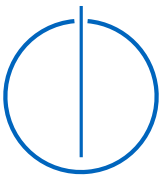
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Leveraging Noise in Quantum Machine Learning to Improve Model Robustness

Erick Ruben Quintanar Salas



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY — INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Leveraging Noise in Quantum Machine Learning to Improve Model Robustness

Nutzung von Rauschen beim Quantenmaschinellen Lernen zur Verbesserung der Modellrobustheit

Author:	Erick Ruben Quintanar Salas
Supervisor:	Prof. Dr. Claudia Eckert
Advisors:	M. Sc. Pascal Debus / M. Sc. Kilian Tscharke
Submission Date:	October 31st, 2024

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, October 31st, 2024 Erick Ruben Quintanar Salas

Acknowledgments

Abstract

Contents

1 Introduction

In recent years the interest on techniques to utilize quantum mechanics has been rising. One of the many applications is quantum computing, where devices based on the laws of quantum theory are exploited to process information [1]. Although current classical computers have become very powerful, they still struggle to process many applications that quantum computers can in theory easily solve.

Many quantum algorithms for quantum computers have been proposed that highly outperform a classical computer with the best known algorithms [2, 3, 4]. These quantum algorithms solve in polynomial time problems that quickly become intractable to solve in a classical computer, as they normally grow exponentially. The most famous algorithm is Shor's algorithm [2], which can find the prime factors of an integer. It is of special interest because if quantum devices were able to execute it, the current confidentiality and integrity guarantees that the RSA [5] cryptographic mechanism offers would be violated.

1.1 Motivation

Even though the previously mentioned quantum algorithms would surpass the performance of the best classical ones, they still can not be executed on current quantum computers due to noise [6]. This noise occurs because current quantum devices are not completely isolated from the environment and every time we perform an operation on them we introduce a disturbance. This type of device is known as Noisy Intermediate-Scale Quantum (NISQ), meaning that there will be significant noise when operating the quantum device.

In order to reduce the influence of noise in NISQ devices, either the precision in which quantum computers can be manipulated has to improve or error-correcting codes have to be implemented [7]. Currently both techniques are being heavily researched and in conjunction will lead to the next generation of quantum devices, namely fault-tolerant quantum computers.

A technology that right now has gained a bigger presence in our society is Machine Learning (ML). There have been many important breakthroughs for ML in the past few

years, with uses in natural language processing, computer vision, anomaly detection, and many more fields [8]. Nowadays ML has a big impact in society, and even though it depicts big opportunities for improvement in society it also represents significant risks.

Quantum computing and ML are information processing techniques that have improved significantly in recent years. This lead to the natural desire of harnessing the advantages of both and to the emergence of a new field of study denominated Quantum Machine Learning (QML) [9]. QML explores several ideas like whether quantum devices are better at ML than classical computers or if quantum information adds new data that affects how machines recognize patterns.

Returning to the possible risks that ML might encounter, several attacks have been developed to force a ML model to missclassify an input [10]. These attacks are denominated adversarial attacks and are based on crafting specific input data that has been slightly modified to cause the model to erroneously classify the input. These small modifications are imperceptible for humans. At the beginning, when the first adversarial attacks were developed, they needed to know the architecture of the model to be able to fool it. Nevertheless, it was proved that adversarial attacks are transferable between models with the same use case, without knowing the architecture of the model or the dataset it was trained on [11].

Adversarial training was developed in order to defend ML models against adversarial attacks [12, 10]. Adversarial training consists of including adversarial samples into the training of the ML model to better generalize its classification. This mechanism has a tradeoff, in which the accuracy of the model lowers, while increasing the resilience to adversarial attacks [13].

In classical ML, noise in training has been shown to improve generalization performance and local optima avoidance [14]. This property from noise is particularly interesting in NISQ devices, as their inherent noise might be able to improve QML performance, accuracy and resilience against adversarial attacks.

1.2 Research Goals

1. Test the effect of different types of noise in QML regarding robustness. 2. Test the effect of noise in different parts of the QML circuits. 3. Test the effect of noise between VQA and Kernel methods. 4. Test the models with different types of adversarial attacks (FGSM, CaW, PGD)

1.3 Outline

Write here what is the general structure of the thesis.

2 Theoretical Background

In Chapter 2 we will introduce the background information that is required to understand the main ideas of this thesis. First we introduce the basic concepts of quantum computing (Sec. 2.1). Then we will describe advanced concepts regarding quantum noise (Sec. 2.2). We assume some baseline ML knowledge, however, we will provide an outlook into QML (Sec.2.3). Finally, we present several types of Adversarial Machine Learning (AML) (Sec. 2.4).

2.1 Fundamentals of Quantum Computing

In this section, first we introduce the qubit (Subsec. 2.1.1), the Bloch sphere (Subsec. 2.1.2), and multiple qubits (Subsec. 2.1.4.2). Afterwards, we present the operations to manipulate qubits in Subsection 2.1.4. Then, we describe entanglement (Subsec. 2.1.5), a primordial property of quantum mechanics. Thereafter, quantum circuits (Subsec. 2.1.7) and measurements (Subsec. 2.1.6) are explained. Finally, the density operator (Subsec. 2.1.8) is defined to enable the description of noise in quantum systems.

The information from this section is mainly based on Nielsen's book [15], further information regarding quantum computing and quantum information can be found there.

2.1.1 Qubit

The basic computing unit in quantum computing is the *qubit* [16]. Similar to the classical bit, a qubit also has a state. While a bit has either a *0* or *1* state, the qubit can have many more states. The quantum equivalent to the classical bit states would be $|0\rangle$ and $|1\rangle$ (spoken as *ket*) in Dirac notation [17] and they represent the orthonormal

computational basis states in Equation 2.1.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2.1}$$

The complementing element to Dirac's notation is the *bra*: $\langle 0|$ and $\langle 1|$. The *bra* is the conjugate transposed vector from a *ket*. The outer product of two vectors u and v in dirac notation is written as $|u\rangle\langle v|$. While the inner product of the same vectors is expressed as $\langle u|v\rangle$.

What makes the qubit different and more capable than the classical bit is that it can also have different states created by a linear combination or *superposition* from its basis states. The linear combination in Equation 2.2 is the complete representation of a qubit, where α and β are two complex numbers that are denominated *probability amplitudes*. The values α and β represent a distribution, in which with probability $|\alpha|^2$ we will observe a *0* value and with probability $|\beta|^2$ we will observe a *1* value. This distribution is determined by the Born rule [18] and states that $|\alpha|^2 + |\beta|^2 \stackrel{!}{=} 1$. The Born rule thus implies that a qubit state is a unitary vector in a two-dimensional complex vector space. Although the probability amplitudes can take on any complex value as long as they fulfill the Born rule, when we perform a measurement in the computational base on the qubit it *collapses* to one of the two basis states.

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.2}$$

In the real physical world qubits can be implemented by several different mechanisms. However, the mathematical qubit abstraction helps establish a baseline computing unit for quantum computing independent of which mechanism it is being represented by [15]. While in this perfect mathematical description noise does not occur, there are different mechanisms to represent the noise that quantum computers suffer from, e.g. the density operator that will be introduced in Subsection 2.1.8.

2.1.2 Bloch Sphere

The qubit state from Equation 2.2 can be rewritten into Equation 2.3, where e is the Euler number, i is the imaginary number, and γ, φ, and θ are real numbers.

$$|\psi\rangle = e^{i\gamma}\left(\cos\frac{\theta}{2}\,|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\beta\,|1\rangle\right) \tag{2.3}$$

Because for a single qubit the global phase $e^{i\gamma}$ has no observable effects, we can omit it and write the state of a qubit as:

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}\beta|1\rangle \tag{2.4}$$

where θ and φ determine a point in the Bloch sphere [19]. The Bloch sphere (Fig. 2.1) is a helpful visual representation for understanding the state of a qubit. In Section 2.2 this representation will be utilized to show the effects of quantum noise on a quantum state. It can also be used to visualize the effect of the operations performed on quantum states, these operations are called *gates* and they will be introduced in Subsection 2.1.4.

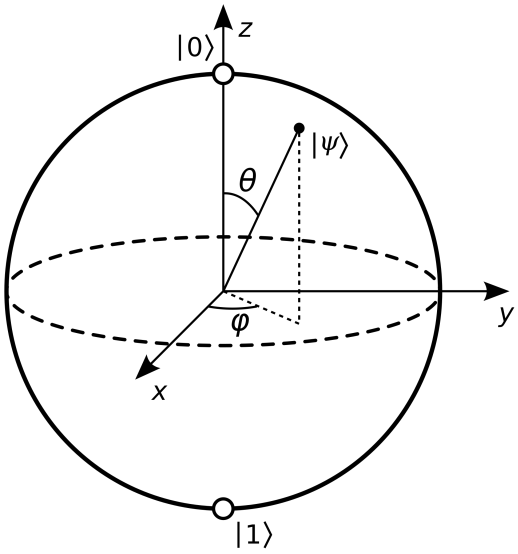


Figure 2.1: Bloch sphere representation of a qubit. Image taken from Wikimedia Commons under the Creative Commons Attribution-Share Alike 3.0 Unported license.

2.1.3 Multiple qubits

To describe multiple qubits we utilize the fundamentals presented in Subsection 2.1.1 and expand them. For two qubits $|00\rangle, |01\rangle, |10\rangle$, and $|11\rangle$ are the com-

putational basis. A general representation for a two qubit system can be found in Equation 2.5, where all the probability amplitudes must follow the Born rule.

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2.5}$$

Due to the Born rule the measurement results for $x = \{00, 01, 10, 11\}$ follow the probability distribution determined by $|\alpha_x|^2$. Similar to a single qubit, once a measurement on both qubits is performed, the state of the qubits will collapse to the measured basis. Nevertheless, with multiple qubits we are able to perform measurements on a subset of qubits. In the case of a two-qubit system, measuring the first qubit will collapse its value. However, the second's qubit state will remain. In the case of Eq. 2.5, if *0* was measured in the first qubit, the amplitudes α_{10} and α_{11} would disappear from the state as they are no longer possible. Furthermore, the remaining amplitudes must be normalized, such as in Eq. 2.6, to fulfill the normalization restriction.

$$|\psi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \tag{2.6}$$

In Equation 2.7 we can find a general representation of a set of n qubits. For a system composed by n qubits there are 2^n amplitudes. If we tried to simulate a quantum system with $n = 50$, assuming that complex numbers require 8 bytes to be stored [20], a classical computer would need approximately 9000 terabytes to store the generated quantum state. This simple calculation shows the reason why quantum computers are so promising and also why classical computers are not able to process quantum information efficiently.

$$|\psi\rangle = \alpha_{00}|0\cdots0\rangle + \cdots + \alpha_{2^{n-1}}|1\cdots1\rangle \tag{2.7}$$

2.1.4 Quantum Gates

Once that quantum states have been defined, performing operations on them is the next step to understand how quantum computing works. These operations are denominated *quantum gates* and they modify the quantum state according to gates' properties. A quantum gate can be represented as a matrix that fulfills one single property, namely that it is a unitary matrix. In Equation 2.8 we can observe that a

unitary matrix is one which when multiplied by its own transpose conjugate is equal to the identity matrix.

$$UU^{\dagger} = I \qquad |\psi\rangle \xrightarrow{U} U|\psi\rangle = |\psi'\rangle \tag{2.8}$$

This property is required because when a quantum gate is used on a quantum state (Eq. 2.8), the resulting quantum state has to be a valid normalized quantum state. By being a unitary matrix, this effect is achieved. Applying a quantum gate to a quantum state can also be mathematically interpreted as a matrix-vector multiplication between the gate's unitary matrix and the quantum state vector. There are infinitely many unitary matrices, however, there are some of specific importance. The most important single-qubit quantum gates will be introduced in Subsection 2.1.4.1, while the most significant multiple-qubits gates will be presented in Subsection 2.1.4.2

2.1.4.1 Single-Qubit Gates

Single-Qubit gates can be described by a two by two unitary matrix. The first three quantum gates introduced are described by the Pauli matrices. They are defined as the X, Y and Z gates because each matrix represents a π rotation around the Bloch sphere in their respective axis.

X Gate

The X gate's matrix and its gate representations can be found in Equation 2.9. In classical computing, the X gate is conceptually equivalent to the NOT gate, thus it is also known as a quantum NOT gate.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \text{—}\boxed{X}\text{—} \qquad \text{—}\oplus\text{—} \tag{2.9}$$

In Equation 2.10 we can see the effect of the X gate, where the computational basis states are flipped. This phenomena is known as a *bit flip*.

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle, \qquad X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \tag{2.10}$$

Z Gate

Unlike the X gate, there is no conceptually equivalent gate for the Z gate in classical computing. In Equation 2.11 we can observe the matrix and symbol representation of the Z gate.

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \qquad -\boxed{Z}- \tag{2.11}$$

The effects of the Z gate can be found in Equation 2.12. We can observe that the $|0\rangle$ state remains unmodified while the $|1\rangle$ state is negative after the operation. This phenomena is known as a *phase flip*.

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |0\rangle, \qquad Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = -|1\rangle \tag{2.12}$$

Y Gate

The Y gate also doesn't have a conceptually equivalent gate in classical computing. More interestingly the Y gate can be described by the product of the X and Z matrix up to a global phase. The matrix and symbol representation of the Y gate can be found in Equation 2.13.

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad -\boxed{Y}- \tag{2.13}$$

In Equation 2.14 we can see how the Y gate modifies the computational basis states. In it we can observe that a bit flip and a phase flip have been executed, while i has been added as a global phase.

$$Y|0\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = i|1\rangle, \qquad Y|1\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = -i|0\rangle \tag{2.14}$$

Rotational Gates

As a complement to the Pauli gates, there are gates that perform a specific θ rotation around each axis of the Bloch sphere. They are known as $R_X(\theta)$, $R_Y(\theta)$ and $R_Z(\theta)$. Their unitary matrices and symbols can be found in Equation 2.15.

$$R_X(\phi) = \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) & -i\sin\left(\frac{\phi}{2}\right) \\ -i\sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) \end{pmatrix} \qquad R_Z(\phi) = \begin{pmatrix} e^{-i\frac{\phi}{2}} & 0 \\ 0 & e^{i\frac{\phi}{2}} \end{pmatrix}$$
$$R_Y(\phi) = \begin{pmatrix} \cos\left(\frac{\phi}{2}\right) & -\sin\left(\frac{\phi}{2}\right) \\ \sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) \end{pmatrix} \qquad \text{—}\boxed{R_X(\phi)}\text{—}\boxed{R_Y(\phi)}\text{—}\boxed{R_Z(\phi)}\text{—} \tag{2.15}$$

More importantly, each Pauli matrix can be derived up to a global phase from their respective rotational gate for a given angle $\theta = \pi$. Additionally, if two rotation gates of the same type are contiguously applied to a quantum state, they can be summarized by one gate of the same type executing a rotation of the both angles' sum, such that $R(\alpha)R(\beta)|\psi\rangle = R(\alpha+\beta)|\psi\rangle$ is valid.

Hadamard Gate

The Hadamard gate is one of the most useful single-qubit gates. It will be later used in Subsection 2.1.5 to create *entanglement* between two different qubits.

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad \text{—}\boxed{H}\text{—} \tag{2.16}$$

The effects of the Hadamard gate can be observed in Equation 2.17. The resulting states from applying the Hadamard gate to the computational basis are denominated $|+\rangle$ and $|-\rangle$ respectively and they represent a superposition with equal probability for both computational basis to be measured.

$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle \qquad H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |-\rangle \tag{2.17}$$

Another interesting property of the previously mentioned single-qubit gates is that all of them are involutory. An involutory matrix is a square matrix that is its own inverse. This implies that performing twice the same gate will not modify the quantum state as the identity transformation would have been applied.

2.1.4.2 Multiple-Qubits Gates

Multiple-Qubits gates are not that different from single-qubit gates, they still have to be unitary. Nevertheless, depending on the qubits that the gate is trying to modify, the size of the matrix will change. Formally, if a gate is applying a transformation to n qubits, then the dimensions of the gate's matrix $M \in \mathbb{C}^{2^n \times 2^n}$. Therefore, if $n = 2$ then M would have the size $\mathbb{C}^{4\times4}$.

Controlled NOT Gate

The Controlled NOT (CNOT) gate is a two-qubit quantum gate. It has a *control* qubit and a *target* qubit. In Equation 2.18 we can observe the matrix and symbol representation where the first qubit is the control qubit and the second is the target qubit.

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{aligned} CNOT\,|00\rangle &= |00\rangle \\ CNOT\,|01\rangle &= |01\rangle \\ CNOT\,|10\rangle &= |11\rangle \\ CNOT\,|11\rangle &= |10\rangle \end{aligned} \tag{2.18}$$

The effects of the CNOT gate can be seen in Equation 2.18, where depending on the value of the control qubit, a NOT gate will be applied on the target wire. This can also be represented as $|A, B\rangle \rightarrow |A, B \oplus A\rangle$ performing an XOR between the control and target qubits and storing the value in the second qubit.

Regarding the CNOT matrix in Equation 2.18, an intuiton can be gained regarding the columns of the matrix, the computational basis, and the effects of the gate. The ordering of the columns each represent the input state with regards to the computational basis. The first column represents $|00\rangle$, the second column $|01\rangle$, the third column $|10\rangle$ and so on. Moreover, the values that the vectors represent in each column are associated with the output values after the gate has been applied, e.g. $\begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}^\top = |00\rangle$, $\begin{pmatrix} 0 & 1 & 0 & 0 \end{pmatrix}^\top = |01\rangle$, $\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}^\top = |10\rangle$, etc.

We can observe this phenomena between the third and fourth column in Eq. 2.18, where the columns and values swap. Also this intuiton helps us construct the matrix from the CNOT gate when the control and the target qubits have been inverted. In Equation 2.19 we construct the matrix from the effects the CNOT gate would have in

the different computation basis states.

$$\begin{aligned} CNOT\,|00\rangle &= |00\rangle \\ CNOT\,|01\rangle &= |11\rangle \\ CNOT\,|10\rangle &= |10\rangle \\ CNOT\,|11\rangle &= |01\rangle \end{aligned} \qquad \text{  } \qquad CNOT_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{2.19}$$

SWAP gate

The SWAP gate's effect are introduced in Equation 2.20. It shows that the value of the first qubit is swapped with the value of the second qubit. Therefore, the gate will only have an effect if the qubits' values are different.

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{  } \qquad \begin{aligned} SWAP\,|00\rangle &= |00\rangle \\ SWAP\,|01\rangle &= |10\rangle \\ SWAP\,|10\rangle &= |01\rangle \\ SWAP\,|11\rangle &= |11\rangle \end{aligned} \tag{2.20}$$

The introduced two-qubit gates are also involutory. Therefore, applying them twice would not modify the original quantum state.

2.1.5 Entanglement

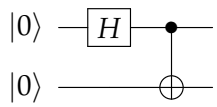
One of quantum mechanics most important concepts is entanglement, it is also what fundamentally enables quantum computing to potentially perform better than classical computing. Entanglement occurs when two or more particles' quantum states are correlated and cannot be described or measured independently of each other.

For two unentangled qubits with quantum states $|\phi\rangle_A$ and $|\psi\rangle_B$ respectively, we can describe the composite system formed by both qubits with the tensor product between their quantum states (Eq. 2.21). Quantum states represented in this manner are referred to as separable states. Entangled systems can't be represented as separable states.

$$|\Psi\rangle_{AB} = |\phi\rangle_A \otimes |\psi\rangle_B \tag{2.21}$$

In quantum computing the simplest and most common sequence of gates to entangle two qubits can be found in Equation 2.22, where the Hadamard gate puts

the first qubit in a superposition and the CNOT gate creates a dependent relationship between both qubits.



$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \tag{2.22}$$

Reminiscing the fact that measuring a qubit will collapse its quantum state to one of the computational basis. In the case of $|\psi\rangle$, according to the effects of Equation 2.6, measuring the first qubit will eliminate one of the two possible states. If the measured value of the first qubit is *0*, then $|\psi'\rangle = |00\rangle$. If *1* is measured, then $|\psi'\rangle = |11\rangle$. In the case where we measured *0*, the second qubit will be *0* when it is measured, else the second qubit would be *1*. Therefore, without measuring the second qubit we can know the value it will take if it was measured.

$$\begin{aligned} |00\rangle &\implies |\Phi^+\rangle = \tfrac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) & |01\rangle &\implies |\Psi^+\rangle = \tfrac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right) \\ |10\rangle &\implies |\Phi^-\rangle = \tfrac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) & |11\rangle &\implies |\Psi^-\rangle = \tfrac{1}{\sqrt{2}}\left(|01\rangle - |10\rangle\right) \end{aligned} \tag{2.23}$$

The quantum states created by the circuit in Equation 2.22 with differring basis inputs are called Bell's states or EPR states [21]. The Bell's states represent the maximally entangled state between two qubits. In Equation 2.23 we can observe the four resulting states, where in the Φ states the value of the measured qubit will be the same as the non-measured qubit. For the Ψ states, the value of the measured qubit will be the opposite of the non-measured qubit. This relationship holds independent on whether the first or the second qubit in the system is measured.

Entanglement is one of the most powerful tools in quantum mechanics. Some of the documented uses of entanglement (mainly using EPR states) are superdense coding [22] and quantum teleportation [23]. Quantum teleportation describes transmitting quantum information from a sender to a receiver in a different location by sending only classical data. Superdense coding refers to transferring a certain amount of classical bits of information using a lesser number of qubits, therefore, sending more information with less resources.

2.1.6 Quantum Measurement

So far performing a measurement will collapse the quantum state to one of the two computational basis. Nonetheless, a quantum state can be measured in several different basis depending on the measurement operator that is being used. Quantum

measurements are defined by a set M_m of measurement operators. This set must fulfill the *completeness equation* property (Eq. 2.24) where all the operators must add to the identity matrix. This property can be interpreted as all the probability amplitudes from the quantum state must add up to one.

$$\sum_m M_m^\dagger M_m = I \implies \sum_m p(m) = \sum_m \langle\psi| M_m^\dagger M_m |\psi\rangle = 1 \tag{2.24}$$

Given a quantum state $|\psi\rangle$ right before a measurement, the probability of measuring m is defined as $p(m) = \langle\psi| M_m^\dagger M_m |\psi\rangle$. The quantum state after the measurement $|\psi'\rangle$ is described in Equation 2.25.

$$|\psi'\rangle = \frac{M_m |\psi\rangle}{\sqrt{\langle\psi| M_m^\dagger M_m |\psi\rangle}} \tag{2.25}$$

We can apply this framework to the computational basis. The measurement operators of the computational basis are $M_0 = |0\rangle\langle 0|$ and $M_1 = |1\rangle\langle 1|$. Both of the operators are Hermitian, thus they are equal to their conjugate transpose and $M_m^\dagger M_m = M_m^2$. Solving the square matrix of both measurement operators yields the original matrix $M_m^2 = M_m$. We can observe now that the completeness property is fulfilled in Equation 2.26.

$$M_0^\dagger M_0 + M_1^\dagger M_1 = M_0 + M_1 = I \tag{2.26}$$

For a state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ we can utilize this new framework to explain what the probability of measuring *0* or *1* is. In Equation 2.27 we can observe that the probability of measuring *0* is $|\alpha|^2$, equal to the magnitude mentioned in Subsection 2.1.1.

$$p(0) = \langle\psi| M_0^\dagger M_0 |\psi\rangle = \langle\psi| M_0 |\psi\rangle = |\alpha|^2 \tag{2.27}$$

With Equation 2.27 we can also easily derive that the probability for measuring the value *1* is $p(1) = |\beta|^2$. In Equation 2.28 we calculate how measuring affects the quantum state, we use Equation 2.25 with the derived probability values. Recalling that a global phase has no effect in a qubit (Subsection 2.1.2), we can assume that $\frac{\alpha}{|\alpha|} |0\rangle = |0\rangle$ and $\frac{\beta}{|\beta|} |1\rangle = |1\rangle$.

$$|\psi_0'\rangle = \frac{M_0 |\psi\rangle}{|\alpha|} = \frac{\alpha}{|\alpha|} |0\rangle \qquad |\psi_1'\rangle = \frac{M_1 |\psi\rangle}{|\beta|} = \frac{\beta}{|\beta|} |1\rangle \tag{2.28}$$

One would assume that knowing the magnitude of the probability amplitudes is enough to reconstruct a quantum state, however, this is not the case. For the states $|+\rangle$ and $|-\rangle$ the probability of measuring the value *0* is $p(0) = \frac{1}{2}$ and the probability of measuring the value *1* is $p(1) = \frac{1}{2}$. Nevertheless, the quantum states $|+\rangle$ and $|-\rangle$ are not equivalent. This occurs because there isn't any quantum measurement capable of discerning between quantum states unless they are orthonormal.

To be able to differentiate $|+\rangle$ and $|-\rangle$ we need to measure in a different basis composed by orthonormal vectors. The normalization property assures that they will be valid quantum states. The orthogonality property guarantees linear independence between vectors, thus, they can describe any quantum state in terms of the basis. In this case we can use $|+\rangle$ and $|-\rangle$ as the new vectors for the basis. We ensure that the vectors are normalized by calculating the inner product of the vectors with themselves $\langle +|+\rangle = \langle -|-\rangle = 1$. We guarantee that the vectors are orthogonal by calculating the inner product of the vectors with each other $\langle +|-\rangle = \langle -|+\rangle = 0$.

$$M_+ = |+\rangle\langle +| = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix} \qquad M_- = |-\rangle\langle -| = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{pmatrix} \tag{2.29}$$

The measurement operators derived from the new basis can be found in Equation 2.29. These operators can now be used to calculate the probability to measure either $|+\rangle$ or $|-\rangle$. Similar to the computational basis case, both measurement projectors are Hermitian. Thus, we can derive the expression for the probability to measure $x \in \{+, -\}$ where $p(x) = \langle \psi | M_x | \psi \rangle$. With this new expression we can clearly differentiate in Equation 2.30 the $|+\rangle$ and $|-\rangle$ quantum states.

$$\begin{aligned} p(+) &= \langle +| M_+ |+\rangle = 1 & \qquad p(-) &= \langle +| M_- |+\rangle = 0 \\ p(+) &= \langle -| M_+ |-\rangle = 0 & \qquad p(-) &= \langle -| M_- |-\rangle = 1 \end{aligned} \tag{2.30}$$

2.1.6.1 Projective Measurement

Projective measurements are a special case of quantum measurements. They are characterized by an *observable*, denoted as M, which is a Hermitian operator acting on the state space of the observed system. The observable's spectral decomposition is in Equation 2.31, where we can see that M is composed by the sum of the projector P_m multiplied by the corresponding eigenvalue m. The observable's eigenvectors form an

orthonormal basis and its eigenvalues are the possible values for the measurement.

$$M = \sum_m m P_m \tag{2.31}$$

From the observable we can derive an expression in Equation 2.32 for the expectation value. This expression is often denoted as $\langle M \rangle$. The expectation value denotes the value of the average measurement with respect to the observable. More importantly, the expectation value can be interpreted as the weighted average of what would be measured over many experiments, thus, analytically calculating the result of several trials.

$$\mathbb{E}(M) = \sum_m m * p(m) = \sum_m m \langle \psi | P_m | \psi \rangle = \langle \psi | \left(\sum_m m P_m \right) | \psi \rangle = \langle \psi | M | \psi \rangle \tag{2.32}$$

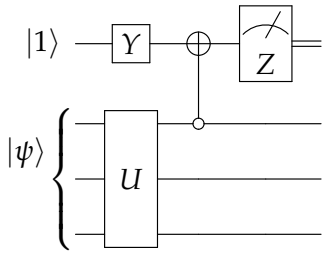
The Pauli Z matrix is of special importance as an observable. In Equation 2.33 we can find the spectral decomposition of the Pauli Z matrix. It has the eigenvalues *1* and *-1*, that correspond to its eigenstates $|0\rangle$ and $|1\rangle$. The eigenstates match the computational basis and form its measurement operators. Hence, the Pauli Z matrix is widely used as an observable for the computational basis.

$$Z = 1 * |0\rangle\langle 0| + (-1) * |1\rangle\langle 1| \tag{2.33}$$

2.1.7 Quantum Circuit

Quantum circuits have already been informally introduced (Eq. 2.22), however, formalizing their structure is essential to understand quantum computing. The main components are the qubit (Subsec. 2.1.1), quantum gates (Subsec. 2.1.4), and quantum measurements (Subsec. 2.1.6). In Equation 2.34 the qubits are represented as *wires* and are abstracted from their physical implementation. The quantum gates in the wire act from left to right and there can be no loops. If there are any gates before a control operation, the gates must be executed before, therefore, creating a dependency between

operations and wires.



(2.34)

If there is no input state on the circuit's left side, the initial quantum state is $|0\rangle$. Otherwise, an initial quantum state affecting one wire can be represented with a ket. Additionally, a multi-qubit initial quantum state can be characterized by a ket containing the quantum state and a curly brace referencing the affected wires. A unitary matrix U acting as a quantum gate on several qubits can be represented as rectangle that covers the modified wires.

When using a control gate (e.g. CNOT) the control wire is represented with a black dot. A white dot can be used if the control value must be $|0\rangle$, which is equivalent to performing an X gate before and after the value is controlled. Finally, the metronome symbol is utilized to indicate a measurement of the qubit value in the wire it appears. An observable can be specified, else, the measurement is performed with respect to the computational basis. The result is an eigenvalue from the obersvable and becomes a classical value, which is represented by using a double-lined wire.

2.1.8 Density Operator

The current definition of quantum states as amplitude state vectors is enough to explain closed quantum systems. However, quantum computers are open quantum systems, meaning that they are influenced by their environment. This influence changes the notion from *pure* into *mixed* quantum states. Mixed quantum states present a probabilistic perspective of quantum states and are represented by the density operator ρ. The density operator is vital for the thesis, as it is utilized to model noise in open quantum systems.

The density operator describes a quantum system's state that is not fully known. In Equation 2.35 we can find the expression for the density operator, where the system is found in the set of states $|\psi_i\rangle$ with probability p_i where $i \in [1, N]$.

$$\rho = \sum_{i=1}^{N} p_i \, |\psi_i\rangle\langle\psi_i| \tag{2.35}$$

The density operator is also called a density matrix and it has three main properties. The first one is that all the density operators are Hermitian. Secondly, the density matrix's trace must be equal to 1. Finally, all density operators are positive semidefinite. Thus, for any quantum state $|\psi\rangle$ we have that $\langle\psi|\,\rho\,|\psi\rangle \geq 0$. For a given density operator ρ, we can quantify how pure the quantum state is. The *purity* γ of ρ is defined as $\gamma(\rho) = \text{tr}\left(\rho^2\right)$. For a pure state ρ, $\gamma(\rho) = 1$, while for a mixed state the purity is less than 1. The maximally mixed state is proportional to the identity matrix and has a purity of $1/d$, where d is the dimension of the Hilbert space where ρ is defined.

In Equation 2.8 we showed how a unitary operation U modifies the quantum state $|\psi\rangle$. Given a state represented by the density operator ρ, in Equation 2.36 we can observer the effects of performing a unitary operation U to the state $|\psi'\rangle$ that appears with probability p_i.

$$\rho \xrightarrow{U} \sum_{i=1}^{N} p_i U\,|\psi_i\rangle\langle\psi_i|\,U^\dagger = U\rho U^\dagger = \rho_U \tag{2.36}$$

The measurement operations also have to be adapted. We perform a measurement with the operator M_m. The probability of measuring m given an initial state $|\psi\rangle$ is described in Equation 2.37.

$$p(m|i) = \langle\psi_i|\,M_m^\dagger M_m\,|\psi_i\rangle = \text{tr}\left(M_m^\dagger M_m\,|\psi_i\rangle\langle\psi_i|\right) \tag{2.37}$$

Utilizing the law of total probability, Equation 2.35 and 2.37 we can further specify in Equation 2.38 the probability of measuring m for the density operator ρ.

$$p(m) = \sum_i p(m|i)\,p_i = \sum_i p_i\,\text{tr}\left(M_m^\dagger M_m\,|\psi_i\rangle\langle\psi_i|\right) = \text{tr}\left(M_m^\dagger M_m \rho\right) \tag{2.38}$$

To finalize the measurement adaptations, ρ_m is the state of the density operator after measuring m. It is described by Equation 2.39.

$$\rho_m = \frac{M_m \rho M_m^\dagger}{\text{tr}\left(M_m^\dagger M_m \rho\right)} \tag{2.39}$$

If M is an observable, using Equation 2.32, we derive the expression for the expectation value of M given state ρ in Equation 2.40. This derivation also takes

advantage from the linearity property of the trace.

$$\langle M \rangle = \sum_i p_i \langle \psi_i | M | \psi_i \rangle = \sum_i p_i \operatorname{tr}\left(|\psi_i\rangle\langle\psi_i| M \right) = \operatorname{tr}\left(\sum_i p_i |\psi_i\rangle\langle\psi_i| M \right) = \operatorname{tr}(\rho M) \tag{2.40}$$

2.2 Quantum Noise

In this section we introduce the fundamental concepts to understand how noise affects quantum systems. The existing types of quantum noise will be presented in Subsection 2.2.1. Furthermore, we present the theory of quantum operations and the different classes of quantum operations regarding noise in Subsection 2.2.2.

Significant quantum noise commonly originates from the interaction of the quantum system with its environment. While a completely isolated quantum system would considerably reduce the noise from the environment, it would prevent us from performing any operation to the quantum state. This concept is known as the coherence-controllability trade-off [24].

Due to this trade-off, open quantum systems theory has been developed. Designing a model for the interactions between a quantum system and its environment is key to accurately describe quantum operations that happen in the real world.

2.2.1 Types of Quantum Noise

Quantum noise can be categorized into two types, coherent and incoherent noise [25]. Coherent noise is most commonly derived from systematic errors like miscalibration, inaccurate control signal, or coupling to low-frequency fields [26]. These disturbances are constant, reversible and don't cause any information loss. Coherent noise can be characterized by unitary matrices being applied to the quantum state.

A specific case of coherent noise is miscalibration on Pauli rotational gates. Given a desired rotation θ, the rotational gate incurs an error ϵ. In Equation 2.41 we can notice how one miscalibrated gate can be interpreted as applying the desired rotation θ and then applying an additional rotation by ϵ. In this case the unitary matrix which describes the coherent error is the matrix describing the rotational gate by the ϵ angle.

$$R_x(\theta + \epsilon) = R_x(\theta) R_x(\epsilon) \tag{2.41}$$

Contrarily, incoherent noise is stochastic and emerges from insufficiently isolating the quantum system from its surroundings. This in turn results on irreversible information loss. Incoherence is often represented as a probabilistic distribution of unitary processes resulting from experimental parameters that change gradually over time [27]. Therefore, incoherent noise cannot be explained by a single unitary matrix but by quantum noise operations. Quantum noise operations will be introduced in Subsection 2.2.2 and are described as completely positive superoperators.

An important difference between coherent and incoherent noise is the magnitude of the influence that they have on a quantum system. Coherent noise's influence on a quantum system in some cases grows quadratically with respect to the circuit's length [28]. Conversely, for incoherent noise, the influence on a quantum system grows at worst linearly with the circuit's size.

Once we know which type of noise we might encounter performing operations in a quantum system, it is of special importance to know when and where it can occur. There are three main places in a quantum algorithm where noise can be present. The first occasion were we might encounter is when preparing the initial state of the quantum system. The second location where noise can be found is when performing any gate quantum gate. Finally, every time a measurement is performed we will come across noise. It is remarkable to note that both, coherent and incoherent noise, can appear in any of these places, even at the same time.

While noise in every part of the quantum system is significant, there are some operations that are constantly repeated and therefore will result in noisier calculations [29]. Noise when executing quantum gates is, on average, responsible for most of the induced noise, as these operations are constantly repeated depending on the length of the circuit. Normally, state preparation and measurements are performed once at the beginning and at the end of the quantum circuit respectively. Consequently, the noise from both operations has a diminished impact on the quantum system in comparison to the quantum gates' noise.

2.2.2 Types of Quantum Noise Operations

Quantum operations are utilized to characterize the effect of the environment on open quantum systems. In Equation 2.42 we derive the expression to describe a quantum operation, this form is called the operator-sum representation [30]. This equation extends the expression in Eq. 2.36 with Kraus operators $\{K_i\}$ [31].

$$\Phi(\rho) = \sum_i K_i \rho K_i^\dagger \tag{2.42}$$

The quantum operation Φ is a completely positive map which describes the change of quantum state ρ with Kraus operators $\{K_i\}$. For non-trace-preserving quantum operations (like noise) a relaxation of the completeness equation (Eq. 2.24) for the Kraus operators is displayed in Equation 2.43.

$$\sum_i K_i^\dagger K_i \leq I \tag{2.43}$$

The operator-sum representation encapsulates the quantum system's dynamics without needing explicit consideration of environmental properties. All the relevant information is condensed into the Kraus operators. Therefore, if we can derive the Kraus operators, we can observe the effects of the environment on the quantum system.

Bit Flip

A bit flip is characterized by the Kraus operators in Equation 2.44. These operators indicate that a bit flip will happen to the quantum state with probability p, while the state will not change with probability $1-p$. If $p=1$ then the operation will just be like applying a X gate.

$$K_0 = \sqrt{1-p}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \qquad K_1 = \sqrt{p}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \tag{2.44}$$

The effects of quantum operations can be illustrated on the Bloch sphere. This depiction is very helpful to understand consequence of noise on quantum states. In Figure 2.2 we can observe the effects of the bit flip channel with $p=0.3$. The distorted bloch sphere shows that the x-axis remains unaffected but the $y-z$ plane is compressed, limiting the possible values for the quantum state.

Phase Flip

A phase flip is defined by the Kraus operators detailed in Equation 2.45. These operators signify that a phase flip may occur on the quantum state with a probability of p, whereas the state will remain unchanged with a probability of $1-p$. If $p=1$ then the operation will essentially apply a Z gate.

$$K_0 = \sqrt{1-p}\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \qquad K_1 = \sqrt{p}\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \tag{2.45}$$

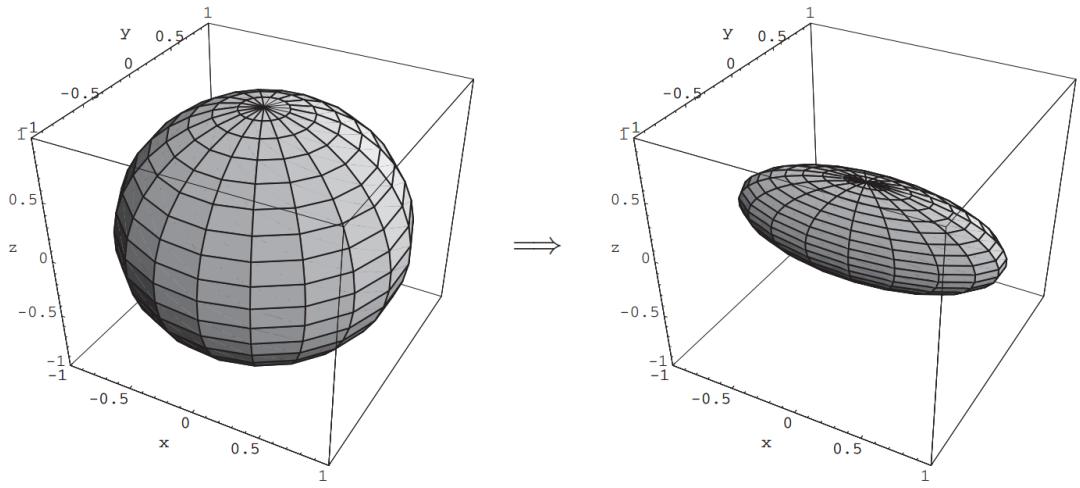


Figure 2.2: Effects of a bit flip channel on the Bloch sphere. Taken from [15].

Figure 2.3 depicts the impact of the bit flip channel when $p = 0.3$. We can appreciate that in the modified Bloch sphere the z-axis is left untouched while the $x - y$ plane is reduced. Thus, the quantum state becomes less pure.

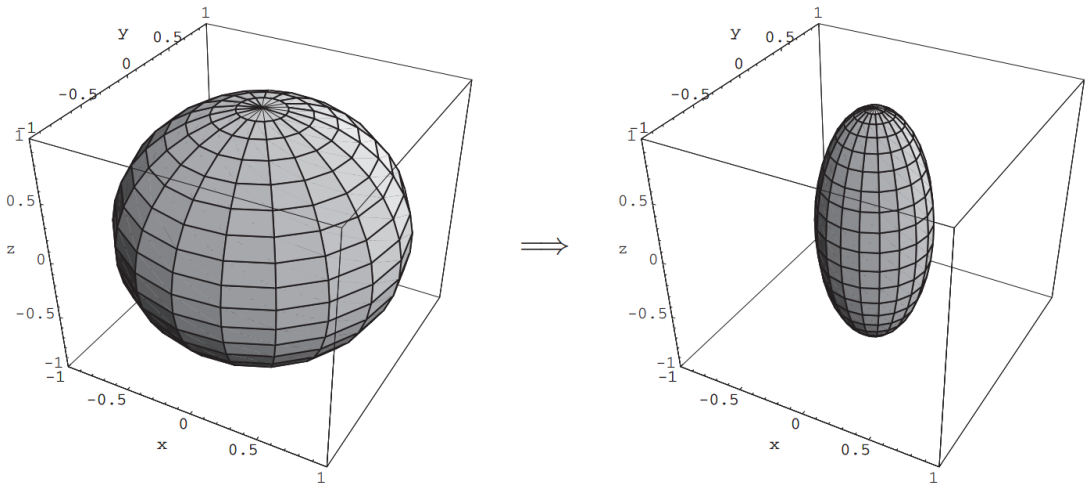


Figure 2.3: Effects of a phase flip channel on the Bloch sphere. Taken from [15].

Depolarizing Channel

The effects that the depolarizing channel has on a quantum state are described by the Kraus operators in Equation 2.46. The operators describe that with probability $p/3$ each of the operators X, Y, and Z will be performed. Therefore, the quantum state is not modified with probability $1-p$. We can further define the effect of these operators as *depolarizing* the quantum state with probability p. Depolarizing a quantum state means substituting ρ for the maximally mixed state.

$$\begin{aligned} K_0 &= \sqrt{1-p}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & K_1 &= \sqrt{p/3}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ K_2 &= \sqrt{p/3}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} & K_3 &= \sqrt{p/3}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \end{aligned} \tag{2.46}$$

In Figure 2.4, we observe the effects of the depolarizing channel with a probability of $p = 0.5$. Notably, the magnitude shrinks evenly in the modified Bloch sphere. Consequently, the quantum state experiences a decrease range from all axis.

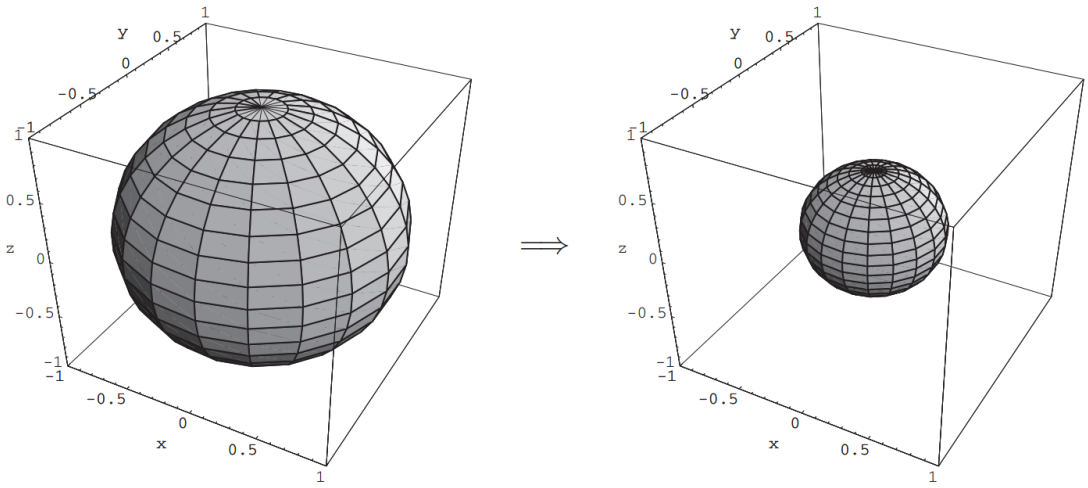


Figure 2.4: Effects of a depolarizing channel on the Bloch sphere. Taken from [15].

Amplitude Damping

The effects of amplitude damping can be equated to *energy dissipation* in the quantum system. Interaction with the environment might lead to physycal phenomena like scattering, dissipation, attenuation, and spontaneous emission, which are all described by the effects of amplitude damping. An amplitude damping channel is defined by the Kraus operators outlined in Equation 2.47. The K_0 operator leaves $|0\rangle$ unchanged but reduces the amplitude of $|1\rangle$. This occurs because there was no loss of energy to the environment, leading the environment to believe the system is probably in $|0\rangle$ rather than $|1\rangle$. The K_1 operator performs a a bit flip to $|1\rangle$ and reduces its amplitude by a factor of $\sqrt{\gamma}$, where γ is the damping strength. This operation signifys that the state lost energy to the environment.

$$K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix} \qquad K_1 = \begin{pmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{pmatrix} \tag{2.47}$$

Figure 2.5 illustrates the impact of the amplitude damping channel with a probability of $p = 0.8$. Remarkably, the magnitude contracts across the modified Bloch sphere and is pulled towards the $|0\rangle$ pole. As a result, the quantum state undergoes a reduction in possible quantum states values.

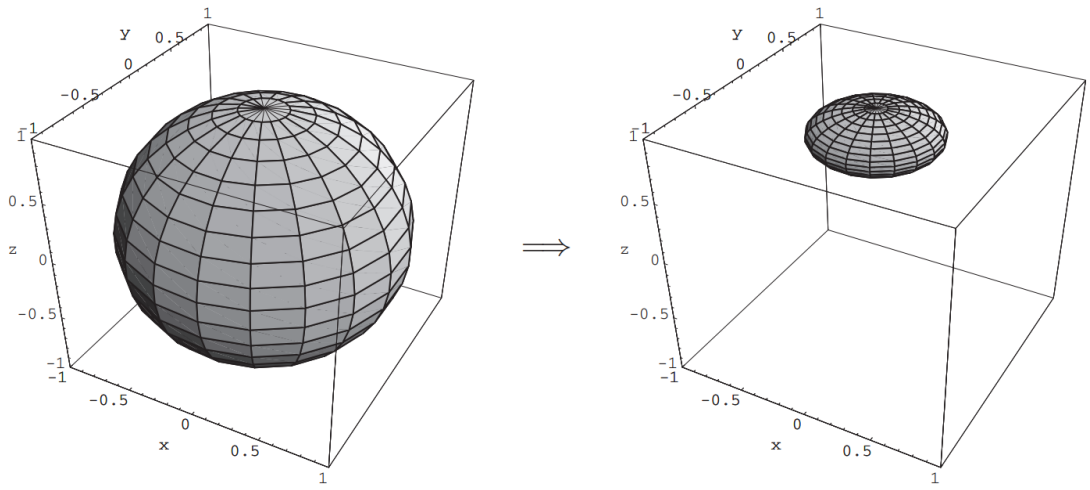


Figure 2.5: Effects of an amplitude damping channel on the Bloch sphere. Taken from [15].

Phase Damping

The effects of phase damping are unique to quantum mechanics, where a loss of information occurs without losing any energy. A phase damping channel is characterized by the Kraus operators presented in Equation 2.48. Similar to amplitude damping's K_0, $|0\rangle$ remains unchanged and the amplitude of $|1\rangle$ is reduced. However, K_1 consumes $|0\rangle$ and reduces the amplitude of $|1\rangle$.

$$K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{pmatrix} \qquad K_1 = \begin{pmatrix} 0 & 0 \\ 0 & \sqrt{\gamma} \end{pmatrix} \tag{2.48}$$

To better understand the effects of phase damping, we can derive equivalent but differently expressed Kraus operators. In Equation 2.49 with $\alpha = \left(1 + \sqrt{1-\lambda}\right)$ we can rewrite the Kraus operators such that they correspond to the phase flip's operators with the probabilities inversed. This is important because while the physical quantum process don't match, the transformation of the quantum state due to the noise is the same. Thus, the effect of phase damping on the Bloch sphere can be observed in Figure 2.3.

$$K_0' = \sqrt{\alpha} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad K_1' = \sqrt{1-\alpha} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.49}$$

2.3 Quantum Machine Learning

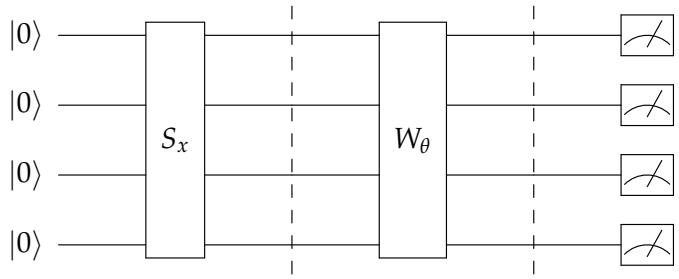
In this section we introduce the fundamental concepts to understand the intersection between quantum computing and machine learning, for short QML. A brief introduction to QML and its main structure follows suit. Furthermore, in Subsection 2.3.1 Variational Quantum Algorithm (VQA) will be presented, taking a focus on the Circuit-Centric ansatz.

The main task of Machine Learning (ML) is to find patterns in data. Consequently, Quantum Machine Learning (QML) takes this goal and extends it with the use of quantum algorithms to find patterns in data. Small quantum algorithms can produce statistical distributions that are difficult for ML to identify. Thus, it is possible that QML might identify patterns that classical ML cannot [32]. Another interesting prospect of QML is that it could potentially identify patterns with less computational effort than ML, this is known as a *quantum speedup*.

There are four possible combinations between the intersection of ML and quantum computing. These approaches depend on mixing the type of data and the processing

device being utilized [9]. In this thesis we refer to QML as the use of a quantum device to handle classical data. For this modality to work, QML requires a data interface to encode classical data into the quantum device. Current NISQ devices enable small QML models with few qubits and quantum gates. This implies that the whole QML pipeline is not able to run on a NISQ device. Consequently, the training of QML models is implemented in a classical device.

In Equation 2.50 we can find a quantum circuit representing a QML model. In general, QML models consist of three main parts [33]. At the beginning, each model must have an encoding component S_x to transform classical data into a suitable input for a quantum device. This is followed by the processing subroutine W_θ, which is composed by several quantum gates and is responsible for modifying the input according to the task. Finally, a measurement component can be found. The measurement's statistical properties are subsequently interpreted as the model's output.



(2.50)

The encoding component S_x symbolizes all possible strategies to represent a data point in an appropiate quantum state. In quantum computing, this process is achieved by performing a state preparation routine. Some encoding schemes are basis, angle and amplitude encoding [34]. While each scheme has its own characteristics, we will mainly focus on amplitude encoding. Amplitude encoding [35] embeds the data point features into the amplitude probabilities of a quantum state. For a data point $x \in \mathbb{C}^N$ the corresponding quantum state $|\psi_x\rangle$ can be derived in Equation 2.51:

$$|\psi_x\rangle = \sum_{i=0}^{N-1} x_i |i\rangle \tag{2.51}$$

The data point x must fulfill two prerequisites. The first requirement is that it must be normalized such that $\sum_i x_i^2 = 1$. The second condition is that the data point must be padded such that N is a power of 2. These prerequisites ensure that $|\psi_x\rangle$ is a valid quantum state according to the Born rule. Additionally, for N features the QML model will need $\log(N)$ qubits to properly encode them.

QML model training is performed similarly to classical ML. The parameters that conform the processing subroutine W_θ are updated by trying to minimize a cost funtion.

This update happens iteratively in a feedback loop, either by a black box mechanism or by looking into W_θ's properties (e.g. gradients) and adapting them accordingly.

With regards to the result's interpretation of QML models, the most common technique is to measure the expectation value of m qubits and assign them to the k possible different classes. The relationship between m and k is $m = \lceil \log(k) \rceil$, therefore the maximum number of classes is bound to the number of qubits the model utilizes.

$$|\psi_{x,\theta}\rangle = W_\theta |\psi_x\rangle , \qquad f_\theta(x) = \langle \psi_{x,\theta}|M|\psi_{x,\theta}\rangle = \langle M \rangle_{x,\theta} \tag{2.52}$$

In Equation 2.52 we can observe that the model can be represented by a function f with parameters θ and input x. In order to perform a prediction we need to prepare the quantum state according to x (Eq. 2.51). Then, the resulting state is processed by the W_θ subroutine. Finally, with the observable M we measure the required amount of qubits. Each qubit measurement value will lay between the range of the observable's eigenvalues. Therefore, a threshold function is utilized to map the input to 0 or 1. Afterwards, all the measurements are binary decoded into the model's classes.

2.3.1 Variational Quantum Algorithm

VQA is a specific type of quantum algorithm. It attempts to gain a quantum advantage with NISQ devices. Currently, there are VQAs that have many diverse applications, in our specific case they can be used for QML models as classifiers or generative models [36].

The required processing subroutine W_θ for QML is denominated an *ansatz*. Ansätze can be tailored specifically for the classification problem or they can be application agnostic. Different classical techniques can be used to train the parameters from the ansatz, these include gradient-based and gradient-free methods [37]. A popular option is ADAM [38], a first-order gradient-based optimization process.

In this thesis we will focus on the Circuit-Centric Quantum Classifier [39] ansatz. A characteristic of this VQA variation is that it utilizes a shallow quantum circuit, this enables it to be run on NISQ devices. However, the trade-off is that only a small subset of Hilbert space will be represented, limiting the flexibility of the classifier.

Each layer in the ansatz is formed by a sequence of R_Z, R_Y and R_Z rotational gates with arbitrary rotational parameters applied to each wire. These rotational gates are followed by $CNOT$ gates that entangle each pair of wires. The expressivity of the ansatz increases with the number of layers. Nevertheless, the complexity and difficulty of training increases as well.

2.4 Adversarial Machine Learning

In this section we introduce the fundamental concepts of adversarial machine learning. A brief introduction to the characterization of AML follows suit. Furthermore, in Subsection 2.4.1 the Fast Gradient Sign Method (FGSM) attack will be presented, while in Subsection 2.4.2 the Projected Gradient Descent (PGD) attack will be introduced.

Adversarial Machine Learning is a subfield of ML and focuses on finding vulnerabilities of models to adversarial examples (Figure 2.6). These adversarial examples are imperceptible specific perturbations applied to a test input to force a ML model to misclassify the input [10]. These perturbations are created by optimizing the input by maximizing the model's prediction error while minimizing the magnitude of the perturbation.

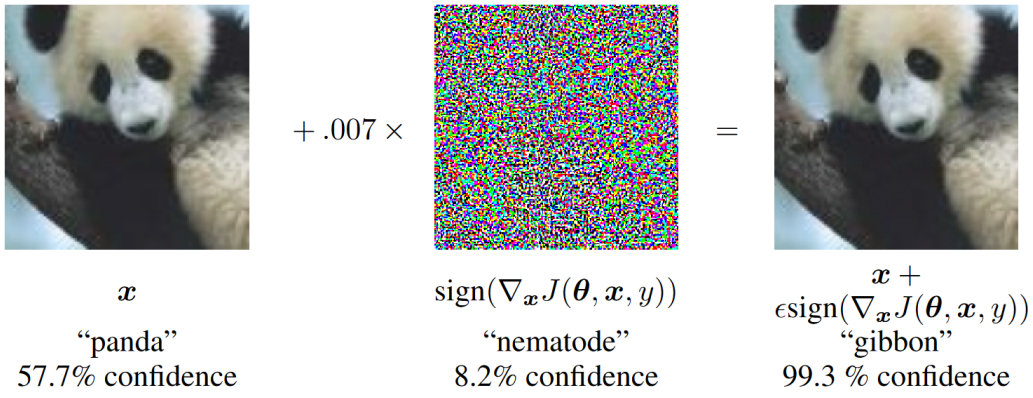


Figure 2.6: Adversarial example where an imperceptible perturbation is added to the input picture to force a misclassification of the ML model. Image taken from [12].

There are two attack variants to generate adversarial examples, namely, targeted and untargeted attacks [40]. For untargeted attacks,the goal is to find a perturbation δ such that $f_\theta(x+\delta) \neq y$, where f is the ML model with trained θ parameters, x is the input data, and y the corresponding target class. For targeted attacks, a perturbation δ is sought such that $f_\theta(x+\delta) = y^*$ where y^* is the target misclassification class different than y. The difference between both types of attacks is that for targeted attacks the adversarial example is misleading the model to predict a specific target label, while

untargeted attacks force an erroneous prediction to any class.

Another category to characterize adversarial attacks is based on the available information for the attacker. If the attacker has access to the model's architecture, its training data, and parameters, it is denominated a white-box attack. Otherwise, it would be considered a black-box [41]. While getting access to a target ML model might be impossible, transferability from adversarial examples between ML models that have been trained on disjoint datasets and with different hyperparameters has been found [11]. Therefore, an attacker only needs to train its own network and perform a white-box attack on it to generate adversarial examples that also affect the target ML model.

Adversarial examples uncover the weaknesses of high accuracy ML models, revealing that the models are not generalizing well, as small modifications to the original input lead to significant errors in classification. Furthermore, a trade-off between model robustness and model accuracy exists [42], thus the need for techniques that do not significantly decrease the model's accuracy while making it as robust as possible. Understanding and defending against adversarial attacks is essential in critical applications utilizing ML, as exploiting the model's vulnerabilities might result in severe real-world harm.

Moreover, several different techniques have been developed in order to generate more efficient and/or more powerful adversarial examples. In this thesis we will focus on FGSM in Subsection 2.4.1 (due to its simplicity and efficiency), and on PGD in Subsection 2.4.2 (due to its high effectiveness with smaller perturbations.)

2.4.1 Fast Gradient Sign Method

The FGSM [12] technique is one of the earliest methods created to quickly develop adversarial examples. FGSM exploits the gradients of the loss function with respect to the input data to manufacture adversarial perturbations.

We can find the definition of an adversarial example according to FGSM's methodology in Equation 2.53, where given a ML model trained on parameters θ, an input data x and its corresponding label y, and the cost function $J(\theta, x, y)$ to train the model, we can obtain an adversarial example x'.

$$x' = x + \epsilon \cdot \text{sign}(\nabla_x J(\theta, x, y)) \tag{2.53}$$

Additionally, a strength regulating factor ϵ is added to control the magnitude of the adversarial perturbation to x. The higher ϵ becomes, the bigger the adversarial

perturbation will be. This leads to adversarial examples that have a higher probability to be misclassified, however, the modified input will start to become more evident.

One of the main benefits of FGSM is that this technique is computationally efficient, as it only needs to calculate once the gradient of the loss function to obtain an adversarial example. Another advantage is that this technique is very simple, therefore, it is straightforward to implement and execute. Nevertheless, FGSM generates adversarial examples in just one step, which does not allow for fine-tuning adversarial examples.

2.4.2 Projected Gradient Descent

The PGD [43] technique is a more powerful method to generate adversarial examples than FGSM, it is considered the strongest attack using the network's local first order information. PGD is derived by iteratively applying the gradient updates to the input data while projecting the calculated perturbation into a predetermined area.

In Equation 2.54 we can find the definition of an adversarial example after $k+1$ iterations, where $x^{(k)}$ is the previous adversarial example after k iterations, α is a strength regulator factor and Π_{x+S} is the projection function to limit the perturbation to an allowed region.

$$x^{(k+1)} = \Pi_{x+S}\left(x^{(k)} + \alpha \cdot \text{sign}(\nabla_x J(\theta, x^{(k)}, y))\right) \tag{2.54}$$

We can observe that the formulation is very similar to FGSM, however, the iterative and the projection component enable PGD to find more inconspicuous adversarial examples that still effectively fool the target ML model. The effectivity of PGD depends on the number of iterations, where higher iterations lead to a higher probability to force a misclassification. Therefore, PGD is able to create adversarial examples that are harder to defend from. Nevertheless, PGD requires more resources to craft an adversarial example than FGSM. Another disadvantage from PGD is that it becomes more sensitive to the values from the iterations and the strength regulator factor.

Finally, in this thesis we will focus on white-box untargeted attacks utilizing both the FGSM and PGD methods to attack a QML model based on a VQA ansatz. In Chapter 4 the implementation of this techniques will be presented. In Chapter 5 we can observe the effects of the adversarial attacks on the different QML models trained with different quantum noise sources. Understanding AML attacks is essential for developing more robust models and improving the security of machine learning systems in practice.

3 Design

Describe how the experiments have been chosen and created: i. Possible mixes with different types of noise, in different places, with different datasets and QML mechanisms.

3.1 Datasets

In this section we will present which datasets are used in the thesis and the reasons why they were included. Current capabilities of quantum computers do not allow for big and complex quantum circuits to be executed. Thus, most of the datasets will have a smaller feature set in comparison to datasets used to benchmark classical ML models.

The dataset selection was heavily influenced by the datasets utilized in [44]. All the datasets but the Plus-Minus Dataset were retrieved from OpenML [45] using Scikit-learn [46] in Python. All data points that had missing features were removed, as well as all duplicate data points. Finally, all the datasets (except the Iris Flower Dataset 3.1.1) have been normalized per feature using the standard deviation and the resulting values were rescaled into the range between [0,1].

3.1.1 Iris Flower Dataset

The Iris Flower Dataset [47] is a collection of 150 data points regarding three different species of the Iris flower. Four features were collected, namely the length and width of the sepals and petals. For this dataset we made a special adaptation such that we would replicate the dataset used by [48]. This adjustment is removing all data points with the class label *virginica* and setting the feature *petal width* to 0.

In Figure 3.1 we can observe the purpose of these modifications. By removing the *virginica* class we can linearly separate the dataset by using two different features, e.g. the *petal length* and *sepal length*. Furthermore, removing the feature *petal width* makes

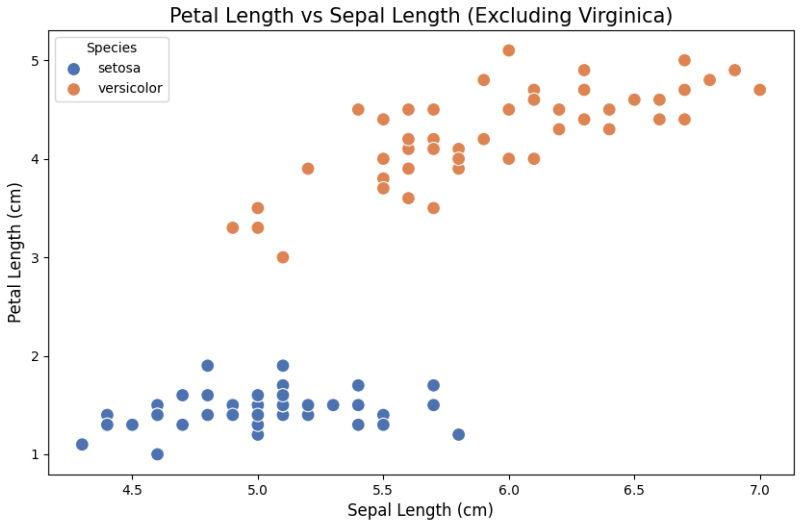


Figure 3.1: Data plot of the Iris dataset's *Sepal Length* and *Sepal Length* features.

the model focus on the other features, eliminating overhead and ensuring the QML will converge to an optimal solution quicker.

3.1.2 Pima Indians Diabetes Dataset

The Pima Indians Diabetes (PID) Dataset [49] contains 768 data points collecting 8 features that were considered risk factors for the onset of diabetes. This dataset presents a class imbalance. To remediate the asymmetry in the classes distribution we removed 232 data points from the *negative* class. The removal was performed randomly with a fixed seed to always keep the same data points and make the results reproducible.

A class imbalance can lead to reduced performance of a ML model [50]. This decrease in accuracy could probably be translated to QML models as well. Therefore, with the removal of the data points from the majority class, we obtain a completely balanced dataset and model performance should be maximized.

3.1.3 Wisconsin Breast Cancer (Diagnostic) Dataset

The Wisconsin Breast Cancer Dataset [51] is composed by 699 data points collecting

10 features of breast masses that could potentially be carcinogenic. While this set is also unbalanced as the original PID Dataset 3.1.2, the asymmetry is not as pronounced. We left the class distribution untouched to check whether the QML models would overcome class distribution imbalance.

It is interesting to note that in [44] they reported that normalizing the dataset per feature with the standard deviation hindered the convergence of the QML models. However, we didn't experience this when training our QML models, thus, we do normalize the data with the standard deviation per feature.

3.1.4 Plus-Minus Dataset

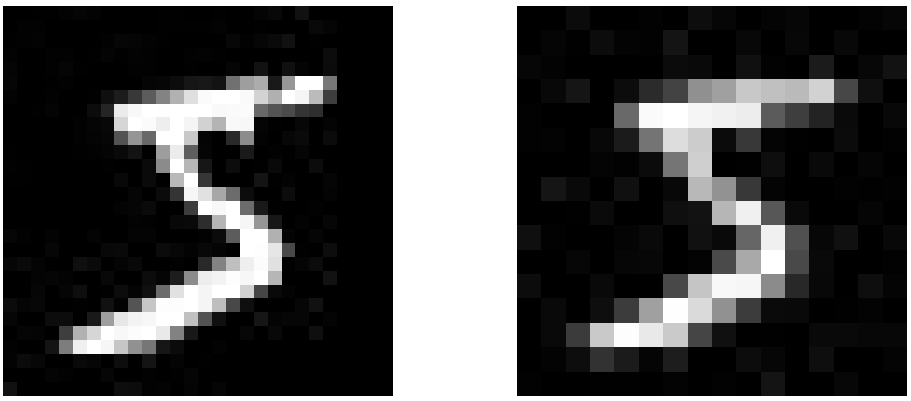
The Plus-Minus Dataset [52] is made of 1,200 grayscale images, each containing $18 \times 18 = 324$ pixels. Inside each image we can observe four different classes, respectively the symbols $-$, $+$, $\vdash$ and $\dashv$. By utilizing this dataset, we will be able to interpret the perturbations caused by the adversarial attacks better. For example, if an adversarial attack trying to force the ML model into misclassifying $\vdash$ into a $+$, the most obvious semantical modification would be to add an horizontal line.

In [53], they observed that QML models focus on different more robust features in the input data than ML models. This dataset will help us verify this claim and provide deeper insight into the semantic meaning of adversarial attacks, specifically in the quantum setting.

3.1.5 MNIST Dataset

The MNIST Dataset [54] is composed by 70,000 images representing handwritten individual digits between the range from 0 to 9. Each image is a monocromatic gray scale image shaped by $28 \times 28 = 746$ pixels. A QML model would require 10 qubits to encode the features with the amplitude embedding technique. Because the amount of operations required for the QML model to process the data increments exponentially, the model quickly becomes intractable to train in commonly available hardware. Therefore, we downsample the images by half to $16 \times 16 = 256$ and the QML model would now only require 8 qubits. In Figure 3.2 we can observe that downsampling the images does not significatively visually affect the image recognition.

Similar to [55] we further divide the MNIST set to simulate a binary classifier and a 4-class classifier task. The resulting datasets are named MNIST2 and MNIST4 respectively. To obtain the MNIST2 dataset, we exclude all classes but the images that



(a) Original image. (b) Downsampled image.

Figure 3.2: Visual comparison between both images.

contain a one or a nine. Furthermore, to retrieve the MNIST4 dataset we kept all the images with digits one, three, seven and nine. With this extension we can now compare how the QML models accuracy varies with the amount of classes to categorize.

3.2 State Preparation with Coherent Noise

In the next chapter (Sec. 4.1) the trained variational circuit classifiers utilize the amplitude embedding technique to encode classical data into the QML model. Pennylanes's `AmplitudeEmbedding` [56] is utilized to encode the data accordingly. In this subsection we will present the impact of state preparation with coherent noise on quantum circuits.

In order to simulate coherent noise we used Qiskit's Aer noise simulator. Qiskit's noise models can be linked with Pennylane by declaring them as an argument when creating a device. We attach coherent noise per gate with the noise's corresponding unitary matrix. The unitary matrices are derived with their respective rotational matrices R_Y, R_Z and CR_X with a small $\theta = \epsilon$. In this case we set ϵ to be around 0.175 to simulate a 10° gate miscalibration.

The test circuits are composed first by an `AmplitudeEmbedding` state preparation layer. Then, a single quantum gate follows that will characterize the test circuit. In total there are three test circuits that describe the effect of coherent noise on the R_Y, R_Z and CNOT gates. These gates were chosen because they are utilized by the strongly entangling layers ansatz. For the rotational gates, a $\theta = \pi$ angle was chosen to simulate their equivalent non-rotational gates.

In Table 3.1 we can observe the ideal noiseless results for the $R_Y(\pi)$ gate in the third column. This table yields the same result as applying the Pauli Y gate. We use the Pauli Z matrix as the observable. The projective measurement values will be between the range of -1 and 1.

Quantum State	$R_Y(\pi)$	$\langle Z \rangle$	Expected $R_Y(\pi + \epsilon)$	Obtained $R_Y(\pi + \epsilon)$
$\|0\rangle$	$i\|1\rangle$	-1	-0.985	-0.985
$\|1\rangle$	$-i\|0\rangle$	1	0.985	0.940
$\|+\rangle$	$-i\|-\rangle$	0	0.174	0.340

Table 3.1: Expectation value with Pauli Z observable of R_Y gate with $\theta = \pi$ and $\epsilon = 10°$. The fourth column represents the calculated effects of coherent noise without state preparation. The fifth column represents the results of coherent noise with state preparation and Qiskit's Aer device.

The calculated expectation values for the $R_Y(\theta + \epsilon)$ gate can be found in the fourth column of Table 3.1. We can observe that the measurements from the resulting quantum state are perturbed. For the computational basis states the expected value is equally reduced in magnitude. For the superposition state $|+\rangle$, the expected value 0 is shifted by the coherent noise. The magnitude of the disturbances from the measurements is directly proportional to ϵ.

The experimental results from adding coherent noise to the $R_Y(\theta)$ gate are presented in the fifth column of Table 3.1. While for the $|0\rangle$ state the result does match the calculated -0.985 value, for the other two tested quantum states there is a notable difference between the calculated and the experimental results.

In relation to the test from the $R_Z(\theta + \epsilon)$ gate, we are not able to observe any effect of either the rotation or the noise. This is because any Z transformation only modifies the phase of the quantum state, which cannot be measured with the Pauli Z observable. However, it is important to note that in the experimental results a difference of around 0.001 was observed in the result of the $|+\rangle$ state measurement.

For the CNOT gate circuit we tested both cases (a control qubit on the first wire and a target qubit on the second wire and vice versa). For the first case of the CNOT gate circuit test, the noiseless expectation values with the Pauli Z observable can be found in the third column of Table 3.2. Similarly to the $R_Y(\theta)$ circuit test, the expectation values lie between a range of -1 and 1. Contrarily, for the case of the CNOT gate we measure at the end of the circuit the two qubits from the quantum state.

In Table 3.2 we can find in the fourth column the calculated projective measure-

Quantum State	CNOT	$\langle Z \rangle$	Expected $\langle Z \rangle + \epsilon$	Obtained $\langle Z \rangle + \epsilon$
$\|00\rangle$	$\|00\rangle$	[1, 1]	[1, 1]	[1, 1]
$\|01\rangle$	$\|01\rangle$	[1, -1]	[1, -1]	[0.992, -0.996]
$\|10\rangle$	$\|11\rangle$	[-1, -1]	[-1, -0.985]	[-0.985, -1]
$\|11\rangle$	$\|10\rangle$	[-1, 1]	[-1, 0.985]	[-0.992, 0.996]

Table 3.2: Expectation value with Pauli Z observable of CNOT gate with $\epsilon = 10^{\circ}$. The fourth column represents the calculated effects of coherent noise without state preparation. The fifth column represents the results of coherent noise with state preparation and Qiskit's Aer device.

ments with the Pauli Z matrix for the CNOT gate. First of all we can observe that for the quantum states $|00\rangle$ and $|01\rangle$ the values don't incur in any noise as the control qubit is set to 0 and the Pauli X rotation is not performed on the target qubit. Notwithstanding, for $|10\rangle$ and $|11\rangle$, where the control qubit is set, noise should be present in the second qubit's measurements with an equal magnitude.

The experimental results from adding coherent noise to the CNOT gate are presented in the fifth column of Table 3.2. Although the result for the $|00\rangle$ state is the same as the previously calculated value, the experimental values obtained for the other three states differ. For the $|01\rangle$ state there should be no noise present, however, both qubit measurements show disturbances in their value. Regarding the $|10\rangle$ state the measurement values for both qubits are inverted. Finally, for the $|11\rangle$ state the first qubit shows signs of noise when it should not, while the second qubit exhibits a different noise magnitude than calculated.

In theory the effects of coherent noise should be deterministic, as their effects can be described by a unitary matrix applied every time once a gate has been executed. In spite of that, the observed results from the experimental tests indicate that coherent noise with Qiskit's noise models plugged into Pennylane are not deterministic and fluctuate between a small margin range.

All the gates that we tested presented differences with the calculated and obtained results. After further investigation we determined that the additional noise arises from the state preparation routine. In Pennylane's documentation we found that the `AmplitudeEmbedding` function inherits from the `StatePrep` function. Furthermore, there is a note that indicates if `StatePrep` is not supported by the device, the method developped by Möttönen et al. [57] will be utilized.

The state preparation method proposed by Möttönen et al. utilizes a combination of $R_Y(\theta)$, $R_Z(\theta)$ and CNOT gates to transform a quantum state $|a\rangle$ into another quantum state $|b\rangle$. For n qubits this method has an upper bound of $2^{n+2} - 4n - 4$ CNOT gates

and $2^{n+2} - 5$ one-qubit elementary rotation gates (both $R_Y(\theta)$ and $R_Z(\theta)$). These bounds grow exponentially, therefore their impact in conjunction with quantum noise will be greatly augmented with the number of qubits used. On the other hand, the bounds can be halved for special cases, e.g. when the initial or the final quantum state is a basis vector.

We performed some tests to see the impact state preparation has on a quantum circuit running on Qiskit's Aer noise simulator. The experiment consists on transforming the initial quantum state $|0_n\rangle$ (where n is the number of qubits) to the quantum state where the magnitude of all the possible amplitude probabilities is equal.

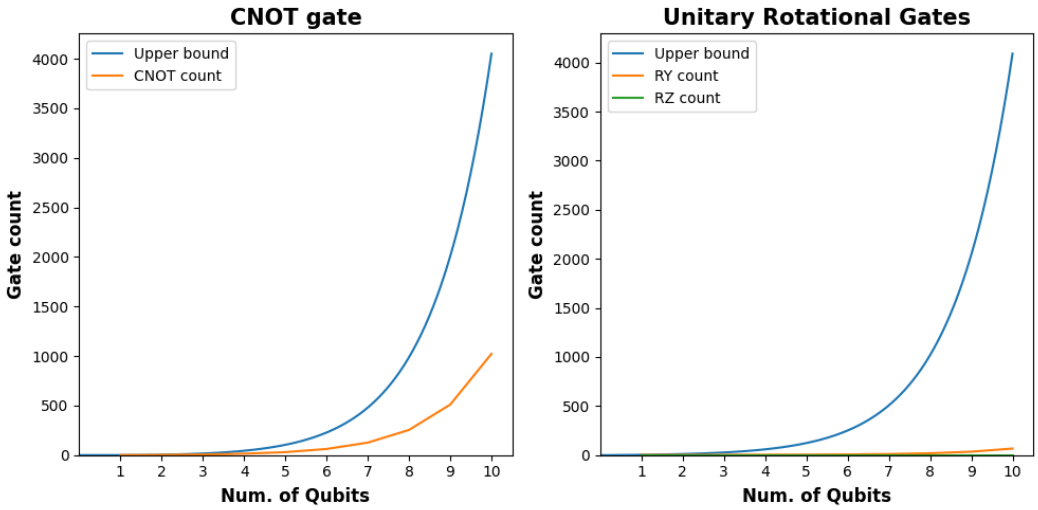


Figure 3.3: Observed required gates to perform state preparation with Möttönen's method.

In Figure 3.3 the theoretical upper bounds of Möttönen's state preparation method for both types of gates can be distinguished from the obtained gate counts in the experiment. While we can experimentally appreciate that the actual gate count does not grow as fast as the theoretical upper bounds (mainly because of the special case where we start from a computational basis vector), we can still observe an exponential growth in the CNOT gate count. Thus, if we perform state preparation with Möttönen's method, it will represent a significant source of noise impacting the QML model.

To better comprehend the previously obtained coherent noise experimental results we created an artificial coherent noise simulation. This simulation utilizes Pennylane's `default.qubit` device, which does not incur in any type of noise. To artificially

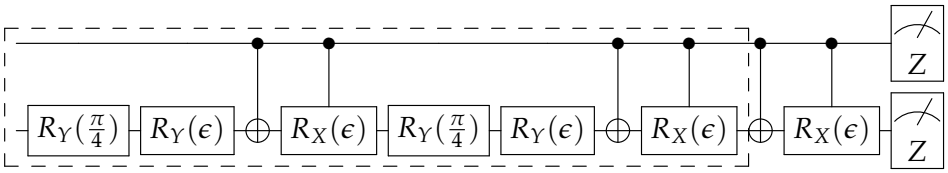
introduce coherent noise into the quantum circuit we extract the performed quantum gates Möttönen's method would use for state preparation and after each gate introduce a corresponding rotational gate with $theta = \epsilon$.

The results of the artificial coherent noise CNOT gate test are found in Table 3.3. Although they still don't match the calculated ideal measurements from Table 3.2, we can attribute the differences to noise incurred during state preparation using Möttönen's method. There should be no noise in any first qubit measurement, however, for the quantum states $|10\rangle$ and $|11\rangle$ state preparation for the first qubit causes noise in the measurements. Regarding the second qubit measurements, $|10\rangle$ and $|11\rangle$ should show less signs of noise. For $|01\rangle$, there should not be any incurred noise. Again, the differences can be attributed to noise sustained during noise preparation.

Quantum State	$\langle Z \rangle$
$\|00\rangle$	[1, 1]
$\|01\rangle$	[1, -0.940]
$\|10\rangle$	[-0.985, -0.852]
$\|11\rangle$	[-0.985, 0.929]

Table 3.3: Obtained artificial expectation value with Pauli Z observable of CNOT gate with $\epsilon = 10°$. The results include coherent noise induced by state preparation.

We can look into a specific case to further specify the influence of coherent noise during state preparation. In Equation 3.1 the quantum circuit describing the state preparation using Möttönen's method and then applying one CNOT gate is illustrated. The state preparation part from the quantum circuit is enclosed by a dotted line box. The initial quantum state is $|00\rangle$ and it is being transformed into $|01\rangle$ by utilizing two $R_Y(\theta)$ and two CNOT gates. We can observe that after each operation there is a rotational gate that correspond to the coherent noise.



(3.1)

In this specific case, because the control qubit is set to 0, none of the control gates will perform any modification to the target wire. The control gates also don't modify the control qubit, thus, there are no perturbations in the measurement. Furthermore, we can simplify all the $R_Y(\theta)$ gates into one by adding up their angles. The resulting

gate is then $R_Y(\frac{\pi}{2} + 2\epsilon)$. If we then apply the gate, we obtain the quantum state $|\psi\rangle = R_Y(\frac{\pi}{2} + 2\epsilon)\,|0\rangle$. We can now calculate the expectation value $\langle\psi|Z|\psi\rangle$ and the result will be -0.940 for an $\epsilon = 0.175$, which is exactly what we get as a result in Table 3.3.

One important conclusion regarding this experiment is that the quantum state $|0\rangle$ does not require to be modified during the state preparation routine, consequently, it will not display any coherent noise disturbance. In addition, any control gate will not introduce any coherent noise to the target qubit if the control qubit is set to 0.

Finally, in the artificial test we get deterministic measurements that better represent coherent noise. Hence, for any further experiment utilizing coherent noise, we will combine Pennylane's `default.qubit` with corresponding rotational gates after each quantum gate to achieve a model closer to reality.

4 Implementation

In Chapter 4 we will introduce the methods used to implement the experiments to answer the research goals. In Section 4.1 we will present how the QML models are trained. Futhermore, in Section 4.2 the attack methodology on the VQA model will be described.

4.1 Variational Quantum Algorithm Model Training

In this section we will specify how the VQA models were trained. In Subsection 4.1.1, we will describe the dataset preprocessing methodology required for the training. Later on in Subsection 4.1.2, the configuration files utilized by the training pipeline will be defined. Afterwards, in Subsection 4.1.3 we will present how the noise is injected to the QML models during training and evaluation. Finally, the developed pipeline for the training will be presented in Subsection 4.1.4.

4.1.1 Datasets Preprocessing

In order to reduce the execution time of the model training, all the datasets mentioned in the Section 3.1 were stored preprocessed. All the datasets but the Plus-Minus dataset were retrieved using OpenML [45]. We obtained the Plus-Minus dataset by asking for it to the authors of [52].

The preprocessing methodology encompasses for all the datasets the removal of data points that are missing at least one feature. Then, duplicate data points are removed to avoid overemphasizing these elements and creating a bias in the data distribution. In case the target labels are non-numeric, we modified them to represent numerical classes. Finally, all the datasets except the Iris Flower dataset are normalized using the standard deviation and rescaled between a range of 0 and 1. This is done with Scikit-learn [46] and its purpose is to prevent feature dominance and to quicken training convergence.

There are two last details with regards to the preprocessing of the datasets. Firstly, as mentioned in Subsection 3.1.5, the images from MNIST were downsampled to be able to process them with a VQA. Secondly, the majority class from the PID dataset was reduced to obtain a balanced dataset with regards to the class distribution.

4.1.2 Configuration File

The configuration files are essential to the training pipeline. Its contents define the dataset to be used, which type of QML model is going to be trained, which noise model will be injected, and some hyperparameters regarding the QML model's architecture. These hyperparameters are the number of qubits, the number of layers, the number of classes, the learning rate, the batch size, and the epochs.

```
{
  "dataset" : "diabetes",
  "qml_model" : "pqc",
  "noise_model" : "none",
  "num_qubits" : 3,
  "num_layers" : 40,
  "num_classes" : 2,
  "learning_rate" : 0.0005,
  "batch_size" : 16,
  "epochs" : 10
}
```

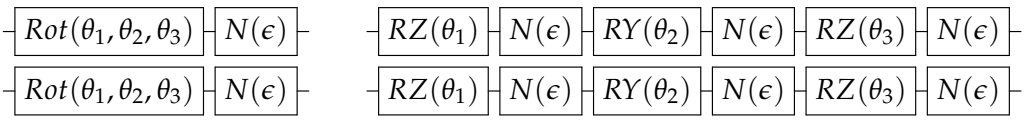
Listing 4.1: Example configuration file.

An example configuration file can be seen in Listing 4.1. This configuration file is given to the training pipeline. Once the training is finished, two more fields will be added to the configuration file. These fields are the model accuracy on the test set and an *id* to help identify the resulting model weights. The configuration files are helpful to automate the training and prepare the adversarial attacks on all the trained QML models.

4.1.3 Noise Injection

The Circuit-Centric ansatz is built upon a combination of Pennylane's arbitrary rotation gate `Rot` and `CNOT` gates. In the case of Pennylane, the *Rot* gate is native to the framework, therefore, there is no need to decompose it into other supported gates.

Nevertheless, if other frameworks or devices are utilized, the gate decomposition is required and it would modify the resulting trained model.



(4.1)

Pennylane's $Rot(\theta_1, \theta_2, \theta_3)$ gate can be decomposed in three rotational gates, namely the sequence of $R_Z(\theta_1)$, $R_Y(\theta_2)$, and $R_Z(\theta_3)$. If we injected the noise after every gate, the influence of noise on the quantum circuit would increase almost three times (Eq. 4.1). Consequently, we manually decompose the *Rot* gate into its three corresponding gates to generate a single layer in the Circuit-Centric ansatz. This will not only allow for the pipeline in the future to be expanded and tested with new devices and frameworks, but also will make the resuls of the thesis valid for all of the devices and frameworks.

In Equation 4.1 the gate with the letter N simulates the injected noise to the circuit. The type of noise model to be injected to the QML model is provided in the previously described configuration file. The configuration file must then provide one of the 6 possible noise models and either the corresponding probability of noise to occur or the angle miscalibration ϵ.

In order to inject noise to a QML model we utilize Pennylane's `transform` module. We utilize it in two different ways to create QML models either with coherent or incoherent noise. To simulate coherent noise we create a Pennylane noise model object that will later on be applied to the quantum circuit. The noise model is conformed by the addition of $R_Y(\epsilon)$, $R_Z(\epsilon)$, or $CRX(\epsilon)$ gates to the noiseless $R_Y(\theta)$, $R_Z(\theta)$, or $CNOT$ gates respectively (where ϵ represents the miscalibration angle). On the other hand, to simulate incoherent noise we insert the noisy operation depending on its type directly to the quantum circuit's gates. This means that after every quantum gate noise might occur with a given probability ϵ.

For the incoherent noise QML models, the chosen values for the probability of noise occurring are 2%, 4%, 6%, 8%, and 10%. These values were chosen to be enough to notice an influence in the training but not to overwhelm the classifier and drown down any potential learning from the data. The same reasons apply to the chosen values for the miscalibration angle for coherent noise, the values are 2°, 4°, 6°, 8°, and 10°.

4.1.4 Training Pipeline

The stepping stones for the training pipeline were presented in the previous subsections. The pipeline has been created in order to simplify and automatize the training of the QML models. The pipeline is based on the configuration files. Each QML model requires a configuration file to be trained, therefore, the amount of configuration files depends on the number of models to be trained.

In this thesis we are training QML models for 7 datasets, with 6 noise types with each one having 5 different miscalibrations or probabilities. Adding up the 7 noiseless models, we end up with 217 configuration files and QML models to be trained. Manually creating the configuration files would take a considerable amount of time. Therefore, we coded a script to automatically create the configuration files of a given dataset.

After the configuration files have been created, we can start with the training. Training 217 QML models individually is also time consuming. Thus, we created a script that retrieves all the configuration files from a specific dataset and performs the model training. More importantly, the resulting weights of the model are automatically saved in conjunction with the modified configuration files. This serves two purposes. Firstly, we need the model weights to evaluate their accuracy when performing the adversarial attacks. Secondly, the pipeline checks if the model has already been previously trained and can skip retraining it. Therefore, the automated training can be interrupted and resumed without losing any information about the trained models.

The actual model training is performed by using a training loop based on the PyTorch Lightning [58] workflow. To enable the Lightning workflow we created a helper file with utility functions that help train and evaluate the QML model. In this file we define the training, validation and testing steps. Additionally, we implement helper functions to calculate the accuracy and a threshold method to be able to evaluate the QML model. Finally, we define the model architecture based on the weights, number of target labels to classify and a forward pass.

The Lightning workflow simplifies and streamlines the training. Nevertheless, in order to perform the training loop, a small setup before is required. First of all, based on the configuration file we retrieve the datasets. If the dataset has already been fetched, we simply load the preprocessed dataset from the local files. Later on, we define the QML model based on the given hyperparameters. Moreover, the required weights for training the module are randomly initialized.

Once the preprocessed dataset has been loaded, we divide it into different sets to facilitate the training loop. For most of the datasets we have a 70/30 split between the training set and the test set. The test set is stratified according to the data distribution from the training set. For the MNIST and Plus-Minus datasets only 20% of the dataset is used for training, as theses datasets are substantially more computationally demanding than the other datasets. Moreover, for the MNIST datasets we have a 70/15/15 training,

validation, and test set split respectively.

The last requirements for the training loop to be prepared are the loss functionand the optimizer used to the adapt the QML model. In our specific case we utilize ADAM, a gradient-based optimizer, as it will lead to faster convergence than just using a gradient descent. For the loss function we utilize PyTorch's implementation of the Cross Entropy Loss, therefore, we interpret the resulting measurements from the QML model similar to a probability distribution.

Once everything is set up, we let the *Trainer* module perform the training of the QML model. The *Trainer* module is responsible for loading the data points in the given batch number, calculating the loss, and optimizing the QML models weights towards convergence. We also implemented an early stopping mechanism when the training accuracy reaches 100% to avoid overfitting the simpler models.

With this implemented pipeline it is simple to perform any further updates. The helping files are modularly designed to be reused, modified, and updated. Incorporating new and different types of QML models with noise injection is simple. Furthermore, the chosen noise types and values can be effortlessly modified and expanded. Not only does the pipeline enable the replication of results, but the creation of new experimentats.

4.2 Adversarial Attacks on Variational Quantum Algorithm Model

In this section we will specify how the adversarial attacks and evaluation are performed. In Subsection 4.2.1, we will describe the process to obtain adversarial examples with the FGSM and PGD techniques. Afterwards in Subsection 4.2.2, the evaluation procedure of the trained QML models against adversarial examples will be defined.

4.2.1 Adversarial Attacks

Similar to the training pipeline introduced in the previously section, we tried to automate and simplify as much as possible the creation of the adversarial examples. In our thesis we are utilizing two different adversarial techniques, namely FGSM (Subsec. 2.4.1) and PGD (Subsec. 2.4.2).

Instead of manually implementing the attacks, there are several Python libraries that already have coded functions according to different techniques to create adversarial

examples. In our case we utilize Cleverhans [59] due to its easeness of use and its interface with PyTorch models. This allows our previous PyTorch QML model definition to be used as an input for the attack functions.

The function to create adversarial examples using the FGSM technique requires the previously mentioned PyTorch QML model. Additionally, it requires the input data point to attack. We can provide as an argument whether it is a targeted attack or not. In our specific case we are creating untargeted adversarial examples, therefore, we don't have to specify a target label. Moreover, we indicate the L_∞ norm to be used, as it causes small uniform distributions across all the features that are harder to perceive. Finally, we must determine the attack strength ϵ.

The function that implements the PGD method requires all the previously mentioned arguments for the FGSM technique. Additionally, it also needs the iterative step size and the number of iterations to recurringly create the adversarial examples. These specific arguments are set to 0.01 and 40 respectively.

The noiseless QML model is used as the required PyTorch model for crafting the adversarial examples. Thus, we retrieve the pretrained weights according to the QML model and the dataset we are targeting. This will enable us to create a baseline attack and check whether the noisy training improves the resilience of the models against adversarial attacks.

Once the dataset has been retrieved, we perform both attacks on the test set to create the adversarial examples needed to evaluate the noisy QML models. We evaluate directly the performance of the noiseless model with the adversarial examples to verify that the performed attacks are correctly implemented and that they are misclassified by the model they are intended to fool. This process is repeated 5 times, each with a different value for the attack strength ϵ. The result of this process is obtaining 10 different variations of the adversarial examples with increasing difficulty to be correctly classified. This will allow us to quantify how resilient the noisy QML models are in case they are actually more robust against adversarial attacks.

While for image data the modifications to obtain the adversarial examples are conceptually simple to understand (Fig. 2.6), for the tabular data it might be more abstract to understand the changes. In Figure 4.1 we show the effects of FGSM on two dimensions of a data point for the Iris dataset. We use two dimensions only to simplify the understanding of what disturbances are added to tabular data.

We can observe that the data point wihout any modifications is classified with the label 0. With an increasing attack strength the data point is modified such that the *Sepal Lenght* is reduced while the *Petal Length* is increased. The red line in this case represents the classification threshold, where the QML classifier starts labeling the data point as 1 instead of 0. Once the features have been adapted accordingly to exceed the

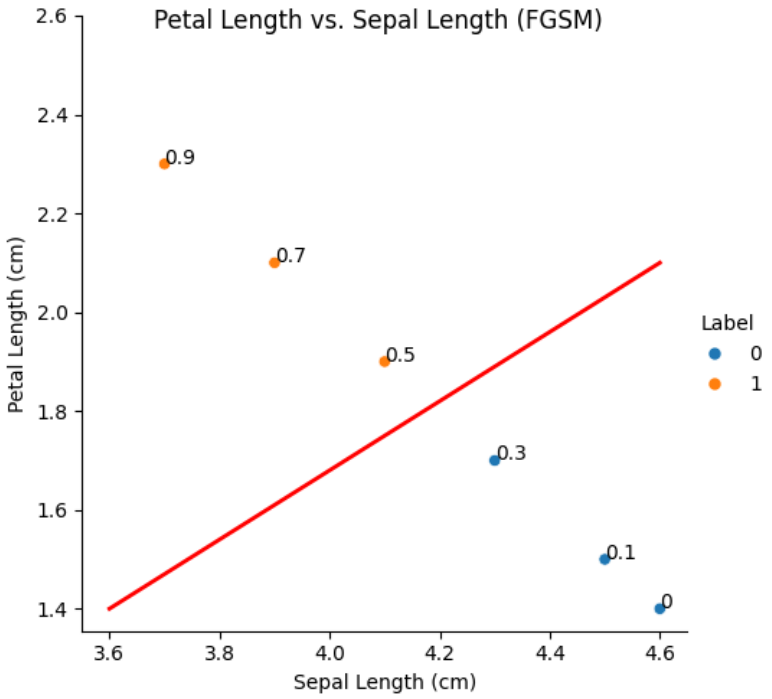


Figure 4.1: Effects of FGSM on the Iris dataset's *Petal Length* and *Sepal Length* features for different attack strengths.

classification threshold, the QML model will start to misclassify the data point.

In the case of attacking a binary classificator, utilizing a technique implementing an untargeted attack is equivalent to executing a targeted attack aiming to obtain a misclassification with the datapoint's contrary class. Therefore, for binary classificators there is no distinction between performing a targeted or an untargeted attack. However, this is not the case when the QML model is trying to differentiate between more than two classes, as the result of an untargeted attack can now be classified into more than one category.

The attack strength regulates the magnitude of the perturbations added to the original input to create the adversarial examples. In this thesis we chose 5 different attack strengths, specifically the values $[0.1, 0.3, 0.5, 0.7, 0.9]$. This allows us to slowly increase the attack strength while checking over a big range how the noisy and noisless QML models react. Another factor that influenced this range is that the attacks are being applied to the preprocessed data points that have been normalized and rescaled.

4.2.2 Automated Evaluation

Continuing the idea of the pipeline, evaluating the trained QML models should be simple to execute. The evaluation requires two arguments, namely the dataset and a given QML model type to assess. The evaluation code will then automatically calculate the accuracy scores of all the 31 (Subsection 4.1.4) trained models. Additionally, it also evaluates the QML models against the two different adversarial attacks with five differing attack strength. The evaluation pipeline then results per dataset on $310 + 31 = 341$ data points to be analysed.

In order to avoid performing a redundant assessment, first, the pipeline will check if the evaluation and the figures have already been created. If any of the two is missing, the pipeline will automatically calculate the QML models' accuracies and draw the figures from the resulting data. To do this, it verifies that all the adversarial samples have already been crafted.

The evaluation process iterates over all the noise models (and its miscalibrations or noise probabilities) and assesses each model with the modified adversarial examples. The adversarial evaluation calculates first the QML model's accuracy against the unmodified test set. Later on, the evaluation script retrieves the adversarial examples for both adversarial attack techniques and their variations and computes the adversarial accuracy of the model. The results of this evaluation are saved in a file for further use when creating the figures comparing the accuracy of the different models in differents scenarios. When the evaluation is performed, the weights of the noisy models are retrieved, however, the evaluation is performed with a noisless model to stop the calculations from being doubly impacted by the quantum noise.

The resulting file from the evaluation process stores the accuracies and the characteristics of the models. Per dataset, we can retrieve 31 clean accuracies without any attack performed and 310 adversarial accuracies, this corresponds to the previosly mentioned 341 number. This information is required by the figure creation process.

The graph creation process will output 16 different figures belonging to 2 different types of graphs. Each type of graph tries to convey a different comparison between the different QML models. The first type of graph deals with the accuracy of the the models on the clean dataset, with this graph we can observe the accuracy of the noiseless model as well as how the noise probabilities or miscalibrations affect the noisy models.

In Figure 4.2 we can find a sample of this type of graphs, all the noise types start at the 0 point on the x axis with the same value. This value represents the baseline performance of the noiseless model on a chosen dataset. For the incoherent noise models, all the noise models appear in one single graph to be able to also compare

them between each other. We separate the coherent noise model graphs from the incoherent noise ones because the x axis corresponds to a different variable.

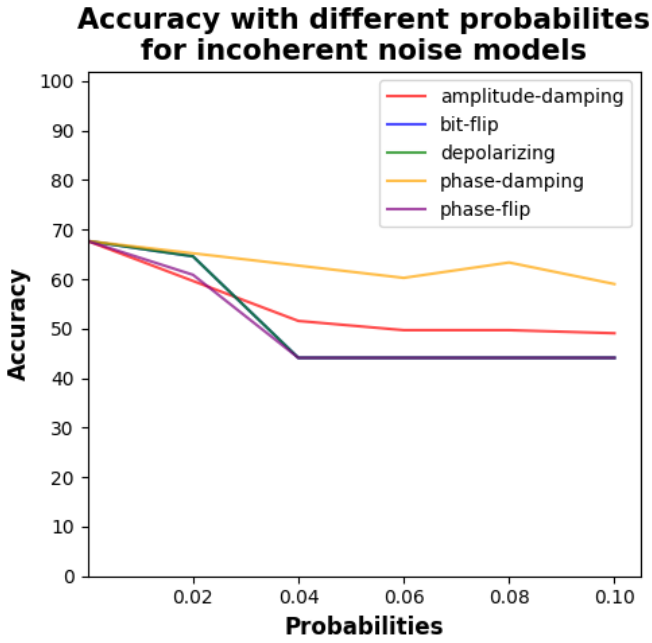


Figure 4.2: Sample graph showing the effects on the model's accuracy of the noise probability modifications to the different noise models.

The second type of graph can be found in Figure 4.3. This graph demonstrates the changes that a given adversarial attack with differing attack strengths provokes in the accuracy of differently trained models. Thus, there is one graph per type of attack for each noise model, which results in 14 graphs to be analyzed. The graphs for the noiseless models are used to verify the effectiveness and the correct working of both adversarial attack techniques.

Furthermore, each graph has all the different variations of noise intensity that affected during training each noise model. Additionally, the baseline performance of the noiseless model can also be found with the 0 modification label on the noisy graphs. Therefore, we can determine whether the noise improves, decreases or doesn't modify the robustness of the trained QML models.

With the automatic creation of the graphs we terminate the pipeline for prepro-

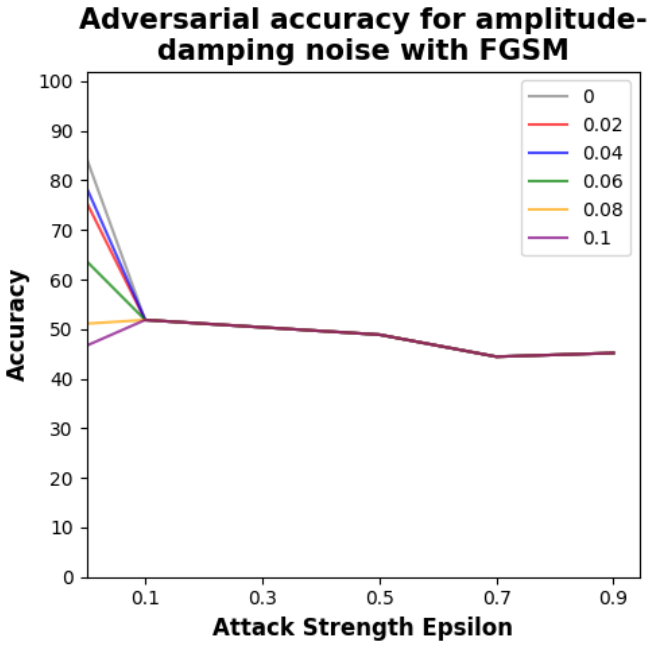


Figure 4.3: Sample graph showing the effects on the model's accuracy of the attack strength modifications to the different noise models.

cessing, training, storing, attacking, and evaluating the accuracy of noiseless and noisy QML models. The pipeline implementation reduces to the minimum the user interaction and allows us to automatically perform the training of resource intensive QML models. Additionally. we can observe the effect of different types of noise and noise intensity on model robustness against several adversarial attacks. Furthermore, the pipeline allows us also to easily modify it by implementing new architectures of QML models or implementing new adversarial attack techniques. Thus, further research of the effects of noise on QML model robustness is simple to perform.

5 Results

5.1 Section

1. Explain why decoherent noise is used and not coherent.
a. Coherent noise will probably just shift the bias.
b. Coherent noise might add too much noise (quadratic growth).
c. True random noise is required to improve generalization.

5.2 Variational Quantum Algorithm Model Accuracy

5.3 Variational Quantum Algorithm Model Adversarial Accuracy

6 Future Work

When creating the weights to be trained, a proper selection might increase the accuracy of the models or avoid barren plateaus.

Investigate QML model training with different optimizers, maybe even no gradient-based methods.

Investigate the influence of the different loss functions when training a QML model. How to interpret correctly the measurements at the end of the QML model, can it be assumed to be a probability distribution?

Do experiments with new ansätze, maybe quantum kernels.

Do experiments with new encoding mechanisms, angle, etc. (encoding might have a big influence in how the features are interpreted)

Perform other different attacks like carlini and wagner

Perform transfer attacks between noisy models including the noiseless models. Basically create adversarial examples from noisy models and see how effective they are against other noisy models or the noiseless model.

7 Style

7.1 Section

Citation test [1]. Acronyms must be added in `main.tex` and are referenced using macros. The first occurrence is automatically replaced with the long version of the acronym, while all subsequent usages use the abbreviation.

E.g. `\ ac{tum}, \ ac{tum}` ⇒ Technical University of Munich (TUM), TUM
For more details, see the documentation of the `acronym` package[1].

7.1.1 Subsection

See Table 7.1, Figure 7.1, Figure 7.2, Figure 7.3.

Table 7.1: An example for a simple table.

A	B	C	D
1	2	1	2
2	3	2	3

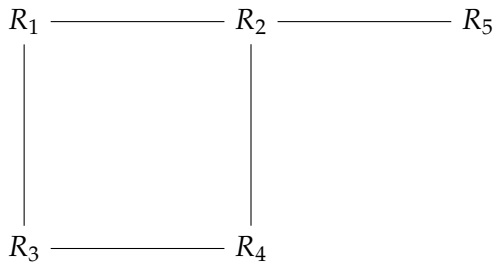


Figure 7.1: An example for a simple drawing.

[1] `https://ctan.org/pkg/acronym`

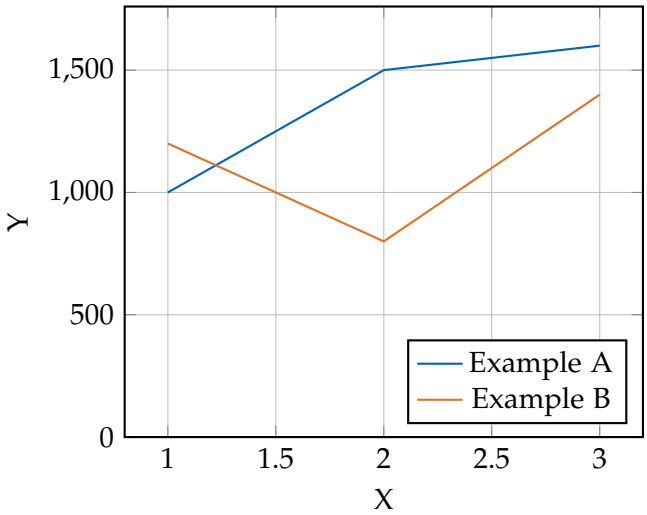


Figure 7.2: An example for a simple plot.

```
1 SELECT * FROM tbl WHERE tbl.str = "str"
```

Figure 7.3: An example for a source code listing.

Abbreviations

TUM Technical University of Munich

NISQ Noisy Intermediate-Scale Quantum

ML Machine Learning

QML Quantum Machine Learning

CNOT Controlled NOT

PID Pima Indians Diabetes

VQA Variational Quantum Algorithm

AML Adversarial Machine Learning

FGSM Fast Gradient Sign Method

PGD Projected Gradient Descent

List of Figures

List of Tables

Bibliography

[1] bibinitperiod M. National Academies of Sciences Engineering. *Quantum Computing: Progress and Prospects*. Google-Books-ID: ATH3DwAAQBAJ. National Academies Press, Mar. 27, 2019. 273 pp. ISBN: 978-0-309-47972-1.

[2] P. W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." In: *SIAM Journal on Computing* 26.5 (Oct. 1997). Publisher: Society for Industrial and Applied Mathematics, pp. 1484–1509. ISSN: 0097-5397. DOI: 10.1137/S0097539795293172.

[3] W. Van Dam, S. Hallgren, and L. Ip. "Quantum Algorithms for Some Hidden Shift Problems." In: *SIAM Journal on Computing* 36.3 (Jan. 2006), pp. 763–778. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S009753970343141X.

[4] S. Hallgren. "Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem." In: *Journal of the ACM* 54.1 (Mar. 2007), pp. 1–19. ISSN: 0004-5411, 1557-735X. DOI: 10.1145/1206035.1206039.

[5] R. L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems." In: *Communications of the ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/359340.359342.

[6] J. Preskill. "Quantum Computing in the NISQ era and beyond." In: *Quantum* 2 (Aug. 6, 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. arXiv: 1801.00862[cond-mat,physics:quant-ph].

[7] P. W. Shor. "Quantum Computing." In: *Documenta Mathematica - Extra Volume ICM 1998*, pp. 467–486.

[8] R. Bommasani, D. A. Hudson, E. Adeli, et al. *On the Opportunities and Risks of Foundation Models*. July 12, 2022. arXiv: 2108.07258[cs].

[9] M. Schuld and F. Petruccione. *Machine Learning with Quantum Computers*. Quantum Science and Technology. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-83097-7 978-3-030-83098-4. DOI: 10.1007/978-3-030-83098-4.

[10] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. *Intriguing properties of neural networks*. Feb. 19, 2014. arXiv: 1312.6199[cs].

[11] N. Papernot, P. McDaniel, and I. Goodfellow. *Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples*. May 23, 2016. arXiv: 1605.07277[cs].

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy. *Explaining and Harnessing Adversarial Examples*. Mar. 20, 2015. arXiv: 1412.6572[cs,stat].

[13] A. Kurakin, I. Goodfellow, and S. Bengio. *Adversarial Machine Learning at Scale*. Feb. 10, 2017. arXiv: 1611.01236[cs,stat].

[14] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig. "Quantum machine learning: a classical perspective." In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474.2209 (Jan. 2018), p. 20170551. ISSN: 1364-5021, 1471-2946. DOI: 10.1098/rspa.2017.0551.

[15] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Higher Education from Cambridge University Press. ISBN: 9780511976667 Publisher: Cambridge University Press. Dec. 9, 2010. DOI: 10.1017/CBO9780511976667. URL: https://www.cambridge.org/highereducation/books/quantum-computation-and-quantum-information/01E10196D0A682A6AEFFEA52D53BE9AE (visited on 03/28/2024).

[16] B. Schumacher. "Quantum Coding." In: *Physical Review A* 51.4 (Apr. 1, 1995), pp. 2738–2747. ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.51.2738.

[17] P. a. M. Dirac. "A New Notation for Quantum Mechanics." In: *Mathematical Proceedings of the Cambridge Philosophical Society* 35.3 (July 1939), pp. 416–418. ISSN: 1469-8064, 0305-0041. DOI: 10.1017/S0305004100021162.

[18] M. Born. "Quantenmechanik der Stoßvorgänge." In: *Zeitschrift für Physik* 38.11 (Nov. 1, 1926), pp. 803–827. ISSN: 0044-3328. DOI: 10.1007/BF01397184.

[19] F. Bloch. "Nuclear Induction." In: *Physical Review* 70.7 (Oct. 1, 1946), pp. 460–474. ISSN: 0031-899X. DOI: 10.1103/PhysRev.70.460.

[20] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. "Array programming with NumPy." In: *Nature* 585.7825 (Sept. 2020). Publisher: Nature Publishing Group, pp. 357–362. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2649-2.

[21] A. Einstein, B. Podolsky, and N. Rosen. "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?" In: *Physical Review* 47.10 (May 15, 1935), pp. 777–780. ISSN: 0031-899X. DOI: 10.1103/PhysRev.47.777.

[22] C. H. Bennett and S. J. Wiesner. "Communication via One- and Two-Particle Operators on Einstein-Podolsky-Rosen States." In: *Physical Review Letters* 69.20 (Nov. 16, 1992), pp. 2881–2884. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.69.2881.

[23] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. "Teleporting an Unknown Quantum State via Dual Classical and Einstein-Podolsky-Rosen Channels." In: *Physical Review Letters* 70.13 (Mar. 29, 1993), pp. 1895–1899. ISSN: 0031-9007. DOI: 10.1103/PhysRevLett.70.1895.

[24] J. Yoneda, K. Takeda, T. Otsuka, T. Nakajima, M. R. Delbecq, G. Allison, T. Honda, T. Kodera, S. Oda, Y. Hoshi, N. Usami, K. M. Itoh, and S. Tarucha. "A Quantum-Dot Spin Qubit with Coherence Limited by Charge Noise and Fidelity Higher than 99.9%." In: *Nature Nanotechnology* 13.2 (Feb. 2018). Publisher: Nature Publishing Group, pp. 102–106. ISSN: 1748-3395. DOI: 10.1038/s41565-017-0014-x.

[25] M. A. Pravia, N. Boulant, J. Emerson, A. Farid, E. M. Fortunato, T. F. Havel, R. Martinez, and D. G. Cory. "Robust Control of Quantum Information." In: *The Journal of Chemical Physics* 119.19 (Nov. 15, 2003), pp. 9993–10001. ISSN: 0021-9606. DOI: 10.1063/1.1619132.

[26] N. Kaufmann, I. Rojkov, and F. Reiter. *Characterization of Coherent Errors in Noisy Quantum Devices*. July 17, 2023. arXiv: 2307.08741[quant-ph].

[27] N. Boulant, S. Furuta, J. Emerson, T. F. Havel, and D. G. Cory. "Incoherent Noise and Quantum Information Processing." In: *The Journal of Chemical Physics* 121.7 (Aug. 15, 2004), pp. 2955–2961. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.1773161. arXiv: quant-ph/0312116.

[28] J. K. Iverson and J. Preskill. "Coherence in Logical Quantum Channels." In: *New Journal of Physics* 22.7 (July 1, 2020), p. 073066. ISSN: 1367-2630. DOI: 10.1088/1367-2630/ab8e5c.

[29] S. Resch and U. R. Karpuzcu. "Benchmarking Quantum Computers and the Impact of Quantum Noise." In: *ACM Computing Surveys* 54.7 (Sept. 30, 2022), pp. 1–35. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3464420.

[30] H.-P. Breuer and F. Petruccione. *The Theory of Open Quantum Systems*. Oxford University Press, Jan. 25, 2007. ISBN: 978-0-19-170634-9. DOI: 10.1093/acprof:oso/9780199213900.001.0001.

[31] K. Kraus. "General State Changes in Quantum Theory." In: *Annals of Physics* 64.2 (June 1, 1971), pp. 311–335. ISSN: 0003-4916. DOI: 10.1016/0003-4916(71)90108-4.

[32] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. "Quantum machine learning." In: *Nature* 549.7671 (Sept. 2017), pp. 195–202. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature23474.

[33] M. Schuld. *Supervised Quantum Machine Learning Models are Kernel Methods*. Apr. 17, 2021. arXiv: 2101.11020[quant-ph,stat].

[34] M. Schuld, R. Sweke, and J. J. Meyer. "Effect of Data Encoding on the Expressive Power of Variational Quantum-Machine-Learning Models." In: *Physical Review A* 103.3 (Mar. 24, 2021), p. 032430. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.103.032430.

[35] P. Rebentrost, M. Mohseni, and S. Lloyd. "Quantum Support Vector Machine for Big Data Classification." In: *Physical Review Letters* 113.13 (Sept. 25, 2014), p. 130503. ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.113.130503.

[36] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles. "Variational Quantum Algorithms." In: *Nature Reviews Physics* 3.9 (Aug. 12, 2021), pp. 625–644. ISSN: 2522-5820. DOI: 10.1038/s42254-021-00348-9. arXiv: 2012.09265[quant-ph,stat].

[37] M. Benedetti, E. Lloyd, S. Sack, and M. Fiorentini. "Parameterized quantum circuits as machine learning models." In: *Quantum Science and Technology* 4.4 (Nov. 2019). Publisher: IOP Publishing, p. 043001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab4eb5.

[38] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. arXiv: 1412.6980[cs].

[39] M. Schuld, A. Bocharov, K. Svore, and N. Wiebe. "Circuit-Centric Quantum Classifiers." In: *Physical Review A* 101.3 (Mar. 6, 2020), p. 032308. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.101.032308. arXiv: 1804.00633[quant-ph].

[40] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar. "Adversarial machine learning." In: ().

[41] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. *Robust Physical-World Attacks on Deep Learning Models*. Apr. 10, 2018. arXiv: 1707.08945[cs].

[42] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry. *Robustness May Be at Odds with Accuracy*. Sept. 9, 2019. arXiv: 1805.12152[cs,stat].

[43] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. *Towards Deep Learning Models Resistant to Adversarial Attacks*. Sept. 4, 2019. arXiv: 1706.06083[cs,stat].

[44] D. Winderl, N. Franco, and J. M. Lorenz. *Quantum Neural Networks under Depolarization Noise: Exploring White-Box Attacks and Defenses*. Dec. 21, 2023. arXiv: 2311.17458[quant-ph].

[45] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo. "OpenML: Networked Science in Machine Learning." In: *ACM SIGKDD Explorations Newsletter* 15.2 (June 16, 2014), pp. 49–60. ISSN: 1931-0145, 1931-0153. DOI: 10.1145/2641190.2641198. arXiv: 1407.7722[cs].

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. ISSN: 1533-7928.

[47] R. A. Fisher. "The Use of Multiple Measurements in Taxonomic Problems." In: *Annals of Eugenics* 7.2 (1936). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1936.tb02137.x, pp. 179–188. ISSN: 2050-1439. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

[48] Y. Du, M.-H. Hsieh, T. Liu, D. Tao, and N. Liu. "Quantum noise protects quantum classifiers against adversaries." In: *Physical Review Research* 3.2 (May 27, 2021). Publisher: American Physical Society, p. 023153. DOI: 10.1103/PhysRevResearch.3.023153.

[49] J. W. Smith, J. Everhart, W. Dickson, W. Knowler, and R. Johannes. "Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus." In: *Proceedings of the Annual Symposium on Computer Application in Medical Care* (Nov. 9, 1988), pp. 261–265. ISSN: 0195-4210.

[50] C. Drummond and R. C. Holte. "C4.5, Class Imbalance, and Cost Sensitivity: Why Under-Sampling beats Over-Sampling." In: ().

[51] W. N. Street, W. H. Wolberg, and O. L. Mangasarian. "Nuclear Feature Extraction for Breast Tumor Diagnosis." In: IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology. Ed. by R. S. Acharya and D. B. Goldgof. San Jose, CA, July 29, 1993, pp. 861–870. DOI: 10.1117/12.148698.

[52] M. Wendlinger, K. Tscharke, and P. Debus. *A Comparative Analysis of Adversarial Robustness for Quantum and Classical Machine Learning Models*. Apr. 24, 2024. arXiv: 2404.16154[quant-ph].

[53] M. T. West, S. M. Erfani, C. Leckie, M. Sevior, L. C. L. Hollenberg, and M. Usman. "Benchmarking Adversarially Robust Quantum Machine Learning at Scale." In: *Physical Review Research* 5.2 (June 23, 2023), p. 023186. ISSN: 2643-1564. DOI: 10.1103/PhysRevResearch.5.023186.

[54] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. "Comparison of Classifier Methods: a Case Study in Handwritten Digit Recognition." In: *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5)*. Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3 - Conference C: Signal Processing (Cat. No.94CH3440-5). Vol. 2. Oct. 1994, 77–82 vol.2. DOI: 10.1109/ICPR.1994.576879.

[55] S. Lu, L.-M. Duan, and D.-L. Deng. "Quantum adversarial machine learning." In: *Physical Review Research* 2.3 (Aug. 6, 2020), p. 033212. ISSN: 2643-1564. DOI: 10.1103/PhysRevResearch.2.033212.

[56] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, J. M. Arrazola, U. Azad, S. Banning, C. Blank, T. R. Bromley, B. A. Cordier, J. Ceroni, A. Delgado, O. Di Matteo, A. Dusko, T. Garg, D. Guala, A. Hayes, R. Hill, A. Ijaz, T. Isacsson, D. Ittah, S. Jahangiri, P. Jain, E. Jiang, A. Khandelwal, K. Kottmann, R. A. Lang, C. Lee, T. Loke, A. Lowe, K. McKiernan, J. J. Meyer, J. A. Montañez-Barrera, R. Moyard, Z. Niu, L. J. O'Riordan, S. Oud, A. Panigrahi, C.-Y. Park, D. Polatajko, N. Quesada, C. Roberts, N. Sá, I. Schoch, B. Shi, S. Shu, S. Sim, A. Singh, I. Strandberg, J. Soni, A. Száva, S. Thabet, R. A. Vargas-Hernández, T. Vincent, N. Vitucci, M. Weber, D. Wierichs, R. Wiersema, M. Willmann, V. Wong, S. Zhang, and N. Killoran. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. July 29, 2022. arXiv: 1811.04968[physics,physics:quant-ph].

[57] M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa. *Transformation of Quantum States Using Uniformly Controlled Rotations*. July 1, 2004. arXiv: quant-ph/0407010.

[58] W. Falcon. *PyTorch Lightning*. original-date: 2021-03-18T18:57:21Z. 2019.

[59] N. Papernot, F. Faghri, N. Carlini, I. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, R. Long, and P. McDaniel. *Technical Report on the CleverHans v2.1.0 Adversarial Examples Library*. June 27, 2018. DOI: 10.48550/arXiv.1610.00768. arXiv: 1610.00768[cs,stat].