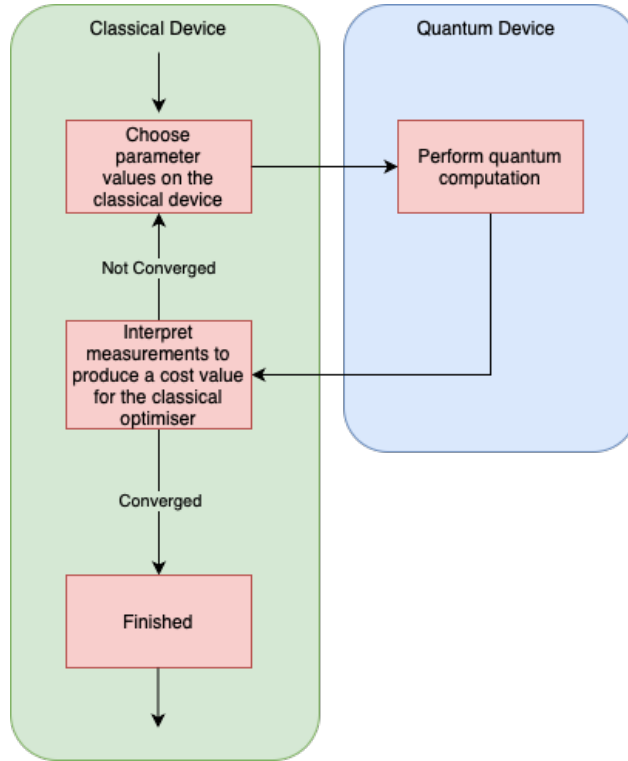


Figure 1-4: A flow diagram demonstrating the PQC optimization process.

mizer deems that the metric value has converged or reached alternate stopping conditions, the optimization loop ends.

4. If the metric has not converged, the optimizer suggests new values for the PQC parameters and the algorithm continues from step 2.

An interesting observation is that the process of PQC optimization is very similar to classical neural network training; in both PQC optimization and neural network training, a parameterized model of fixed structure is defined which is believed to be of general enough form to implement a desired computation. Then, the model has its trainable parameters iteratively updated by a classical optimization algorithm to improve some output metric.

### 1.3.2 Genetic Algorithms

Genetic algorithms are a widely-studied class of algorithms that can be applied to a wide variety of combinatorial optimization problems [6]. They are a type of nature-

inspired algorithm that mimic the natural process of evolution with digital analogues of genetic information, reproduction, random mutation, and natural selection.

The concept of genetic information is mimicked by representing solutions to the target problem as encoded data. This encoded data is often in the form of a binary string. Genetic algorithms are initialized with a collection of solutions called a population. In principle these can be derived from any source, such as another optimization process or simple random sampling. A decoding function mapping from the chosen representation to problem solutions must be defined on a per-problem basis. An objective function to optimize must also be defined. Applying the decoding to individuals in the population allows evaluating the objective function for each individual.

The solution representations undergo unary and binary genetic operations which produce a new population of solutions. These operations model mutation and reproduction respectively. The new population is referred to as the next generation of solutions. For binary strings, mutation of a solution is typically performed by flipping some bits in its bit string and reproduction is typically performed by concatenating subsequences of the two parent solutions. Solutions are typically selected for undergoing genetic operations with probability proportionate to their objective values. As a result, the genetic information of well-performing solutions is more likely to be used in the construction of the new population. This mimics the effects of natural selection and biases the generation of the new population towards producing better-performing solutions than the parent generation. Although binary strings are typically used to represent solutions, an alternate representation tailored to a problem can also be defined, so long as genetic operators can be implemented for it.

The process of decoding solution representations, evaluating solutions, ranking solutions by objective values, and producing new solutions through reproduction and mutation is performed iteratively until a predefined stopping condition is met. There are many variations of genetic algorithms due to the open-ended nature of implementing evolution-inspired genetic operations and representing solutions [6].

Genetic algorithms have previously been applied to the problem of quantum circuit

design [2, 3, 7, 19, 24].

## Non-dominated Sorting Genetic Algorithm II

Non-dominated Sorting Genetic Algorithm II (NSGA-II) [10] is a variation of the genetic algorithm which customizes the genetic selection and fitness evaluation operations to optimize multiple fitness functions simultaneously. This is done by preferentially selecting non-dominated solutions as parents when creating new individuals. While non-domination is defined for problems with an arbitrary number of objective functions, we define it here for only the case of two objective functions as our work does not require more.

In a minimization problem with two fitness functions, a solution  $s$  with fitness values  $(a, b)$  is considered non-dominated with respect to another set of  $n$  solutions with fitness values  $\{(f_i, g_i) | i \in \{1, 2, \dots, n\}\}$  if and only if

$$\forall i \in \{1, 2, \dots, n\}, (f_i < a) \implies (g_i > b),$$

i.e. if and only if there are no solutions in the set with all fitness values superior to the corresponding fitness values of  $s$ .

NSGA-II also makes use of elitism to guarantee the preservation of the solutions that best optimize at least one of the individual fitness functions. The metric of crowding distance is used in the selection process to promote diversity in the achieved fitness values. This can help avoid early convergence in a local minimum.

NSGA-II has been applied to quantum circuit design in previous works due to the fact that multi-objective optimization allows minimizing circuit sizes (for NISQ suitability) in addition to creating circuits that perform well at a task [2].

## 1.4 Prior Works

The experiments in our work build mainly off of two prior works to do with QSVM feature map circuit design and optimization. The first work, *Automatic Design of*

*Quantum Feature Maps* [2], details and implements a genetic algorithm which automates the design of feature map circuits tailored to a specific dataset. We reproduce this system as a baseline for comparison. The second work, *Training Quantum Embedding Kernels on Near-Term Quantum Computers* [15], investigates using the metric of kernel-target alignment to optimize feature map circuits containing trainable parameters in addition to the usual data embedding parameters of a feature map circuit. Kernel-target alignment is a metric that measures the degree of agreement between the kernel function outputs for each pair of data points in a training data set and a hypothetical ideal kernel output for that pair. We use this same technique with multiple metrics (including kernel-target alignment) to optimize the circuits produced by our genetic algorithm implementation.

### 1.4.1 Automatic Design of Quantum Feature Maps

*Automatic design of quantum feature maps* [2] implemented an NSGA-II genetic algorithm for automatically designing quantum feature map circuits for QSVM classification. Given a target dataset, NSGA-II is used to pick a feature map circuit that simultaneously minimizes the number of gates used and maximizes classification accuracy when used in a QSVM classifier on that target dataset.

Before running the genetic optimization algorithm, parameters  $M$  and  $N$  must be selected. These designate the maximum number of qubits and maximum number of gate layers in generated circuits respectively, and they are fixed for the duration of the optimization.

For the purposes of applying genetic operations, the work encodes feature map circuits as a concatenation of encodings of the  $MN$  potential gates in the circuit. The encoding of a single gate is a string of 5 bits, which are interpreted independently as shown in Table 1.1. This means each solution is a binary string of length  $5MN$ . When decoding the bit strings, the groups of 5 bits are decoded sequentially to fill in the gates for each qubit in each layer of the feature map sequentially. Stated explicitly, for each consecutive group of 5 gate bits, the  $i$ th gate description will be applied to qubit  $i \bmod M$  and will encode feature  $i \bmod N$  if it performs a parameterized

Bits	Gate	Bits	Parameter
000	H	00	$\pi$
001	CNOT	01	$\pi/2$
010	I	10	$\pi/4$
011	$R_x$	11	$\pi/8$
100	$R_z$		
101	I		
110	I		
111	$R_y$		

Table 1.1: When decoding a 5 bit sequence that encodes a single gate, the first 3 bits of the sequence determine which type of gate the sequence decodes to, according to the left side of the table. H, CNOT, and I gates refer to the Hadamard, controlled not, and identity gates respectively. The  $R_x$ ,  $R_y$ , and  $R_z$  gates are parameterized gates defined by the equations  $R_x(\theta) = \cos(\theta/2)I - i\sin(\theta/2)X$ ,  $R_y(\theta) = \cos(\theta/2)I - i\sin(\theta/2)Y$  and  $R_z(\theta) = \cos(\theta/2)I - i\sin(\theta/2)Z$ . In the case of the first 3 bits selecting a parameterized gate, the last 2 bits select a proportionality parameter from the right side of the table. Input data features are multiplicatively scaled by this proportionality parameter before being substituted into the parameterized gate. If the multi-qubit CNOT gate is selected to be applied to qubit  $i$ , it will use qubit  $i$  as the control and qubit  $i + 1$  as the target qubit, with addition being modulo the number of qubits in the feature map circuit.

rotation.

While accuracy maximization was performed directly on the measured accuracy metrics for each feature map, size minimization was instead performed on a proxy size metric with importance weighted to scale with accuracy. Before weighting, the size metric SM is computed as

$$SM = \frac{N_{\text{local}} + 2N_{\text{CNOT}}}{M},$$

where  $N_{\text{local}}$  is the number of single qubit gates in the circuit and  $N_{\text{CNOT}}$  is the number of multi qubit CNOT gates. With accuracy weighting, the weighted size metric WS is computed as

$$WS = SM + SM \cdot \text{accuracy}^2.$$

According to the authors, this weighting shifts the focus towards decreasing circuit

size once good classification accuracy has been achieved.

The work found that the approach was suitable for designing high accuracy feature map circuits with a low number of gates that make little use of entanglement [2].

### 1.4.2 Training Quantum Embedding Kernels on Near-Term Quantum Computers

Hubregtsen *et al.* [15] apply parameterized quantum circuit optimization to the task of optimizing existing feature map circuits for a target dataset. In contrast to the previous work [2], this approach does not automate the selection of the gates used in the circuit or the circuit size. The only parts of the feature map circuit that are modified are the values of its trainable parameters. The trainable parameter values were optimized using a Stochastic Gradient Ascent optimization algorithm to maximize the kernel-target alignment of the corresponding kernel functions. Given a set of training data points  $\{x_1, x_2, \dots, x_n\}$  and labels  $\{y_1, y_2, \dots, y_n\}$  where  $\forall i, y_i \in \{-1, 1\}$ , and a kernel function  $\kappa(x_i, x_j)$ , the kernel-target alignment of  $\kappa$  on the data set is defined by the equation

$$\text{KTA} = \frac{\langle K, O \rangle_F}{\sqrt{\langle K, K \rangle_F \langle O, O \rangle_F}},$$

where  $O$  and  $K$  are  $n \times n$  matrices with  $O_{ij} = y_i y_j$ ,  $K_{ij} = \kappa(x_i, x_j)$ , and  $\langle \cdot, \cdot \rangle_F$  is the Frobenius inner product between matrices. This maximization of kernel-target alignment was performed to improve the suitability of feature maps for target datasets, as increased kernel-target alignment has previously been shown to correlate with improved classification ability by Cristianini *et al.* [8] .

The work made use of both classical noise-free simulations of quantum computers and real NISQ computers to run their experiments. The authors reported improvements in classification accuracy after training feature map circuits to maximize kernel-target alignment for the target dataset [15].

While the previous works demonstrate that quantum feature maps can be designed and optimized with automated methods, they do not answer the question of whether

designing the circuit structure with an automated algorithm can be effectively applied in tandem with feature map parameter optimization. This would potentially allow for maximizing the performance of the small feature map circuits suitable for execution on NISQ devices.

Another issue with the proposed methods for automating the structural design of feature map circuits is that they make use of genetic algorithms with fitness functions that are slow to evaluate. This is compounded by the fact that genetic algorithms are a population-based algorithm which may need to evaluate many generations of solutions, meaning the evaluation speed of the cost function is very often the limiting factor to how fast they run.

In this work, we intend to investigate whether making use of a hybrid approach incorporating both genetic algorithms and feature map parameter optimization will demonstrate the benefits of both algorithms simultaneously. We also aim to investigate alternative fitness functions for genetic algorithm based feature map design to improve the running time of the algorithm without degrading the performance of the generated feature map circuits.

## 1.5 Methods

In this work, we set out to investigate potential improvements that could be made to the genetic algorithm for quantum feature map design defined in [2]. Our investigation focused on the use of alternative metrics to accuracy in the genetic algorithm, as well as performing parameter optimization on the circuits generated by the original approach. In particular, we use the COBYLA [22] classical optimizer to optimize circuit parameters.

The experimental core of the work involves evaluating the performance of nine variations and extensions of the algorithm in the original work, including a reproduction of the original algorithm itself. Our first three variations are determined by the metric maximized by the genetic algorithm: in the original work this was classification accuracy, but we also use kernel-target alignment and an approximation of

kernel-target alignment that is faster to compute.

These three variations form the base for an additional six variations for comparison; we form two additional variations for each of the three base variations by further optimizing the proportionality parameters of the feature map circuits output by genetic algorithm to either minimize the root mean squared error (RMSE) of the SVM decision function output or maximize kernel-target alignment. We use a modified formulation of the error calculation for RMSE as the SVM decision function has no fixed optimal target value.

The nine variations are evaluated over nine datasets to determine the classification accuracies and margin sizes achieved by the produced circuits in each case. This data is used to determine whether each approach is effective, as well as the effectiveness of each approach relative to the others. The code for running the experiments is made available on GitHub [20].

## RMSE Calculation

In our RMSE calculations, we calculate the error for a decision function output  $a$  and training label  $b$  using the following rule:

$$\text{error}(a, b) = \begin{cases} (b - a), & \text{if } b = 1 \text{ and } a < b, \\ (a - b), & \text{if } b = -1 \text{ and } a > b, \\ 0, & \text{otherwise.} \end{cases}$$

This choice of error function means that the decision function is penalized for distance from the class label, unless it exceeds the magnitude of the class label while still making a correct classification, in which case the error is 0.

The RMSE is calculated over a dataset by taking the root of the mean of the squares of the errors of the SVM decision function outputs relative to the known class labels.



## Kernel-Target Alignment Approximation

Our kernel-target alignment approximation for a feature map circuit and dataset is simply calculated by averaging the kernel-target alignment of the feature map circuit over some number of disjoint subsets  $k$  of the dataset. In Chapter 2, we prove mathematically that this results in a constant factor reduction of  $k$  in the number of kernel evaluations required to compute the metric.

## 1.6 Contributions

This work makes three main contributions:

1. Experimentally verifying the effectiveness of kernel-target alignment as a metric for genetic algorithm based quantum feature map design.
2. Proposing an approximation of kernel-target alignment that requires fewer quantum kernel evaluations to compute, and experimentally demonstrating its effectiveness in genetic algorithm based quantum feature map design.
3. Demonstrating the synergistic effectiveness of applying both genetic feature map circuit design and feature map circuit parameter optimization in the task of automating quantum feature map circuit design.

With regards to the first contribution, our findings showed that using kernel-target alignment as a replacement for accuracy in the design of quantum feature maps by genetic algorithm does produce feature maps that are effective for classifying the target datasets with accuracy comparable to the original approach and larger margin sizes. However, the circuits produced by this approach are notably larger than those produced when using accuracy as the metric to maximize. We hypothesize that this could be due to the fact that the limiting value of kernel-target alignment is harder to achieve than the limiting value of accuracy, meaning the genetic optimization spends less time trying to minimize the circuit size than when accuracy is used as the metric to maximize.

Our kernel-target alignment approximation in the second contribution performs similarly to kernel-target alignment in the genetic algorithm feature map generation task, while requiring fewer kernel evaluations to compute. Like un-approximated kernel-target alignment, it also achieves similar accuracy to and larger margin sizes than the original approach when used as the metric to maximize in the genetic optimization process.

Our third contribution shows that further optimizing the proportionality parameters of the circuits generated by the genetic algorithm using classical optimizers noticeably improves the margin sizes achieved by the circuits. This also implies that the choice of circuit parameter values made by the genetic algorithm is not optimal and could be improved.

## 1.7 Research Outputs

The paper in chapter 2 was uploaded to arXiv.org (<https://arxiv.org/abs/2302.02980> [21]) and has been submitted to the Springer Nature journal Quantum Machine Intelligence (<https://www.springer.com/journal/42484>).

The experiment and graphing code developed during this work has been made publicly available on GitHub [20].

## 1.8 Thesis Outline

The second chapter presents the main output of the work where the above contributions are demonstrated, and is based on the publication uploaded to arXiv.org [21]. The paper contains an introduction section framing the problem, followed by a methods section explaining the technical details of the algorithms and techniques used in the work. The third chapter concludes the thesis and proposes ideas for further study.

## Chapter 2

# Hybrid Genetic Optimisation for Quantum Feature Map Design

# Hybrid Genetic Optimisation for Quantum Feature Map Design

Rowan Pellow-Jarman<sup>1,2\*</sup>, Anban Pillay<sup>1,3</sup>, Ilya Sinayskiy<sup>1,2</sup>  
and Francesco Petruccione<sup>4,2</sup>

<sup>1\*</sup>University of KwaZulu-Natal.

<sup>2</sup>National Institute for Theoretical and Computational Science  
(NITheCS).

<sup>3</sup>Centre for Artificial Intelligence Research (CAIR).

<sup>4</sup>Stellenbosch University.

\*Corresponding author(s). E-mail(s): [rowanpellow@gmail.com](mailto:rowanpellow@gmail.com);

Contributing authors: [pillayw4@ukzn.ac.za](mailto:pillayw4@ukzn.ac.za);

[sinayskiy@ukzn.ac.za](mailto:sinayskiy@ukzn.ac.za); [petruccione@sun.ac.za](mailto:petruccione@sun.ac.za);

## Abstract

Kernel methods are an important class of techniques in machine learning. To be effective, good feature maps are crucial for mapping non-linearly separable input data into a higher dimensional (feature) space, thus allowing the data to be linearly separable in feature space. Previous work has shown that quantum feature map design can be automated for a given dataset using NSGA-II, a genetic algorithm, while both minimizing circuit size and maximizing classification accuracy. However, the evaluation of the accuracy achieved by a candidate feature map is costly. In this work, we demonstrate the suitability of kernel-target alignment as a substitute for accuracy in genetic algorithm-based quantum feature map design. Kernel-target alignment is faster to evaluate than accuracy and does not require some data points to be reserved for its evaluation. To further accelerate the evaluation of genetic fitness, we provide a method to approximate kernel-target alignment. To improve kernel-target alignment and root mean squared error, the final trainable parameters of the generated circuits are further trained using COBYLA to determine whether a hybrid approach applying conventional circuit parameter training can easily complement the genetic structure optimization approach. A total of eight new approaches are compared to the original across nine varied binary classification problems from the UCI machine learning

repository, showing that kernel-target alignment and its approximation produce feature map circuits enabling comparable accuracy to the previous work but with larger margins on training data (in excess of 20% larger) that improve further with circuit parameter training.

**Keywords:** Quantum Computing, Machine Learning, SVM, QSVM, Genetic algorithm, NSGA-II, Kernel-target alignment

## 1 Introduction

Quantum computers leverage quantum properties such as entanglement, and promise the potential of a speed advantage over classical algorithms when applied to specialized problems. Some algorithms such as Shor’s algorithm for factorization (Shor, 1997) and Grover’s search algorithm (Grover, 1996) have been shown theoretically to outperform all known classical algorithms applied to the same tasks. Quantum algorithm design is made difficult by the unintuitive nature of quantum entanglement which must be used effectively to achieve an advantage over classical algorithms. Quantum machine learning seeks to apply quantum computation to machine learning tasks to achieve a quantum advantage over classical machine learning.

Quantum machine learning and classical machine learning show promise for automating many practical tasks that would otherwise require human intelligence, including disease diagnosis (Myszczyńska et al, 2020), natural language processing (Khurana et al, 2022), and image classification (Horak and Sablatnig, 2019). Machine learning algorithms are designed to learn functions from data (Goodfellow et al, 2016). These algorithms can be separated into three categories: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning algorithms make use of pre-labeled training data while unsupervised learning algorithms do not (Alloghani et al, 2020). In reinforcement learning, agents learn behaviour by interacting with an environment that provides rewards and punishments to guide them.

Kernel methods are an important class of techniques in both classical and quantum machine learning. The Support Vector Machine (SVM) (Boser et al, 1996) is an important classical supervised kernel method, due to its theoretical relations to other learning models (Jacot et al, 2018; Schuld, 2021) and results regarding its generalisation ability (Vapnik, 1998; Vapnik and Chervonenkis, 2015). It is built around an optimization algorithm for finding an optimal linear hyperplane that separates data points into two classes. The hyperplane is selected to maximize the minimum margin of any point in the training dataset, where the margin of a point is defined as the distance of the point from the separating hyperplane. Larger margin sizes have been theoretically linked to improved generalisation performance (Vapnik, 1998; Vapnik and Chervonenkis, 2015). Non-linear decision boundaries can be achieved by mapping non-linear data to a higher dimensional feature space. The mapping function used is called

a feature map, and the range of a feature map is called a feature space. By use of a technique known as the “*kernel trick*”, the decision boundary optimization problem can be reformulated in terms of a kernel function that computes the similarity of a pair of data points in the feature space (Boser et al, 1996). This obviates the need to explicitly compute feature map outputs for data points, so long as the corresponding kernel function can be computed.

The feature map must be carefully selected for effective separation of the data. For any kernel function and labelled training set combination, a quantity known as the kernel-target alignment of the kernel can be calculated. This indicates the degree of agreement between the kernel function and a hypothetical oracle kernel induced by the training labels that is well suited to the training data (Cristianini et al, 2001). A high kernel-target alignment has been shown in other works to correlate with improved classification performance (Cristianini et al, 2001) and it has been proposed for use as a metric for selecting suitable kernels for a dataset in classification problems (Cristianini et al, 2001; Hubregtsen et al, 2022).

The QSVM algorithm enhances the SVM algorithm by implementing the feature map function as a quantum circuit (see section 2.1.2). While quantum feature map circuits are parameterized in the feature values of a single data point, they can also contain additional trainable parameters which can be optimized to improve the suitability of the kernel circuit for a specific dataset (Hubregtsen et al, 2022). A quantum circuit containing trainable parameters is called an *ansatz*. Prior work has used classical optimizers on trainable parameterized quantum kernels to maximize their kernel-target alignment, which resulted in positive effects on the classification accuracy of the resulting SVM models (Hubregtsen et al, 2022).

Quantum feature map circuits of fixed structure that make use of trainable parameter values are reported in Hubregtsen et al (2022). The trainable parameter values are optimized using Stochastic Gradient Ascent to maximize the kernel-target alignment of the corresponding kernel functions. This was performed to test whether kernel-target alignment optimization could improve the performance of a fixed-structure quantum feature map on a given dataset. Increased kernel-target alignment had previously been shown in Cristianini et al (2001) to correlate with improved classification ability. The technique described can be applied either to tailor an existing feature map to a dataset or to fully generate a feature map for a dataset from a feature map *ansatz* that has a predetermined circuit structure and parameter placement. The work made use of both classical noise-free simulations of quantum computers and real NISQ computers to run quantum circuits, reporting improvements in classification accuracy after kernel-target alignment maximization (Hubregtsen et al, 2022).

A second model training metric applicable to quantum kernel classifiers is a classifier’s root mean squared error (RMSE). This is often optimized to train parameterized models for classification tasks. It can be better suited to training than direct evaluation of accuracy since it accounts for the magnitudes

of misclassification errors rather than simply the number of errors that occur. Accuracy can be too insensitive to circuit parameter changes to show when a circuit has slightly improved if a data set is not sufficiently large.

Another approach to optimizing the choice of kernel circuit for a dataset is to optimize the selection of circuit gates used in the feature map in addition to the values of trainable circuit parameters. This approach has been applied to optimizing circuits applied to other problems ([Ostaszewski et al, 2021](#)). Genetically inspired algorithms have also been applied to circuit structure optimization since they are capable of combinatorial optimization ([Lukac and Perkowski, 2002](#); [Bautu and Bautu, 2007](#); [Rasconi and Oddi, 2019](#)).

[Altares-López et al \(2021\)](#) detailed the implementation of a genetic algorithm for automated feature map circuit design for use with QSVM classifiers that both maximizes classification accuracy and minimizes circuit size. The optimization was performed using a variation of genetic algorithm named NSGA-II ([Deb et al, 2002](#)) which customizes the usual genetic selection and fitness evaluation operations of the genetic algorithm (as outlined in section 2.3) to evaluate multiple fitness functions and preferentially select non-dominated solutions for crossover.

In a minimization problem with two fitness functions, a solution  $s$  with fitness values  $(a, b)$  is considered non-dominated with respect to another set of  $n$  solutions with fitness values  $\{(f_i, g_i) | i \in \{1, 2, \dots, n\}\}$  if and only if

$$\forall i \in \{1, 2, \dots, n\}, (f_i < a) \implies (g_i > b)$$

, i.e. if and only if there are no solutions in the set with all fitness values superior to the corresponding fitness values of  $s$ . NSGA-II also makes use of elitism to guarantee the preservation of the solutions that best optimize at least 1 of the individual fitness functions.

In order to apply a genetic algorithm to quantum feature map design, the work also defined a binary string representation for encoding feature map circuits. The encoding strategy can be found in [Altares-López et al \(2021\)](#) and is summarised here.

Each circuit gate is encoded in a sequence of 5 bits, the first three of which encode the type of gate applied and the last two of which encode a proportionality parameter for use in the case that a parameterised rotation gate was selected by the first three bits. The mapping of bits to gates and proportionality parameters is shown in Table 1.

Hyperparameters  $M$  and  $N$  which designate the maximum number of qubits and maximum number of gate layers respectively must be chosen before applying the genetic algorithm and are fixed for the duration of the genetic optimization. A single solution is represented as a bit string of length  $5MN$ , which holds the concatenation of the encodings of the individual gates in the feature map.

The gates in the encoding are applied successively with the target qubit and feature value to potentially encode being selected in a round-robin fashion.

Bits	Gate	Bits	Parameter
000	H	00	$\pi$
001	CNOT	01	$\pi/2$
010	I	10	$\pi/4$
011	Rx	11	$\pi/8$
100	Rz		
101	I		
110	I		
111	Ry		

**Table 1:** Mapping used on each consecutive 5-bit sequence of bits encoding a feature map gate to determine the gate type and a proportionality parameter value used in the case of parameterised gates. The available gates for the encoding to select from are Hadamard (H), CNOT, identity (I), and parameterised rotations around the X, Y, and Z axis of the Bloch sphere which are used to encode data point feature values into the circuit. We define the  $R_x(\theta)$  gate as

$$\cos(\theta/2)I - i\sin(\theta/2)X,$$

the  $R_y(\theta)$  gate as

$$\cos(\theta/2)I - i\sin(\theta/2)Y,$$

and the  $R_z(\theta)$  gate as

$$\cos(\theta/2)I - i\sin(\theta/2)Z.$$

If a parameterised rotation gate  $R_a$  around axis  $a$  is selected, the parameter selected by the last two of the five gate bits will be used to selected a proportionality parameter  $p$ . When the gate  $R_a$  is applied to encode a feature value  $x_i$ , the gate applied will be  $R_a(px_i)$ . In the case of a CNOT gate being selected and this gate being applied to qubit  $i$ , qubit  $i$  will be used as the control qubit and qubit  $(i + 1) \bmod M$  will be used as the target qubit.

Stated explicitly, for each consecutive group of 5 gate bits, the  $i$ th gate description will be applied to qubit  $i \bmod M$  and will encode feature  $i \bmod N$  if it performs a parameterised rotation. This encoding strategy was selected for simplicity, although other strategies could also feasibly be attempted.

Two fitness functions were optimized in the work: accuracy on a test set was maximized and a weighted size metric was simultaneously minimized. The unweighted size metric SM was calculated in terms of the number of qubits  $M$ , the number of single qubit gates  $N_{\text{local}}$ , and the number of entangling gates  $N_{\text{CNOT}}$ , by the expression

$$\text{SM} = \frac{N_{\text{local}} + 2N_{\text{CNOT}}}{M}.$$

The weighted size metric WS supplied to the genetic algorithm was given by the expression

$$\text{WS} = \text{SM} + \text{SM} \cdot \text{accuracy}^2.$$



The work was able to demonstrate the effectiveness of using NSGA-II with the devised feature map binary string encoding strategy, accuracy maximization, and weighted size minimization to automatically produce quantum feature map circuits for QSVM classification using only a dataset and a few hyperparameters as input. The generated circuits were also experimentally shown to generalise to unseen data. In addition to using few qubits and quantum gates, the circuits produced by the approach were observed to make little to no use of entanglement, meaning they could be efficiently simulated classically and the approach could constitute a quantum-inspired classical machine learning algorithm.

Some attempts have been made to enhance the genetic optimization and compare the approach to others. The work done in [Chen and Chern \(2022\)](#) is based on the algorithm put forward in [Altares-López et al \(2021\)](#). They used a modified encoding scheme which encoded the proportionality parameter values using three bits instead of only two, doubling the number of encodable parameter values. A restricted choice of parameter values was one of the potential limitations of the algorithm designed in [Altares-López et al \(2021\)](#). The algorithm was also further modified to optimize gate cost and classification accuracy in a single objective expression, using a single objective genetic algorithm instead of a multi-objective genetic algorithm such as NSGA-II. Notably, this introduced a new hyperparameter used to weight the focus of the optimization between circuit size and accuracy, which is not done with NSGA-II.

The feature map generated by the genetic algorithm was compared with two other choices of ansatz: a hardware efficient ansatz proposed in [Kandala et al \(2017\)](#), and a unitary decomposition ansatz proposed in [Shende et al \(2006\)](#). Each ansatz was trained with COBYLA ([Powell, 1994](#)), directly evaluating classification accuracy as a cost function. It was found that the feature map circuits generated by the genetic algorithm variation used in the work performed similarly to the hardware efficient ansatz, depending on the circuit depth hyperparameter selected when generating the hardware efficient ansatz. However, both were beaten by the unitary decomposition ansatz in accuracy, which achieved the highest accuracy at the cost of having a large, fixed size.

Our work investigates kernel-target alignment as a metric for automating quantum feature map design for the Quantum-Enhanced Support Vector Machine (QSVM) algorithm ([Schuld and Killoran, 2019](#); [Rebentrost et al, 2013](#)). Our work has two goals: investigate the suitability of using alternative cost functions to accuracy in the genetic optimization process described in [Altares-López et al \(2021\)](#) and secondly, investigate whether the problem of limited circuit parameter choices in the genetic algorithm can be addressed by a hybrid process of genetic and circuit parameter training.

We address the first goal by evaluating two alternative cost functions to the accuracy metric in [Altares-López et al \(2021\)](#) : firstly, kernel-target alignment for the genetic optimization step and secondly, a heuristic estimation of the kernel-target alignment performing a fraction of the kernel evaluations.

For the second goal, a hybrid method involving further optimising the final choice of trainable circuit parameter values after the genetic algorithm terminates for each of the above approaches is evaluated. This final optimization uses COBYLA (Powell, 1994) to maximize either kernel-target alignment or RMSE. The new approaches are compared to the original across several binary classification problems of varying difficulties. We show that even though the kernel-target alignment metric is less computationally expensive to compute in terms of quantum kernel evaluations and avoids the training of an SVM classifier, the performance of the constructed classifiers is comparable to the original approach and often achieves a better margin distribution on training data. It has been demonstrated Theoretically that increased margin sizes indicates better generalisation ability (Vapnik, 1998; Vapnik and Chervonenkis, 2015). The kernel-target alignment approximation heuristic is shown to perform marginally worse than exact kernel-target alignment optimization but at a fraction of the computational cost. The hybrid approaches are shown to improve margin sizes over the original. The original approach is also shown to sometimes overfit to the test data used to evaluate its accuracy metric, particularly on difficult problems.

In the following section, we give a more detailed explanation of the background topics involved in understanding this work and related works, and explain our experimental setup. This is followed by a section covering our findings and interpretations. In the final section we give an overview of the contributions made and suggest ideas for further research.

## 2 Methods

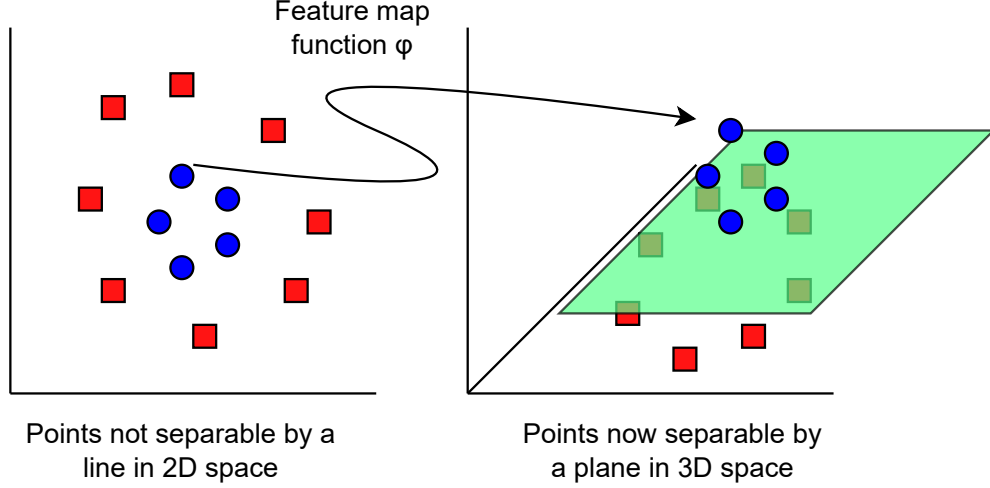
### 2.1 Binary classifiers using quantum kernels

#### 2.1.1 Support Vector Machine (SVM)

The SVM algorithm is a classical supervised machine learning algorithm for binary classification problems that works by finding an optimal separating hyperplane between two classes of data points. The SVM algorithm is applicable when the data points can be represented by real-valued feature vectors. For simplifying definitions, the class labels are usually replaced with positive and negative one.

The *margin* of a single data point is defined as the distance from the data point to the SVM’s chosen hyperplane. The margin of an SVM classifier refers to the minimum of the margins of the data points. The data points with minimum margin are known as the *support vectors*. The hyperplane chosen by the SVM is optimal in the respect that it maximises the minimum of the margins of the training set data points by solving a quadratic programming optimization problem.

The SVM algorithm is also capable of classifying datasets with classes that are not linearly separable. This can be achieved by first mapping the data points to a higher dimensional space in such a way that they become linearly



**Fig. 1:** An example illustrating how a feature map function could be used to make non-linearly separable points linearly separable in a higher dimensional space. In this case, the feature map could be implemented as a function that adds a third dimension to the points with decreasing value as distance from the central region of the points increases.

separable in the higher dimensional space (see Figure 1 for an illustration). A function used to perform this mapping is called a *feature map* and the range of the function is called the *feature space*. The choice of feature map must be suited to the dataset in order to classify it well, since it determines whether the data will become linearly separable after transformation.

A feature map  $\phi(x)$  that maps a point into feature space has a corresponding *kernel function*

$$\kappa(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$$

which computes the inner product of a pair of data points in the feature space. The margin optimization problem can be equivalently reformulated as a dual problem in terms of the kernel function (Boser et al, 1996), which can sometimes avoid the explicit computation of the feature map. This advantage is often referred to as the “*kernel trick*”. Seeing that in many cases where a feature map can not be efficiently computed but the corresponding kernel function can be, the dual formulation of the problem increases the number of potential feature maps that can be applied to a dataset.

In the dual form of the SVM, the classifier output for a given class is determined using a coefficient sequence  $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$  and offset sequence  $b = \{b_1, b_2, \dots, b_n\}$  chosen by the SVM algorithm during the hyperplane optimization. The decision function df outputs an indication of the distance of its input point from the hyperplane after mapping into feature space. It is defined in terms of the kernel function  $\kappa$ , the training samples  $\{x_1, x_2, \dots, x_n\}$ , the  $\alpha$

coefficients, and the  $b$  offsets as follows

$$\text{df}(x) = \sum_{i=1}^n (\alpha_i \kappa(x, x_i) + b_i).$$

The sign of  $\text{df}(x)$  is used to determine the predicted class of the argument point  $x$ :

$$\text{Class}(x) = \text{sign}(\text{df}(x)).$$

### 2.1.2 Quantum-enhanced Support Vector Machine

The Quantum-enhanced Support Vector Machine (QSVM) algorithm extends the SVM algorithm by performing the kernel computation on a quantum computer (Schuld and Killoran, 2019; Rebentrost et al, 2013). A quantum circuit parameterised in the values of a single data point is used as a feature map to map the data points to a high dimensional quantum state in a quantum Hilbert space.

For a quantum feature map encoding data points into  $q$  qubits, the dimensionality of the feature Hilbert space is  $2^q$ . Although a quantum computer can efficiently compute the quantum state feature space representation of data point, in general the quantum state cannot be efficiently represented classically due to the exponentially increasing dimensionality of the feature space. To work around this classical limitation, the kernel function is computed directly on the quantum computer and the kernel-based formulation of the SVM is used. The kernel computation for a pair of data points can be efficiently performed by measuring the overlap of their corresponding states in the quantum feature space.

To train a QSVM model, the *Gram matrix*  $K_{n \times n}$  of the training points must be computed. For  $n$  training points  $\{x_1, x_2, \dots, x_n\}$  and a quantum kernel function  $\kappa$ , the Gram matrix is defined by  $K_{ij} = \kappa(x_i, x_j)$  where  $i, j \in \{1, 2, \dots, n\}$ . In the case of a noise free quantum computer or simulator being used to execute  $\kappa$ , the symmetric property  $\kappa(x_i, x_j) = \kappa(x_j, x_i)$  and the property that  $\kappa(x_i, x_i) = 1$  can be used to reduce the number of required evaluations.

Assuming the stated properties, the  $n$  main diagonal entries of  $K$  ( $K_{11}$ ,  $K_{22}$ , ...,  $K_{nn}$ ) do not require kernel evaluations, since  $K_{ii} = \kappa(x_i, x_i) = 1$ . For the remaining  $n^2 - n$  entries  $K_{ij}$  which are not on the main diagonal, there is a symmetric entry  $K_{ji}$  with the same value, since

$$K_{ij} = \kappa(x_i, x_j) = \kappa(x_j, x_i) = K_{ji}$$

. This means that only half of the entries need to be explicitly computed by kernel evaluations. In effect, only  $\frac{n^2 - n}{2}$  kernel evaluations must be performed to construct  $K$ . In the case of a NISQ computer, this technique could potentially

be applied if measures were taken to mitigate noise and correct the kernel matrix such as in [Hubregtsen et al \(2022\)](#).

Since the only difference between the SVM and QSVM algorithms is how the kernel computation is performed, the potential advantage of the QSVM algorithm lies in enabling the computation of kernel functions that are hard to estimate classically ([Liu et al, 2021](#)). While examples of such kernels have been discovered ([Liu et al, 2021](#)) for artificial datasets, it is an open question how best to design quantum feature maps to achieve a useful kernel with a quantum speed advantage.

## 2.2 Kernel quality metrics

### 2.2.1 Kernel-target alignment

Kernel-target alignment is a heuristic for kernel quality that measures the degree of similarity between two kernels, or the degree of agreement between a kernel and a dataset ([Cristianini et al, 2001](#)). It is calculated using a matrix inner product between the Gram matrix constructed from the training samples and an oracle matrix constructed from the training labels, where the oracle matrix acts as a stand-in for the Gram matrix of a hypothetical kernel which is very well-suited to the data.

For a set of training points  $\{x_1, x_2, \dots, x_n\}$  with corresponding labels  $\{y_1, y_2, \dots, y_n\}$  with  $\forall i, y_i \in \{-1, 1\}$ , a kernel function  $\kappa(x_i, x_j)$ , and with the *Frobenius* inner product for matrices defined as

$$\langle A, B \rangle_F = \sum_{i,j} A_{ij} B_{ij},$$

the kernel-target alignment can be computed as follows ([Cristianini et al, 2001](#))

1. Compute the Gram matrix  $K_{n \times n}$  using the kernel function and training points by the rule

$$K_{ij} = \kappa(x_i, x_j).$$

2. Compute the oracle matrix  $O_{n \times n}$  using the training labels by the rule

$$O_{ij} = y_i y_j.$$

3. Compute the kernel-target alignment KTA using the Frobenius inner product as

$$\text{KTA} = \frac{\langle K, O \rangle_F}{\sqrt{\langle K, K \rangle_F \langle O, O \rangle_F}}.$$

A high kernel-target alignment has been shown in other works to correlate with improved classification performance ([Cristianini et al, 2001](#)) and it has

been proposed for use as a metric for selecting applicable kernels for a dataset in classification problems (Cristianini et al, 2001; Hubregtsen et al, 2022).

### 2.2.2 Root Mean Squared Error

Root Mean Squared Error (RMSE) is a common metric for measuring the error of a model. It is calculated as the square root of the mean of the squared errors of a classifier's predictions on each training set data point. In this work, the RMSE of a classifier is calculated using the errors of the decision function on training data with an adjustment to the error calculation. The adjustment is to account for there not being a definitively correct output of the SVM decision function for a given sample and label pair. The error is measured relative to a positive target decision function output  $m$ , which we set to one in this work.

We calculate the error for a decision function output  $a$  and training label  $b$  using the following rule:

$$\text{error}(a, b) = \begin{cases} (m - a) & \text{if } b = 1 \text{ and } a < m \\ (a - m) & \text{if } b = -1 \text{ and } a > -m \\ 0 & \text{otherwise} \end{cases}$$

This choice of error function means that only points not classified to the desired degree of confidence  $m$  contribute to the error calculation, and the errors of the considered points increase with distance from the target output.

For a set of training points  $\{x_1, x_2, \dots, x_n\}$  with corresponding labels  $\{y_1, y_2, \dots, y_n\}$  with  $\forall i, y_i \in \{-1, 1\}$ , the RMSE is calculated in terms of this adjusted error function and the decision function  $\text{df}$  by the following rule

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n \text{error}(\text{df}(x_i), y_i)^2}{n}}$$

## 2.3 Overview of genetic algorithms

Genetic algorithms are flexible metaheuristic algorithms inspired by the real-world evolutionary principles of natural selection, genetic inheritance, and random mutation. They are a popular choice of algorithm for optimizing complex objective functions in cases where algorithms known to produce a global optimum are unknown or infeasible.

Implementing a genetic algorithm first requires designing a solution representation on which genetic operations can be performed. The solution representation often (but not always) takes the form of a binary string, which is mapped by a problem-specific decoding function to a usable solution. A genetic algorithm manages a set of these solution representations, which is called a population. The initial population can be a set of randomly generated solutions or chosen according to a problem-specific heuristic.

The optimization process is iterative and typically repeats until a suitable solution is found, a desired number of iterations has passed, or the rate of

improvement of the solutions becomes low. Each iteration, genetically inspired operations are applied to the population to create a new replacement population. The population creation process involves fitness evaluation, selection, crossover, and mutation operations.

Fitness evaluation is performed by decoding a solution representation into a solution, then evaluating a numeric score of its suitability for the problem. This is performed for the entire population, after which point a selection operation is applied to select some solutions for crossover and mutation. It is important that better solutions are more likely to be selected for crossover, since this is the main mechanism driving improvement between generations. The solutions selected for crossover are referred to as “*parents*”.

The crossover operation is performed between two parent solutions to produce one or more new solutions, called child solutions. This is usually performed by taking a simple combination of the solution representations of the parents. In the case of a binary string solution representation, a simple crossover can be performed by combining two non-overlapping subsequences of the parents, taking a random number of bits from the first and the remainder from the corresponding positions in the second. A mutation operation can be applied to a child solution by randomly editing its representation by a small amount. This simulates the random mutation which occurs in real life and affects the diversity of available genetic material in the population.

A strategy often employed when determining which individuals will make up the next generation is to preserve the best performing of the solutions among the current generation and the newly created children. This is known as *elitism* and ensures that solutions can survive through multiple generations and potentially indefinitely, so long as they continue to outperform newer ones. This helps prevent regression of the achieved fitness due to chance as generations pass.

The general idea for a genetic algorithm is flexible enough that many variations and extensions of the discussed components have also been studied ([Chahar et al, 2021](#)).

## 2.4 Experiments

The algorithm for automated feature map design described in [Altares-López et al \(2021\)](#) was reimplemented using the Julia programming language ([Bezanson et al, 2017](#)), the Yao quantum simulator framework ([Luo et al, 2020](#)), and the pymoo ([Blank and Deb, 2020](#)) implementation of NSGA-II. All experiments were run with the maximum qubit count and feature map depth hyperparameters set to 6. The genetic algorithm population size was set to 100, with 15 new individuals being produced every generation. 30% of the new individuals were produced by crossover; the rest were chosen randomly from the parents. In each generation, 70% of the population underwent mutation. When mutation occurred, 20% of the bits in the mutated solution were flipped. All experiments were run using a noise free quantum simulator provided by Yao.



Dataset	Class -1	Class 1	Features (PCA)	Train	Test	Validation
Moons	Top left	Bottom right	2 (N/A)	210	90	500
Cancer	Benign	Malignant	30 (10)	210	90	124
Iris	Versicolor	Virginica	4 (N/A)	42	18	40
Digits	Eight	Nine	64 (10)	140	60	148
Circles	Outer	Inner	2 (N/A)	210	90	500
Random	Red	Blue	2 (N/A)	210	90	N/A
Voice	Acceptable	Unacceptable	309 (10)	28	12	44
SUSY	Background	Signal	18 (10)	210	90	500
SUSY reduced	Background	Signal	8 (N/A)	210	90	500

**Table 2:** Table showing the characteristics of the datasets and sample splits used. Not all points in the base datasets were used to ensure the sample split remained balanced in each of the sample sets. Other considerations in determining the data splits were experiment runtime while maintaining a sufficiently large ratio of test points to train points and a sufficiently large number of validation points. All datasets with more than 10 feature values were reduced to 10 features using Principle Component Analysis (PCA). The moons, circles, and random datasets are artificial, with the moons and circles datasets being generated with Scikit-learn (Pedregosa et al, 2011). The rest of the datasets are sourced from the UCI Machine Learning Repository (Dua and Graff, 2017) either directly or indirectly through Scikit-learn (Pedregosa et al, 2011).

Three configurations of the original algorithm were run on nine different datasets of varying difficulty (see Table 2) to compare their effectiveness. The first configuration maximized accuracy on a test set and minimized weighted size, as in the original work (Altares-López et al, 2021). The second configuration maximized kernel-target alignment on the training data, ignoring the test data, and minimized the unweighted size metric also defined in Altares-López et al (2021). The third configuration maximized an approximation of kernel-target alignment on the training data, and minimized the same unweighted size metric.

Before performing the genetic optimization the datasets are split into three disjoint subsets, namely training data, testing data, and validation data. The training data is used to evaluate kernel-target alignment and its approximation, as well as train the QSVM model for a given feature map circuit. The testing data is only used to evaluate the accuracy metric in the first approach. The validation data is used to determine the generalisation ability of the generated models, and must be separate from the testing data since the first approach can indirectly access the testing data through the accuracy metric and potentially overfit to it.

In order to calculate the kernel-target alignment approximation for  $n$  training points, the  $n$  points are divided into  $a$  disjoint complementary subsets of size roughly  $n/a$ . The number of subsets  $a$  can be adjusted based on the number of training points to balance speed and precision. The kernel target alignment is calculated on each of the subsets in turn, then averaged.



Assuming the properties of the kernel function are not used to accelerate the Gram matrix computation,  $n^2$  kernel evaluations are required to compute the exact kernel-target alignment, and only

$$a(n/a)^2 = (n^2)/a$$

evaluations are required to compute the approximation, giving a factor  $a$  speedup. If the kernel properties are used, then

$$(n^2 - n)/2 = n^2/2 - n/2$$

kernel evaluations are required to compute the exact kernel-target alignment. Therefore, the number of kernel evaluations required when evaluating the kernel-target alignment approximation can be derived as

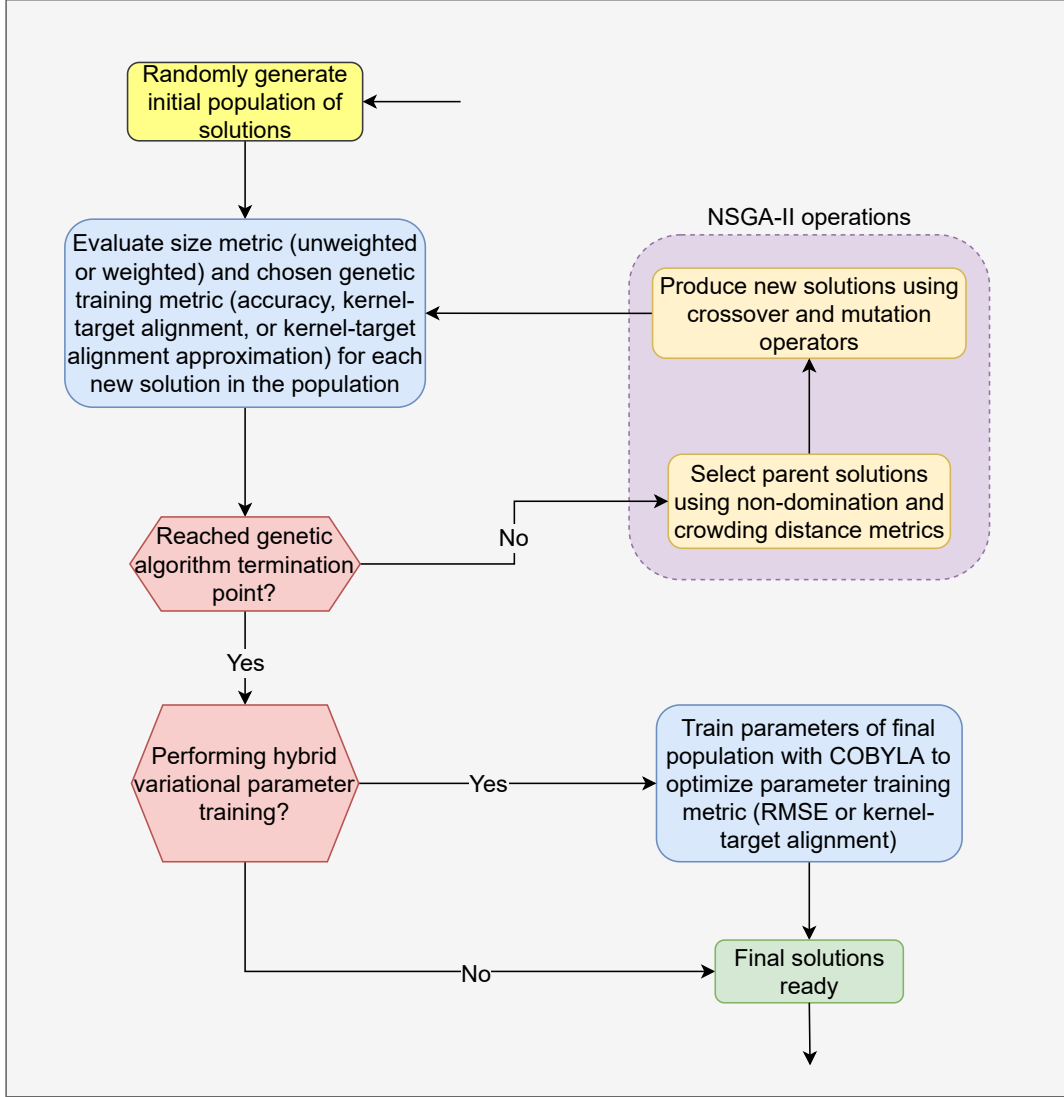
$$a\left(\frac{\left(\frac{n}{a}\right)^2 - \frac{n}{a}}{2}\right) = \frac{n^2}{2a} - \frac{n}{2},$$

meaning the factor speedup by evaluating the approximation is larger than  $a$ , but should approach  $a$  as  $n$  increases. In this work, we use a value of  $a = 5$  in all experiments involving the kernel-target alignment approximation.

After the genetic optimization in each configuration completes, we attempt further improvement by further optimizing just the proportionality parameters encoded in the last two bits of the gate representation using an implementation of COBYLA (Powell, 1994) provided by the NLopt optimization library (Johnson, 2011). This allows the parameter values to not be restricted to one of only four possibilities. This optimization aims to either minimize RMSE or maximize kernel-target alignment using the training set to evaluate the metrics. The COBYLA optimizer is allowed one hundred evaluations of the cost function to perform the optimization. A flow diagram outlining the algorithmic process can be seen in Figure 2.

The COBYLA cost functions for RMSE and kernel-target alignment both require computing a Gram matrix each evaluation, meaning the number of kernel evaluations performed is  $\frac{100(n^2 - n)}{2}$ . This can be contrasted with the genetic optimization of accuracy or kernel-target alignment, where at least as many kernel evaluations are performed to evaluate the fitness of just the first generation of 100 solutions in the genetic algorithm. In the subsequent 1199 generations  $15 \times 1199 = 17985$  more Gram matrix evaluations are performed for a total of 18085, meaning the final parameter training for the entire output population requires roughly 55% percent of the number of kernel evaluations performed in the genetic optimization in the cases of genetically optimizing accuracy or the exact kernel-target alignment.

We name the three base approaches 1, 2, and 3, respectively. Each approach has two additional sub-approaches defined for further training of RMSE or



**Fig. 2:** A flow diagram outlining the algorithm followed to genetically train quantum feature map circuits. The diagram also shows how a hybrid method involving circuit parameter training can be performed after genetic optimization.

kernel-target alignment, for a total of nine approaches. The RMSE and kernel-target alignment variations are named with a .1 and .2 suffix respectively. We graph the classification accuracies, average margins, ROC curves, feature map circuits, and confusion matrices of the best models produced by each approach, where the best model of a population is taken to be the one achieving the highest validation set accuracy. For two dimensional datasets, decision boundaries are also graphed.

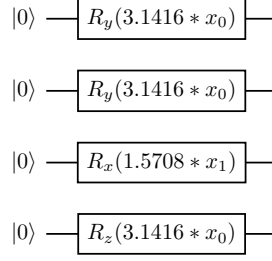
The code implementing the experiments and result graphing can be found on GitHub ([Pellow-Jarman, 2022](#)).

### 3 Results

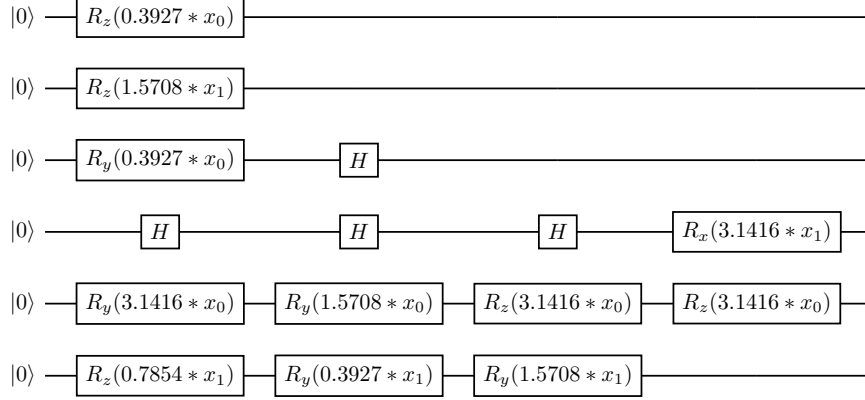
As in the original work by [Altares-López et al \(2021\)](#), the feature map circuits produced by each of the approaches tend to make little to no use of entangling gates (see Figure 3). However, the circuits produced by optimizing the kernel-target alignment based metrics tend to produce significantly larger circuits overall (see Figure 3). This could be explained by the fact that the weighted size metric in the genetic optimization was replaced with an unweighted size metric in those approaches due to the weighted size metric depending on the test set accuracy, which was not evaluated. The optimization of the circuit size did not converge in the allocated 1200 generations in approaches 2 and 3, which can be inferred from the presence of redundant gates. This could possibly be addressed by allowing more generations to pass or using a size metric weighted in kernel-target alignment instead of accuracy, similarly to the original approach. Another possible explanation for the larger circuit size is that a circuit achieving perfect accuracy may still be able to improve its kernel-target alignment; in the accuracy maximization case, the genetic algorithm is able to shift focus to minimizing circuit size after achieving 100% accuracy, but the same cannot be done as easily when maximizing kernel-target alignment since its limiting value of one is more difficult to achieve. Additionally, the high mutation rate of 70% could be reduced to attempt to reach convergence in the allocated 1200 generations.

Our experiments show that substituting kernel-target alignment or approximated kernel-target alignment for accuracy in the genetic optimization process produces feature map circuits with accuracy comparable to the original approach across all datasets (see Figures 4 and 6 for example). Further optimizing the final population’s trainable parameters using COBYLA was often able to improve the average margin sizes of the classifiers (see Figure 5) on training data and sometimes able to improve validation set classification accuracy (see Figures 4 and 9), showing that a hybrid approach performing further optimization of the final populations parameter values is worth attempting despite the computational cost if improving accuracy is important. Training the parameters of a single solution for 100 cost evaluations requires only half a percent of the kernel evaluations as the genetic optimization process, so a smaller subset of the final solutions could be trained at a much lower cost. Additionally, the untrained parameters encoded in the solution binary strings are not lost if further training is performed and can still be used if they happen to perform better than the trained ones.

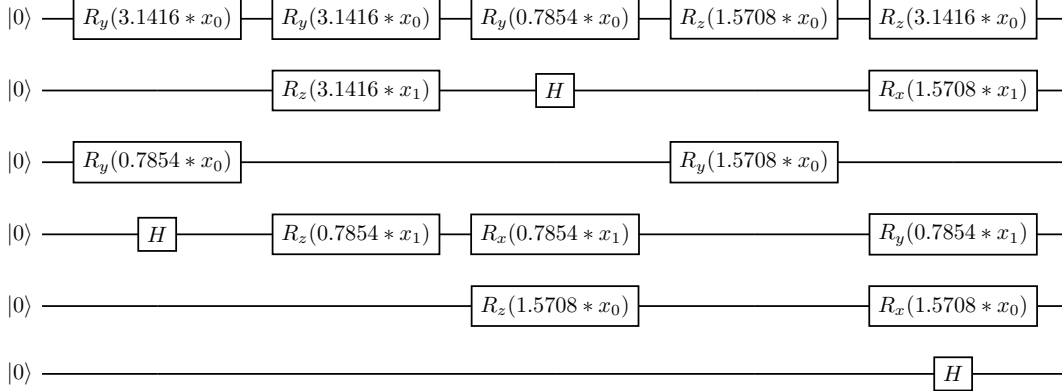
The results demonstrate that on difficult datasets such as the SUSY, SUSY reduced features, Voice, and Random datasets, the original approach’s models can overfit to the testing data used to evaluate the accuracy metric (see Figure 7). This is likely due to the fact that test set accuracy is directly optimized in the genetic algorithm without regard to training set accuracy. Since the kernel-target alignment approaches make use of only the training data during the genetic optimization they do not suffer from the same drawback, although they do not show improvement on validation data for the difficult problems.



(a) Approach 1 - Accuracy, Weighted size

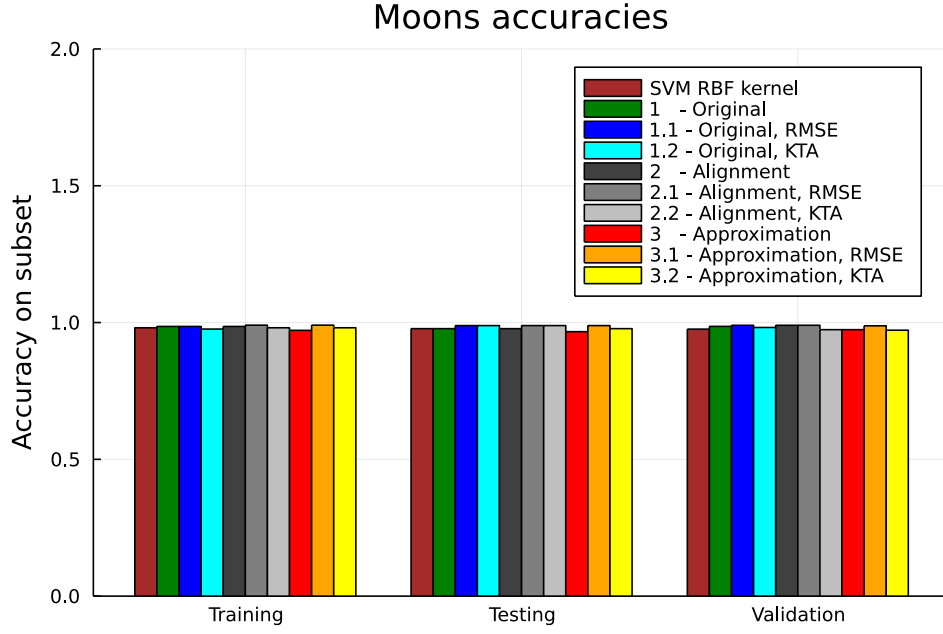


(b) Approach 2 - Kernel-target alignment, Unweighted size

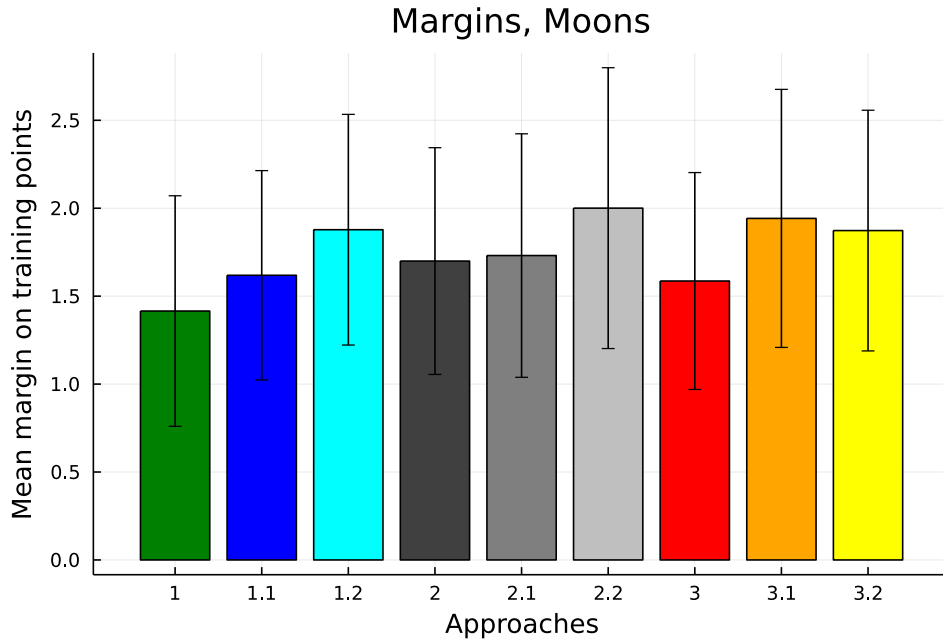


(c) Approach 3 - Kernel-target alignment approximation, Unweighted size

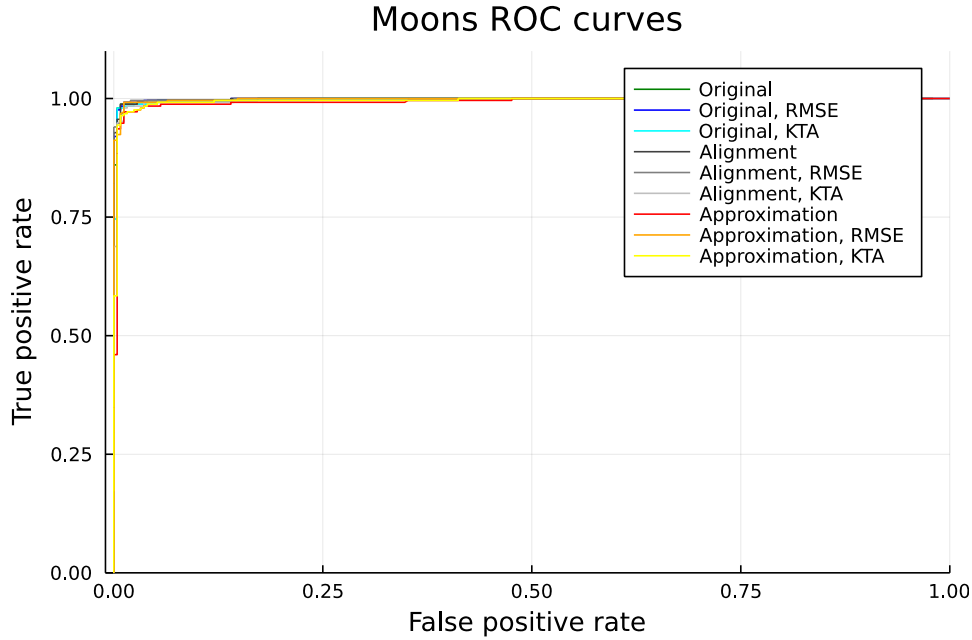
**Fig. 3:** The circuits with highest validation set accuracy produced by the three base genetic approaches when creating quantum feature maps for the Moons dataset. (a) shows the best produced circuit when training to maximize accuracy and minimize weighted size as in the original work, (b) shows the best circuit when training to maximize the exact kernel-target alignment and minimize unweighted size, and (c) shows the best circuit when training to maximize the approximation of the kernel-target alignment and minimize unweighted size. Circuits (b) and (c) are significantly larger. Unused gate layers and qubits are omitted from the diagrams.



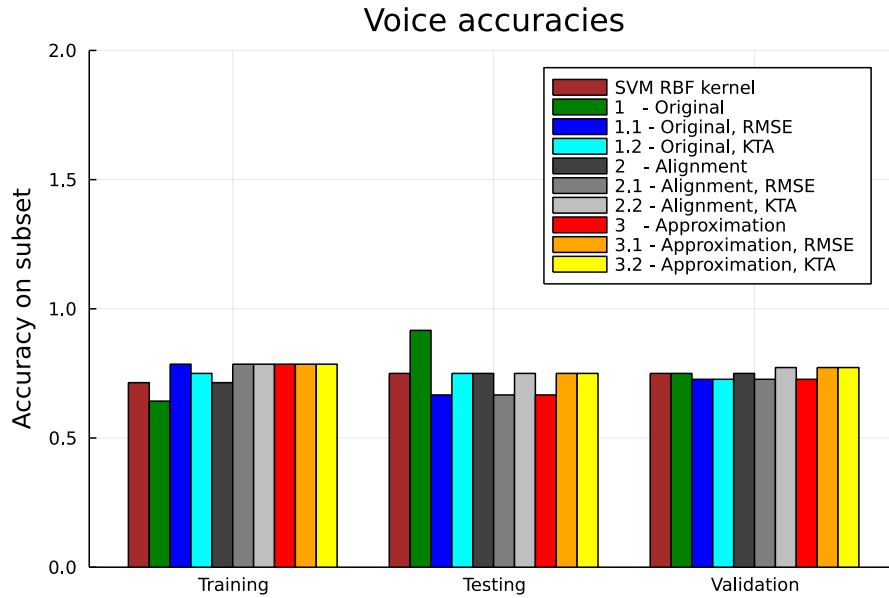
**Fig. 4:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Moons dataset, compared with a classical RBF kernel for reference. All approaches can be seen to achieve comparable accuracy across the different subsets.



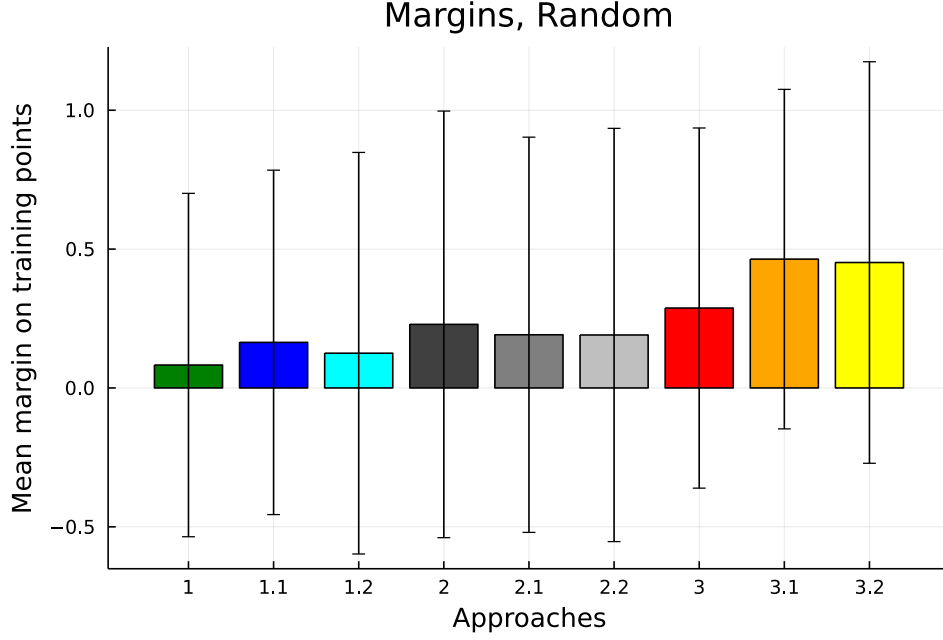
**Fig. 5:** A graph showing the mean margin of the Moons training set points for the best classifiers produced by each approach, with errors bars showing standard deviation. Circuit parameter training and genetic training of kernel-target alignment are both shown to increase the mean margin size. The approach numbering corresponds to the numbering used in Figure 4.



**Fig. 6:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Moons dataset. All of the produced models are shown to perform similarly on the dataset.



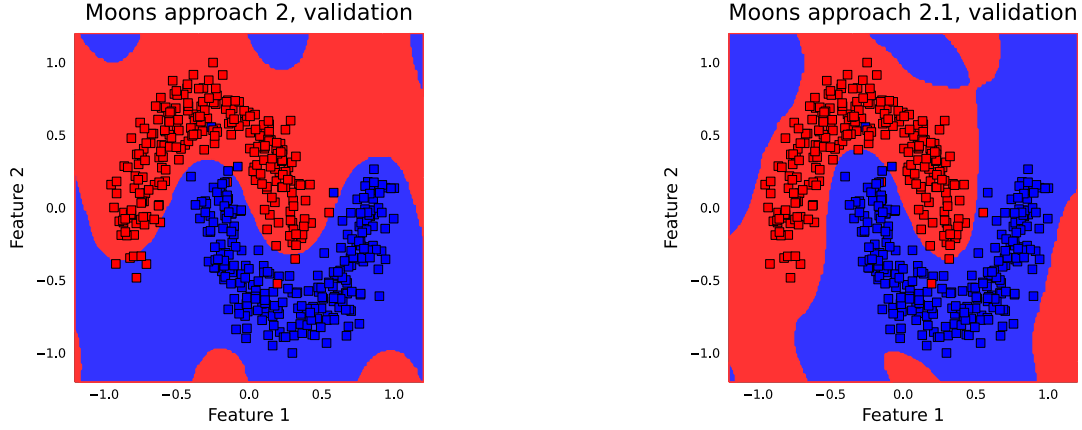
**Fig. 7:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Voice dataset, compared with a classical RBF kernel for reference. Genetic accuracy maximization is shown to overfit to the testing data used to evaluate the accuracy metric, justifying the necessity of a separate validation set.



**Fig. 8:** A graph showing the mean margin of the Random training set points for the best classifiers produced by each approach, with errors bars showing standard deviation. The approach numbering corresponds to that in Figure 4.

This problem could possibly be avoided by shuffling the training and testing data each generation, although this would make the accuracy metric depend on the generation at which the accuracy was evaluated and could prevent caching of solution fitnesses in the genetic algorithm. A second possible solution is to average the accuracy over subsets of the data. Given a dataset of  $n$  points, this can be performed while requiring at most  $\frac{n^2-n}{2}$  kernel evaluations in the worst case, since the Gram matrix for the entire dataset can be computed once and used as a cache to look up the kernel output for any pair of points when creating models with arbitrary choices of training and testing subsets.

The margins of classifiers trained with the second and third approach tended to be larger than those trained with the first (see Figures 5 and 8, as well as Table 3). This could be due to the fact that the kernel-target alignment metric and its approximation are evaluated on the training subset as opposed to accuracy which is evaluated on the testing subset, leading to the former two approaches having higher confidence on the training subset. In either case, increased margin size is an indicator of improved generalisation ability according to theoretical works (Vapnik, 1998; Vapnik and Chervonenkis, 2015) showing that margin size bounds VC dimension, and VC dimension bounds expected generalisation error. Further parameter training on the final generation also tended to show some improvements in margin sizes, even on easier datasets such as the Moons and Circles datasets where there was not much effect on overall classification accuracy. This improvement in margin can be seen visually in the decision boundary graphs of the classifiers (see Figure 9).



**Fig. 9:** Decision boundaries of the best classifiers for approaches 2 and 2.1 on the Moons validation data. Further parameter training on training data to minimize RMSE after genetically designing the feature maps to maximize kernel-target alignment is shown to improve classification ability on validation data.

Approach	Average margin	Absolute Change	Percentage Change
1-Accuracy (Original work)	0.838	N/A	N/A
Accuracy, RMSE training	0.993	+0.154	+18.42%
Accuracy, KTA training	0.971	+0.133	+15.87%
2-Alignment	1.043	+0.205	+24.46%
Alignment, RMSE training	1.016	-0.028	-2.66%
Alignment, KTA training	1.090	+0.047	+4.49%
3-Approximation	1.065	0.226	+26.99%
Approximation, RMSE training	1.145	+0.080	+7.54%
Approximation, KTA training	1.124	+0.059	+5.59%

**Table 3:** Table showing the average margin size of the best classifier produced by each approach, averaged across the nine datasets with equal weighting given to each dataset. For this purpose, the best classifier is defined as the classifier achieving the highest validation set accuracy for the target dataset. The improvement columns for the base approaches (2 and 3) show how genetic optimization of kernel-target alignment and its approximation improve on the margins achieved in the original work. For the hybrid approaches (with additional RMSE and KTA parameter training), the columns show change relative to the base approaches.

## 4 Conclusion

In this paper, we compared our implementation of the approach defined in [Altares-López et al \(2021\)](#) with adjustments to the genetic algorithm cost functions. These adjustments were aimed at investigating the suitability of kernel-target alignment as an alternative metric to test set accuracy and at reducing the number of kernel evaluations required by the approach. The new



approaches were shown to still be effective at designing accurate classifiers with fewer kernel evaluations, although at the cost of increased circuit size. They were also shown to often produce classifiers with better margins on training data. We also put forward a hybrid approach extending the original work by applying COBYLA (Powell, 1994) to further optimize the trainable parameters of the produced quantum feature map circuits after the termination of the genetic algorithm to attempt further improvement, at a lower additional computational cost than the genetic algorithm’s base cost. This parameter training was also shown to be capable of improving margin sizes and sometimes accuracy without increasing the circuit gate cost.

There is still more work to be done in accelerating the genetic algorithm while keeping gate costs low. A potential avenue to achieving this goal is the use of a multi-phase genetic algorithm in which the cost function is initially easy to evaluate but increases in precision after a set number of generations passes. For example, the  $a$  parameter of the kernel-target alignment approximation could be made to decrease as generations pass for the approximation to become more accurate at the cost of more kernel evaluations, or the cost function could be switched from kernel-target alignment to classification accuracy to reduce gate cost once a predetermined kernel-target alignment has been achieved.

The original approach could also further be extended to have the gate encoding for parameterised gates select a classical data encoding function such as those used in Suzuki et al (2020) to introduce classical nonlinearity to the encoding, potentially allowing for even lower gate cost or higher accuracy circuits to be produced.

## Declarations

### Competing interests

Francesco Petruccione the Chair of Scientific Board and Co-Founder of QUNOVA computing. The authors declare no other competing interests.

### Authors’ contributions

RPJ wrote the main manuscript text and prepared figures. All authors reviewed the manuscript.

### Funding

We would like to thank the National Institute of Theoretical and Computational Sciences (NITheCS) and the Center for Artificial Intelligence Research (CAIR) for funding this research.

## Availability of data and materials

All code and data associated with the current submission is available at <https://github.com/RowPJ/hybrid-genetic-optimisation-for-quantum-feature-map-design>.

## References

- Alloghani M, Al-Jumeily Obe D, Mustafina J, et al (2020) A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science, pp 3–21. [https://doi.org/10.1007/978-3-030-22475-2\\_1](https://doi.org/10.1007/978-3-030-22475-2_1)
- Altare-López S, Ribeiro A, García-Ripoll JJ (2021) Automatic design of quantum feature maps. *Quantum Science and Technology* 6(4):045,015. <https://doi.org/10.1088/2058-9565/ac1ab1>, URL <https://doi.org/10.1088/2058-9565/ac1ab1>
- Bautu A, Bautu E (2007) Quantum circuit design by means of genetic programming
- Bezanson J, Edelman A, Karpinski S, et al (2017) Julia: A fresh approach to numerical computing. *SIAM Review* 59(1):65–98. <https://doi.org/10.1137/141000671>, URL <https://epubs.siam.org/doi/10.1137/141000671>
- Blank J, Deb K (2020) pymoo: Multi-objective optimization in python. *IEEE Access* 8:89,497–89,509
- Boser B, Guyon I, Vapnik V (1996) A training algorithm for optimal margin classifiers. In: *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp 144–152
- Chahar V, Katoch S, Chauhan S (2021) A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications* 80. <https://doi.org/10.1007/s11042-020-10139-6>
- Chen BS, Chern JL (2022) Generating quantum feature maps for svm classifier. <https://doi.org/10.48550/ARXIV.2207.11449>, URL <https://arxiv.org/abs/2207.11449>
- Cristianini N, Shawe-Taylor J, Elisseeff A, et al (2001) On kernel-target alignment. In: Dietterich T, Becker S, Ghahramani Z (eds) *Advances in Neural Information Processing Systems*, vol 14. MIT Press, URL <https://proceedings.neurips.cc/paper/2001/file/1f71e393b3809197ed66df836fe833e5-Paper.pdf>
- Deb K, Pratap A, Agarwal S, et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary*

- Computation 6(2):182–197. <https://doi.org/10.1109/4235.996017>
- Dua D, Graff C (2017) UCI machine learning repository. URL <http://archive.ics.uci.edu/ml>
- Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press
- Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. Association for Computing Machinery, New York, NY, USA, STOC '96, pp 212–219, <https://doi.org/10.1145/237814.237866>, URL <https://doi.org/10.1145/237814.237866>
- Horak K, Sablatnig R (2019) Deep learning concepts and datasets for image recognition: overview 2019. p 100, <https://doi.org/10.1117/12.2539806>
- Hubregtsen T, Wierichs D, Gil-Fuster E, et al (2022) Training quantum embedding kernels on near-term quantum computers. Phys Rev A 106:042,431. <https://doi.org/10.1103/PhysRevA.106.042431>, URL <https://link.aps.org/doi/10.1103/PhysRevA.106.042431>
- Jacot A, Gabriel F, Hongler C (2018) Neural tangent kernel: Convergence and generalization in neural networks. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, NIPS'18, pp 8580–8589
- Johnson SG (2011) The NLOpt nonlinear-optimization package. URL <http://ab-initio.mit.edu/nlopt>
- Kandala A, Mezzacapo A, Temme K, et al (2017) Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. Nature 549(7671):242–246. <https://doi.org/10.1038/nature23879>, URL <https://doi.org/10.1038%2Fnature23879>
- Khurana D, Koli A, Khatter K, et al (2022) Natural language processing: State of the art, current trends and challenges. Multimedia Tools and Applications <https://doi.org/10.1007/s11042-022-13428-4>
- Liu Y, Arunachalam S, Temme K (2021) A rigorous and robust quantum speed-up in supervised machine learning. Nature Physics 17(9):1013–1017
- Lukac M, Perkowski M (2002) Evolving quantum circuits using genetic algorithm. In: Proceedings 2002 NASA/DoD Conference on Evolvable Hardware, pp 177–185, <https://doi.org/10.1109/EH.2002.1029883>
- Luo XZ, Liu JG, Zhang P, et al (2020) Yao.jl: Extensible, Efficient Framework for Quantum Algorithm Design. Quantum 4:341. <https://doi.org/10.22331/>

- q-2020-10-11-341, URL <https://doi.org/10.22331/q-2020-10-11-341>
- Myszczyńska MA, Ojamies PN, Lacoste AMB, et al (2020) Applications of machine learning to diagnosis and treatment of neurodegenerative diseases. *Nature Reviews Neurology* 16(8):440–456. <https://doi.org/10.1038/s41582-020-0377-8>, URL <https://doi.org/10.1038/s41582-020-0377-8>
- Ostaszewski M, Grant E, Benedetti M (2021) Structure optimization for parameterized quantum circuits. *Quantum* 5:391. <https://doi.org/10.22331/q-2021-01-28-391>, URL <https://doi.org/10.22331/q-2021-01-28-391>
- Pedregosa F, Varoquaux G, Gramfort A, et al (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830
- Pellow-Jarman R (2022) Hybrid genetic optimisation for quantum feature map design. <https://github.com/RowPJ/hybrid-genetic-optimisation-for-quantum-feature-map-design>
- Powell MJD (1994) A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation, Springer Netherlands, Dordrecht, pp 51–67. [https://doi.org/10.1007/978-94-015-8330-5\\_4](https://doi.org/10.1007/978-94-015-8330-5_4), URL [https://doi.org/10.1007/978-94-015-8330-5\\_4](https://doi.org/10.1007/978-94-015-8330-5_4)
- Rasconi R, Oddi A (2019) An innovative genetic algorithm for the quantum circuit compilation problem. In: *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, AAAI’19/IAAI’19/EAAI’19, <https://doi.org/10.1609/aaai.v33i01.33017707>, URL <https://doi.org/10.1609/aaai.v33i01.33017707>
- Rebentrost P, Mohseni M, Lloyd S (2013) Quantum support vector machine for big data classification. *Physical Review Letters* 113. <https://doi.org/10.1103/PhysRevLett.113.130503>
- Schuld M (2021) Quantum machine learning models are kernel methods
- Schuld M, Killoran N (2019) Quantum machine learning in feature hilbert spaces. *Phys Rev Lett* 122:040,504. <https://doi.org/10.1103/PhysRevLett.122.040504>, URL <https://link.aps.org/doi/10.1103/PhysRevLett.122.040504>
- Shende V, Bullock S, Markov I (2006) Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25(6):1000–1010. <https://doi.org/10.1109/tcad.2005.855930>, URL <https://doi.org/10.1109%2Ftcad.2005.855930>

Shor PW (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* 26(5):1484–1509. <https://doi.org/10.1137/s0097539795293172>, URL <http://dx.doi.org/10.1137/S0097539795293172>

Suzuki Y, Yano H, Gao Q, et al (2020) Analysis and synthesis of feature map for kernel-based quantum classifier. *Quantum Machine Intelligence* 2(1). <https://doi.org/10.1007/s42484-020-00020-y>, URL <http://dx.doi.org/10.1007/s42484-020-00020-y>

Vapnik V (1998) *Statistical Learning Theory*. Wiley-Interscience

Vapnik V, Chervonenkis A (2015) On the uniform convergence of relative frequencies of events to their probabilities, pp 11–30. [https://doi.org/10.1007/978-3-319-21852-6\\_3](https://doi.org/10.1007/978-3-319-21852-6_3)

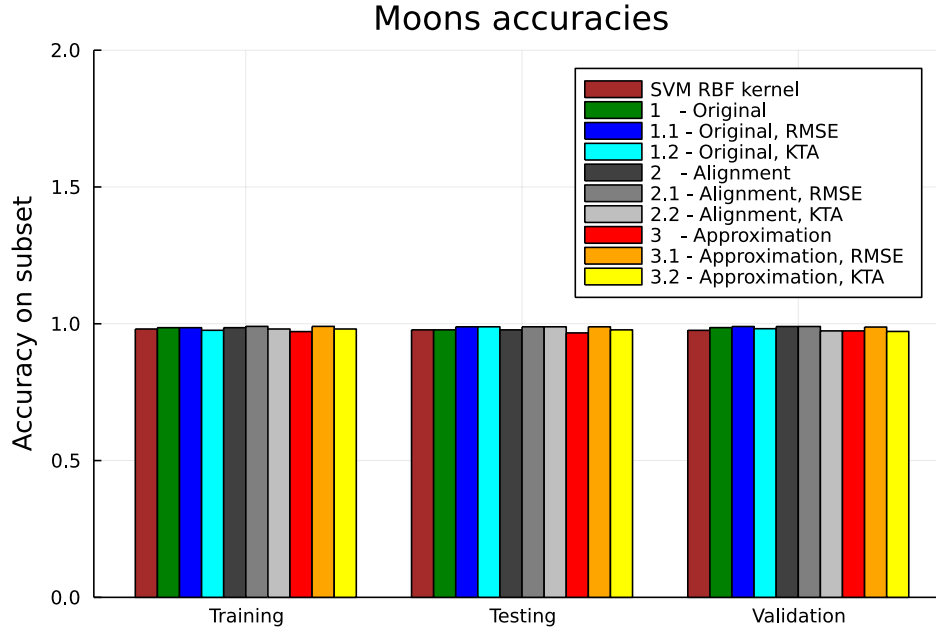
## A Appendix

For Figures 10 - 18, we show the accuracy of the best produced kernels on each dataset, with the classical RBF kernel also shown for comparison. These figures show that the newly proposed approaches achieve comparable classification accuracy to the original approach, despite not directly evaluating classification accuracy during training. The graphs for the difficult problems (Random, Voice, SUSY, SUSY reduced features) show that the original approach has a tendency to overfit to the testing data used to evaluate the accuracy metric, and that a separate validation set must be reserved to estimate the generalisation of its models. The approaches based on maximizing the approximation of kernel-target alignment are also seen to overfit to training data in these cases more than the other approaches do.

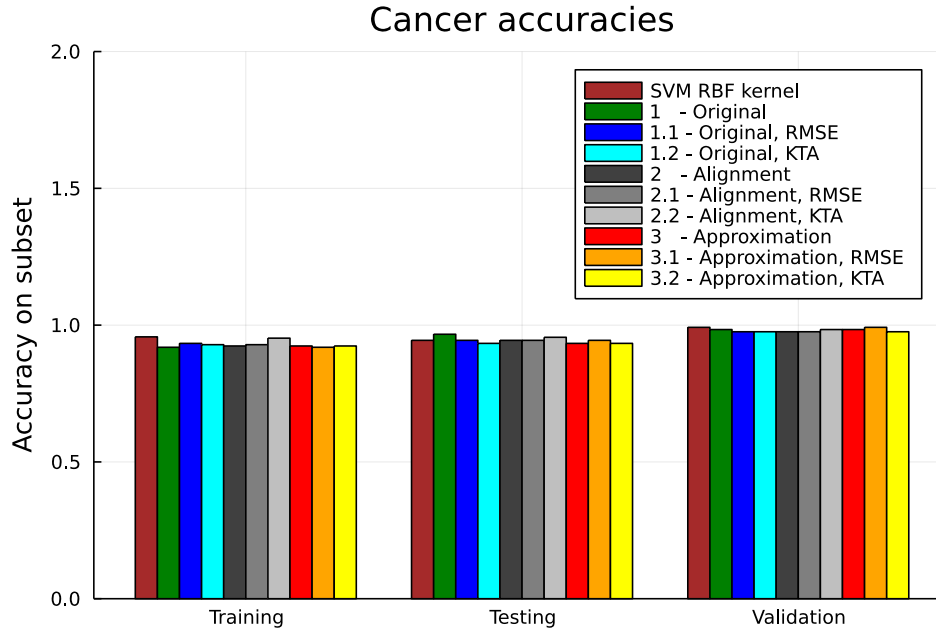
Figures 19 - 27 show the margins of the same best produced kernels on the training data from each dataset. The approaches based on maximizing kernel-target alignment and its approximation tend to have larger average margin sizes, and final parameter training for RMSE and kernel-target alignment also tends to increase the margin sizes. The exception to the trend was the Iris dataset, which was also the second smallest dataset with only 42 training set points.

Figures 28 - 36 show the ROC curves of the best produced models for each dataset. Across the datasets, the curves are mostly similar when comparing approaches.

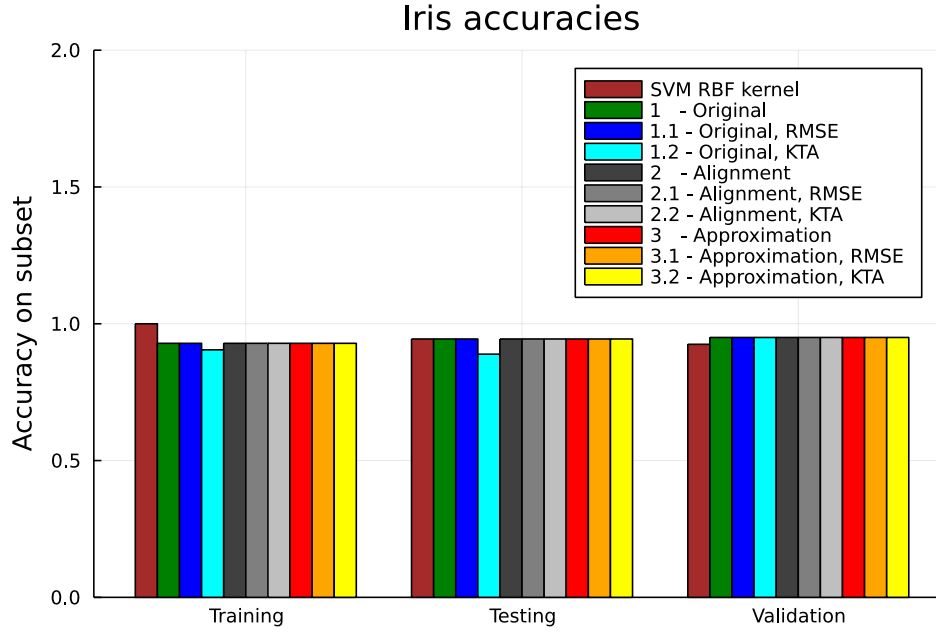
Figures 37 - 39 show the decision boundaries resulting from the best feature map produced by each approach, for the datasets with two dimensional feature vectors. These can be inspected to see the improvements made by parameter training as well as the complexity of the decision rules produced by each approach.



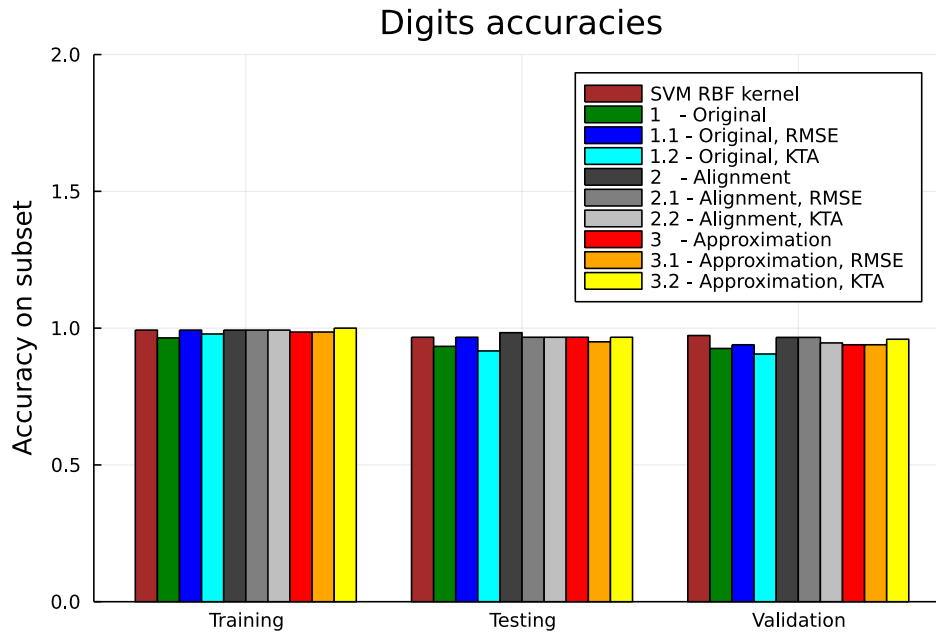
**Fig. 10:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Moons dataset, compared with a classical RBF kernel for reference.



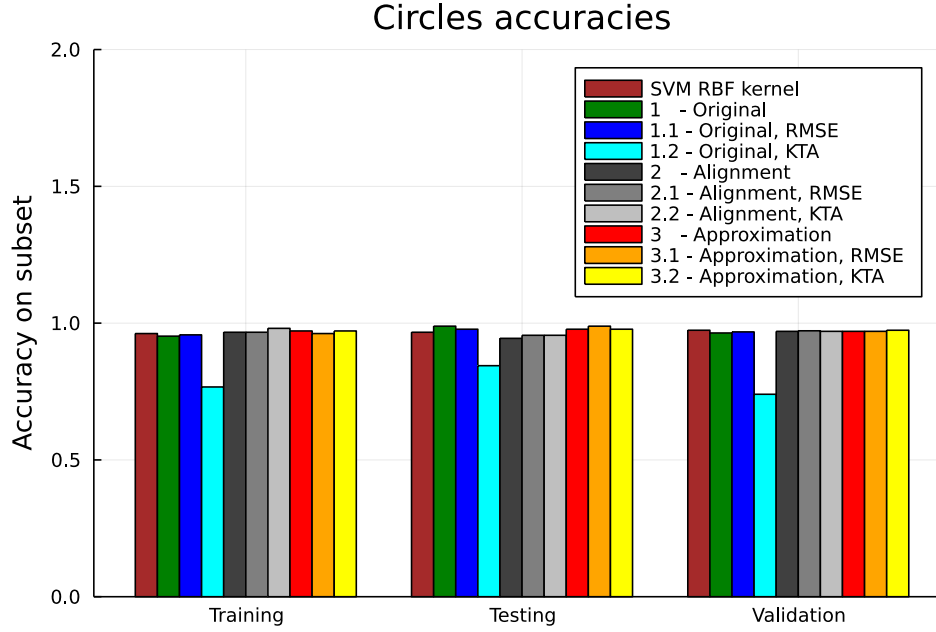
**Fig. 11:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Cancer dataset, compared with a classical RBF kernel for reference.



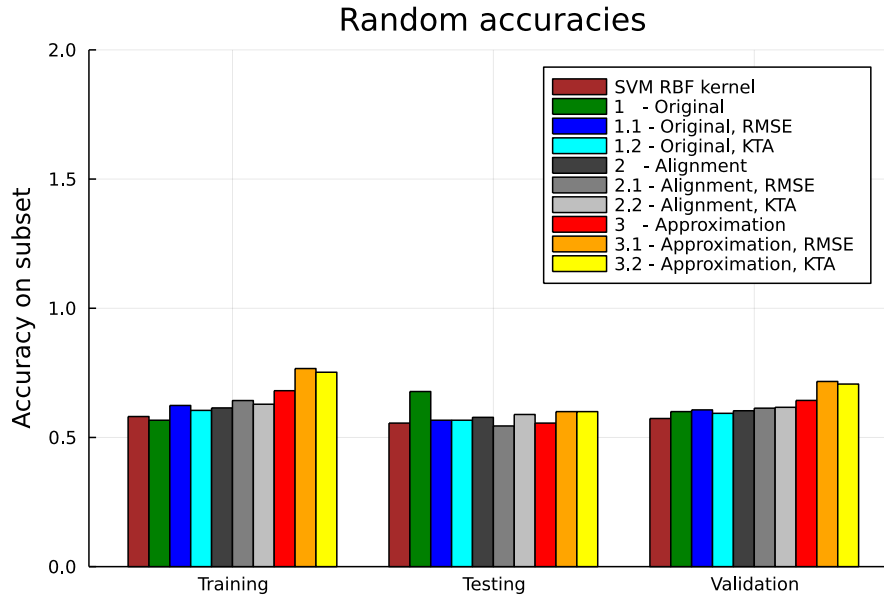
**Fig. 12:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Iris dataset, compared with a classical RBF kernel for reference.



**Fig. 13:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Digits dataset, compared with a classical RBF kernel for reference.

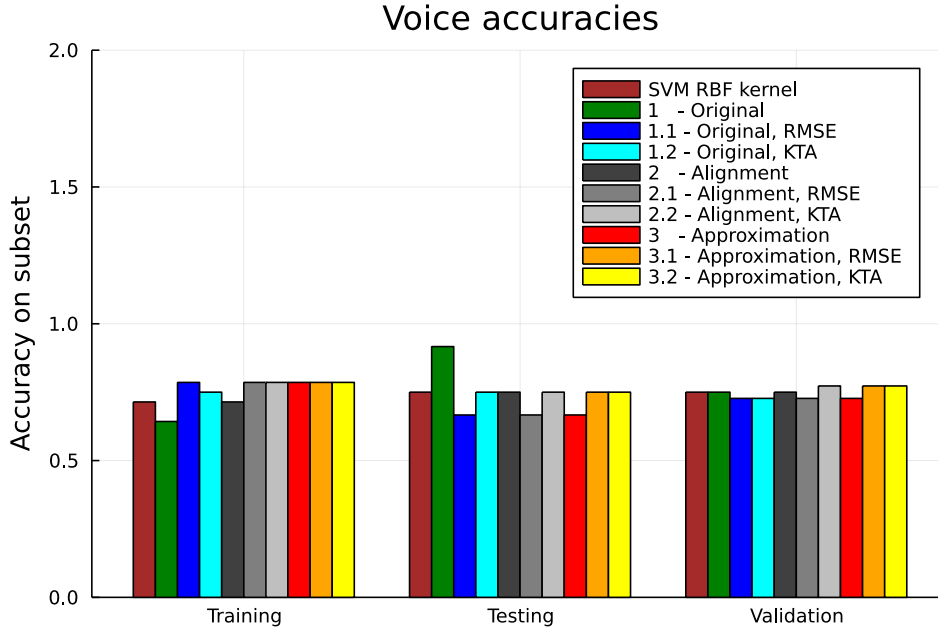


**Fig. 14:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Circles dataset, compared with a classical RBF kernel for reference.

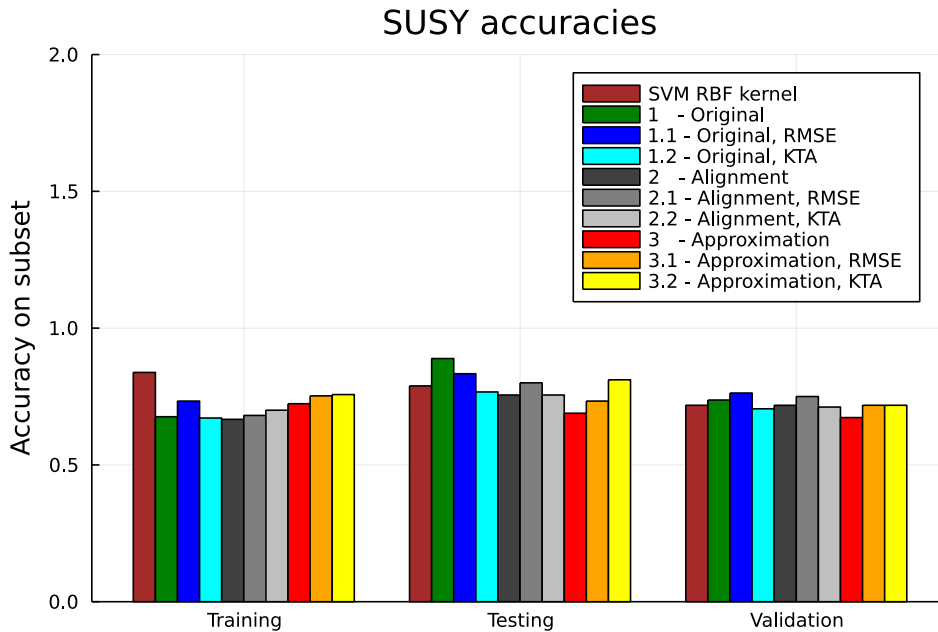


**Fig. 15:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Random dataset, compared with a classical RBF kernel for reference. In the case of this dataset, the validation set is the union of the training and testing points. This gives an idea of the extent to which the approach was able to memorize all the points.

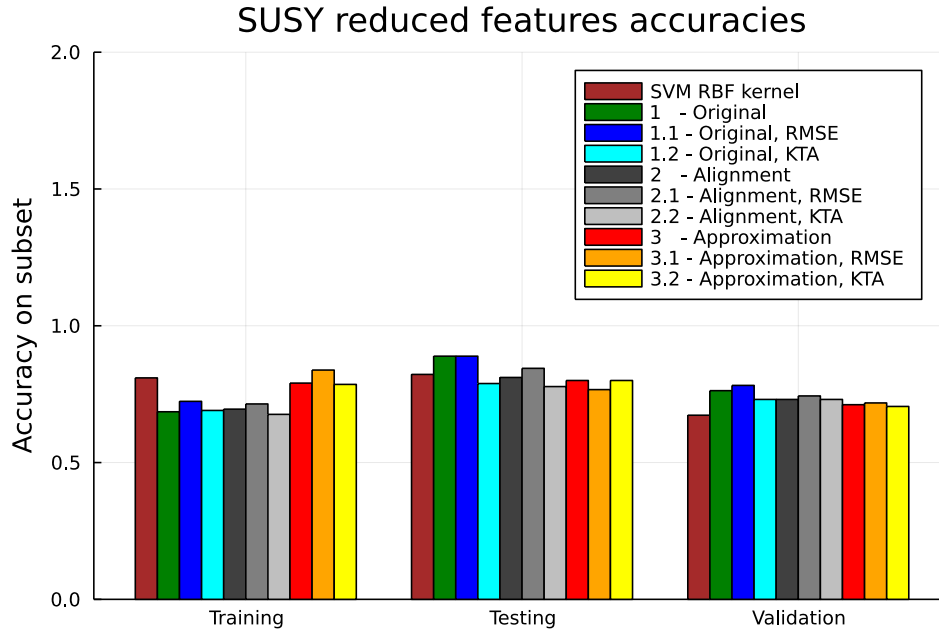




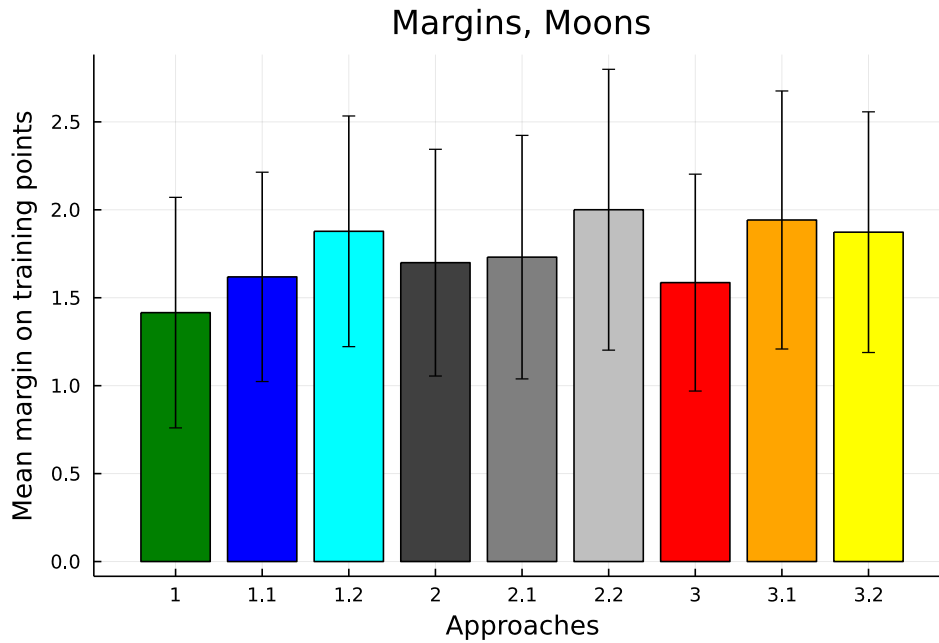
**Fig. 16:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the Voice dataset, compared with a classical RBF kernel for reference.



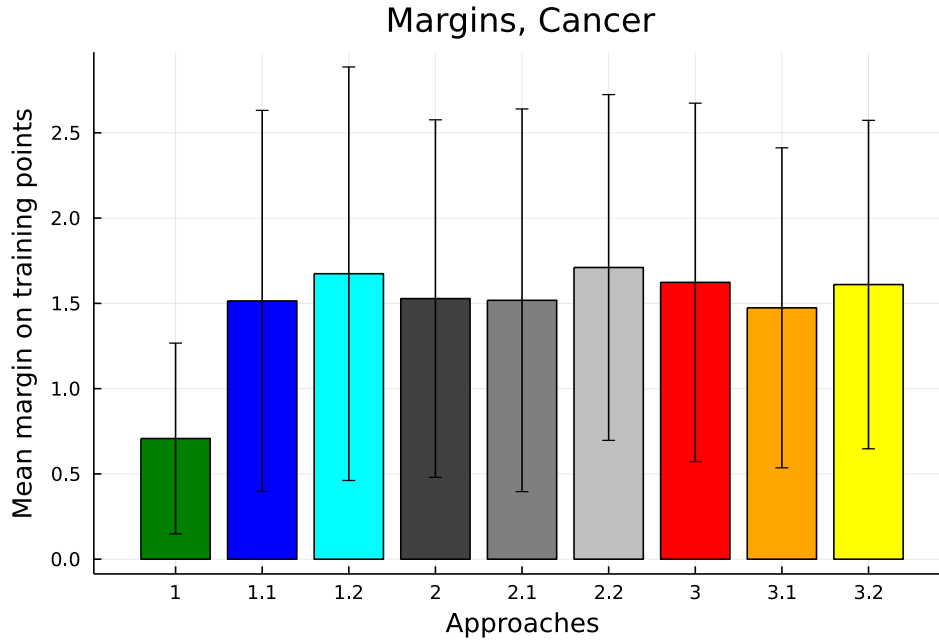
**Fig. 17:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the SUSY dataset, compared with a classical RBF kernel for reference.



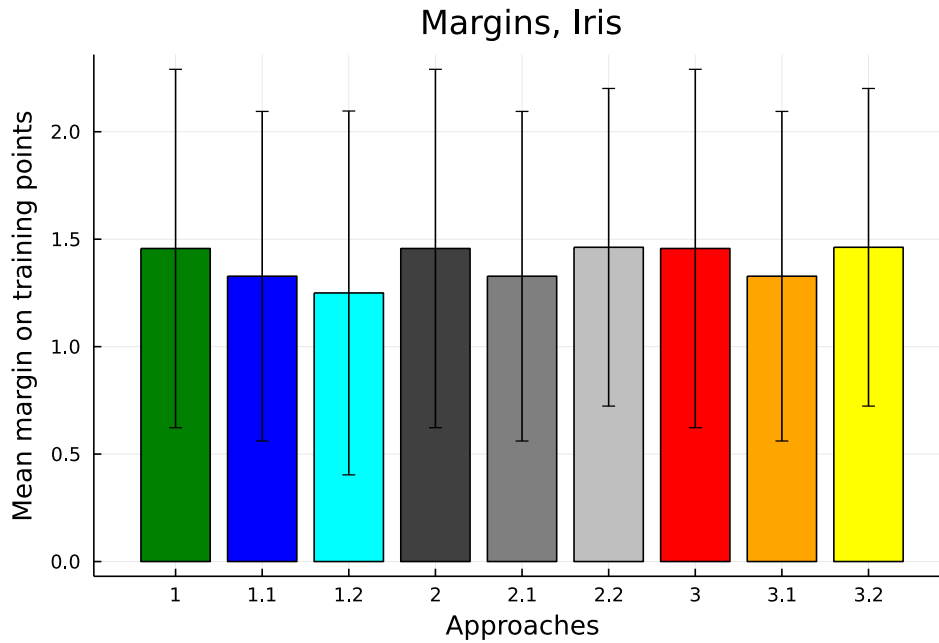
**Fig. 18:** A graph showing the classification accuracies of the best models produced by various approaches of quantum feature map design on the SUSY reduced features dataset, compared with a classical RBF kernel for reference.



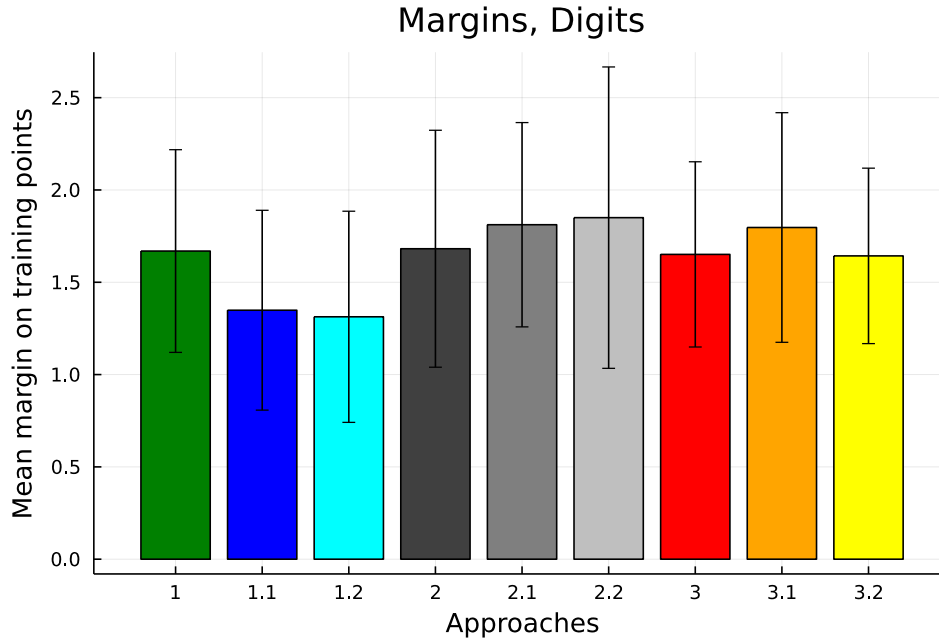
**Fig. 19:** A graph showing the mean margin of the Moons training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



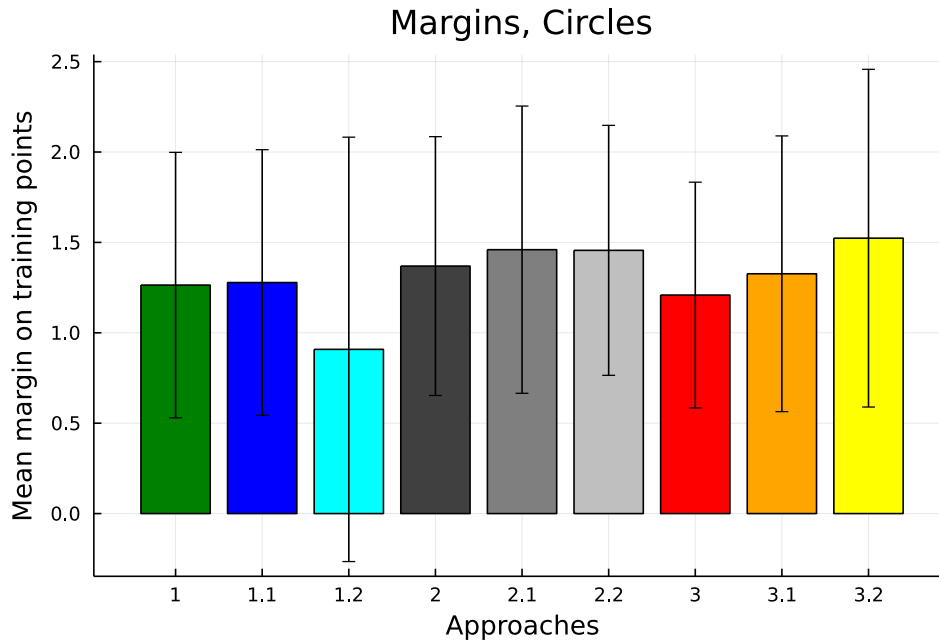
**Fig. 20:** A graph showing the mean margin of the Cancer training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



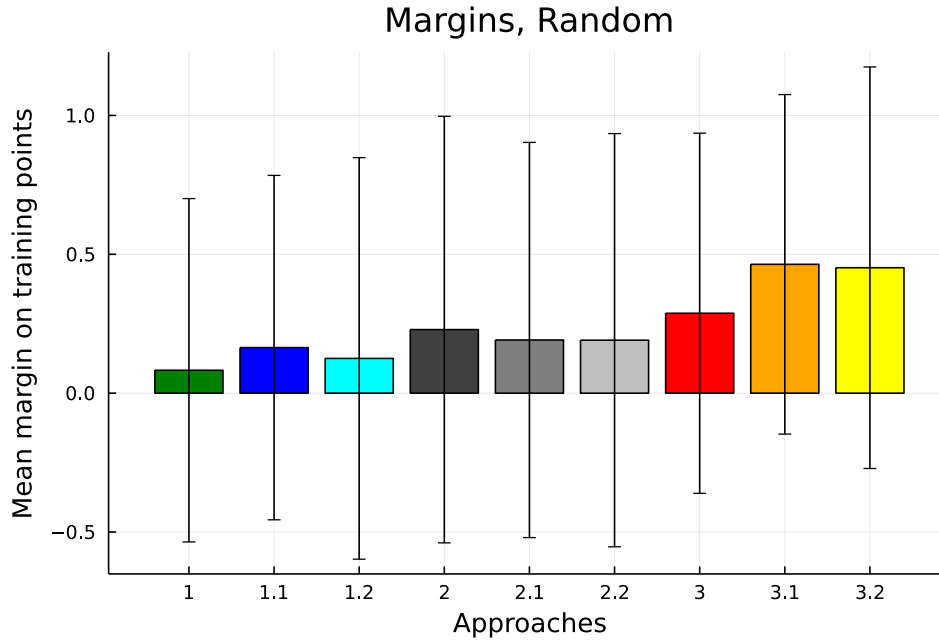
**Fig. 21:** A graph showing the mean margin of the Iris training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



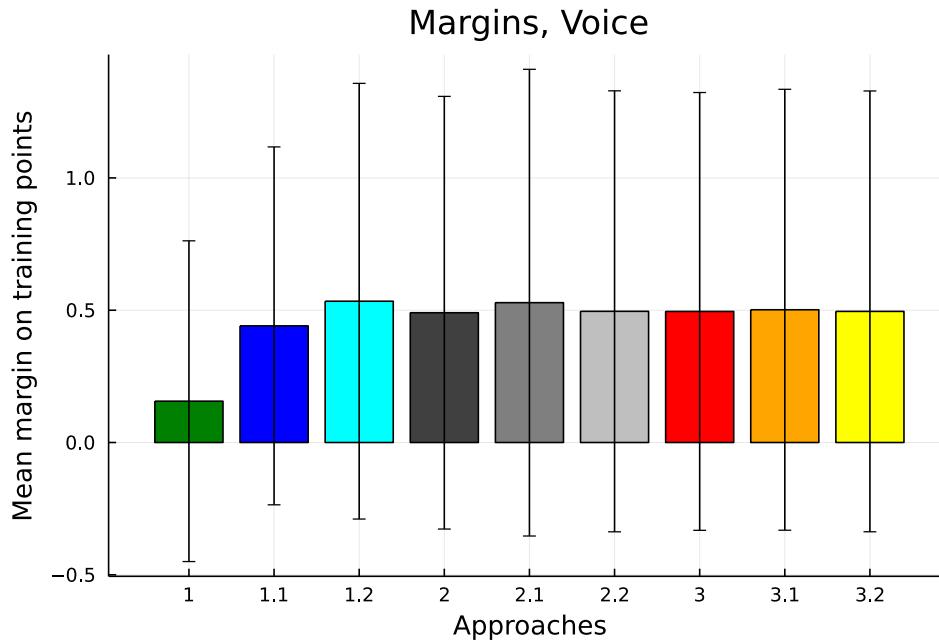
**Fig. 22:** A graph showing the mean margin of the Digits training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



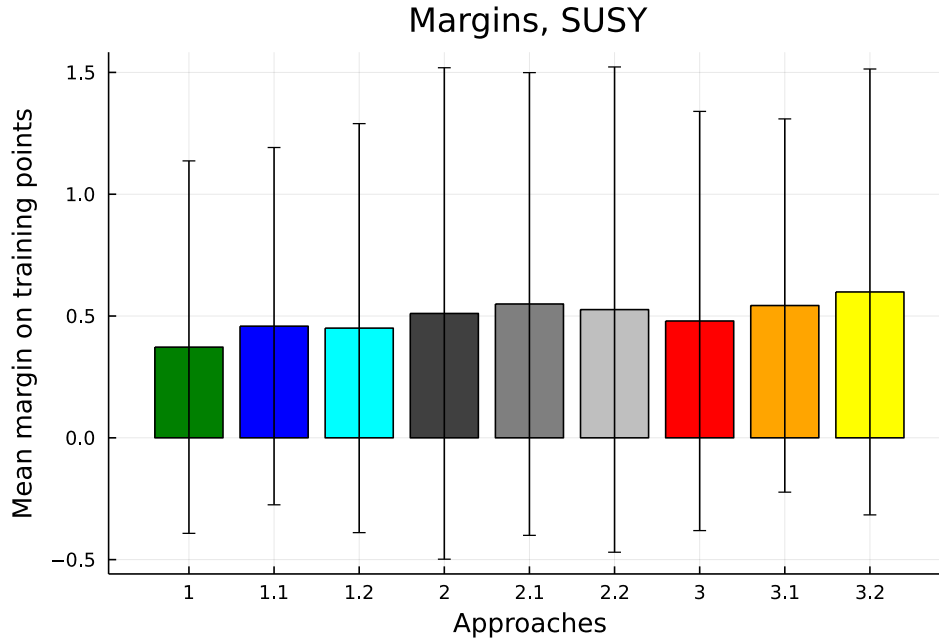
**Fig. 23:** A graph showing the mean margin of the Circles training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



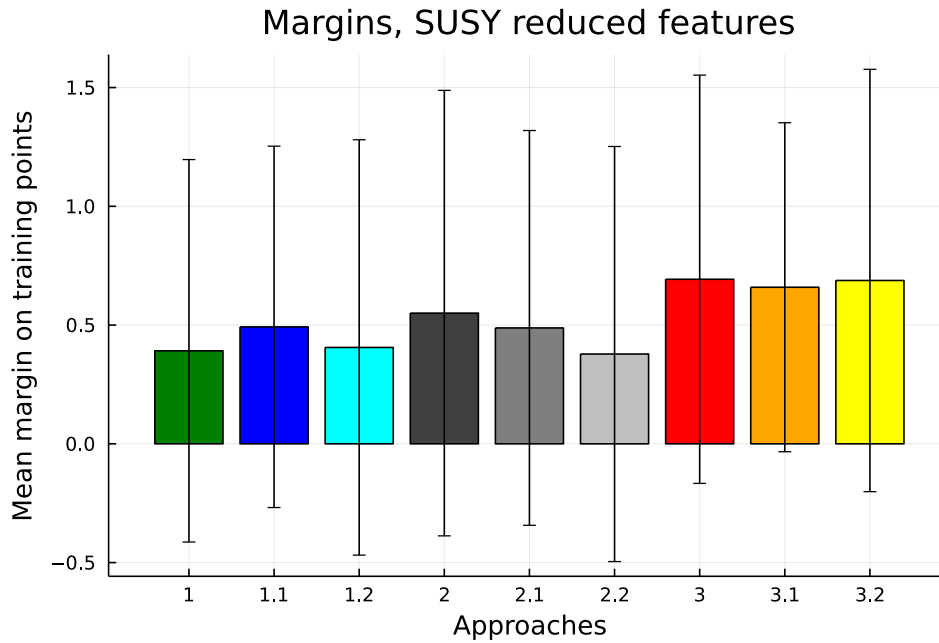
**Fig. 24:** A graph showing the mean margin of the Random training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



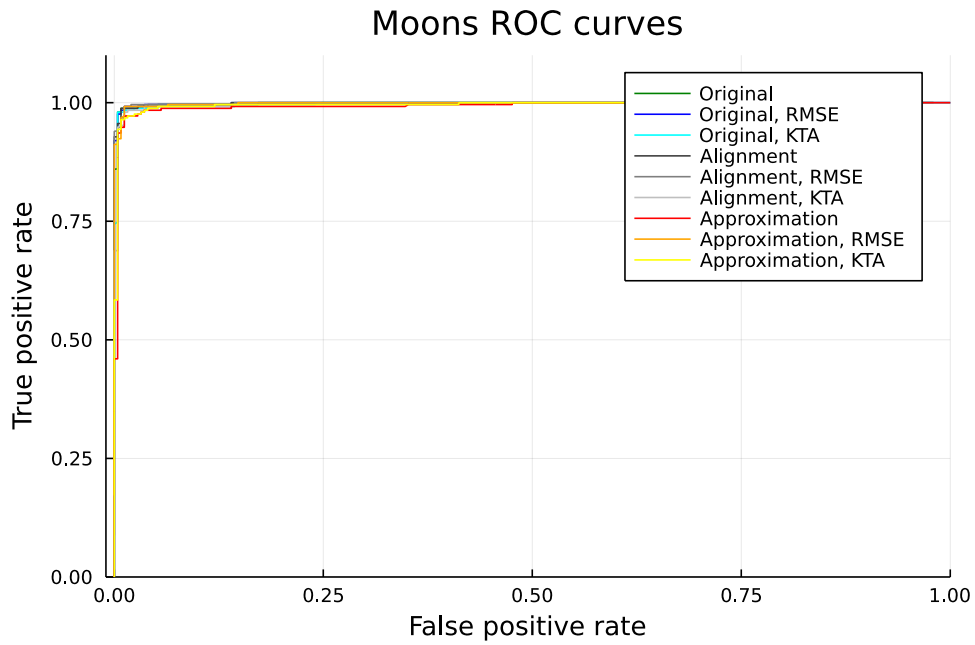
**Fig. 25:** A graph showing the mean margin of the Voice training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



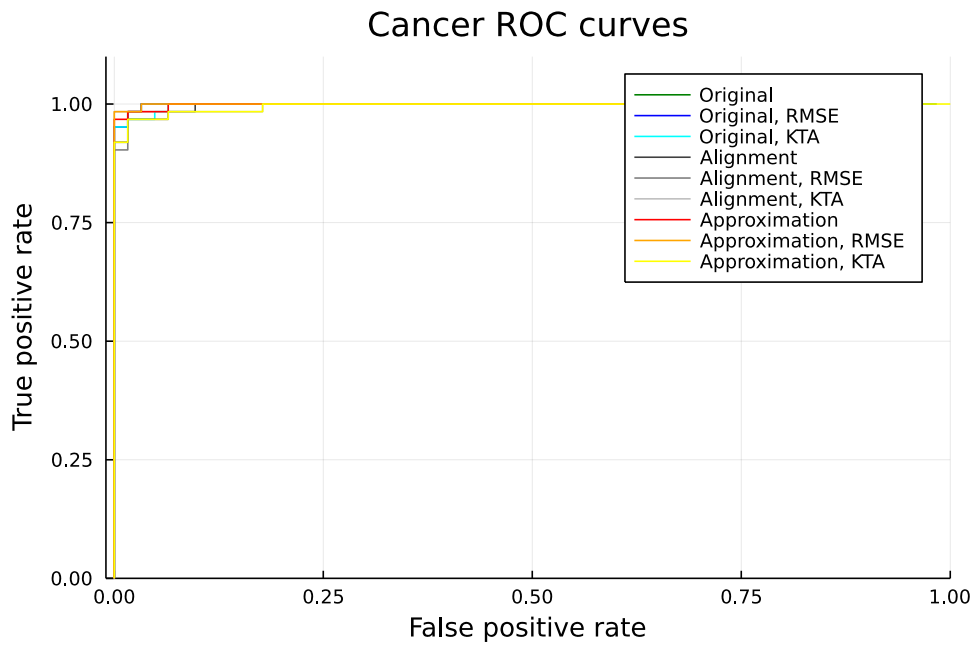
**Fig. 26:** A graph showing the mean margin of the SUSY training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



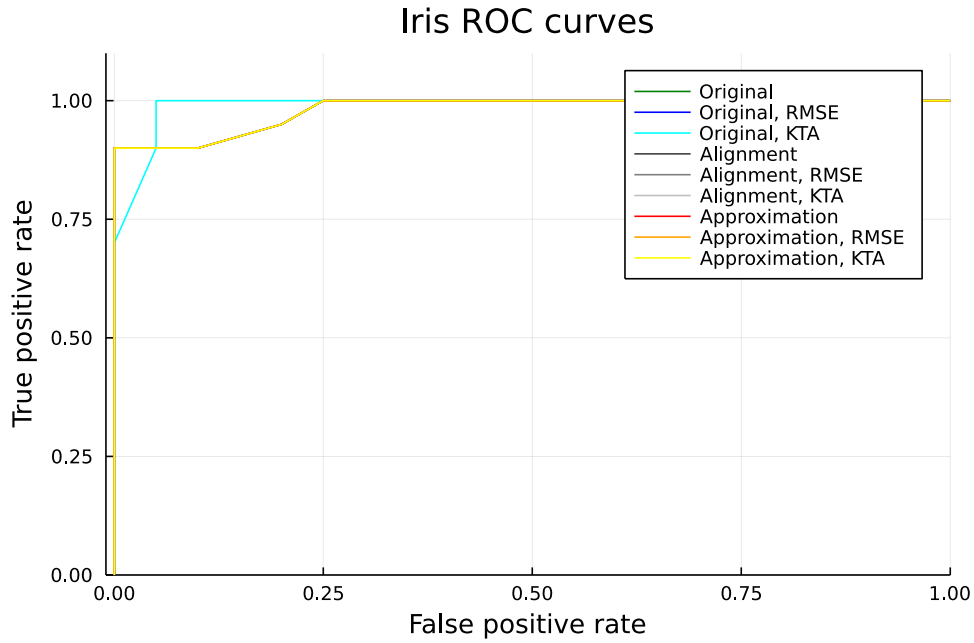
**Fig. 27:** A graph showing the mean margin of the SUSY reduced features training set points for the best classifiers produced by each approach, with errors bars showing standard deviation.



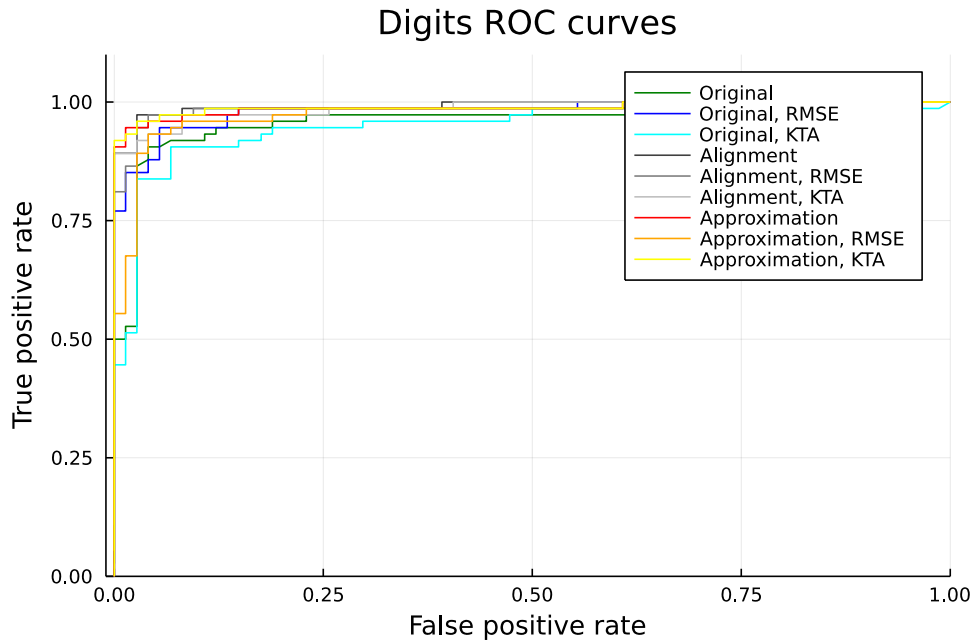
**Fig. 28:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Moons dataset.



**Fig. 29:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Cancer dataset.

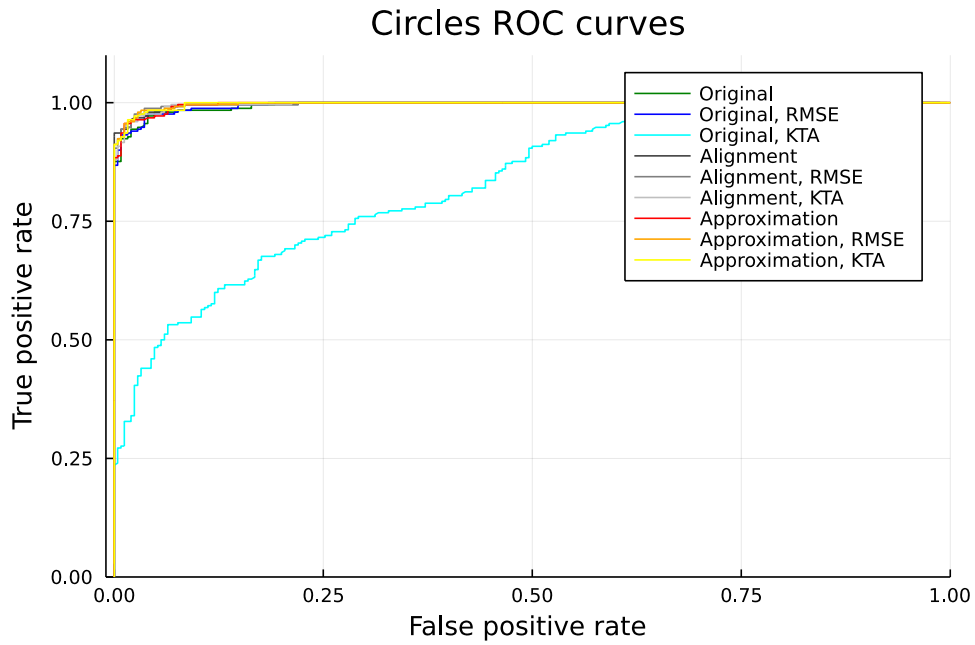


**Fig. 30:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Iris dataset.

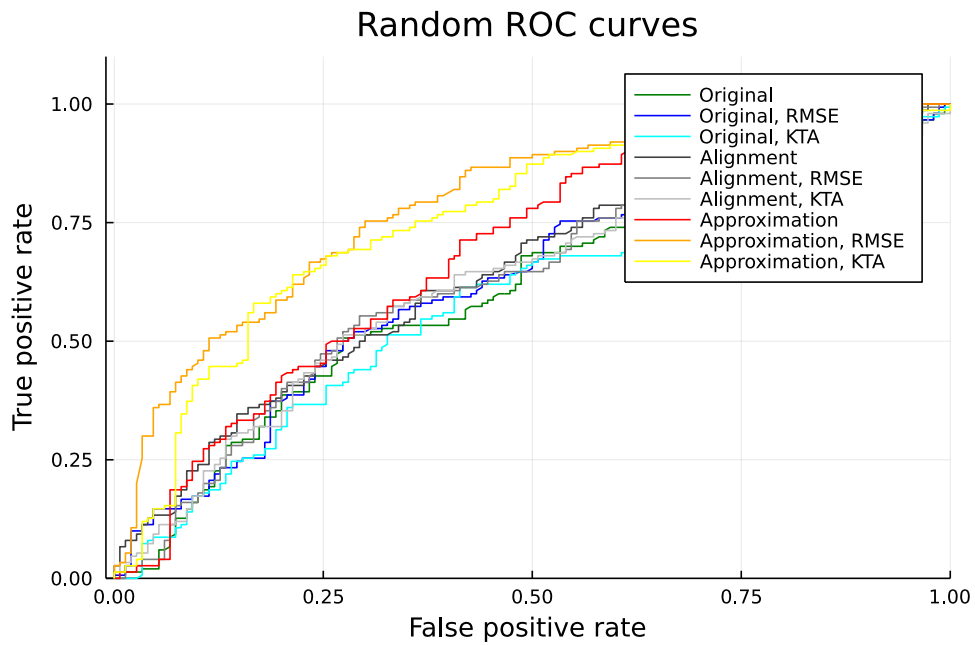


**Fig. 31:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Digits dataset.

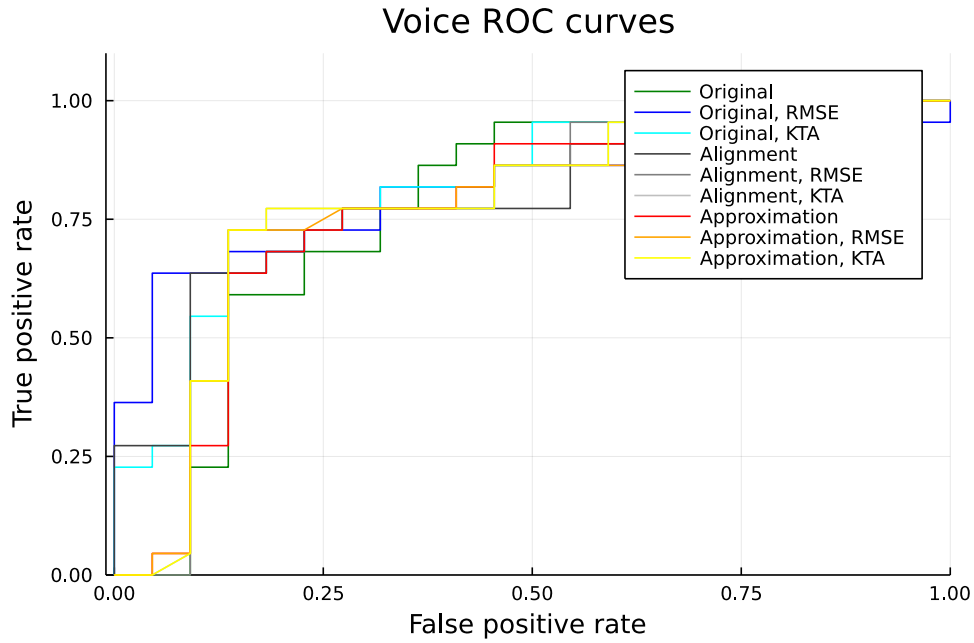




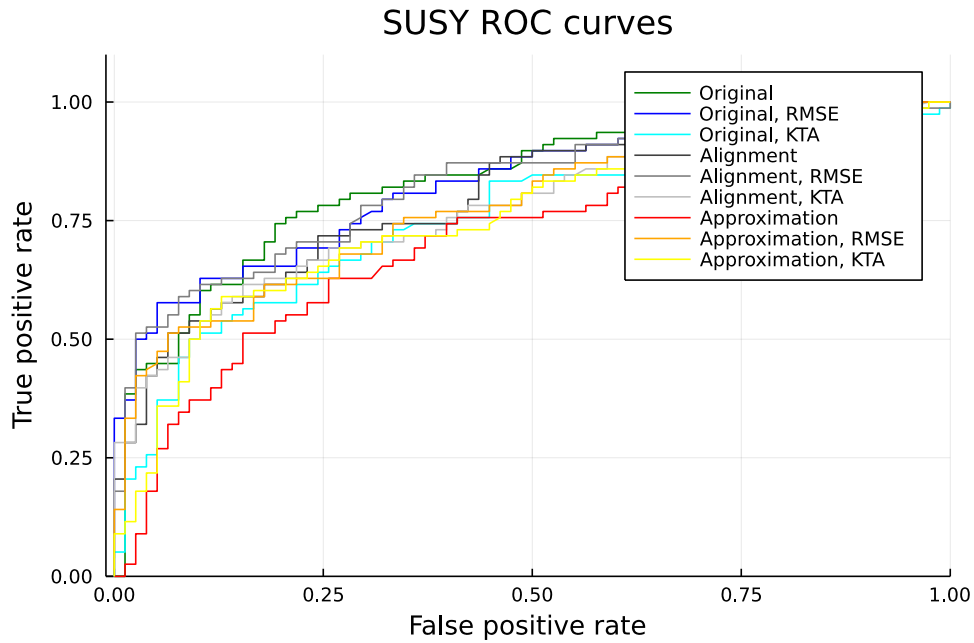
**Fig. 32:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Circles dataset.



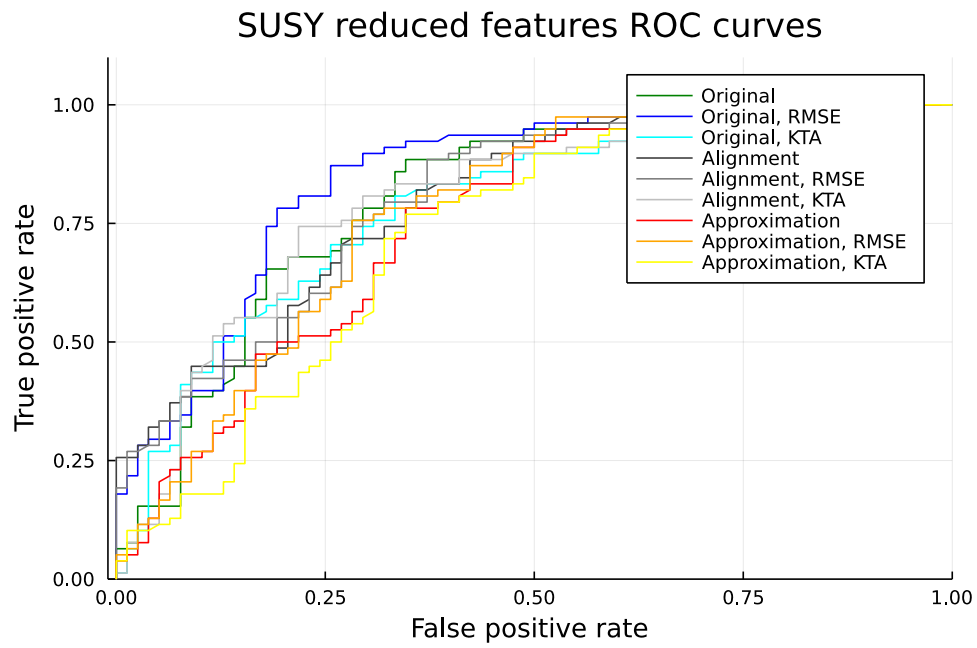
**Fig. 33:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Random dataset.



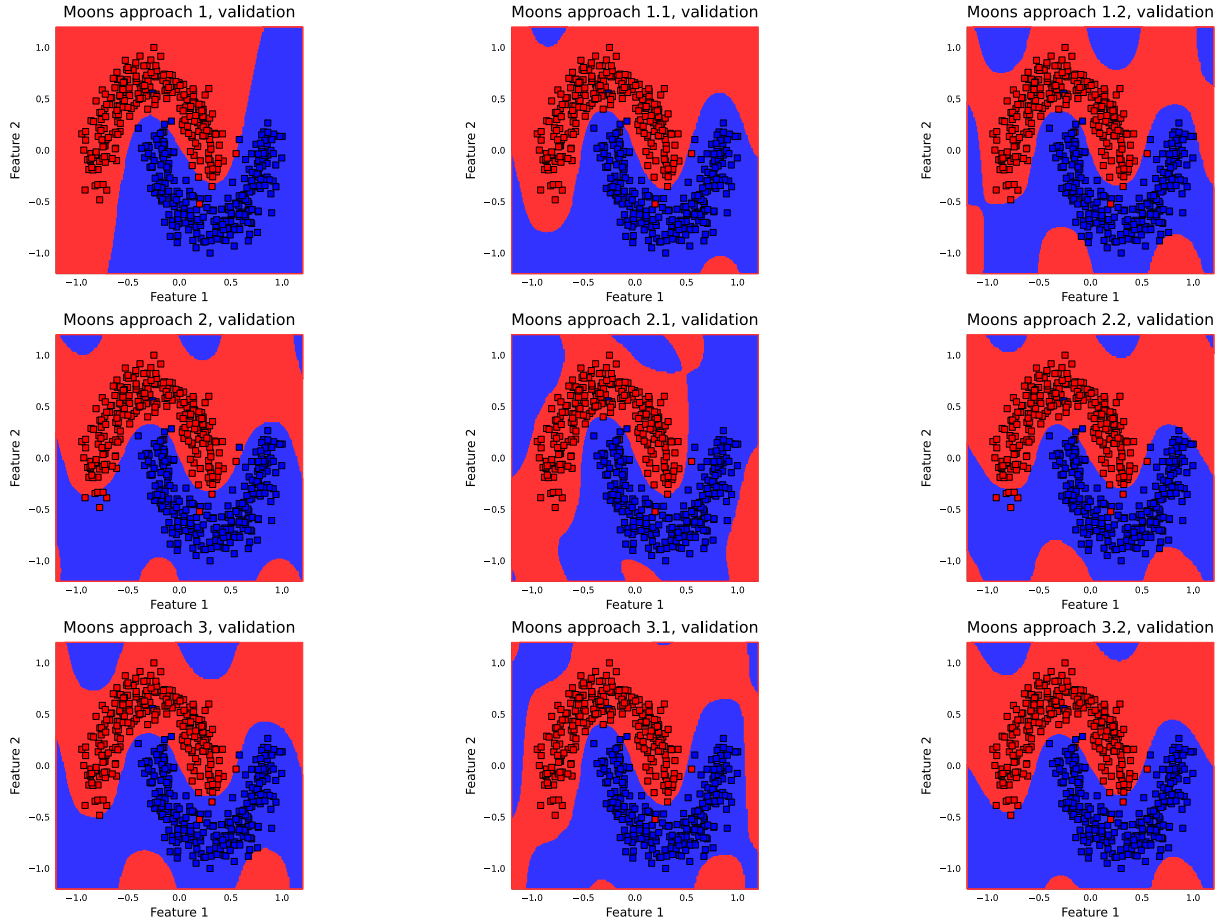
**Fig. 34:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the Voice dataset.



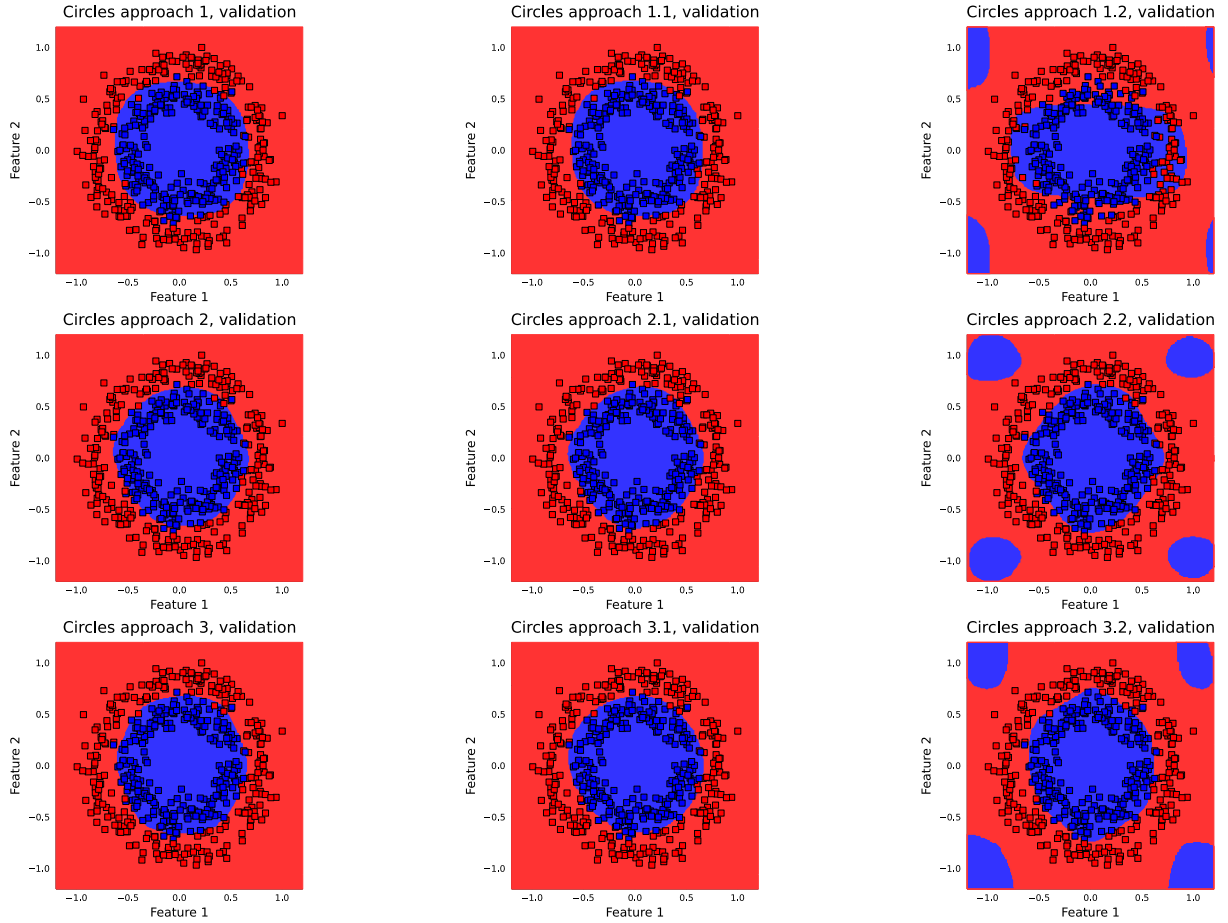
**Fig. 35:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the SUSY dataset.



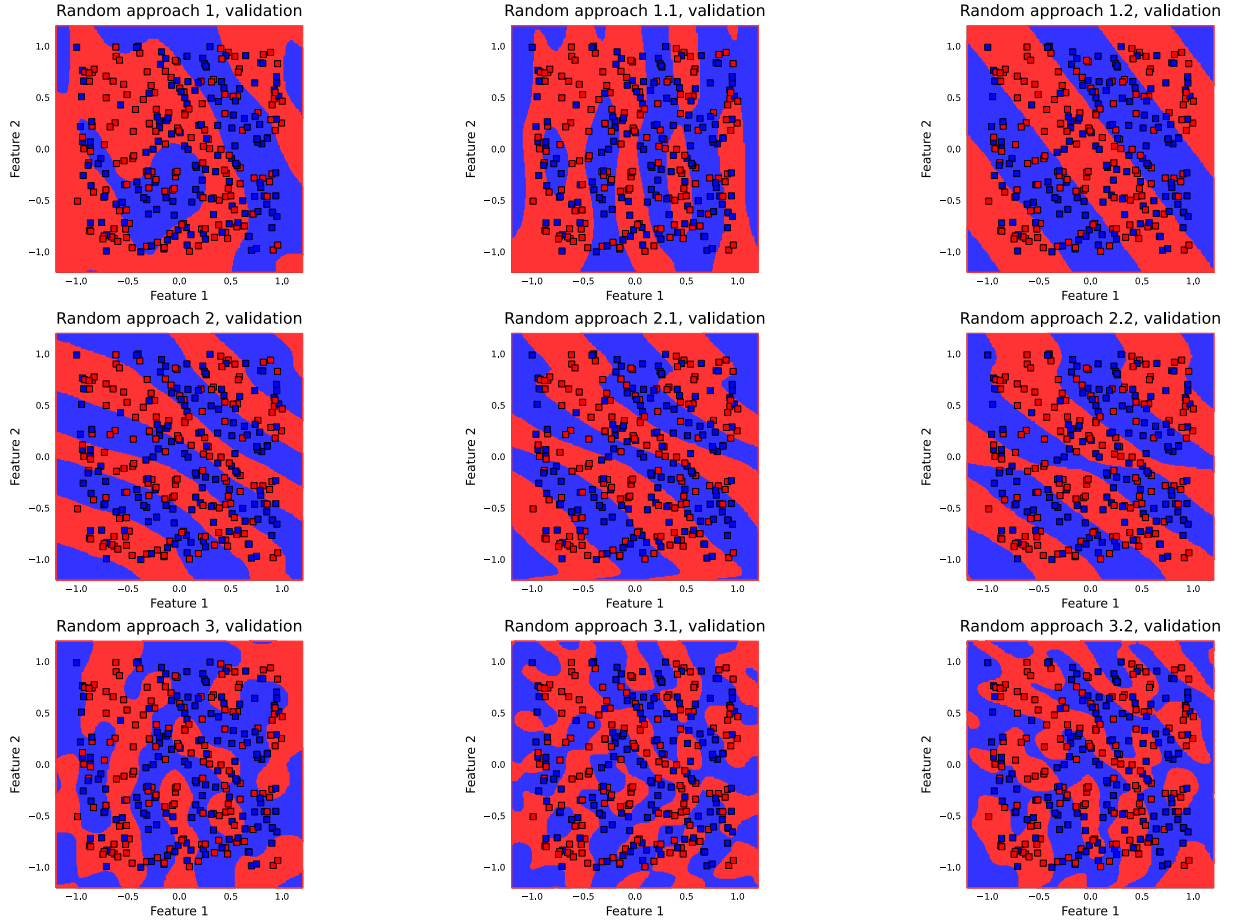
**Fig. 36:** A graph showing the ROC curves of the best models produced by various approaches of quantum feature map design on the SUSY reduced features dataset.



**Fig. 37:** Decision boundaries for each approach on the Moons validation data.



**Fig. 38:** Decision boundaries for each approach on the Circles validation data.



**Fig. 39:** Decision boundaries for each approach on the Random validation data. In the case of the Random dataset, the validation data does not consist of unseen randomly generated points as would be the case in other datasets. Under the assumption that the models cannot generalise to new pseudorandom data, the validation dataset instead shows the concatenation of the training and testing points to give an idea of the different approaches' capacity for memorizing a random assignment of labels to random points.

# Chapter 3

## Conclusion

### 3.1 Summary of the Work and Key Findings

In this thesis, we set out to investigate and improve algorithms for the automated design of quantum feature map circuits for use with the QSVM algorithm, attempting to find ways to make them faster and produce better performing circuits.

Chapter 1 provided the context and background to the work. It began with an introduction to the subject of quantum computing. This introduction covered the fundamental principles of quantum computing, including quantum information, operating on quantum states, quantum circuit diagrams as a language for quantum algorithms, a walk-through of a real quantum algorithm that demonstrates a provable advantage over any classical method, and the challenges of running quantum algorithms on current quantum hardware.

The introduction also reviewed material relevant to machine learning and included discussions on classification tasks, the importance of generalization, the classical Support Vector Machine (SVM) algorithm and the significance of margins. The central concepts of this work relate to the field of quantum machine learning. We gave an overview of relevant techniques used in this work that included: the quantum extension of the SVM algorithm, the Quantum-enhanced Support Vector Machine (QSVM), the optimization of Parameterized Quantum Circuits (PQCs) to improve their suitability for a given task, and noted the similarity of this process to classical

neural network training. The background concluded with a discussion on the use of genetic algorithms for solving combinatorial optimization problems.

This was followed with accounts of the major prior works this thesis built on:

1. **Automatic Design of Quantum Feature Maps** [2], in which the authors devised and tested a multi-objective genetic algorithm for automatically generating efficient quantum feature maps for a target dataset by simultaneously maximizing achieved accuracy and minimizing circuit size in the generated feature map circuits.
2. **Training Quantum Embedding Kernels on Near-Term Quantum Computers** [15], in which the authors applied PQC optimization to quantum feature map circuits to maximize kernel-target alignment, improving the suitability of feature map circuits for a target dataset without modifying their structure.

Next the methods used in our own work were detailed, which can be summarized as testing nine variations of the genetic algorithm defined in [2] across nine datasets. Eight of the variations were new, while one was a reproduction of the original approach. The eight new variations were derived by replacing accuracy maximization in the genetic algorithm with either kernel-target alignment or an approximation of it, and by following up the genetic optimization configurations with an additional step of circuit parameter optimization aimed at either maximizing kernel-target alignment or minimizing root mean squared error.

We then detailed the main contributions of our work, which we restate briefly here:

1. Experimentally verifying the effectiveness of kernel-target alignment as a metric for genetic algorithm based quantum feature map design, showing it produces circuits achieving similar accuracy to the original approach while being faster to evaluate and, notably, achieves larger margin sizes.
2. Proposing an approximation of kernel-target alignment that requires fewer quantum kernel evaluations to compute, and experimentally demonstrating its effec-



tiveness in genetic algorithm based quantum feature map design. This effectiveness includes the advantage of larger margin sizes seen when fully evaluating kernel-target alignment.

3. Experimentally demonstrating the synergistic effectiveness of applying both genetic feature map circuit design and feature map circuit parameter optimization in the task of automating quantum feature map circuit design.

The second chapter of the work is a reprint of the main output of the work: a journal article which provides experimental justification for the contribution claims made in this thesis.

## 3.2 Implications and Significance

The research outputs detailed in this thesis have several implications for the ongoing development of quantum feature map design. Firstly, we demonstrate that genetic algorithm based approaches can generate effective feature maps even when the metric maximized is relatively fast to evaluate. This opens the door for testing other metrics that are efficient to evaluate while still producing effective feature maps. Furthermore, the results provide further evidence of the effectiveness of optimizing metrics based on kernel-target alignment in particular.

In addition, our results show that a hybrid approach combining genetic algorithm optimization and circuit parameter optimization has performance advantages over just using a genetic algorithm. This is an encouraging sign for the potential of hybrid optimization methods in automatically designing quantum circuits for other tasks.

## 3.3 Future Work

There remains considerable scope for further research. Potential ideas for future work include:

1. Modifying the genetic algorithm to be better able to select proportionality parameters. This could be done by encoding the proportionality parameters of gates using continuous values instead of a bit sequence selecting from one of four fixed values. This would also require modifying the genetic crossover and mutation operations, but could potentially make the application of a classical optimizer for feature map circuit parameter optimization unnecessary.
2. Using metric approximations that dynamically trade off execution speed and accuracy of approximation. An inaccurate approximation of a metric may still be as useful as the metric itself in the early stages of the genetic optimization process. An approximation metric could be devised that gradually increases the accuracy of its approximation throughout the optimization process to improve run time, potentially without affecting the quality of the final result.
3. Optimizing multiple genetic algorithm metrics could be done in sequential phases. This might solve the problem encountered in our work (kernel-target alignment producing larger circuits than accuracy) while still maintaining some speed advantage. This idea, like the previous one, would involve the initial phase of the genetic optimization being accelerated with a metric that is fast to compute like approximated kernel-target alignment. After a set number of iterations, the genetic algorithm can switch to maximizing accuracy as a metric in an attempt to produce the smaller circuits observed when using accuracy.
4. The genetic algorithm could be modified to select nonlinear classical functions to apply to feature values before substituting them into the feature map circuit. Such functions have previously been shown to change the effectiveness of feature maps [30]. This could greatly increase the expressive power of a feature map without changing the quantum resources required to perform a single kernel estimation. Selecting a classical function can be cast as a combinatorial optimization problem, meaning a genetic algorithm would be well suited to performing this task.

### 3.4 Closing Thoughts

This thesis has contributed to the rapidly evolving field of quantum machine learning in general and automated quantum feature map design in particular. As the capabilities of real quantum hardware continue to improve, we move slowly toward a future where quantum-enhanced algorithms may redefine what is computationally possible. The approaches detailed in this work and its successors may one day be used to help harness the potential of powerful quantum machines to solve complex machine learning tasks.

# Bibliography

- [1] Mohamed Alloghani, Dhiya Al-Jumeily Obe, Jamila Mustafina, Abir Hussain, and Ahmed Aljaaf. *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*, pages 3–21. 2020.
- [2] Sergio Altares-López, Angela Ribeiro, and Juan José García-Ripoll. Automatic design of quantum feature maps. *Quantum Science and Technology*, 6(4):045015, 2021.
- [3] Andrei Bautu and Elena Bautu. Quantum circuit design by means of genetic programming. 2007.
- [4] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152.
- [5] Zhenyu Cai, Ryan Babbush, Simon C. Benjamin, Suguru Endo, William J. Huggins, Ying Li, Jarrod R. McClean, and Thomas E. O’Brien. Quantum error mitigation. *Rev. Mod. Phys.*, 95:045005, 2023.
- [6] Vijay Chahar, Sourabh Katoch, and Sumit Chauhan. A review on genetic algorithm: Past, present, and future. *Multimedia Tools and Applications*, 80, 2021.
- [7] Bang-Shien Chen and Jann-Long Chern. Generating quantum feature maps for svm classifier. *arXiv.org*, 2022.
- [8] Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz Kandola. On kernel-target alignment. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.
- [9] Christopher M. Dawson and Michael A. Nielsen. The solovay-kitaev algorithm. *Quantum Info. Comput.*, 6(1):81–95, jan 2006.
- [10] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

- [11] David Deutsch and Richard Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, 439(1907):553–558, 1992.
- [12] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng. Digital zero noise extrapolation for quantum error mitigation. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 306–316, Los Alamitos, CA, USA, 2020. IEEE Computer Society.
- [13] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, New York, NY, USA, 1996. Association for Computing Machinery.
- [14] Karel Horak and Robert Sablatnig. Deep learning concepts and datasets for image recognition: overview 2019. In *Deep learning concepts and datasets for image recognition: overview 2019*, page 100, 2019.
- [15] Thomas Hubrigtsen, David Wierichs, Elies Gil-Fuster, Peter-Jan H. S. Derks, Paul K. Faehrmann, and Johannes Jakob Meyer. Training quantum embedding kernels on near-term quantum computers. *Phys. Rev. A*, 106:042431, 2022.
- [16] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8580–8589, Red Hook, NY, USA, 2018. Curran Associates Inc.
- [17] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, 2022.
- [18] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, 2021.
- [19] M. Lukac and M. Perkowski. Evolving quantum circuits using genetic algorithm. In *Proceedings 2002 NASA/DoD Conference on Evolvable Hardware*, pages 177–185, 2002.
- [20] Rowan Pellow-Jarman. Hybrid genetic optimisation for quantum feature map design. <https://github.com/RowPJ/hybrid-genetic-optimisation-for-quantum-feature-map-design>, 2022.
- [21] Rowan Pellow-Jarman, Anban Pillay, Ilya Sinayskiy, and Francesco Petruccione. Hybrid genetic optimisation for quantum feature map design, 2023.
- [22] M. J. D. Powell. *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolation*, pages 51–67. Springer Netherlands, Dordrecht, 1994.

- [23] John Preskill. Quantum Computing in the NISQ era and beyond, 2018. arXiv:1801.00862v3.
- [24] Riccardo Rasconi and Angelo Oddi. An innovative genetic algorithm for the quantum circuit compilation problem. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI’19/IAAI’19/EAAI’19. AAAI Press, 2019.
- [25] Patrick Rebentrost, M. Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical Review Letters*, 113, 2013.
- [26] Joschka Roffe. Quantum error correction: an introductory guide. *Contemporary Physics*, 60(3):226–245, 2019.
- [27] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Phys. Rev. Lett.*, 122:040504, 2019.
- [28] Maria Schuld and Francesco Petruccione. *Quantum Models as Kernel Methods*, pages 217–245. Springer International Publishing, Cham, 2021.
- [29] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.
- [30] Yudai Suzuki, Hiroshi Yano, Qi Gao, Shumpei Uno, Tomoki Tanaka, Manato Akiyama, and Naoki Yamamoto. Analysis and synthesis of feature map for kernel-based quantum classifier. *Quantum Machine Intelligence*, 2(1), 2020.
- [31] V. N. Vapnik and A. Y. Chervonenkis. *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*, pages 11–30. Springer International Publishing, Cham, 2015.
- [32] V.N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.
- [33] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023.