People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

IBN KHALDOUN UNIVERSITY OF TIARET

Dissertation

Presented to:

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTEMENT OF COMPUTER SCIENCE

in order to obtain the degree of :

MASTER

Specialty: software engineer

Presented by:
Mokhtari Hadj Mohamed

On the theme:

Water quality prediction based on quantum machine learning

Defended publicly on / /2024 in Tiaret in front the jury composed of:

Mr Aid Lahcene	MCA	Tiaret University	Chairman
Mr Meghazi Hadj Madani	MCB	Tiaret University	Supervisor
Mr Moustefaoui kadda	MAA	Tiaret University	Examiner

2023-2024

Abstract

The quantum computer is the future of the next generation of computers with his qualifications in the use of entanglement and superposition, and using it for machine learning what is called quantum machine learning in the quantum world, we will see a great change in the world of data science with good results in small data as well in very big data that classical computers can't handle, we have got the problem of water quality and apply we try to apply on it quantum machine learning algorithm that named VQC algorithm from Qiskit library we have to get the result that good with the data that we have.

الحاسوب الكمي هو مستقبل الجيل القادم من الحواسيب بفضل مؤهلاته في استخدام التشابك والتراكب، واستخدامه في التعلم الآلي فيما يعرف بالتعلم الآلي الكمي في عالم الكم. سنشهد تغييرًا كبيرًا في عالم علم البيانات مع نتائج جيدة في البيانات الصغيرة وكذلك في البيانات الكبيرة جدًا التي لا تستطيع الحواسيب التقليدية معالجتها. لدينا مشكلة جودة المياه ونحاول تطبيق خوارزمية التعلم الآلي الكمي المعروفة باسم خوارزمية VQC من مكتبة Qiskit للحصول على نتائج جيدة مع البيانات التي لدينا.

Table of Contents

Table of Figures

Introduction

We will introduce a short introduction to quantum machine learning, quantum programming, and quantum computers. Research has been intensifying in recent years, starting with research about quantum computers. After that, we chose the title “Water Quality Prediction Based on Quantum Machine Learning" because, in the next 10 years, potable water will be decreasing every day. So, we chose two themes: quantum computers and the revolution that many computer scientists predict will change the world, and the problem of water. We chose quantum computers for their speed and utility. We are still at the beginning of the quantum computer era, so in the next five years, more technology will emerge in this field, including programming languages and quantum computers with higher qubit numbers.

We will see more development in machine learning for quantum computers and more algorithms in this category of science. We hope to find more solutions and perform simulations of special problems in the real world that we cannot solve now. We expect to see more secure encryption technology with the end of RSA and the rise of quantum computers. The evolution of quantum computers mirrors the early development of classical computers in 1822. We started this research years ago, and we tried multiple quantum machine learning algorithms like QNN, QSCV, and VQC. However, some algorithms face challenges due to outdated libraries and technology.

We chose potable water and applied machine learning algorithms because, as We said in the last section, potable water is decreasing every day. Regions on Earth, like Africa, have problems with potable water. Many organizations are trying to find solutions. With dryness and random consumption of groundwater, countries like Algeria will also face problems. So, we chose this theme to find some scientific solutions to the water crisis.

We will present this paper in the following structure:

- **Chapter One:** Discusses quantum computers, including all technology and quantum mechanisms.
- **Chapter Two:** Explores quantum machine learning and compares quantum and classical algorithms.
- **Chapter Three:** Demonstrates a model for water quality prediction using quantum machine learning algorithms and how to use it for the first time.

We faced many problems in this research, including issues with the libraries used, datasets, and available technology. We asked, "How can quantum machine learning help humanity?" and specifically, "How can quantum machine learning and quantum computers determine if water is potable or not?"

By addressing these questions, we hope to advance the field of quantum machine learning and contribute to solving critical global issues such as water scarcity.

Chapter I: Quantum Computing

1. Introduction:

Quantum mechanics, a branch of physics, traces its origins to a series of scientific discoveries in the late 19th century, and it has been actively evolving ever since. While the roots of quantum computing can be traced back to the 1980s, when physicists began actively exploring the potential of computing with quantum systems [1].

Quantum computing deals with the manipulation of quantum systems. The physical details of this are dependent on the quantum computer's hardware design [1].in this chapter presents the higher-level abstractions employed in quantum computing, beginning with the fundamental concept of a quantum state representing any quantum system. We will explore the building blocks of qubits and gates, Including the intriguing principles of superposition and entanglement. Furthermore, we will examine the processes of measuring quantum circuits and the pivotal role of quantum algorithms. Finally, we will introduce the powerful IBM Qiskit library, a vital tool for harnessing the potential of quantum computing.

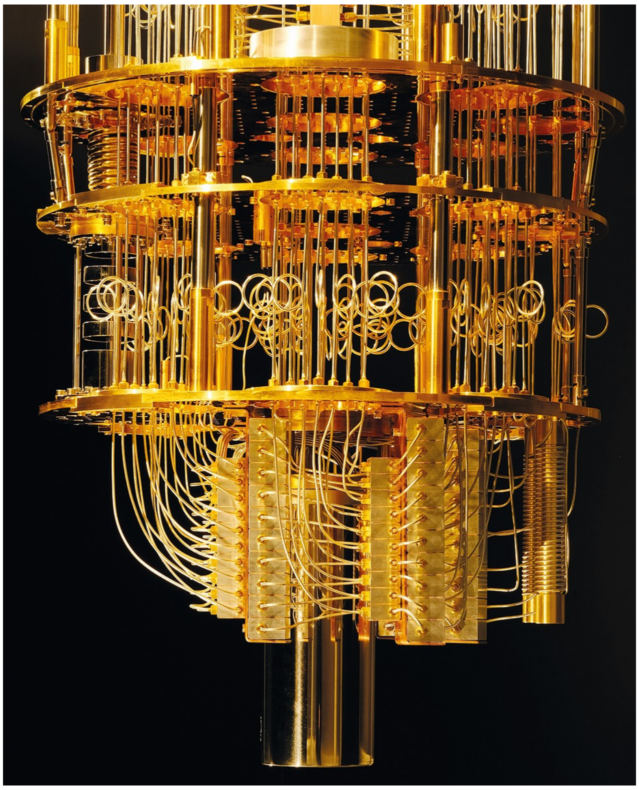


Figure 1: Real Quantum Computer IBM

1.1. History of quantum computing.

In 1982, the history of quantum computing took a significant turn with Richard Feynman's lectures on the potential advantages of utilizing quantum systems for computation[1].

Three years later, in 1985, David Deutsch published the concept of a "universal quantum computer," laying the theoretical foundation for this emerging field[1].

A breakthrough moment came in 1994 when Peter Shor presented an algorithm that could efficiently find the prime factors of large numbers, significantly outperforming classical algorithms and potentially undermining the foundations of modern encryption. This algorithm, now known as Shor's algorithm, highlighted the immense potential of quantum computing [1].

In 1996, Lov Grover introduced an algorithm for quantum computers that promised more efficient database searching, now referred to as Grover's search algorithm. That same year, Seth Lloyd proposed a quantum algorithm capable of simulating quantum-mechanical systems[1].

The late 1990s and early 2000s witnessed several significant developments, including the founding of D-Wave Systems by Geordie Rose in 1999, the development of the idea for adiabatic quantum computing by Eddie Farhi at MIT in 2000, and the first implementation of Shor's algorithm by IBM and Stanford University in 2001, factoring 15 into its prime factors on a 7-qubit processor[1].

In 2010, D-Wave released the D-Wave One, the first commercial quantum computer (annealer), marking a milestone in the commercialization of quantum computing technology[1].

IBM made quantum computing available on the IBM Cloud in 2016, further democratizing access to this powerful technology[1].

In 2019, Google claimed to have achieved quantum supremacy, a term coined by John Preskill in 2012 to describe the ability of quantum systems to perform tasks surpassing those in the classical world, signaling a significant leap forward in the capabilities of quantum computing[1].

2. Double-slit experiment:

One of the most renowned experiments in physics is the double slit experiment. Originating with Thomas Young in 1799, this peculiar demonstration reveals that tiny particles that make up matter can exhibit wavelike behavior. Even more strangely, the very act of observing these particles seems to influence their behavior[2].

Picture a wall with two narrow openings or slits. Imagine throwing tennis balls at this wall - some will bounce off, while others pass through the slits. If there is a second wall behind the first, the balls going through the slits will strike it. Where do you think the ball marks will appear on the back wall? You might expect two vertical lines matching the slit patterns[2].

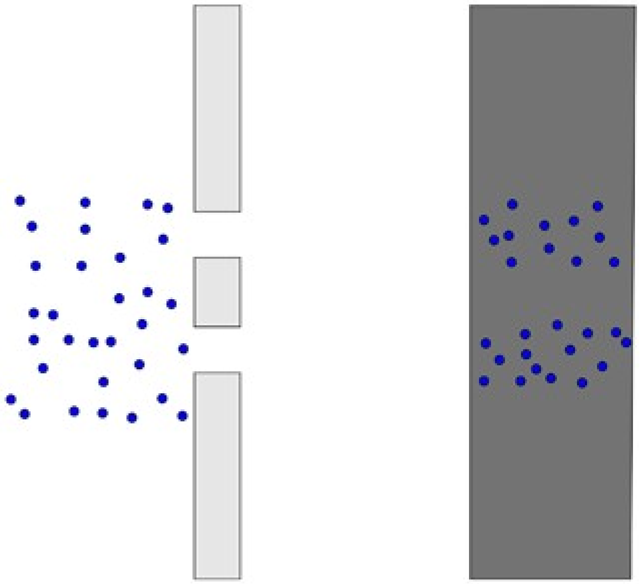


Figure 2: The pattern you get from particles. [2]

However, the diagram shows an unexpected pattern on the second wall viewed from the front. Rather than just two lines, there is an interference pattern of multiple bright and dark bars, much like the wave pattern seen when ripples from two sources overlap in water[2].

Now, envision shining a monochromatic light source (a single wavelength, like a laser) at a wall containing two narrow slits spaced approximately the same distance apart as the light's wavelength. Observe the diagram, which depicts the light wave and the wall from an overhead view, with the blue lines tracing the wave's peaks. As the wave passes through both slits, it divides into two new waves emanating from each opening, which then intermingle. At points where a peak combines with a trough, the waves cancel each other out. However, where peaks intersect with peaks (marked by the crossing blue lines), the waves reinforce each other, producing areas of increased intensity. When this pattern of light strikes a second wall placed behind the first, an alternating pattern of bright and dark bands emerges, known as an interference pattern. The bright stripes occur precisely where the wavefronts are constructively interfering and amplifying each other.[2]

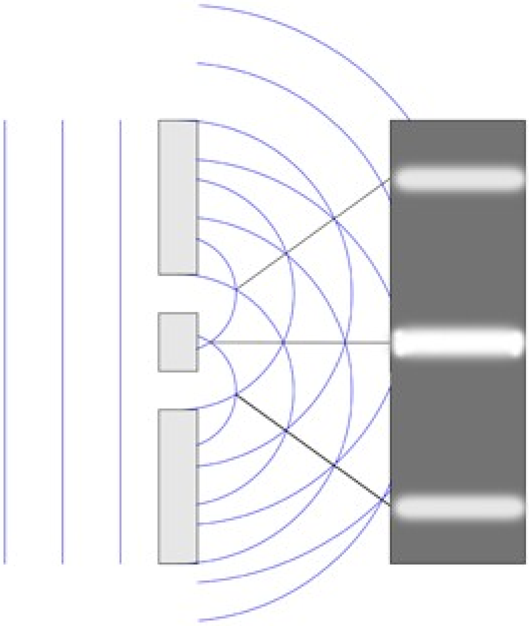


Figure 3: An interference pattern.[2]

The photograph depicts an actual interference pattern observed in experiments. It exhibits a greater number of alternating light and dark fringes compared to the simplified diagram, as the image captures more fine details. (For the sake of accuracy, the pattern also incorporates a diffraction pattern that would arise from a single slit aperture. However, we can set that nuance aside for this discussion.)[2].

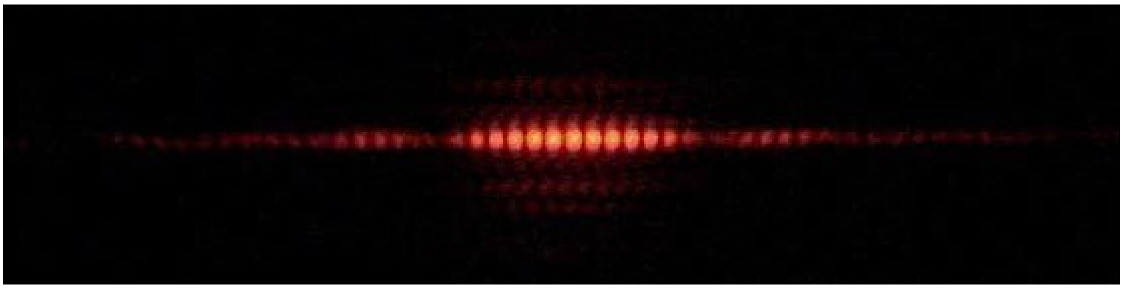


Figure 4: diffraction pattern real experiment [2]

Now, let's explore the quantum realm. Envision firing electrons at our wall containing the two slits, but initially blocking one of the openings. You'll observe that some of the electrons pass through the unobstructed slit and strike the second wall in a pattern reminiscent of the tennis ball scenario: the locations where they impact form a strip roughly matching the shape of the slit opening.[2].

Next, unblock the second slit. One might expect to see two distinct rectangular strips on the back wall, similar to the pattern observed with the tennis balls. However, the actual result is strikingly different: the locations where the electrons strike the wall accumulate in a manner that recreates the interference pattern characteristic of wave behavior. This contradicts the notion that particles like electrons would simply pass through the slits and impact separately[2].

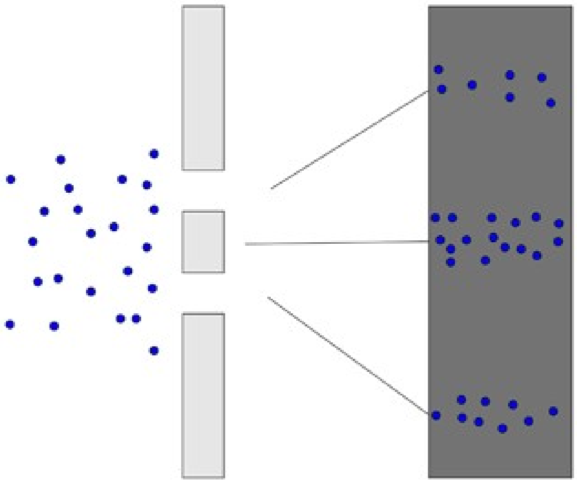


Figure 5: Diffraction pattern [2]

Fig 5

The image depicts the results from an actual double-slit experiment conducted with electrons. As an increasing number of electrons are fired through the dual slits, each successive frame captures the pattern they create upon striking the second wall. What gradually materializes is a distinctive striped interference pattern, with alternating bright and dark bands reminiscent of the wave-like behavior observed when light passes through the same setup.[2].

3. The qubits

A qubit, short for quantum bit, is the fundamental unit of information in a quantum computer system. It can be viewed as the quantum mechanical analogue to the classical binary bit used in conventional computers. More precisely, a qubit is a two-dimensional quantum system whose state can be expressed as a linear combination of two basis states as,[3].

$$|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{1}$$

In this representation, α and β are complex numbers that satisfy the condition $|\alpha|^2 + |\beta|^2 = 1$. The ket notation, also referred to as the Dirac notation, utilizes $|0> = \begin{pmatrix}1\\0\end{pmatrix}$ and $|1> = \begin{pmatrix}0\\1\end{pmatrix}$ as vectors representing the two basis states of a two-dimensional vector space commonly known as Hilbert space. According to this notation, Equation (1) expresses the state of the qubit as a two-dimensional complex vector $\begin{pmatrix}\alpha\\\beta\end{pmatrix}$[3].

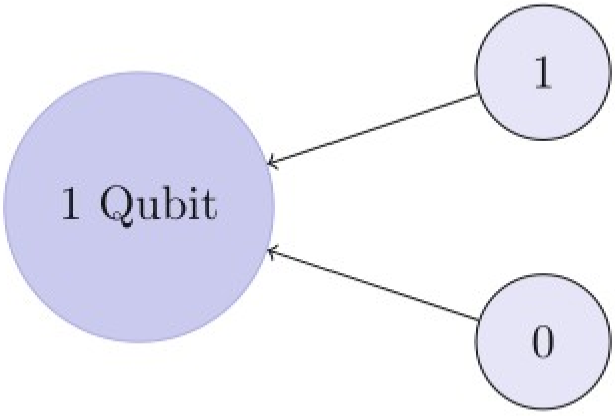


Figure 6: A classical bit can be either 0 or 1. A qubit can be in a superposition of both 0 and 1 [4]

3.1 Example:

The quantum state of a spinning coin can be represented as a superposition of the "heads" and "tails" states. Assigning the ket vector |1> to represent the "heads" state and |0> to represent the "tails" state, the quantum state of the spinning coin can be expressed as a linear combination of these two basis vectors:[4]

$$|\text{coin}> = \frac{1}{\sqrt{2}}(|1> + |0>) \quad (2)$$

What is the probability of getting heads?

The amplitude of |1> is $\beta = 1 / \sqrt{2}$, so $|\beta| = (1/\sqrt{2})^2 = 1/2$. Consequently, the probability is 0.5, equivalent to 50% [4].

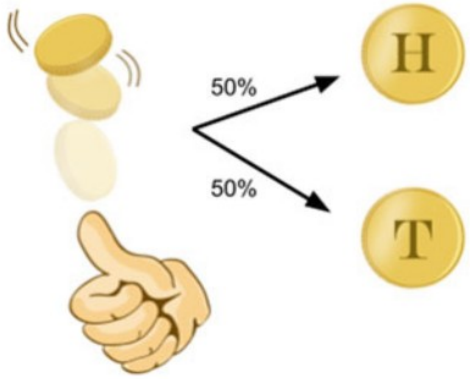


Figure 7: A tossed coin has a 50% chance of landing on heads or tails [4]

3.2. System of qubit

Quantum bits, or qubits, are the essential elements of quantum computing. Significantly, qubits can embody states of 0, 1, or a superposition of both [5]. A single qubit's characteristics can be understood through the Bloch sphere, a visual representation with geometric properties akin to the trigonometric unit circle. Each point on the Bloch sphere signifies a unique potential superposition of a single qubit. Moreover, the upper and lower points on the sphere correspond to the two observable states of the qubit, denoted as $|0\rangle$ and $|1\rangle$ [4].

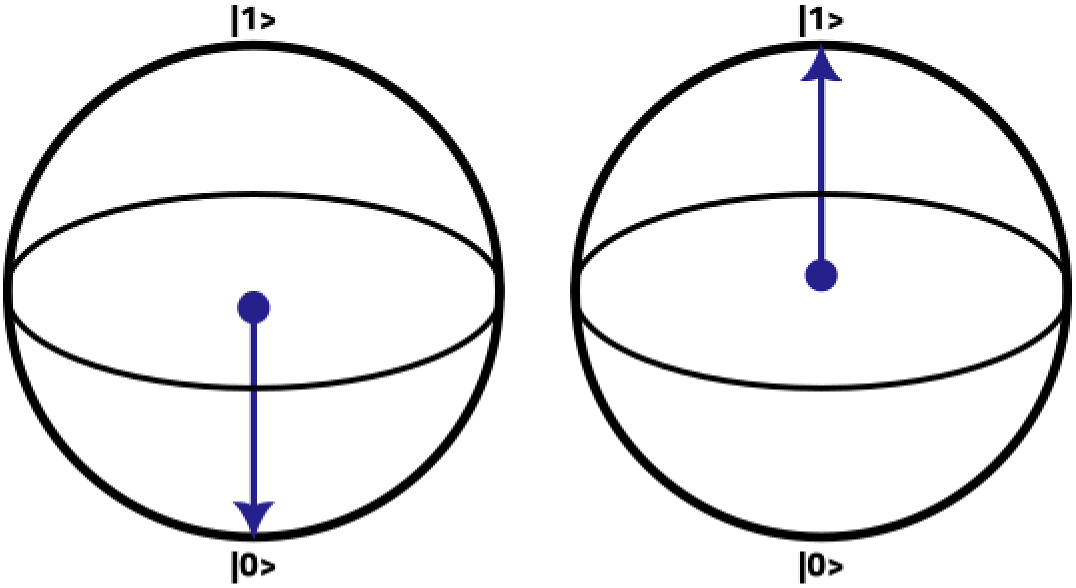


Figure 8: Quatnum Bits

A quantum computer consists of a multitude of qubits. Therefore, it is essential to understand how to construct the combined state of a qubit system based on the individual qubit states. The joint state of a qubit system is determined using the tensor product operation, denoted as ⊗. Mathematically, taking the tensor product of two states is equivalent to taking the Kronecker product of their corresponding vectors. For instance, if we have two single qubit states |ϕ> = $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$ and |ϕ'> = $\begin{pmatrix}\alpha'\\ \beta'\end{pmatrix}$, then the complete state of a system composed of two independent qubits can be expressed as[3],

$$|\phi> \otimes |\phi'>=\begin{pmatrix}\alpha\\ \beta\end{pmatrix}\otimes\begin{pmatrix}\alpha'\\ \beta'\end{pmatrix}=\begin{pmatrix}\alpha\alpha'\\ \alpha\beta'\\ \beta\alpha'\\ \beta\beta'\end{pmatrix} \tag{3}$$

In some cases, the ⊗ symbol may be omitted altogether when representing the tensor product in order to simplify the notation. Instead, the states are enclosed within a single ket. For instance, |ϕ⟩ ⊗ |ϕ ′⟩ can be abbreviated as |ϕϕ ′⟩, and |0⟩ ⊗ |0⟩ ⊗ |0⟩ can be shortened to |000⟩. When dealing with larger systems, the Dirac notation offers a more concise method for calculating the tensor product by utilizing the distributive property of the Kronecker product. For a system consisting of, let's say, three qubits where each qubit is in the state $\gamma j = \alpha j |0\rangle + \beta j |1\rangle$, for j = 1, 2, 3, the combined state is:[3]

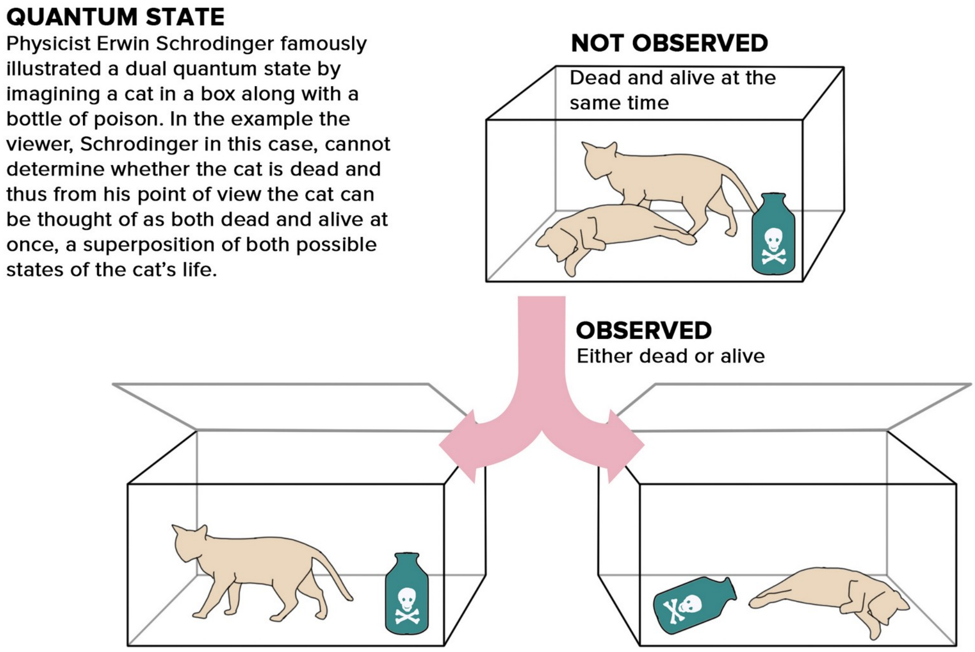
$$|\gamma 1\gamma 2\gamma 3\rangle = |\gamma 1\rangle \otimes |\gamma 2\rangle \otimes |\gamma 3\rangle$$
$$= \alpha 1\alpha 2\alpha 3 |000\rangle + \alpha 1\alpha 2 \beta 3 |001\rangle + \alpha 1 \beta 2\alpha 3 |010\rangle + \alpha 1 \beta 2 \beta 3 |011\rangle + \beta 1\alpha 2\alpha 3 |100\rangle + \beta 1\alpha 2 \beta 3 |101\rangle + \beta 1 \beta 2\alpha 3 |110\rangle + \beta 1 \beta 2 \beta 3 |111\rangle \quad (4)$$

The measurement of all three qubits has the potential to yield any of the eight (2^3) possible bit-strings, which are linked to the eight basis vectors. These examples demonstrate that the dimension of the state space increases exponentially as the number of qubits, denoted as n, grows, and the number of basis vectors is equal to 2^n. [3]

4. Superposition and entanglement.

4.1 Superposition:

Superposition, a fundamental principle in quantum mechanics, explains how quantum systems can exist in multiple states simultaneously until they are observed. In our previous discussion, we examined this idea using examples such as the double-slit experiment and coin flip. To further explore this concept, let's consider Schrӧdinger's well-known thought experiment: a cat confined within a box alongside a vial of poison. As long as the box remains closed, the cat exists in a peculiar state where it is both alive and dead, representing the intriguing nature of superposition.[6]



QUANTUM STATE
Physicist Erwin Schrodinger famously illustrated a dual quantum state by imagining a cat in a box along with a bottle of poison. In the example the viewer, Schrodinger in this case, cannot determine whether the cat is dead and thus from his point of view the cat can be thought of as both dead and alive at once, a superposition of both possible states of the cat's life.

Figure 9: Erwin Schrodinger illustrated of Quantum superposition [6]

4.1. Entanglement

Quantum entanglement is a natural occurrence that arises when multiple qubits exhibit correlation. This entanglement can lead to peculiar and beneficial outcomes, potentially enabling quantum computers to outperform classical computers in terms of speed. By entangling qubits, intricate quantum information is unveiled, a feature absent in the realm of classical computing. Such entanglement stands as a key asset of the quantum domain![4]

It was discovered that their states can be characterized by a two-dimensional complex vector, while the observables and evolution operators are represented by 2×2 matrices. However, what happens when we consider two photons, two electrons, or two atoms? Or even three?[7]

One illustration of the peculiar nature of entanglement can be seen with the scenario of two fair coins. In a classical setting, after flipping two fair coins multiple times, you would observe the results HH, HT, TH, or TT, each happening with a 25% chance.[4]

By entangling these two fair coins through quantum means, it becomes feasible to generate a state represented as (1/ √2)(|H H> + |T T>) as depicted in Figure 10.[4]

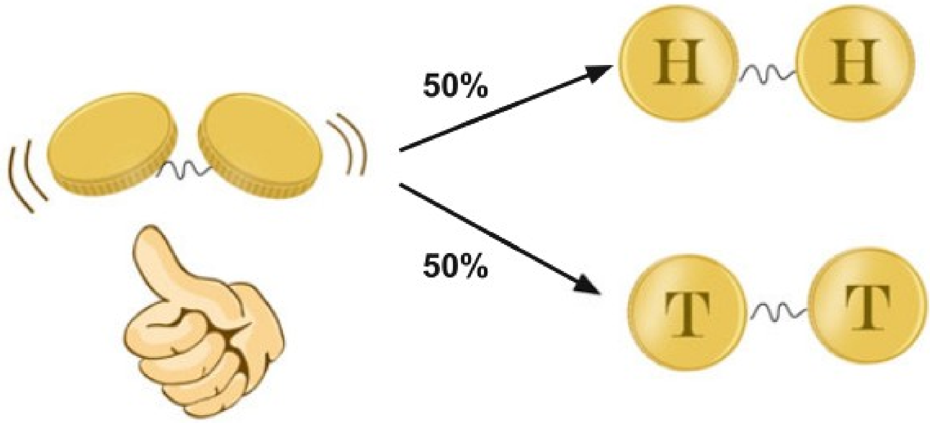
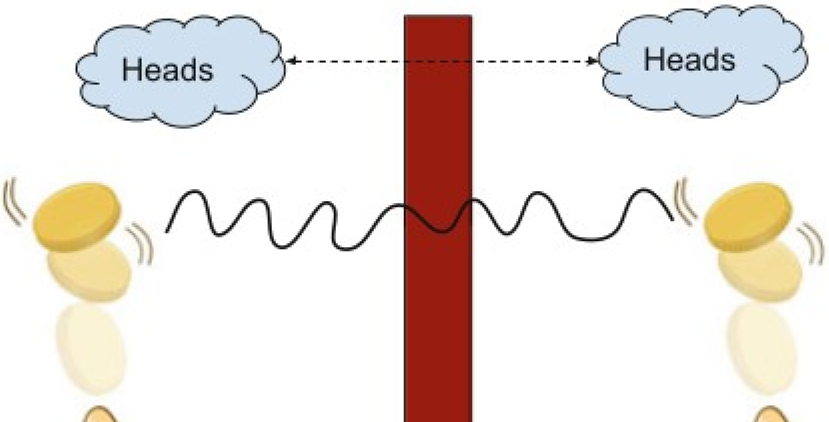


Figure 10: Two coins that are entangled [4]

There exist numerous other varieties of entangled states, however, a well-known illustration is the Bell state. In the case of this entangled pair of coins being flipped, they are intertwined in a manner that permits only two potential measurement results: (1) both coins landing on heads, or (2) both coins landing on tails, each outcome having an equal probability of 50%. The combinations HT or TH would never be observed[4].

Additionally, in the scenario where the two entangled coins are placed far apart, one coin can be flipped and its outcome measured. If the measured coin shows heads, it indicates that the other coin will also land on heads. Conversely, if the measured coin shows tails, it means that the other coin will also land on tails. This phenomenon hints at the possibility of instantaneous transmission of information between the two coins, potentially surpassing the speed of light, which is considered the fastest speed in the Universe.In accordance with Figure 11, should the two coins be flipped simultaneously, they mysteriously manage to land on the same side as each other despite the absence of any classical communication between them.[4]

Let us



examine a scenario involving two electrons, assuming that they are sufficiently distant from each other to be identified as "electron 1" and "electron 2" without any confusion. Expressing the state of electron 1 can be done effortlessly as [7]:

$$|\psi>_1 = a|\uparrow>_1 + b|\downarrow>_1 \text{ with } |a|^2 + |b|^2 = 1. \tag{5}$$

To emphasize our reference to electron 1, we incorporate a subscript "1". Likewise, we can express the state of electron 2 [7]:

$$|\varphi>_2 = c|\uparrow>_2 + d|\downarrow>_2 \text{ with } |c|^2 + |d|^2 = 1. \tag{6}$$

We're asking about the condition of the combined system comprising these two electrons.

The states of the combined system can be generated by utilizing the states of the separate systems. Within the ket, the symbols and writing serve as convenient labels for measurement results. Consequently, when the spin of each electron in the z-direction is measured, there are four possible measurement outcomes [7]:

$$\begin{gathered}|\text{electron } 1 = \uparrow, \text{ electron } 2 = \uparrow>,\\ |\text{electron } 1 = \uparrow, \text{ electron } 2 = \downarrow>,\\ |\text{electron } 1 = \downarrow, \text{ electron } 2 = \uparrow>,\\ |\text{electron } 1 = \downarrow, \text{ electron } 2 = \downarrow>.\end{gathered} \tag{7}$$

This procedure is not particularly convenient, hence, as an alternative, we could represent it as [7]:

$$|\uparrow_1, \uparrow_2> , |\uparrow_1, \downarrow_2> , |\downarrow_1, \uparrow_2> , |\downarrow_1, \downarrow_2>. \tag{8}$$

According to convention, the arrangement of the upward ↑ and downward ↓ arrows is predetermined, and the comma is unnecessary. This can be further streamlined to [7]:

$$|\uparrow\uparrow> , |\uparrow\downarrow> , |\downarrow\uparrow> , |\downarrow\downarrow>. \tag{9}$$

The four quantum states are meaningful when measuring Sz on individual electrons. In quantum mechanics, superpositions of these states can be created [7]:

$$|\psi> = 1/2|\uparrow\uparrow> + 1/2|\uparrow\downarrow> + 1/2|\downarrow\uparrow> + 1/2|\downarrow\downarrow>. \tag{10}$$

Now, let us take into account that the electrons are in the states mentioned in (6.1) and (6.2). How can we represent this using the states from (6.5)? Initially, we analyze the probabilities of measuring Sz for each electron. Due to the complete independence of the electrons, their probabilities multiply [7]:

$$\begin{gathered} \Pr(\uparrow\uparrow) = \Pr(\uparrow_1) \times \Pr(\uparrow_2) = |a|^2\,|c|^2 \\ \Pr(\uparrow\downarrow) = \Pr(\uparrow_1) \times \Pr(\downarrow_2) = |a|^2\,|d|^2 \\ \Pr(\downarrow\uparrow) = \Pr(\downarrow_1) \times \Pr(\uparrow_2) = |b|^2\,|c|^2 \\ \Pr(\downarrow\downarrow) = \Pr(\downarrow_1) \times \Pr(\downarrow_2) = |b|^2\,|d|^2 \end{gathered} \tag{11}$$

It is easy to verify that the probabilities add up to one[8]:

$$\Pr(\uparrow\uparrow) + \Pr(\uparrow\downarrow) + \Pr(\downarrow\uparrow) + \Pr(\downarrow\downarrow) = 1. \tag{12}$$

The spin state of two electrons that aligns with these probabilities is [7]:

$$|\psi> = ac|\uparrow\uparrow> + ad|\uparrow\downarrow> + bc|\downarrow\uparrow> + bd|\downarrow\downarrow>. \tag{13}$$

However, this is merely the result of the two spin states [7]:

$$\begin{gathered} |\psi>1\ |\varphi>2 = (a|\uparrow>_1 + b|\downarrow>_1)(c|\uparrow>_2 + d|\downarrow>_2) \\ \equiv ac|\uparrow\uparrow> + ad|\uparrow\downarrow> + bc|\downarrow\uparrow> + bd|\downarrow\downarrow> \end{gathered} \tag{14}$$

Consequently, the merging of two quantum systems can be accomplished by merging their states into a composite quantum system through the multiplication of states, as illustrated above, while keeping the symbol order in the ket intact. Subsequently, it can be proven that the state in (6.6) is identical to two electrons, each in the state [7]:

$$|\psi> = |\phi> = 1/\sqrt{2}\,|\uparrow> + 1/\sqrt{2}\,|\downarrow>. \tag{15}$$

5. Quantum gate

A quantum gate, also known as a quantum logic gate, is a basic quantum circuit that functions on a limited amount of Qubits.

On the other hand, quantum logic involves different quantum gates such as the Feynman and Peres gates, along with the Toffoli gate, among others, as detailed in Table 1. These gates play a crucial role in building quantum circuits. The 2 × 2 Feynman gate, also known as the CNOT gate, behaves similarly to the XOR logic gate but with added features to guarantee reversibility. [9]

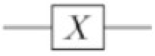
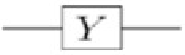
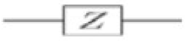
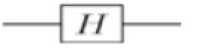
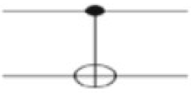
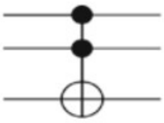
Name of the gates	Gate symbol	Matrix
Pauli-X (X) (NOT)		$\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$
Hadamard (H)		$1/\sqrt{2}\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix}1000\\0100\\0001\\0010\end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix}10000000\\01000000\\00100000\\00010000\\00001000\\00000100\\00000001\\00000010\end{bmatrix}$

Table 1: Basic quantum gates.

5.1. The most used gates.

Let's get what any of this gate do:

There are three Pauli gates available: X, Y, and Z. According to Eq. 16, each gate changes the notation of the Pauli matrices as follows [8]:

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

(16)

The Pauli gates operate on a solitary qubit and are capable of generating superposed qubit states. The Pauli X gate functions similarly to the classical NOT gate on the basis states. In other words, it transforms the state |0> into |1> and vice versa[8].

$$X|0> = |1>; X|1> = |0>$$

The CNOT gate is equivalent to the XOR classical gate illustrated in Figure 18, which is identical to the classical NOT gate. Unitary rotation matrices, applied to a single qubit, are constructed using the Pauli gates. This is based on the information provided in reference [8]

$$X^2 = Y^2 = Z^2 = I \qquad (17)$$

yields

$$Rx(\theta) = \exp i\theta X = \cos(\theta)I + i \sin\theta X$$
$$R y(\theta) = \exp i\theta Y = \cos(\theta)I + i \sin\theta Y$$
$$Rz(\theta) = \exp i\theta Z = \cos(\theta)I + i \sin\theta Z$$

(18)

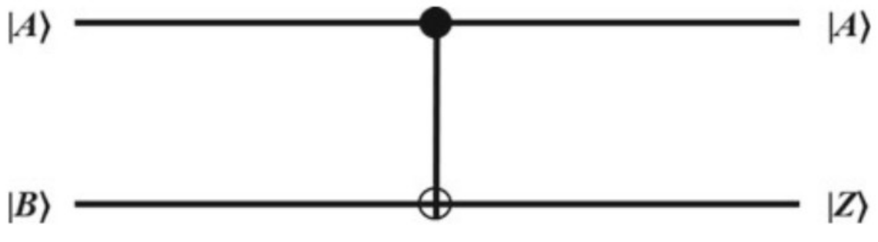


Figure 12: CNOT quantum gate.[4]

The Hadamard gate holds significant importance in the field of quantum computing. When a qubit is initially in a definite |0> or |1> state, the application of the Hadamard gate results in a superposition of both |0> and |1> states. In Figure 16, we demonstrate the utilization of a Hadamard gate on the qubit in the |0> state using the IBM Q simulator, followed by the measurement of the output [4].

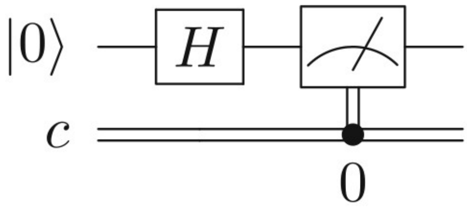


Figure 13: Applying a Hadamard gate and measuring on the IBM Q machine

6. Measurements

The final quantum state is prepared from the initial input quantum state through the utilization of qubits and gates, as discussed in the previous sections. However, the process of computation requires an additional crucial step known as quantum measurement. This section focuses on quantum mechanics[9], specifically the transformation of quantum information stored in a quantum system into classical information. For instance, measuring a qubit usually involves determining whether it is 0 or 1, which corresponds to reading out a classical bit. It is important to note that measurement outcomes in quantum mechanics are probabilistic in nature [3].

In accordance with the given notation for inner products, the probability of observing the state $|0\rangle$ after measurement for the single qubit state mentioned in Equation (1) can be expressed as the squared absolute value of the overlap $|\langle 0|\phi\rangle|^2$ Similarly, the probability of obtaining the state $|1\rangle$ after measurement is $|\langle 1|\phi\rangle|^2$. Therefore, the probabilities associated with measurements can be represented by the squared absolute values of the overlaps. Expanding on this concept, when measuring an n qubit state $|\phi\rangle$, the probability of obtaining the bit string $|x_1 . . . x_n\rangle$ is given by $|\langle x_1 . . . x_n |\phi\rangle|^2$ [3].

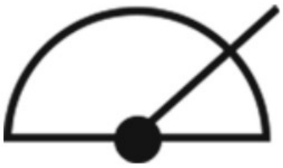


Figure 14: Irreversible measurement gate.[10]

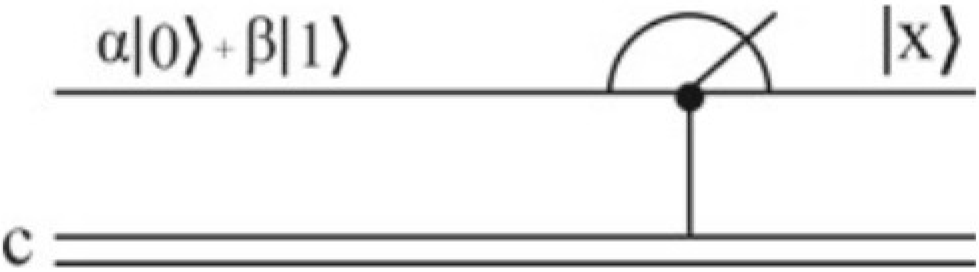


Figure 15: Measurement of a single qubit [10]

Now let's examine a slightly more intricate scenario involving measurement. Let's assume we have a three qubit state, denoted as |ψ ⟩ but we choose to measure only the first qubit while keeping the other two qubits unaffected. What is the likelihood of observing a |0⟩ in the first qubit? This probability can be determined by:[3]

$$\sum_{(x_2 x_3)\in\{0,1\}} \left|\langle 0\, x_2\, x_3|\phi\rangle\right|^2$$

The system's state following this measurement will be determined through the normalization of the state,[3]

$$\sum_{(x_2 x_3)\in\{0,1\}} \langle 0\, x_2\, x_3|\phi\rangle \mid 0\, x_2\, x_3 >$$

When we apply this framework to the state in Eq. (5), it becomes evident that the likelihood of obtaining |0⟩ in the initial qubit will be 0.5. In the event that this outcome is achieved, the system's ultimate state would then transition to |000⟩. [3] Alternatively, if we were to measure the state of |1⟩ in the initial qubit, the resulting state would be |111⟩. [3]

We can also calculate how subsystem measurements impact a state composed of n qubits. There are instances where measurements must be conducted using a basis other than the computational basis. To accomplish this, it is necessary to apply a suitable transformation to the qubit register prior to measurement. Further information on how to perform this task is provided in a later section that covers observables and expectation values.[3]

Measurement is a fundamental concept in quantum computing, which is often more intricate than it appears. For a more in-depth comprehension of advanced concepts in this area, we suggest delving into pertinent books and articles [3, 5, 9, 11].

7. Quantum circuits.

We have encountered a few basic quantum circuits previously. Let's delve deeper into the components of a quantum circuit[11].In this section, we will explain the construction and interpretation of quantum circuits[3]. An illustration of a simple quantum circuit with three quantum gates can be seen in Figure 19. The circuit should be interpreted from left to right, with each line representing a wire within the quantum circuit. These wires may not necessarily correspond to physical wires; they could symbolize the passage of time or the movement of a physical particle like a photon from one point to another in space[11]. Qubits are typically depicted as horizontal lines [3], It is customary to assume that the initial state inputted into the circuit is a computational basis state, often the state comprising all |0>s. While this rule is occasionally disregarded in quantum computation and information literature, it is considered good practice to notify the reader when this occurs[11]. Gates are then depicted on the qubits they operate on.[3].

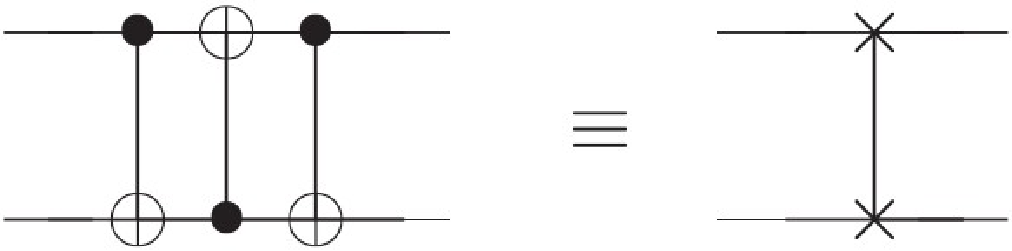


Figure 16: Circuit swapping two qubits, and an equivalent schematic symbol notation for this common and useful circuit. [3]

It is important to observe that when formulating a mathematical expression for the circuit, the gates are listed from right to left according to their sequence of operation.[3].

The principles mentioned above can be effectively demonstrated through an example. Figure 20 showcases a circuit that is utilized to create an entangled two qubit state known as a Bell state starting from the initial state |00⟩[3].

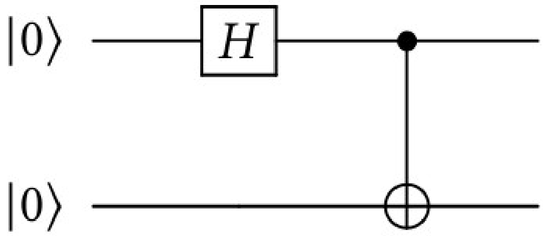


Figure 17: Quantum circuit for preparing a Bell state

The circuit encodes the equation[3],

$$CNOT_{12}(H\otimes I)|00>=\frac{1}{\sqrt{2}}(|00>+|11>)$$

Now, let us meticulously examine the process by which the circuit generates the Bell state. The circuit is read in a sequential manner, from left to right. The qubits are assigned numerical labels, beginning with the topmost qubit. Initially, the H gate is applied to the highest qubit, resulting in a modification of the system's state.[3],

$$(H\otimes I)|00>=(H|0>)\otimes(I|0>)=\frac{(|0>+|1>)}{\sqrt{2}}=\frac{1}{\sqrt{2}}(|00>+|10>)$$

Subsequently, $CNOT_{12}$ operates on both of these qubits. The filled-in dot on the first qubit indicates that this qubit serves as the control qubit for the CNOT operation.

The ⊕ symbol on the second qubit indicates that this qubit is the target of the NOT gate (which is controlled by the state of the first qubit). The result of the CNOT operation is then obtained.[3],

$$CNOT_{12}\left(\frac{1}{\sqrt{2}}(|00>+|11>)\right)|10>=\frac{1}{\sqrt{2}}(CNOT_{12}|00>+CNOT_{12}|10>)=\frac{1}{\sqrt{2}}(|00>+|11>)$$

A qubit's measurement is also represented by a unique gate featuring a meter symbol, as shown in Figure 21. The inclusion of this gate on a qubit indicates that the qubit should be measured in the computational basis.[3].

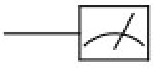


Figure 18: The measurement gate

There are certain characteristics that differentiate classical circuits from quantum circuits. Firstly, quantum circuits are designed to be acyclic, meaning that feedback from one part of the circuit to another is not permitted. In contrast, classical circuits allow for the joining of single wire outputs through an operation called FANIN, resulting in a bitwise OR of the inputs. However, this operation is irreversible and non-unitary, making it unsuitable for quantum circuits. Additionally, the inverse operation, FANOUT, which produces multiple copies of a bit, is also prohibited in quantum circuits. Quantum mechanics dictates that the copying of operations is impossible, and we will explore an example of this in the upcoming qubit, rendering the FANOUT operation infeasible.[11]

8. Quantum algorithms.

We have now provided an introduction to all the fundamental elements essential for the exploration of practical quantum algorithms.[3].

8.2 Quantum algorithms

A quantum algorithm consists of three basic steps[3]:

- The data can be encoded, either classically or quantumly, into the state of a group of input qubits[3].

- A series of quantum gates implemented on this group of input qubits[3].
- At the conclusion, one or more of the qubits are measured to acquire a result that can be interpreted in a classical manner[3].

In this review, we will elucidate the execution of these three stages for a diverse range of quantum algorithms[3].

8.2.1 Deutsh's Algorithm

Deutsch's quantum algorithm, as explained in this section, necessitates just one invocation of a black box for Uf in order to solve the problem. Conversely, any classical algorithm mandates two invocations of a classical black box for Cf, with one for each input value. The crucial aspect of Deutsch's algorithm lies in its nonclassical capability to place the second qubit of the input into a superposition within the black box. [12]

The quantum circuit illustrating the Deutsch algorithm can be found in Figure 22 It is important to highlight that a key component of the Deutsch algorithm is the oracle Uf gate, depicted in Figure 22.b.[8],

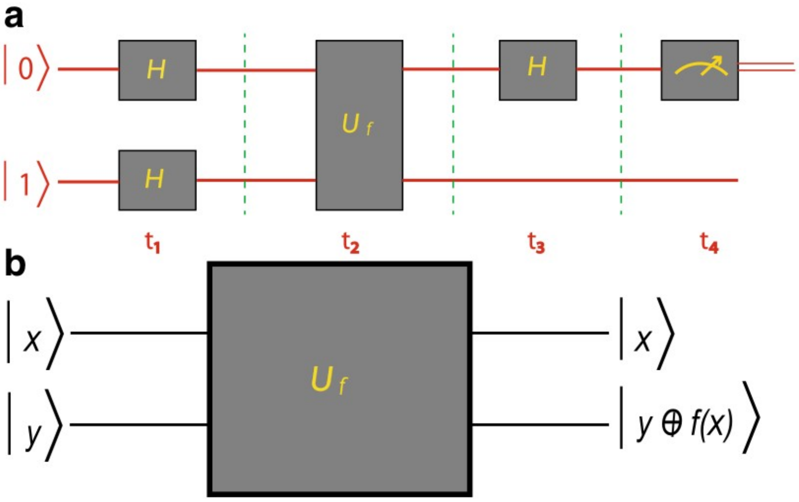


Figure 19: Quantum circuit for the Deutsch algorithm. b The action of the U f gate, also called the oracle. [10]

In order to minimize the number of queries to the function f, which takes a single bit as input and produces a single bit as output, we must ascertain whether f is constant (always yielding the same output) or balanced (returning 0 for one input and 1 for the other). [13]

By utilizing the circuit depicted in Figure 22.a, Deutsch's problem can be resolved through a single inquiry to the oracle. In the event that f is constant, the outcome will be 0; however, if f is balanced, the result will be 1. This assertion can be readily substantiated by considering the state immediately prior to the activation of the oracle gate. [13].

When the input is |x>|y>, Uf will generate |x>|f (x) ⊕ y>, Therefore, if |y> = | 0>, t, applying Uf will result in |x>|f (x)>. The algorithm applies Uf to the two-qubit state |+> |−>, where the first qubit is a superposition of the two values in the domain of f, and the third qubit is in the superposition |−> = √12 (|0> − |1>). This yields the following outcome.

$$U_f(|->|+>)=U_f\left(\frac{1}{2}(|0>+|1>)(|0>-|1>)\right)$$

$$\text{¿}\frac{1}{2}(|0>(|0\oplus f(0)>-|1\oplus f(0)>))+|1>(|0\oplus f(1)>-|0\oplus f(1)>)$$

When $f(x)=0,\frac{1}{\sqrt{2}}\text{¿}$ becomes $\frac{1}{\sqrt{2}}\text{¿}$ when $f(x)=1,\frac{1}{\sqrt{2}}\text{¿}$ becomes $\frac{1}{\sqrt{2}}\text{¿}$ Therefore

$U_f\text{¿}$[12].

When f is constant, the value of (−1)f (x) iis simply a global phase that holds no physical meaning. Therefore, the state can be represented as |+>|−>. On the other hand, when f is not constant, the term (-1)f (x) negates one of the terms in the superposition, resulting in the state |−>|−>. up to a global phase. By applying the Hadamard transformation H to the first qubit and measuring it, we can determine the outcome with certainty. In the first case, we obtain |0> and in the second case, we obtain |1>. This allows us to determine whether f is constant or not with certainty by making a single call to Uf. This marks our first example of a quantum algorithm that surpasses any classical algorithm in performance. [12]

Readers may find it surprising that this algorithm achieves success with absolute certainty. Quantum mechanics is commonly associated with its probabilistic nature, leading people to mistakenly assume that anything involving quantum means must be probabilistic. Furthermore, there is a tendency to believe that anything displaying distinct quantum properties must also be probabilistic. However, our study of quantum analogs to classical computations has already revealed that the first expectation is not valid. The algorithm for Deutsch's problem serves as evidence that even processes that are inherently quantum can defy probabilistic behavior. [12]

8.2.2 Advanced Algorithms

There are advanced algorithms you can see and search about:

- Deutsch-Jozsa’s algorithm
- Grover’s algorithm
- Shor’s algorithm
- Variational algorithms

9. IBM Qiskit.

Qiskit is a quantum computing library that has been developed by IBM as an open-source project. It provides users with the ability to write and execute programs on either IBM's quantum processors or a local simulator, eliminating the need for a graphical interface. This is particularly advantageous when dealing with a large number of qubits, as the graphical interface becomes less practical. Currently, Qiskit allows users to access quantum processors with up to 16 qubits, although smaller processors are also available. Qiskit is a highly robust software development kit (SDK) that consists of multiple components designed to address various challenges associated with practical quantum computing. These components, known as Terra, Aer, Aqua, and Ignis, are responsible for different aspects of quantum software development. In this section, we will provide a brief overview of how to program simple quantum circuits using Qiskit. For a more comprehensive understanding of Qiskit and its extensive capabilities, we recommend visiting the official website at www.qiskit.org. [12]

For our objectives, Qiskit can be considered as a Python library utilized for executing quantum circuits. A typical Qiskit code consists of two main components: circuit design and execution. During the circuit design stage, we instantiate a QuantumCircuit object with the necessary number of qubits and classical bits. Subsequently, gates and measurements are incorporated into this initialized circuit. Gates and measurements are executed in Qiskit as functions of the QuantumCircuit class. Once the circuit is designed, we must select a backend for circuit execution. This can either be a simulator known as the qasm_simulator or one of IBM's quantum processors. To utilize a quantum processor, it is essential to input your IBM Q account details into Qiskit. Illustrated in Figure 24 is a straightforward code for constructing the Bell state. This code represents the Qiskit rendition of the circuit depicted in Figure 23, with an additional measurement at the conclusion to validate our outcomes. [3]

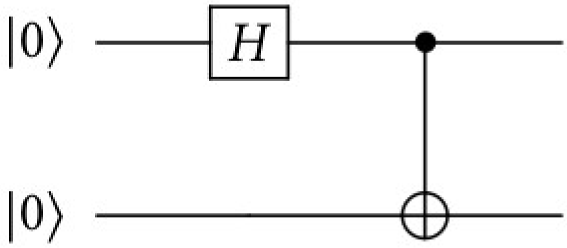
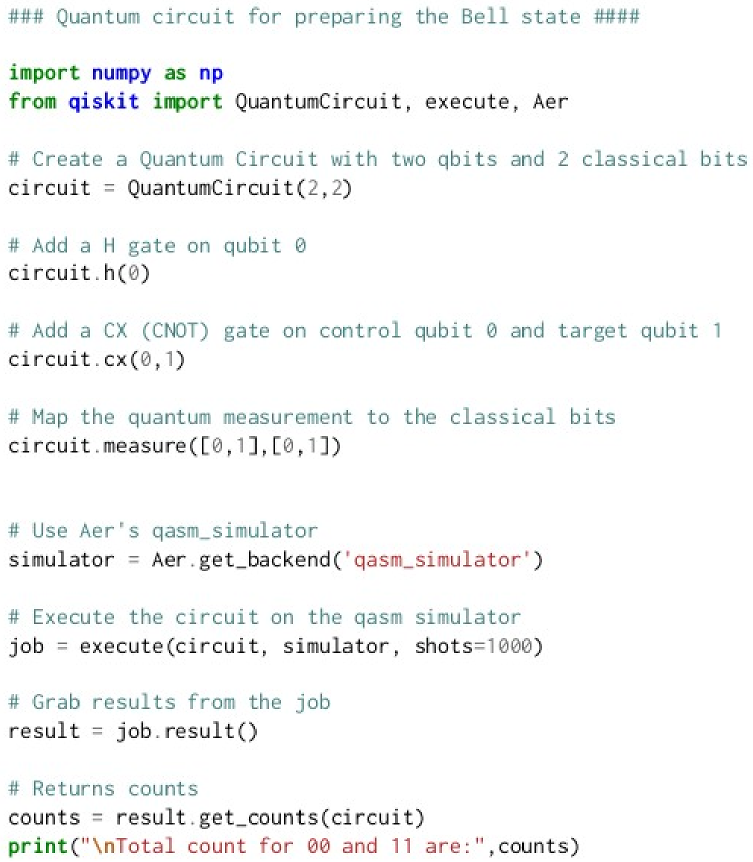


Figure 20: Quantum circuit for preparing a Bell state



```
### Quantum circuit for preparing the Bell state ####

import numpy as np
from qiskit import QuantumCircuit, execute, Aer

# Create a Quantum Circuit with two qbits and 2 classical bits
circuit = QuantumCircuit(2,2)

# Add a H gate on qubit 0
circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0,1)

# Map the quantum measurement to the classical bits
circuit.measure([0,1],[0,1])


# Use Aer's qasm_simulator
simulator = Aer.get_backend('qasm_simulator')

# Execute the circuit on the qasm simulator
job = execute(circuit, simulator, shots=1000)

# Grab results from the job
result = job.result()

# Returns counts
counts = result.get_counts(circuit)
print("\nTotal count for 00 and 11 are:",counts)
```

Figure 21: Qiskit code to create and measure a Bell state. Source: www.qiskit.org

10. Quantum computer platform

The next few years will witness a significant transformation in various sectors due to the advent of quantum computing. However, the availability of commercially viable quantum computers is extremely limited, and the high-performance models are exorbitantly priced for most businesses. Additionally, these advanced machines often require bulky refrigeration units, further adding to the cost and complexity. To address these challenges, cloud-based quantum computing has emerged as a promising solution. [23]

It offers developers, researchers, and businesses a convenient platform to develop and test quantum algorithms using real quantum computers or simulators accessible through the cloud. This type of service, known as quantum-as-a-service (QaaS), is provided by several major IT companies as well as a few smaller firms. While corporate users may find the access costly, some providers offer free access to researchers, fostering innovation and collaboration in the quantum computing field [23], and this are the most popular Quantum computer platform in the market now:

- IBM Quantum
- Google Quantum AI
- Amazon Bracket
- Microsoft Azure Quantum
- Alibaba Cloud
- D-Wave Leap
- Xanadu Cloud
- QuTech Inspire

11. Conclusion.

This chapter introduced us to the exciting field of quantum computing programming. We learned about the history of quantum computing and its key components like qubits, gates, and circuits. Understanding these basics helps us use powerful quantum algorithms and tools such as IBM Qiskit. As we finish this chapter, it's clear that quantum computing has the potential to revolutionize how we approach complex problems and push the limits of technology. Quantum computing opens up new possibilities for solving challenges that were previously unsolvable.

Chapter II: Quantum Machine Learning.

1. Introduction

While machine learning algorithms are used to compute immense quantities of data, quantum machine learning utilizes qubits and quantum operations or specialized quantum systems to improve computational speed and data storage done by algorithms in a program [24].

Following the emergence of quantum computers, a faction of computer scientists began experimenting with running machine learning algorithms on these quantum platforms. Within this chapter, we delve into the realm of Quantum Machine Learning (QML), exploring its algorithms, and various models within the field.

2. Classical Machine learning

Two primary avenues exist for machines to learn: learning from data and learning through interaction. There are four broad categories of learning methods: supervised learning, unsupervised learning, reinforcement learning, and deep learning [25, 26]. The theory underlying machine learning is a pivotal subdiscipline that draws from both artificial intelligence and statistics, with roots extending back to the earliest work on artificial neural networks and AI in the 1950s [27, 28]. In 1959, Arthur Samuel offered a famous definition of machine learning as "the field of study that gives computers the ability to learn without being explicitly programmed" [29].

Within the theory of machine learning, the concept of "learning" is generally categorized into three types (refer to Figure 25) that encompass the broad range of approaches: supervised learning, unsupervised learning, and reinforcement learning. [29]

In supervised learning, a machine outputs a function from a training set of labeled data points. The goal of supervised learning is to determine the relationship between the input and output and to predict the output for the new data or the input values. The prediction probability distribution function (PPDCL) is composed of three steps: model selection, model learning, and inference. In unsupervised learning, the algorithm is provided with data that lacks labels. The training set consists of input values, and the objective is to uncover hidden patterns within the unlabeled information derived from the input data. Clustering is one example of unsupervised problems. Dimensionality reduction involves a three-step process in unsupervised

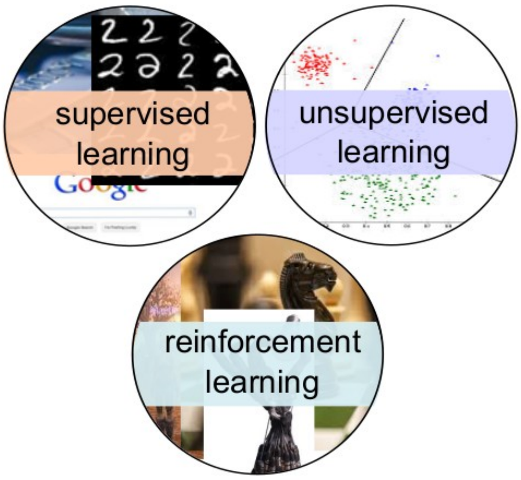


Figure 22: The three types of classical learning [29]

learning, which includes model selection, learning, and the generation of new samples [5, 8]. Reinforcement learning serves as an intermediary approach between supervised and unsupervised learning due to the absence of an immediate correct output for a given input, yet it still involves a form of guidance. Instead of obtaining the desired output for every input, the algorithm receives feedback from the environment. This feedback plays a crucial role in assessing the impact of the chosen steps on the output, whether they have been beneficial or detrimental [8].

3. Quantum Machine Learning Algorithm.

Quantum computing involves the manipulation of quantum systems to carry out information processing. By harnessing the unique property of quantum states to exist in a superposition, it becomes possible to significantly enhance the speed and complexity of computations.[29], Quantum Machine Learning, or QML, is a unique quantum software application that combines quantum hardware architectures with classical and quantum machine learning algorithms [33]. Quantum machine learning involves the development of quantum algorithms to address common machine learning problems by applying the power of quantum computing [29]. Quantum machine algorithms are primarily executed through supervised and unsupervised learning methods. Within the realm of quantum clustering, the utilization of quantum Lloyd's algorithm proves instrumental in addressing k-means clustering challenges [34], The centroid distance in the cluster is calculated using a repetitive method. Quantum algorithms offer a speed advantage over traditional machine learning algorithms in accelerating this process. [32]

3.1 Quantum neural network (QNN)

Quantum neural networks have emerged as a novel advancement that exploits the principles of quantum mechanics to execute specific computations with greater efficiency compared to classical neural networks. These networks are built upon the foundation of quantum computing, a cutting-edge paradigm that employs quantum bits (Qubits) to encode information and carry out computations [35]

Quantum neural networks aim to utilize quantum entanglement and superposition for information processing in a distinct manner. These networks possess the capability to execute specific tasks at a significantly faster rate compared to classical neural networks, particularly for challenges involving extensive parallelism or intricate quantum phenomena.[35]

3.1.1 Basics of Quantum Neural Networks

1- Quantum Bits (Qubits)

2- Quantum Gates

3- Quantum Circuits

4- Quantum Neurons

5- Quantum Layer and Architecture

6- Quantum Training

7- Hybrid Approaches

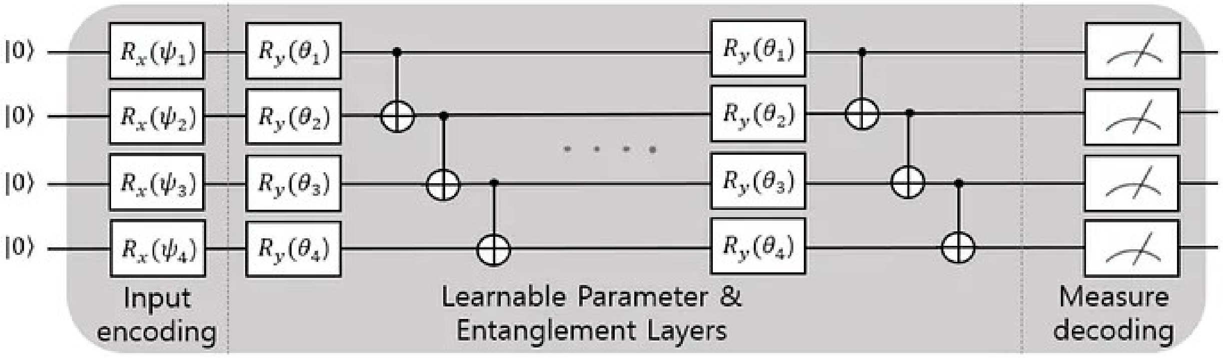
8- Applications

3.1.2 Quantum Circuit Architectures for QNNs

Quantum circuit architectures are essential components in the development and deployment of quantum neural networks (QNNs), which are machine learning models that harness the principles of quantum mechanics for specific purposes. The architecture of a quantum circuit determines how qubits are interconnected and the specific operations performed on them. Here are a few commonly used quantum circuit architectures for QNNs:[35]

- Variational Quantum Circuit (VQC)
- Quantum Convolutional Neural Network (QCNN)

- Quantum Recurrent Neural Network (QRNN)
- Quantum Boltzmann Machine (QBM)
- Quantum Autoencoder (QAE)



Fig

ure 23: Ouantum vs. Classical Neural Networks[36]

Quantum computers hold great power because they are capable of faster computation than their classical counterparts. On that lens, it is important to analyze whether quantum neural networks hold a similar power. IBM describes how the capabilities of a neural network can be described through “effective dimension”, a measure of how useful and non-redundant the neural network is. Then, it was found that the quantum neural networks produced noticeably higher effective dimensions and produced lower loss quicker. Even though there is so much more we don’t know about how quantum neural networks might perform comparatively, these results are promising — just like in other fields of computation, machine learning could be by the power of quantum. [36]

3.2 Quantum support vector machine (QSVM)

In this study, we present two Support Vector Machine (SVM) classifiers that operate on classical data and leverage the quantum state space as the feature space, thereby achieving a quantum advantage. we use a non-linear mapping of the data to a quantum state represented as $\Phi : \vec{x} \in \Omega \rightarrow | \Phi(\vec{x})><\Phi(\vec{x})|$, c.f. Figure 25(a).[41]

We employ both classifiers on a superconducting quantum processor. In the initial method, we utilize a variational circuit, as described in references [37, 38, 39, 40], to generate a separating hyperplane within the quantum feature space. Conversely, in the second approach, we leverage the quantum computer to directly estimate the kernel function of the quantum feature space and subsequently implement a conventional Support Vector Machine (SVM). Achieving a quantum advantage in either approach crucially hinges on the inability to classically estimate the kernel. This remains valid even when employing intricate variational quantum circuits as classifiers. The experiment's main focus is to dissociate the feasibility of hardware implementation of the classifier from the challenge of choosing a suitable feature map for practical datasets. The dataset in question is deliberately structured to enable 100% successful classification as a means of verifying the methodology. [41]

The experimental setup comprises five interconnected superconducting transmons, with only a pair being utilized for the purposes of this study, as illustrated in Figure 29(a). Two co-planar waveguide (CPW) resonators serve as quantum buses to facilitate connectivity within the device. Furthermore, each qubit is equipped with an extra CPW resonator dedicated to control and readout functions. The entanglement within our system is established through CNOT gates, leveraging cross-resonance interactions [42] Additionally, individual qubit operations are utilized as basic building blocks. The quantum processing unit is connected to the mixing chamber plate of a dilution refrigerator for thermal stability. [41]

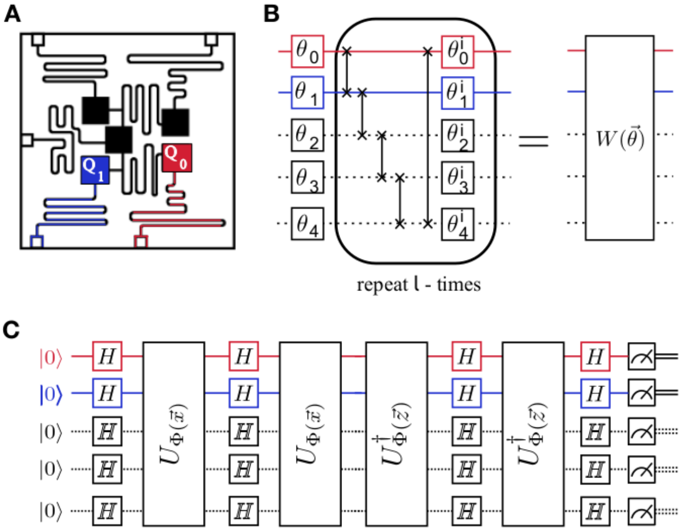


Figure 25: Experimental implementations[41]

FIG. 2. Experimental implementations

3.2.1 Quantum feature map:

The concept of a feature map can be defined as follows: Consider a Hilbert space denoted as F, which is referred to as the feature space. Let X represent an input set, and x be a sample taken from this input set. A feature map is essentially a mapping function φ : X → F that transforms inputs into vectors within the Hilbert space. These resulting vectors, denoted as φ(x) ∈ F are commonly known as feature vectors. [43] Feature maps are of great significance in the field of machine learning as they facilitate the transformation of various types of input data into a space that possesses a clearly defined metric. Typically, this space has a significantly higher dimension compared to the original input data. When the feature map is a nonlinear function, it alters the relative positioning of data points, as illustrated in Figure 27. Consequently, this transformation can greatly simplify the classification of datasets within the feature space. It is worth noting that feature maps are closely linked to kernels [44].

We will demonstrate that classifiers utilizing quantum circuits, like the one depicted in Figure 29(c), do not offer a quantum advantage compared to a traditional support vector machine when the feature vector kernel $K(\vec{x}, \vec{z}) = | < \Phi(\vec{x}) | \Phi(\vec{z}) > |^2$ is overly simplistic. For instance, a classifier employing a feature map that solely produces product states can be readily implemented classically. To surpass classical

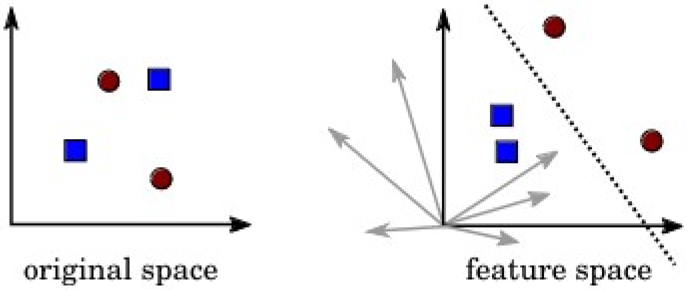


Figure 26: Mapping data points from original space to higher-dimensional feature space for better separability.[44]

methods, it is necessary to employ a map based on circuits that are challenging to simulate classically. Given that quantum computers are not expected to be classically simulable, there exists a wide array of (universal) circuit families to select from. In this context, we suggest utilizing a circuit that performs effectively in our experiments and is not excessively deep. We define a feature map on n-qubits created by the unitary $U_{\Phi}(\vec{x}) = U_{\Phi}(\vec{x})\ H^{\otimes n}\ U_{\Phi}(\vec{x})\ H^{\otimes n}$, where H represents the conventional Hadamard gate. [41]

$$U_{\Phi(\vec{x})}=\exp\left(i\sum_{S\subseteq[n]}\varphi S(\vec{x})\right)\prod_{i\in s}Z_i$$

The gate represented by a diagonal matrix in the Pauli Z - basis, as shown in Figure 1 (b), will operate on the initial state | 0>. The data is encoded using the coefficients $\varphi_S(\vec{x}) \in$ R, to $\vec{x}\in\Omega$. A diagonal unitary $U_{\Phi}(\vec{x})$ can be utilized if it can be implemented efficiently. This is particularly applicable when dealing with interactions of weight $|S| \leq 2$. Computing the inner-product between two states produced by a circuit with a single diagonal layer $U_{\Phi}(\vec{x})$ is #P - hard [45]. However, within the context of additive error approximation in experiments, it is possible to efficiently simulate a single layer preparation circuit using uniform sampling in classical computing [46], Calculating inner products using circuits that involve two basis transformations and diagonal gates is considered challenging, especially when accounting for potential additive errors. For further details, please refer to the supplementary material.[41]

3.2.2 Quantum kernel estimation:

The second classification method solely relies on the utilization of a quantum computer to approximate the kernel $K(\vec{x}_i , \vec{x}_j) = | < \Phi(\vec{x}_i) | \Phi(\vec{x}_j) > |^2$ or all the labeled training data $\vec{x}_j \in T$. Subsequently, the classical optimization problem, is employed to acquire the optimal Lagrange multipliers α_i and support vectors N_S.These support vectors N_s enable the construction of the classifier, as indicated in equation (21). In order to apply the classifier to a new datum $\vec{s} \in S$, it is necessary to estimate the kernel $K(\vec{x}_i , \vec{s})$ between $\vec{s}$ and the support vectors in $i \in N_S$ We explore two approaches to estimate this overlap within our specific framework.[41]

$$\widetilde{m}(\vec{x}) = sign\left(\sum_{i \in N_s} \alpha_i y_j K(\vec{x}_i, \vec{s}) + b\right) \qquad (21)$$

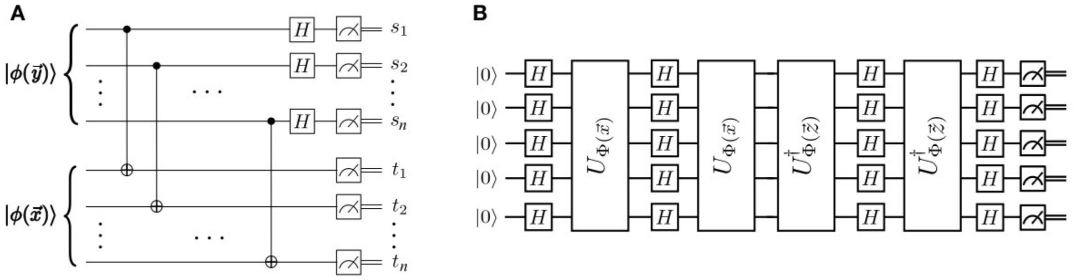


Figure 27: SWAP-Matrix Expectation and Fidelity Estimation

The swap test is commonly employed to estimate the fidelity between two states [47]. This particular circuit, though, does not belong to the category of short depth circuits on a quantum computing architecture with geometrically local gates. It necessitates a series of controlled SWAP operations, commonly referred to as Fredkin gates, all contingent upon the state of a single ancilla qubit. A highly commendable protocol was recently formulated in [48]. The conventional swap test has been extensively studied by the authors, who have discovered various techniques for optimizing its performance. In the context of our algorithm, where only the fidelity value is required, the authors suggest a circuit design that maintains a constant depth.

This proposal is detailed in section III.C of reference [23], and it relies on the parallel execution of pairs of CNOT gates [41].

This particular circuit, as shown in Figure 31.a, calculates the expectation value < ψ | <φ |SWAP| ψ>|φ> in a direct manner. The algorithm's operation can be comprehended in the following manner:[41]

The SWAP gate is recognized as both a unitary gate and a Hermitian observable, denoted as $SWAP^{\dagger} = SWAP$ with eigenvalues ±1. The expectation value when acting on two product states is given by <ψ|<φ |SWAP| ψ>| φ> = | < φ | ψ > $|^2$. This gate can be broken down into a product of two qubit swap gates, SWAP = $\prod_{k=1}^{n}$ =1 SWAP$_{\text{sktk}}$ all operating in parallel. [41]

In order to calculate the expectation value, it is necessary to diagonalize the complete gate. This can be achieved by diagonalizing the individual two-qubit swap gates. It is observed that SWAP$_{ij}$ = CNOT$_{i\to j}$ CNOT$_{j\to i}$ CNOT$_{i\to j}$. Additionally, by utilizing the circuit identity CNOT$_{j\to i}$ = H$_j$ CZ$_{ji}$ H$_j$, it can be deduced that SWAP$_{ij}$ can be diagonalized by CNOT$_{j\to i}$ H$_j$ and possesses eigenvalue (−1)x$_i$ x$_j$. For the entire circuit depicted in Figure S5.a, the initial step involves applying a transversal set of CNOT gates across both registers. This is followed by a single layer of Hadamard gates H on the top register. Subsequently, the output is sampled and the average of the boolean function is calculated.[41]

$$f(s,t)=(-1)^{(s_1t_1+\ldots+s_nt_n)} \quad (22)$$

The results are documented. The bits produced on the upper register are denoted by s ∈ $\{0, 1\}^n$, whereas t ∈ $\{0, 1\}^n$ represents the output sequence on the lower register.[41]

$$|\,\Phi_{\vec{x}}(\vec{x})>¿U_{\Phi(\vec{x})}H^{\otimes n}U_{\Phi(\vec{x})}H^{\otimes n}\,|\,0>¿^{\otimes n}¿ \text{ Where } U_{\Phi(\vec{x})}=\exp\left(i\sum_{S\subseteq[n]}\phi s(\vec{x})\prod_{i\in S}Z_i\right) \quad (23)$$

The effectiveness of this approach is applicable to any input states | ψ>, | φ> Nevertheless, our states possess a specific structure and are exclusively produced by the unitary equation (23) illustrated in Figure 31. Explicitly expressing the kernel as $K(\vec{y},\vec{x})=\left|\left\langle\Phi(\vec{y})\middle|\Phi(\vec{x})\right\rangle\right|^2=\left|\left\langle 0^n\middle|U^{\dagger}_{\Phi(\vec{y})}U_{\Phi(\vec{y})}\middle|0^n\right\rangle\right|^n$ The method of measuring can be determined by applying the circuit to $U^{\dagger}_{\Phi(\vec{y})}U_{\Phi(\vec{y})}$ to the state |0^n>, as shown in Fig 31.b. Subsequently, the resulting state $U^{\dagger}_{\Phi(\vec{y})}U_{\Phi(\vec{y})}$ |0^n> should be sampled R times in the Z basis. By recording the number of observed zero (0, . . . , 0) bit-strings and dividing it by the total number of shots R, the frequency $\nu_{(0,\ldots,0)}$ = #{(0, . . . , 0)}R^{-1} provides an estimator for K($\vec{y}$, $\vec{x}$) with a sampling error $\tilde{\varepsilon}=O\left(R^{-1/2}\right)$.[41]

An approximate estimation for the operator norm || · || of the discrepancy between the resulting estimator K̂ and the true kernel matrix K can be determined by || K − K̂|| ≤ ||K − K̂||$_F$. In this context, $\|A\|_F=\sqrt{\sum_{ij}|A_{ij}|^2}$ represents the Frobenius norm of matrix A. By considering the largest sampling error $\tilde{\epsilon}$ among all matrix entries and setting $||K - \hat{K}||_F \leq \tilde{\epsilon}|T|$, where both matrices of the training set T have dimensions $|T| \times |T|$, a rough upper bound can be obtained. Consequently, in order to ensure a maximum deviation of ϵ with a high level of confidence, a total of $R = O(\epsilon^{-2} |T|^2)$ shots need to be drawn for each matrix entry. Taking into account the symmetry of the K matrix and the trivial diagonals, it is necessary to estimate $|T|(|T| - 1)2^{-1}$ matrix entries. Therefore, the overall sampling complexity is expected to scale as $O(\epsilon^{-2} |T|^4)$. For a more meticulous analysis of the statistical error, one could employ one of the matrix-concentration results [49].

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \vec{x}_i \circ \vec{x}_j \qquad (24)$$

The optimization problem, equation (24), is solely concave if the matrix $K \geq 0$ is positive semi-definite. It is possible that shot noise and other errors in the experiment can result in a $\hat{K}$ that is no longer positive semi-definite. This occurrence has been observed multiple times in our experiments. One potential approach to address this issue is a technique outlined in [50], where an optimization problem is utilized to identify the nearest positive semi-definite K-matrix in trace norm to $\hat{K}$ that adheres to the constraint. Nevertheless, in our experiments, we have determined that this method is unnecessary as the performance has been nearly optimal even without its implementation.[41]

3.2.3 Quantum Variational classification

The initial classification procedure consists of four sequential stages. Initially, the data $\vec{x} \in \Omega$ undergoes a transformation into a quantum state through the utilization of the feature map circuit $U_\Phi(\vec{x})$ depicted in Figure 28(b) with respect to a reference state $| 0>^n$. In addition, the feature state undergoes the application of a brief depth quantum circuit $W(\vec{\theta})$, as depicted in Figure 29(b). This circuit, consisting of l layers, is characterized by the parameter $\vec{\theta} \in R^{2n(l+1)}$, which will be subject to optimization during the training process. Subsequently, a binary measurement $\{M_y\}$ is performed on the state $W(\vec{\theta}) U_{\Phi(\vec{\theta})}$ $|0>^n$ for the purpose of addressing a two-label classification problem, where $y \in \{+1, -1\}$. The measurement is conducted through measurements in the Z-basis, where the resulting bit-string $z \in \{0, 1\}^n$ is inputted into a selected boolean function f : {0, 1}^{n} → {+1, −1}. The measurement operator is represented as M$_y$ = 2−1 (1 + yf), where we have defined $f=\sum_{z\in\{0,1\}^n} f(z)|z>¿z|$. The probability of observing outcome y is denoted as $p_y(\vec{x})=\langle\Phi(\vec{x})|W^{\dagger}(\vec{\theta})M_y W(\vec{\theta})|\Phi(\vec{x})\rangle$. Next, in order to determine the decision rule, R repeated measurement shots are conducted to acquire the empirical distribution p̂$_y$ ($\vec{x}$). A label m̃($\vec{x}$) = yis assigned p̂$_y$ ($\vec{x}$) > p̂$_{-y}$ ($\vec{x}$) − y$_{b.}$ Here, a bias parameter b ∈ [−1, 1] is introduced, which is subject to optimization during the training process.[41]

The circuit $U_{\Phi(\vec{x})}$ for the feature map, along with the boolean classifier, allows us to optimize the parameters ($\vec{\theta}$, b). In order to carry out this optimization process, it is necessary to establish a cost function. We define the empirical risk R$_{emp}$($\vec{\theta}$) as the error probability Pr (m̃($\vec{x}$) ≠ m($\vec{x}$)), which represents the average incorrect label assignment across the samples in the training set T.[41]

$$R_{emp}(\vec{\theta})=\frac{1}{|T|}\sum_{\vec{x}\in T} Pr(\widetilde{m}(\vec{x})\neq m(\vec{x}))$$

In the case of binary classification, the probability of error in assigning the incorrect label can be determined using the binomial cumulative density function (CDF) of the empirical distribution p̂$_y$ ($\vec{x}$). Refer to the supplementary material for the detailed derivation. When the number of samples (shots) R is significantly large R >> 1, the binomial CDF can be approximated by a sigmoid function sig(x) = (1 + e−x) −1.. The approximate probability of incorrectly assigning the label m($\vec{x}$) = y can be calculated using this approximation.[41]

$$Pr(\widetilde{m}(\vec{x})\neq m(\vec{x}))\approx sig\left(\frac{\sqrt{R}\left(\frac{1}{2}-\left(\widetilde{p_y}(\vec{x})-\frac{yb}{2}\right)\right)}{\sqrt{2\left(1-\widetilde{p_y}(\vec{x})\right)\widetilde{p_y}(\vec{x})}}\right)$$

The experiment consists of two phases: initially, the classifier is trained and optimized ($\vec{\theta}$, b)Spall's SPSA [51, 52] stochastic gradient descent algorithm has demonstrated excellent performance in the presence of noise in experimental conditions. Once the parameters have converged to ($\vec{\theta}$*, b*) , the circuit can be utilized as a classifier. Subsequently, during the categorization stage, the classifier allocates tags to unmarked data $\vec{s}\in S$ based on the decision rule m̃($\vec{s}$).[41]

The quantum variational classifier W $(\vec{\theta})$ is implemented in our superconducting quantum processor, with 5 different depths (l = 0 through l = 4), as depicted in Figure 29(b). It is anticipated that increasing the depth will result in a higher success rate for classification. The binary measurement is derived from the parity function f = Z_1 Z_2 . For each depth, we train three distinct data sets, with each training set comprising 20 data points per label. Figure 30(b) showcases one of these data sets, along with the corresponding training set used. Figure 30(a) illustrates the optimization empirical risk R_{emp} $(\vec{\theta})$ for two different training sets and depths. In all experiments conducted in this study, we employed a technique for error mitigation that relies on first-order zero-noise extrapolation [53, 50] To obtain an estimate with zero noise, we executed a duplicate of the circuit on a time scale that was slowed down by a factor of 1.5, as described in the supplemental material. This technique is applied at each trial step, and it is the mitigated cost function that is utilized by the classical optimizer. The empirical risk shown in Figure 30(a) demonstrates convergence to a lower value for depth l = 4 compared to l = 0, albeit with a greater number of optimization steps. While error mitigation does not significantly enhance the results for depth 0 - as the noise in our system is not the limiting factor in that scenario-, it does provide substantial improvement for larger depths. Although Pr($\tilde{m}(\vec{x})$ ≠ m $(\vec{x})$) explicitly in-larager depths. Although Pr($\tilde{m}(\vec{x})$ ≠ m $(\vec{x})$) explicitly accounts for the number of experimental shots taken, we set R = 200 to avoid gradient problems, despite having taken 2000 shots in the actual experiment.[41]

Upon completion of each training session, we utilize the trained set of parameters ($\vec{\theta}$*, b* = 0) to classify 20 distinct test sets. These test sets are randomly selected for each data set. The classification experiments are conducted with 10,000 shots, which is significantly higher than the 2,000 shots used during training. To ensure accuracy, the classification of each data point is error-mitigated and repeated twice. The success ratios obtained from each of the two classifications are then averaged. In Figure 29(c), we present the classification results obtained through our quantum variational approach. It is evident that the classification success improves as the circuit depth increases, as depicted in Figure 30(c). For depths greater than 1, the success rates approach values close to 100%. Remarkably, this high classification success is maintained up to depth 4, despite the presence of decoherence caused by the 8 CNOTs in both the training and classification circuits, for l = 4 [41].

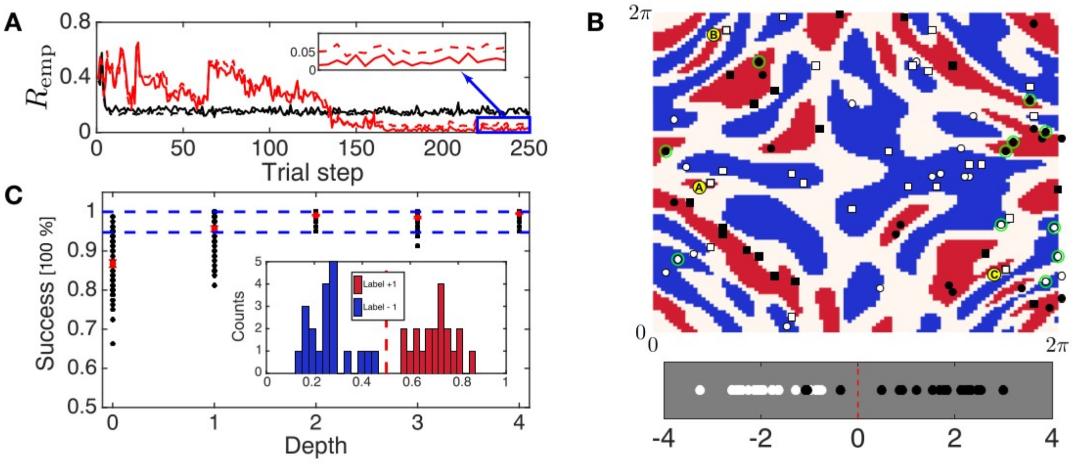


Figure 28: Converaence of the method and classification results [41]

3.2.4 What is the advantage of QSVM over the classical SVM?

Quantum support vector machines (QSVMs) offer several potential advantages over classical support vector machines (SVMs), primarily due to the inherent properties of quantum computation. Some of these advantages include:

1. **Increased computational power:**

Quantum computers can potentially perform certain types of calculations much faster than classical computers. QSVMs leverage quantum algorithms, such as quantum parallelism and quantum superposition, to process information in parallel, which can lead to faster classification times for large datasets.

2. **Ability to handle high-dimensional data efficiently:**

Quantum computers are well suited for processing high-dimensional data due to their inherent parallelism and the ability to represent and manipulate large amounts of information simultaneously. This can be particularly advantageous for tasks where the number of features or dimensions is very high, such as in natural language processing or image recognition.

3. **Potential for improved generalization:**

QSVMs may offer improved generalization performance compared to classical SVMs in certain scenarios. Quantum algorithms could potentially find more optimal hyperplanes for separating classes in high-dimensional feature spaces, leading to better generalization to unseen data.

4. **Capability for quantum data representation:**

QSVMs can exploit quantum states to represent data, which can provide a richer and more expressive representation compared to classical data encoding methods. This could potentially lead to better classification performance, especially for certain types of quantum data or problems where quantum effects play a significant role.

5. **Opportunities for quantum speedup:**

While quantum speedup is not guaranteed for all problems, QSVMs offer the potential for significant speedup in certain cases, particularly for specific quantum algorithms and problem instances. This could enable QSVMs to outperform classical SVMs for certain types of classification tasks.

3.2.5 Steps for Developing a QSVM Model:

Building an effective quantum SVM (QSVM) model involves several crucial steps, taking into account the specific challenges and considerations related to quantum machine learning:

1. Quantum Data Preparation
2. Quantum Kernel Selection
3. Quantum Learning Algorithm Selection
4. Quantum QSVM Model Training
5. Model Evaluation
6. Model Tuning and Improvement
7. Model Deployment and Usage

3.2.6 Types of Quantum Kernels for QSVMs:

In quantum support vector machines (QSVMs), kernels play a crucial role by allowing an implicit projection of quantum states into an infinite-dimensional feature space. Choosing the appropriate quantum kernel is essential for the performance of the quantum SVM model.

Here are some of the most common types of quantum kernels:

3.2.6.1. State Overlap Kernel:

The state overlap kernel calculates the similarity between two quantum states by measuring their overlap.

It is simple to implement and interpret, but may not effectively capture non-linear relationships between quantum states.

Formula: K(ψ1, ψ2) = ⟨ψ1 | ψ2⟩ where ψ1 and ψ2 are quantum states.

3.2.6.2. Fidelity Kernel:

The fidelity kernel calculates the fidelity between two quantum states, measuring their proximity in terms of trace distance.

It is more robust to quantum noise than the state overlap kernel and can capture more complex non-linear relationships.

Formula: K(ψ1, ψ2) = F(ψ1, ψ2)^2 where F(ψ1, ψ2) is the fidelity between the quantum states ψ1 and ψ2.

3.2.6.3. Trace Distance Kernel:

The trace distance kernel calculates the trace distance between two quantum states, measuring their difference in terms of the trace of density matrices.

It is even more robust to quantum noise than the fidelity kernel and can capture even more complex non-linear relationships.

Formula: K(ψ1, ψ2) = 1 - ||Tr(ρ1ρ2)||_1 where ρ1 and ρ2 are the density matrices of the quantum states ψ1 and ψ2, and ||.||_1 is the trace norm.

3.2.6.4. Kernels Based on Quantum Circuits:

More advanced quantum kernels can be constructed using specific quantum circuits.

These kernels can capture complex non-linear relationships by exploiting the power of quantum transformations.

Examples: kernels based on quantum gates, kernels based on quantum measurements.

3.2.6.5. Hybrid Kernels:

Hybrid kernels can be combined from different types of quantum kernels to leverage their respective strengths.

This can allow capturing a wide range of non-linear relationships in quantum data.

3.2.6.6 Choosing the Quantum Kernel:

The choice of the appropriate quantum kernel depends on the nature of the quantum data, the classification task, and the available resources.d the available resources.

3.2.7 Data Representation for QSVM Learning:

3.2.7.1 Vector Representation:

Each data point is transformed into a vector in a feature space. The dimension depends on the number of features of the data. Feature extraction techniques are used for the transformation.

3.2.7.2 Kernel Representation:

A kernel function is used to calculate the similarity between data points. Projection into an infinite-dimensional feature space. Crucial choice of the kernel function for performance.

3.2.7.3 Constraint Representation:

Learning a hyperplane to separate the classes. Minimization of a quadratic cost function under constraints. Constraints for the correct ranking of support points.

3.2.7.4 Important Considerations:

Normalization of data for a uniform scale. Feature selection for the most relevant ones. Appropriate choice of kernel according to the data and task.

3.2.7.5 Hilbert Space:

Crucial role in QSVM theory, but no explicit representation of data. Internal workings of the kernel and mathematical properties. Hilbert space to define geometric concepts in infinite dimensions. Dot product and linear algebra techniques for learning.

3.2.7.6 In summary:

Vector or kernel representation for QSVM learning. Choice of representation and parameters tailored to the problem. Hilbert space important for kernel theory and properties.

4. Validation model

There exist various model validation techniques, and the selection of the appropriate one relies on the nature of your data and the objectives you aim to accomplish with your machine learning model. The following are the prevailing model validation techniques [54]:

1. Train and Test Split or Holdout
2. Resubstitution
3. K-Fold Cross-Validation
4. Random Subsampling
5. Bootstrapping
6. Nested Cross-Validation

5. Performance Metrics

Classification metrics evaluate the effectiveness of machine learning algorithms in classifying data into distinct categories. Their primary objective is to allocate a given data point to a specific predefined category [55]:

1. Accuracy

$$Accuracy=\frac{TP+TN}{TP+FP+TN+FN}$$

2. Confusion Matrix
3. Precision and Recall

$$Precision=\frac{TP}{TP+TF}$$

$$Recall=\frac{TP}{TP+FN}$$

4. F1-score

$$F1\,Score=\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}=\frac{2*Precision*Recall}{Precision+Recall}$$

5. Area Under the Receiver Operating Characteristic Curve (AU-ROC)

6. Conclusion

In summary, quantum machine learning tries to use quantum computers to make machine learning algorithms better. This chapter looked at quantum versions of popular algorithms like support vector machines (QSVM). It discussed how to run these quantum algorithms on real quantum hardware like superconducting qubits. Experiments showed that using deeper, more complex quantum circuits improved the performance of the QSVM, allowing it to correctly classify data points with near 100%

accuracy in some cases. However, to truly get better performance than classical computers, the quantum algorithms need to implement feature maps that are very hard for normal computers to simulate. The chapter also covered ways to validate and test quantum machine learning models, like cross-validation. It explained metrics used to measure a model's performance, such as accuracy, precision, and recall. While quantum machine learning is still a new field, it has the potential to enhance machine learning capabilities in the future when large, powerful quantum computers are available.

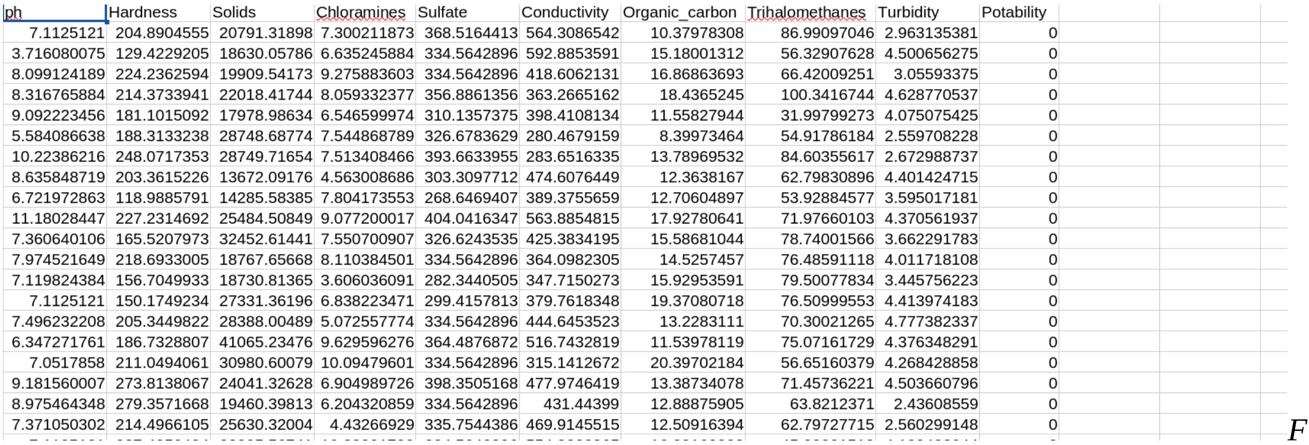
Chapter III: The Water quality Classification Model

1. Introduction

In the previous chapter, we delved into the fundamentals of machine learning and quantum computing. We modeled several QNN circuits as our initial algorithm, and then attempted to use Qiskit's QSVM, which encountered some issues with the library. Moving forward, we will apply these concepts to a practical example: a water quality model. This model will utilize the VQC (variational quantum classification) method, as detailed in Chapter II.3.2.3. We will streamline our dataset and implement the model using the VQC library from Qiskit, programming with Qiskit circuits and Python. Additionally, we will discuss traditional machine learning libraries, their advancements, and the challenges we face.

2. Dataset Description and Preprocessing Phase

We obtained one datasets, one containing 3277 observations [56], sourced from Kaggle. In order to clean the data, we need to remove rows with missing variables. The dataset is illustrated in Figure 33. These datasets differ in terms of the number of rows, columns, and the volume of data they contain.[57]



ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
7.1125121	204.8904555	20791.31898	7.300211873	368.5164413	564.3086542	10.37978308	86.99097046	2.963135381	0
3.716080075	129.4229205	18630.05786	6.635245884	334.5642896	592.8853591	15.18001312	56.32907628	4.500656275	0
8.099124189	224.2362594	19909.54173	9.275883603	334.5642896	418.6062131	16.86863693	66.42009251	3.05593375	0
8.316765884	214.3733941	22018.41744	8.059332377	356.8861356	363.2665162	18.4365245	100.3416744	4.628770537	0
9.092223456	181.1015092	17978.98634	6.546599974	310.1357375	398.4108134	11.55827944	31.99799273	4.075075425	0
5.584086638	188.3133238	28748.68774	7.544868789	326.6783629	280.4679159	8.39973464	54.91786184	2.559708228	0
10.22386216	248.0717353	28749.71654	7.513408466	393.6633955	283.6516335	13.78969532	84.60355617	2.672988737	0
8.635848719	203.3615226	13672.09176	4.563008686	303.3097712	474.6076449	12.3638167	62.79830896	4.401424715	0
6.721972863	118.9885791	14285.58385	7.804173553	268.6469407	389.3755659	12.70604897	53.92884577	3.595017181	0
11.18028447	227.2314692	25484.50849	9.077200017	404.0416347	563.8854815	17.92780641	71.97660103	4.370561937	0
7.360640106	165.5207973	32452.61441	7.550700907	326.6243535	425.3834195	15.58681044	78.74001566	3.662291783	0
7.974521649	218.6933005	18767.65668	8.110384501	334.5642896	364.0982305	14.5257457	76.48591118	4.011718108	0
7.119824384	156.7049933	18730.81365	3.606036091	282.3440505	347.7150273	15.92953591	79.50077834	3.445756223	0
7.1125121	150.1749234	27331.36196	6.838223471	299.4157813	379.7618348	19.37080718	76.50999553	4.413974183	0
7.496232208	205.3449822	28388.00489	5.072557774	334.5642896	444.6453523	13.2283111	70.30021265	4.777382337	0
6.347271761	186.7328807	41065.23476	9.629596276	364.4876872	516.7432819	11.53978119	75.07161729	4.376348291	0
7.0517858	211.0494061	30980.60079	10.09479601	334.5642896	315.1412672	20.39702184	56.65160379	4.268428858	0
9.181560007	273.8138067	24041.32628	6.904989726	398.3505168	477.9746419	13.38734078	71.45736221	4.503660796	0
8.975464348	279.3571668	19460.39813	6.204320859	334.5642896	431.44399	12.88875905	63.8212371	2.43608559	0
7.371050302	214.4966105	25630.32004	4.43266929	335.7544386	469.9145515	12.50916394	62.79727715	2.560299148	0

Figure 29: Dataset is illustrated

3. Data Encoding Phase

In the encoding phase of the proposed model, the classical data that has been processed is converted into a quantum format that can be utilized as input for quantum algorithms. To accomplish this, one commonly employed method is through the utilization of feature maps, with the ZFeatureMap being a well-known technique for encoding classical data into a quantum state. The ZFeatureMap circuit applies a series of Z rotations to the qubits based on the binary representation of the input features. Specifically, the number of Z rotations applied to each qubit is determined by the Hamming weight of the corresponding feature, which refers to the count of non-zero bits in the binary representation of the feature. Subsequently, the resulting quantum state can be employed as input for the quantum kernel in the VQC, enabling the model to operate on the data within a quantum space. [57]

The VQC has the potential to leverage the benefits of quantum computing in classification tasks by encoding classical data into a quantum state through the ZZFeatureMap circuit. These advantages encompass the efficient classification of high-dimensional data and the possibility of enhancing classification performance by utilizing quantum interference within feature space. [57]

4. Classification Phase

After preprocessing the dataset, the subsequent stage in constructing a machine-learning model involves the classification phase. During this phase, the preprocessed data is utilized to train a model capable of effectively categorizing new instances. The preprocessed dataset is partitioned into a training set (70%) and a testing set (30%) for this purpose. The training set is employed to train the VQC algorithm using a quantum kernel that transforms the input numerical data into a feature space of higher dimensions. To enhance the efficiency of the mapping process, the ZFeatureMap with a single repetition and a GPU backend is utilized. Following the training of the VQC model, it becomes possible to predict the class labels of new, unseen data. The primary objective of the classification phase is to develop a reliable and precise model capable of accurately classifying new data. The quantum component of the proposed model is illustrated in Figure 34, This figure demonstrates that the cleaned or processed data serves as input for the data encoder and VQC circuit. Furthermore, it reveals that each feature is represented by a qubit.[57]

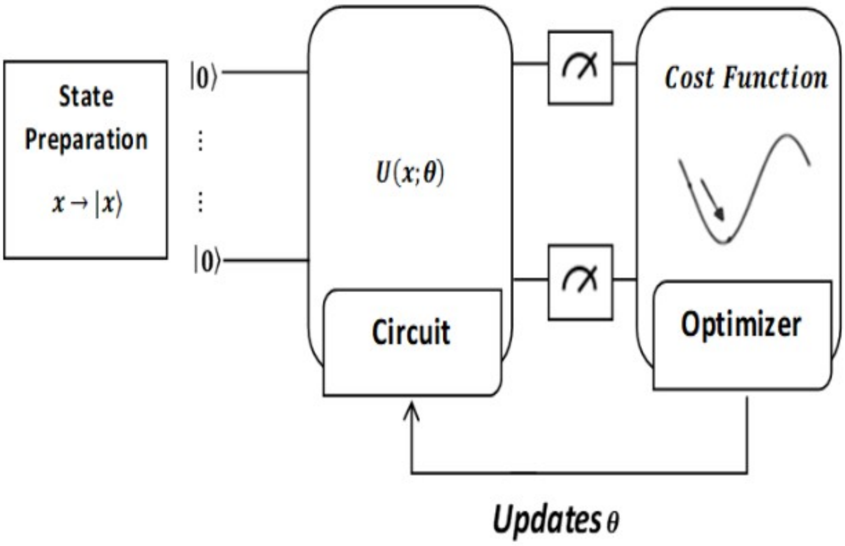


Figure 30: Variational Quantum Classifier[60]

5. Evaluation Phase

In this phase, the performance of the overall proposed intelligent model based on the VQC algorithm is assessed using various metrics. The initial metric is accuracy, which measures the proportion of correctly classified samples out of all the samples. A high accuracy value indicates that the model is performing well. [57]

Accuracy = (T P + T N)/(T P + F N + T N + F P) (25)

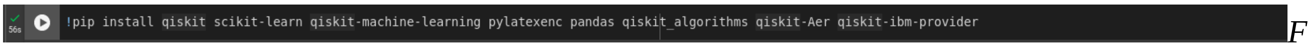
6. Implementation Phase in Qiskit

Within this segment, we shall outline the procedures necessary to operationalize the final sections on an actual quantum computer, showcasing each step in the process. Initially, the procedure commences with the installation of a series of libraries:

Name	**Version**
qiskit	1.0>
scikit-learn	Last version
qiskit-machine-learning	Last version
pylatexenc	Last version
pandas	Last version
qiskit_algorithms	Last version

qiskit-Aer	Last version
qiskit-ibm-provider	Last version

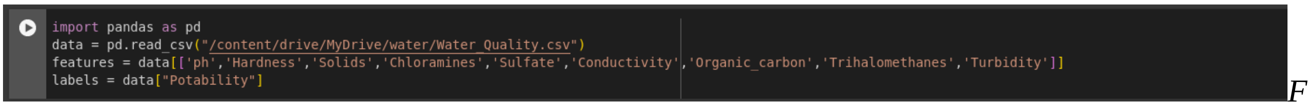
To execute the installation process, simply execute the following command in your interpreter, labeled as Figure 35.



```
!pip install qiskit scikit-learn qiskit-machine-learning pylatexenc pandas qiskit_algorithms qiskit-Aer qiskit-ibm-provider
```

Figure 31: installation aiskit

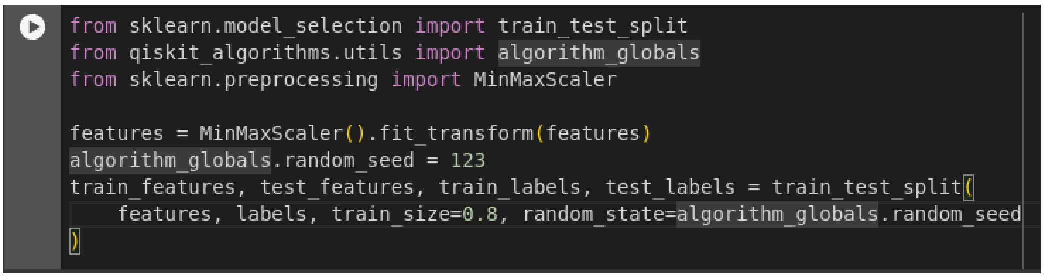
After the installation process is complete, your dataset stored in a CSV file will be imported and undergo a cleaning process.



```
import pandas as pd
data = pd.read_csv("/content/drive/MyDrive/water/Water_Quality.csv")
features = data[['ph','Hardness','Solids','Chloramines','Sulfate','Conductivity','Organic_carbon','Trihalomethanes','Turbidity']]
labels = data["Potability"]
```

Figure 32: Uplead dataset

After importing the data, use MinMaxScaler from scikit-learn can be utilized for this task. When used without any specific parameters, it effectively achieves the desired outcome by scaling and mapping the data onto a specified range [59], and use the features are separated from the labels. The Sklearn library's train_test_split function is then utilized to split the data into 80% training and 20% testing sets. This division allows our model to learn from the training data in order to make predictions on new, unseen data. Refer to figure 08 for a visual representation of this process.

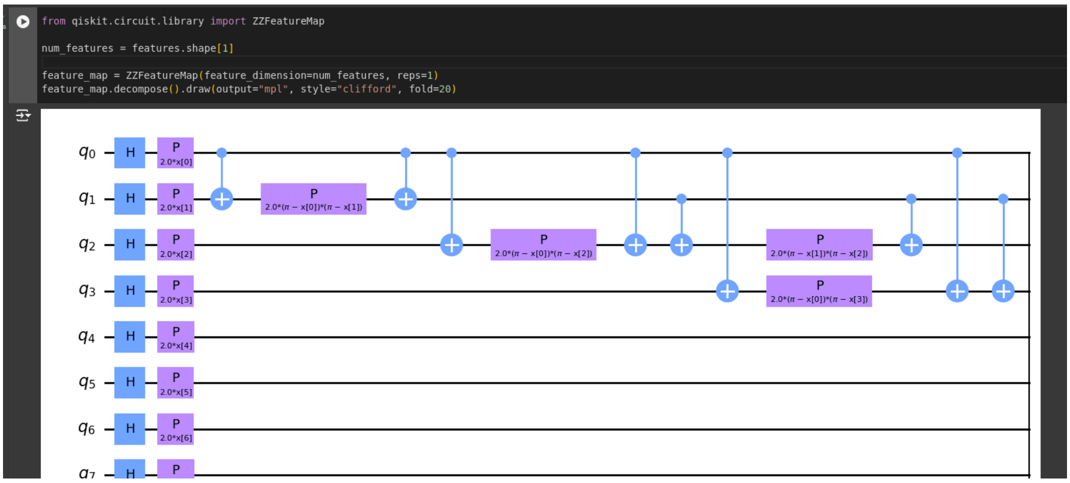


```
from sklearn.model_selection import train_test_split
from qiskit_algorithms.utils import algorithm_globals
from sklearn.preprocessing import MinMaxScaler

features = MinMaxScaler().fit_transform(features)
algorithm_globals.random_seed = 123
train_features, test_features, train_labels, test_labels = train_test_split(
    features, labels, train_size=0.8, random_state=algorithm_globals.random_seed
)
```

Figure 33: train_test_split function

After dividing the data, we commence the process by creating our ZZFeatureMap.

Now we add RealAmplitudes circuit is a heuristic trial wave function used as Ansatz in chemistry applications or classification circuits in machine learning. The circuit consists of alternating layers of YY rotations and CXCX entanglements. The entanglement pattern can be user-defined or selected from a predefined set. It is called

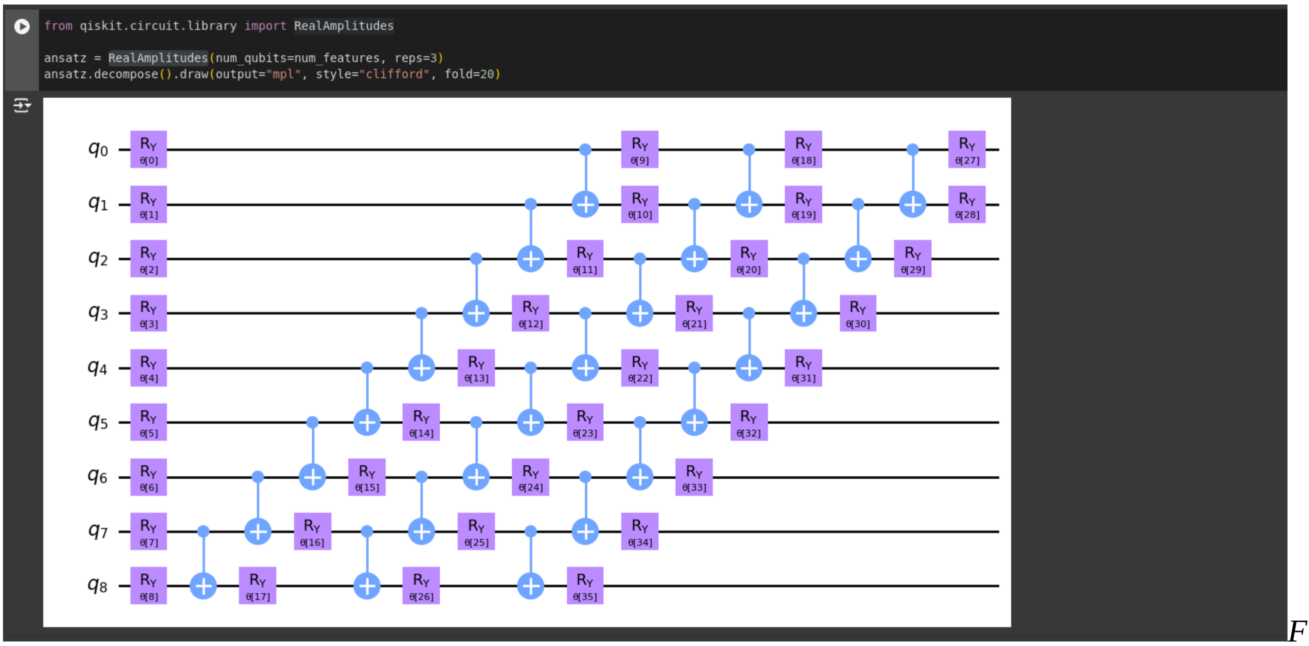


```
from qiskit.circuit.library import ZZFeatureMap

num_features = features.shape[1]

feature_map = ZZFeatureMap(feature_dimension=num_features, reps=1)
feature_map.decompose().draw(output="mpl", style="clifford", fold=20)
```

q0
q1
q2
q3
q4
q5
q6
q7
H
P
2.0*x[0]
2.0*x[1]
2.0*x[2]
2.0*x[3]
2.0*x[4]
2.0*x[5]
2.0*x[6]
2.0*(π − x[0])*(π − x[1])
2.0*(π − x[0])*(π − x[2])
2.0*(π − x[1])*(π − x[2])
2.0*(π − x[0])*(π − x[3])

Figure 34: ZZFeatureMap

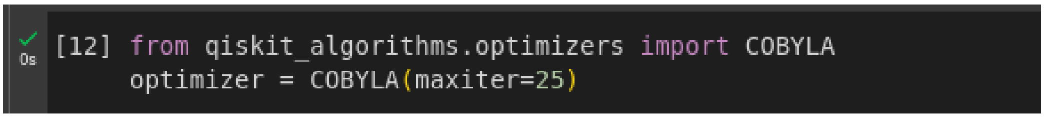
RealAmplitudes since the prepared quantum states will only have real amplitudes, the complex part is always 0.[58]



```
from qiskit.circuit.library import RealAmplitudes

ansatz = RealAmplitudes(num_qubits=num_features, reps=3)
ansatz.decompose().draw(output="mpl", style="clifford", fold=20)
```

Figure 35: RealAmplitudes

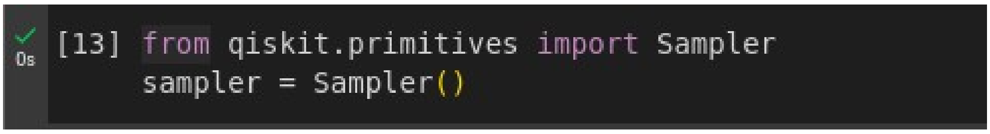
After defining the quantum circuit architecture for the classifier, we select an optimization algorithm to facilitate the training process. This step mirrors the approach taken in classical deep learning frameworks. To accelerate the training, we opt for a gradient-free optimizer, which can significantly reduce the computational overhead. However, Qiskit provides a range of optimizers, and you are free to explore alternative options that may better suit your specific requirements.[59]



```
[12] from qiskit_algorithms.optimizers import COBYLA
     optimizer = COBYLA(maxiter=25)
```

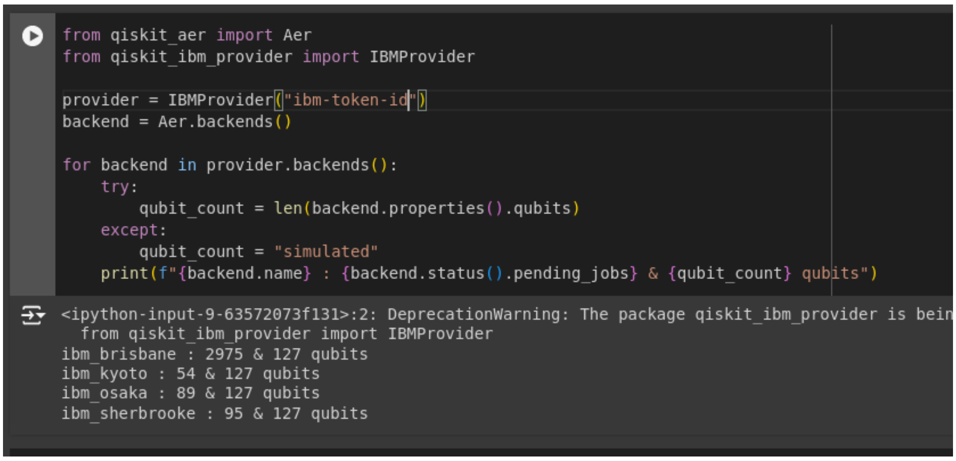
Figure 36: Optimizers

In the subsequent stage, the determination of the training location for our classifier is established. The options available are training on a simulator or a genuine quantum computer. In this case, we will opt for a simulator. To accomplish this, we generate an instance of the Sampler primitive, which serves as the benchmark implementation based on statevector. By utilizing qiskit runtime services, it is feasible to create a sampler that is supported by a quantum computer.[59]



```
[13] from qiskit.primitives import Sampler
     sampler = Sampler()
```

Figure 37: Sampler



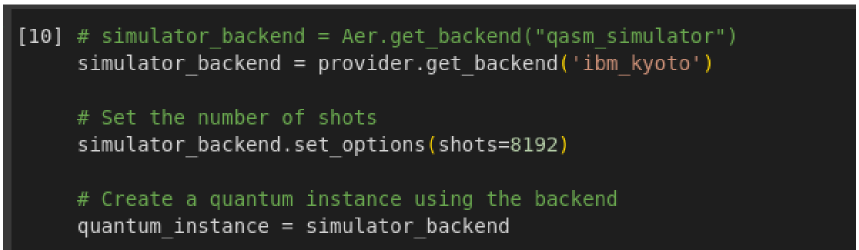
```
from qiskit_aer import Aer
from qiskit_ibm_provider import IBMProvider

provider = IBMProvider("ibm-token-id")
backend = Aer.backends()

for backend in provider.backends():
    try:
        qubit_count = len(backend.properties().qubits)
    except:
        qubit_count = "simulated"
    print(f"{backend.name} : {backend.status().pending_jobs} & {qubit_count} qubits")
```

```
<ipython-input-9-63572073f131>:2: DeprecationWarning: The package qiskit_ibm_provider is bein
  from qiskit_ibm_provider import IBMProvider
ibm_brisbane : 2975 & 127 qubits
ibm_kyoto : 54 & 127 qubits
ibm_osaka : 89 & 127 qubits
ibm_sherbrooke : 95 & 127 qubits
```

Figure 38: Register and show all quantum computer

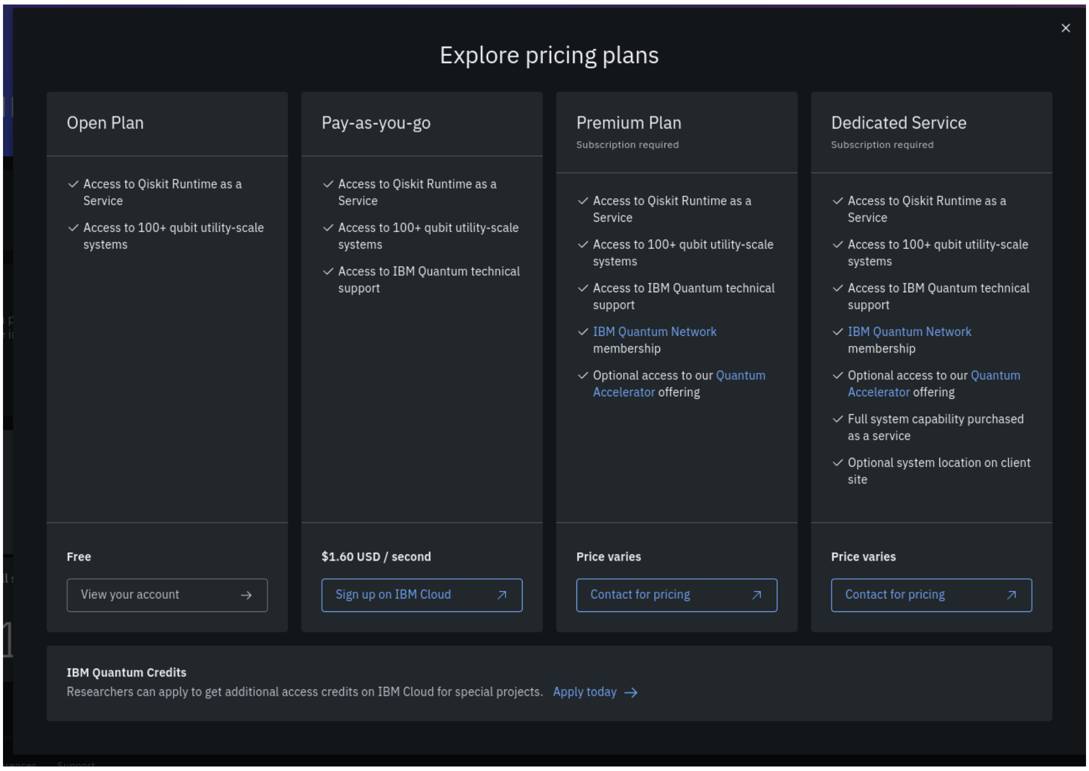
Here is an example of how to determine the authenticity of a quantum computer, Firstly, it is essential to showcase the availability of all quantum computers. The IBM token ID can be obtained from the IBM Quantum platform, as illustrated in Figure 40. The API token is a crucial component in accessing the platform's resources.

Figure 39: API token from IBM web site



```
[10] # simulator_backend = Aer.get_backend("qasm_simulator")
     simulator_backend = provider.get_backend('ibm_kyoto')

     # Set the number of shots
     simulator_backend.set_options(shots=8192)

     # Create a quantum instance using the backend
     quantum_instance = simulator_backend
```

In the upcoming configuration, we will select the quantum computer. Unfortunately, quantum computers are limited for free users to 300 circuits and 10 minutes of use, while our data set contains over 3700 elements.



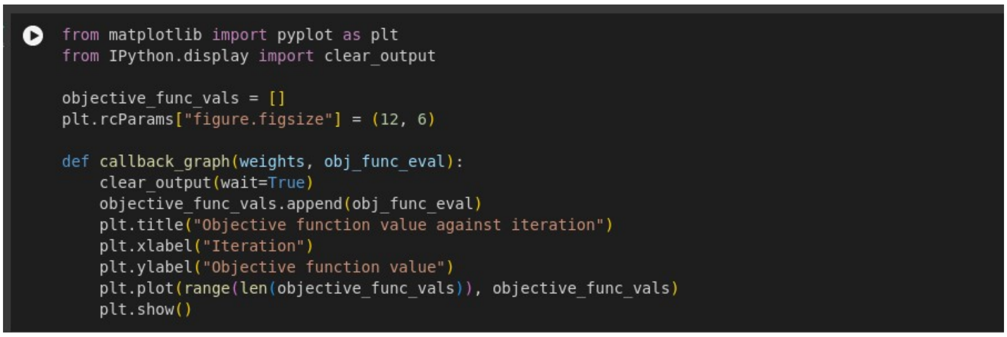
Explore pricing plans

Open Plan	Pay-as-you-go	Premium Plan Subscription required	Dedicated Service Subscription required
✓ Access to Qiskit Runtime as a Service ✓ Access to 100+ qubit utility-scale systems	✓ Access to Qiskit Runtime as a Service ✓ Access to 100+ qubit utility-scale systems ✓ Access to IBM Quantum technical support	✓ Access to Qiskit Runtime as a Service ✓ Access to 100+ qubit utility-scale systems ✓ Access to IBM Quantum technical support ✓ IBM Quantum Network membership ✓ Optional access to our Quantum Accelerator offering	✓ Access to Qiskit Runtime as a Service ✓ Access to 100+ qubit utility-scale systems ✓ Access to IBM Quantum technical support ✓ IBM Quantum Network membership ✓ Optional access to our Quantum Accelerator offering ✓ Full system capability purchased as a service ✓ Optional system location on client site
Free	**$1.60 USD / second**	**Price varies**	**Price varies**
View your account →	Sign up on IBM Cloud ↗	Contact for pricing ↗	Contact for pricing ↗

IBM Quantum Credits
Researchers can apply to get additional access credits on IBM Cloud for special projects. Apply today →

Figure 41: Qiskit pricing plans

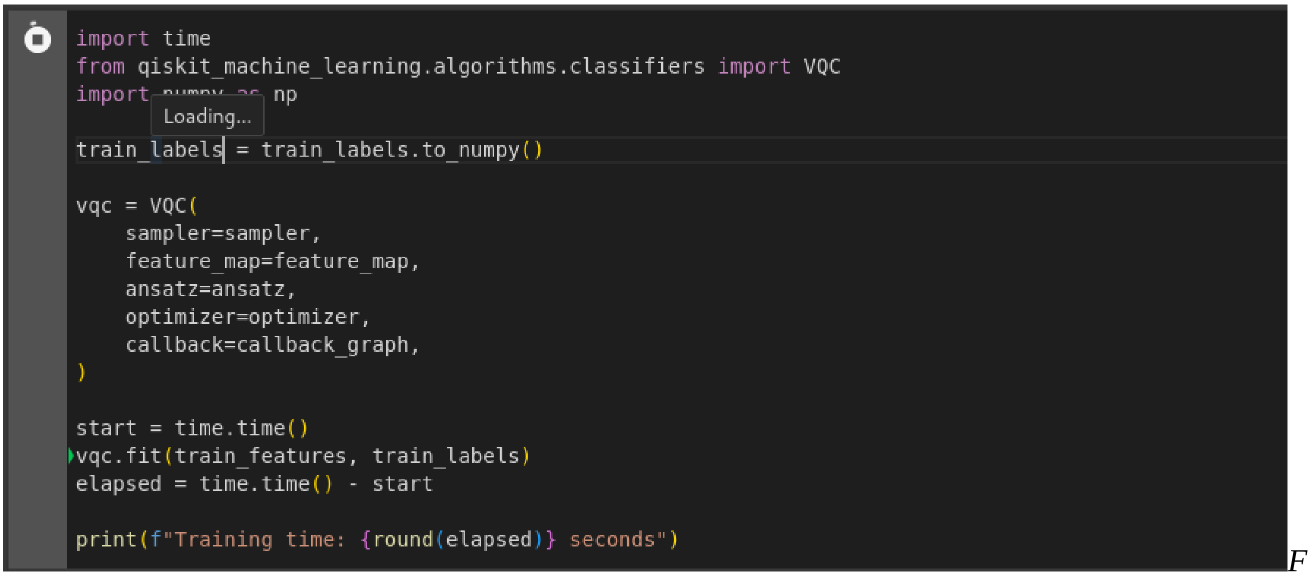
Afterwards, a callback function will be generated to execute the function for each evaluation of the objective function, involving two variables: the current weights and the corresponding value of the objective function at those weights.



```python
from matplotlib import pyplot as plt
from IPython.display import clear_output

objective_func_vals = []
plt.rcParams["figure.figsize"] = (12, 6)

def callback_graph(weights, obj_func_eval):
    clear_output(wait=True)
    objective_func_vals.append(obj_func_eval)
    plt.title("Objective function value against iteration")
    plt.xlabel("Iteration")
    plt.ylabel("Objective function value")
    plt.plot(range(len(objective_func_vals)), objective_func_vals)
    plt.show()
```

Figure 42: callback function

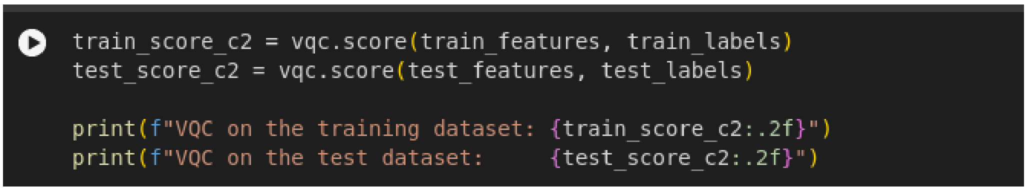
In the subsequent stage, the VQC function is established by incorporating all the necessary parameters. Subsequently, we proceed with the fitting process, and upon completion, we calculate the duration of this iteration.



```
import time
from qiskit_machine_learning.algorithms.classifiers import VQC
import numpy as np

train_labels = train_labels.to_numpy()

vqc = VQC(
    sampler=sampler,
    feature_map=feature_map,
    ansatz=ansatz,
    optimizer=optimizer,
    callback=callback_graph,
)

start = time.time()
vqc.fit(train_features, train_labels)
elapsed = time.time() - start

print(f"Training time: {round(elapsed)} seconds")
```

Figure 43: VOC function

Finally, in the last stage, we evaluate the accuracy of both the training and testing processes, yielding the final outcome.



```
train_score_c2 = vqc.score(train_features, train_labels)
test_score_c2 = vqc.score(test_features, test_labels)

print(f"VQC on the training dataset: {train_score_c2:.2f}")
print(f"VQC on the test dataset:     {test_score_c2:.2f}")
```

Figure 44: The accuracy of both the training and testing processes

7. Experimental Results and Discussion

In this section, the performance of the proposed water quality classification model is evaluated and tested. We obtain a line plot (Fig. 49) depicting the objective function value over multiple iterations during the model's optimization and training process. To assess the model's significance, we apply a score function from the Qiskit library that provides us with accuracy scores.

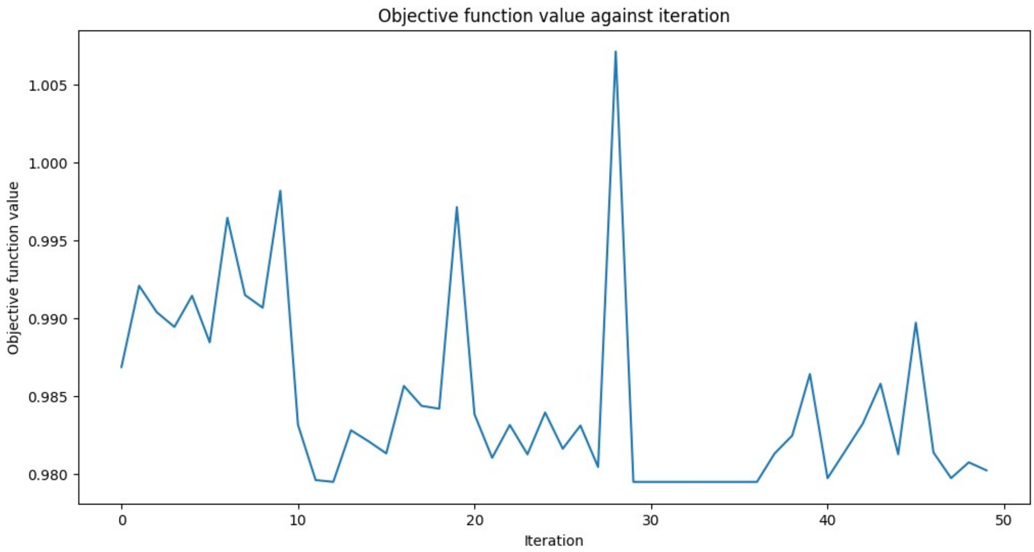


Figure 45: Model's performance

The model's performance was neither particularly not good, achieving an accuracy of 58% on the test set and 59% on the training set. However, the available data was limited, and the model was not ideally suited for this specific task. We explored other promising models like VQC, but they did not work due to some issues with the library, potentially due to constraints in the available data. Additionally, we considered using other datasets, but the free Google Colab instance's RAM limit of 12GB posed a limitation.

8. Conclusion

In this chapter, we have explored the potential of using Variational Quantum Classification (VQC) for water quality classification. Despite the promising theoretical advantages of quantum computing, our experimental results showed that the performance of the proposed model was suboptimal, achieving an accuracy of 58% on the test set and 59% on the training set. Several factors contributed to this outcome, including the limited amount of available data, the specific nature of the task, and computational constraints associated with using free resources on Google Colab.

The model's performance was not ideal for the specific task at hand, and we faced challenges such as library compatibility issues and data constraints. Nevertheless, this study provides valuable insights into the practical application of quantum machine learning for water quality classification and highlights areas for improvement in future research.

Overall, while the current implementation of the VQC model did not meet expectations, it lays the groundwork for further exploration and refinement of quantum machine learning techniques in this domain.

Conclusion

This dissertation has explored the promising intersection of quantum computing and machine learning to address the critical issue of water quality prediction. With the increasing scarcity of potable water, innovative solutions are essential. Quantum computing, with its unparalleled computational capabilities, offers a revolutionary approach to complex problem-solving.

Initially, we provided an in-depth overview of quantum computing fundamentals, including key concepts like superposition, entanglement, and quantum gates. These principles enable quantum computers to outperform classical counterparts in certain computational tasks, setting the stage for advancements in various fields.

The focus then shifted to quantum machine learning, particularly emphasizing algorithms such as Quantum Neural Networks (QNN) and Quantum Support Vector Machines (QSVM). These quantum algorithms promise significant improvements over classical machine learning techniques due to their ability to handle complex data structures and computations more efficiently.

A practical implementation of these concepts was demonstrated through the development of a water quality prediction model using Variational Quantum Classification (VQC). This model, implemented on IBM's Qiskit platform, involved a meticulous process of data preprocessing, quantum data encoding, and algorithm application. Despite the innovative approach, the experimental results yielded modest accuracy rates of 58% on the test set and 59% on the training set. These outcomes underscore the current challenges in the nascent field of quantum machine learning, such as data limitations, library compatibility issues, and computational constraints associated with using free resources.

Despite these challenges, the study highlights the feasibility and potential of quantum machine learning in solving real-world problems. The findings underscore the necessity for further research to overcome existing limitations and improve the robustness and accuracy of quantum models.

In conclusion, this dissertation has laid a foundational framework for integrating quantum machine learning into environmental science applications. The exploration of VQC for water quality prediction represents a significant step towards harnessing the power of quantum computing. While the results are preliminary, they offer valuable insights and set the stage for future advancements. As quantum technology progresses, we anticipate the development of more sophisticated and accurate models, contributing significantly to global challenges such as water quality monitoring and prediction. The continued exploration and refinement in this field are crucial and hold immense potential for transformative impacts on science and technology.

Bibliography

[1] The Quantum Insider. (2024, Apr 27). History of Quantum Computing. https://thequantuminsider.com/2020/05/26/history-of-quantum-computing/

[2] Plus Magazine. (2024 Apr 27.). Physics in a minute: The double-slit experiment. https://plus.maths.org/content/physics-minute-double-slit-experiment-0

[3] Adedoyin, A., Ambrosiano, J., Anisimov, P., Casper, W., Chennupati, G., Coffrin, C., ... & Lokhov, A. Y. (2018). Quantum algorithm implementations for beginners. *arXiv preprint arXiv:1804.03719.*

[4] Hughes, C., Isaacson, J., Perry, A., Sun, R. F., & Turner, J. (2021). *Quantum computing for the quantum curious* (p. 150). Springer Nature.

[5] Chander, S. R. The Emergence of Quantum Advantage and Predictions of Its Limiting Factors.

[6] BYJU'S. (2024 Apr 27). What is the Superposition Principle in Quantum Mechanics, in Layman's Terms? https://byjus.com/question-answer/what-is-the-superposition-principle-in-quantum-mechanics-in-laymans-terms/

[7] Kok, P. (2018). *A first introduction to quantum physics*. Springer International Publishing.

[8] Baaquie, B. E., & Kwek, L. C. (2023). *Quantum computers: Theory and algorithms.* Springer Nature.

[9] Pandey, R., Srivastava, N., Singh, N. K., & Tyagi, K. (Eds.). (2023). *Quantum Computing: A Shift from Bits to Qubits*. Springer.

[10] Published with permission of © Belal E. Baaquie and L. C. Kwek. All Rights Reserved

[11] Nielsen, M. A., & Chuang, I. L. (2010). *Quantum computation and quantum information*. Cambridge university press.

[12] Rieffel, E. G., & Polak, W. H. (2011). *Quantum computing: A gentle introduction.* MIT press.

[13] Serrano, M. A., Pérez-Castillo, R., & Piattini, M. (Eds.). (2022). *Quantum software engineering*. Springer Nature.

[14] Deutsch D, Jozsa R (1992) Rapid solution of problems by quantum computation. Proc R Soc Lond A Math Phys Eng Sci 439(1907):553–558

[15]Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing, STOC ’96. ACM, New York, NY, pp 212–219

[16] Rivest RL, Shamir A, Adleman L (1978) A method for obtaining digital signatures and public-key cryptosystems. Commun ACM 21(2):120–126

[17] Shor P (1994) Algorithms for quantum computation: discrete logarithms and factoring. In:Proceedings of FOCS, pp 124–134

[18] Nielsen MA, Chuang IL (2011) Quantum computation and quantum information, 10th Anni-versary edn. Cambridge University Press

[19] Peruzzo A, McClean J, Shadbolt P, Yung MH, Zhou XQ, Love PJ, Aspuru-Guzik A, O’brien, J.L. (2014) A variational eigenvalue solver on a photonic quantum processor. Nat Commun 5(1):1–7

[20] Farhi E, Goldstone J, Gutmann S (2014) A quantum approximate optimization algorithm. ArXiv preprint arXiv:1411.4028

[21] Schuld M (2018) Supervised learning with quantum computers. Springer

[22] Abbas A, Sutter D, Zoufal CZ, Lucchi A, Figalli AF, Woerner S (2021) The power of quantum neural networks. Nat Computational Sci 1:403–409

[23] IoT World Today. (n.d.). 13 Companies Offering Quantum as a Service. https://www.iotworldtoday.com/industry/13-companies-offering-quantum-as-a-service

[24] *Top Performance Metrics in Machine Learning: A Comprehensive Guide*. (n.d.). Www.v7labs.com. Retrieved May 11, 2024, from https://www.v7labs.com/blog/performance-metrics-in-machine-learning#top-regression-metrics

[25] Murphy, K.P.: Machine Learning: a Probabilistic Perspective. MIT Press (2012)

[26] Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., Lloyd, S.: Quantum Machine Learning. Nature. 549(7671), (November 2016)

[27] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. Artificial intelligence: A modern approach, volume 3. Prentice Hall Englewood Cliffs, 2010.

[28] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological Review, 65(6):386, 1958.

[29] Schuld, M., Sinayskiy, I., & Petruccione, F. (2015). An introduction to quantum machine learning. *Contemporary Physics*, *56*(2), 172-185.

[30] Mishra, N., Kapil, M., Rakesh, H., Anand, A., Warke, A., Sarkar, S., Dutta, S., Gupta, S., Dash, A.P., Gharat, R., Chatterjee, Y., Roy, S., Raj, S., Jain, V.K., Bagaria, S., Chaudhary, S., Singh, V., Maji, R., Dalei, P., Behera, B.K., Mukhopadhyay, S., Panigrahi, P.K.: Quantum Machine Learning : A Review and Current Status Quantum Machine Learning, pp. 1–24. https://doi.org/10.13140/RG.2.2.22824.72964

[31] Wittek, P.: Quantum machine learning. Academic Press. 73-83, (2014). https://doi.org/10.1016/B978-0-12-800953-6.00020-7.

[32] Saini, S., Khosla, P. K., Kaur, M., & Singh, G. (2020). Quantum driven machine learning. *International Journal of Theoretical Physics*, *59*(12), 4013-4024.

[33] Haney, B. S. (2020). Quantum machine learning: a patent review. *Case W. Res. JL Tech. & Internet*, *12*, i.

[34] Lloyd, S.P.: Least squares quantization in pcm. IEEE Transactions on InformationTheory. 28, 129–137 (1982)

[35] Vasuki, M., Karunamurthy, A., Ramakrishnan, R., & Prathiba, G. (2023). Overview of Quantum Computing in Quantum Neural Network and Artificial Intelligence.

[36] Amzhao. (2024, May 23). *Quantum Neural Networks*. MIT 6.S089—Intro to Quantum Computing. https://medium.com/mit-6-s089-intro-to-quantum-computing/quantum-neural-networks-7b5bc469d984

[37] Mitarai, K., Negoro, M., Kitagawa, M. & Fujii, K. Quantum circuit learning. arXiv preprint arXiv:1803.00745 (2018).

[38] Farhi, E. & Neven, H. Classification with quantum neural networks on near term processors. arXiv preprintarXiv:1802.06002 (2018).

[39] Kandala, A. et al. Hardware-efficient variational quantum eigensolver for small molecules and quantum mag nets. Nature 549, 242 (2017).

[40] Farhi, E., Goldstone, J., Gutmann, S. & Neven, H.Quantum algorithms for fixed qubit architectures. ArXiv preprint arXiv:1703.06199 (2017).

[41] Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, *567*(7747), 209-212.

[42] Rigetti, C. & Devoret, M. Fully microwave-tunable universal gates in superconducting qubits with linear couplings and fixed transition frequencies. Phys. Rev. B 81, 134507 (2010). URL https://link.aps.org/doi/10.1103/PhysRevB.81.134507 .

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

[44] B. J Mercer, Phil. Trans. R. Soc. Lond. A 209, 415 (1909).

[45] Goldberg, L. A. & Guo, H. The complexity of approximating complex-valued ising and tutte partition functions. computational complexity 26, 765–833 (2017).

[46] Demarie, T. F., Ouyang, Y. & Fitzsimons, J. F. Classical verification of quantum circuits containing few basis changes. Physical Review A 97, 042319 (2018).

[47] Buhrman, H., Cleve, R., Watrous, J. & De Wolf, R. Quantum fingerprinting. Physical Review Letters 87, 167902 (2001).

[48] Cincio, L., Subaşı, Y., Sornborger, A. T. & Coles, P. J. Learning the quantum algorithm for state overlap. ArXiv preprint arXiv:1803.04114 (2018).

[49] Tropp, J. A. et al. An introduction to matrix concentration inequalities. Foundations and Trends R in Machine Learning 8, 1–230 (2015).

[50] Kandala, A. et al. Extending the computational reach of a noisy superconducting quantum processor. ArXiv preprint arXiv:1805.04492 (2018).

[51] Spall, J. C. A one-measurement form of simultaneous perturbation stochastic approximation. Automatica 33, 109 (1997).

[52] Spall, J. C. Adaptive stochastic approximation by the simultaneous perturbation method. IEEE Transaction on Automatic Control 45, 1839 (2000).

[53] Temme, K., Bravyi, S. & Gambetta, J. M. Error mitigation for short-depth quantum circuits. Physical review letters 119, 180509 (2017).

[54] *Machine Learning Model Validation - The Data-Centric Approach*. (n.d.). Www.appen.com. Retrieved May 11, 2024, from https://www.appen.com/blog/machine-learning-model-validation

[55] *Top Performance Metrics in Machine Learning: A Comprehensive Guide*. (n.d.). Www.v7labs.com. Retrieved May 11, 2024, from

https://www.v7labs.com/blog/performance-metrics-in-machine-learning#top-regression-metrics

[56] *Water Quality*. (n.d.). Www.kaggle.com. https://www.kaggle.com/datasets/adityakadiwal/water-potability

[57] Ahmed, H. K., Tantawi, B., Magdy, M., & Sayed, G. I. (2023). A Quantum Machine Learning Model for Medical Data Classification. In *Machine Intelligence for Smart Applications: Opportunities and Risks* (pp. 95-114). Cham: Springer Nature Switzerland.

[58] *RealAmplitudes*. (n.d.). IBM Quantum Documentation. Retrieved May 30, 2024, from https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.library.RealAmplitudes

[59]*Training a Quantum Model on a Real Dataset - Qiskit Machine Learning 0.7.2*. (n.d.). Qiskit-Community.github.io. Retrieved May 30, 2024, from https://qiskit-community.github.io/qiskit-machine-learning/tutorials/02a_training_a_quantum_model_on_a_real_dataset.html#2.-Training-a-Classical-Machine-Learning-Model

[60] Maheshwari, D., Sierra-Sosa, D., & Garcia-Zapirain, B. (2021). Variational quantum classifier for binary classification: Real vs synthetic dataset. IEEE access, 10, 3705-3715.