# Quantum Classification and Quantum Key Distribution

Jean-Pierre Villacura, Felipe Cisternas[a]

[a]*Universidad Técnica Federico Santa María, Valparaíso, Chile*

**Abstract**

Este trabajo investiga la aplicación de técnicas de Aprendizaje Automático Cuántico en problemas de clasificación y la implementación de Quantum Key Distribution (QKD) para garantizar la seguridad en la transmisión de datos. Se presentan las Redes Neuronales Cuánticas y se comparan varios métodos utilizando conjuntos de datos sintéticos y reales. Este estudio resalta el potencial del Aprendizaje Cuántico y QKD en tareas de clasificación y seguridad de datos, subrayando la importancia de futuras investigaciones para optimizar estos enfoques y explorar su aplicación en problemas más complejos.

*Keywords:* Quantum Computing, Machine Learning, Key Distribution, Clasificación, CyberSecurity

## 1. Introducción

En los últimos años, la computación cuántica ha emergido como un campo revolucionario con el potencial de transformar radicalmente la forma en que procesamos la información y abordamos problemas complejos. Dos áreas de investigación que han capturado considerable atención en este contexto son la Distribución de Clave Cuántica (Quantum Key Distribution, QKD) y las Redes Neuronales Cuánticas (Quantum Neural Networks, QNNs).

La Distribución de Clave Cuántica se ha consolidado como un enfoque prometedor para garantizar la seguridad en la transmisión de datos. Utilizando los principios de la mecánica cuántica, QKD permite el intercambio de claves criptográficas entre dos partes comunicantes de manera segura y confiable. Aprovechando las propiedades intrínsecas de la información cuántica, QKD ofrece una protección inigualable contra ataques criptográficos y asegura la confidencialidad e integridad de la información transmitida.

Por otro lado, las Redes Neuronales Cuánticas representan una nueva frontera en el campo del aprendizaje automático. Estas redes utilizan qubits como unidades fundamentales de procesamiento y operaciones cuánticas para realizar cálculos complejos y modelar fenómenos de manera más precisa. Las QNNs prometen mejorar significativamente la capacidad de las redes neuronales clásicas para resolver problemas difíciles, como el reconocimiento de patrones, la optimización y la simulación de sistemas cuánticos.

A lo largo de este estudio, examinamos como se pueden implementar QKD y las QNNs en código mediante ejemplos prácticos que demuestran el potencial de la combinación de estas dos tecnologías. También discutimos los desafíos actuales y las perspectivas futuras en la integración de QKD y QNNs, y las implicaciones que esto puede tener en la seguridad y el procesamiento de datos en entornos cada vez más complejos y conectados.

En resumen, este trabajo propone un enfoque innovador y multidisciplinario que busca aprovechar las sinergias entre la Distribución de Clave Cuántica y las Redes Neuronales Cuánticas. Esperamos que este estudio impulse nuevas investigaciones y desarrollos en el campo de la computación cuántica y su aplicación en áreas críticas como la seguridad de la información y el aprendizaje automático avanzado.

## 2. Quantum Machine Learning

El aprendizaje automático es un sub-campo de la inteligencia artificial, cuyo objetivo es desarrollar técnicas que permitan que las maquinas *aprendan*. Se dice que un agente aprende cuando su desempeño mejora con la experiencia y mediante el uso de datos. Durante los últimos años, el Machine Learning se ha desarrollado exponencialmente, gracias a avances como las redes neuronales, Transformers, Modelos de lenguaje, modelos Generativos, etc. (Dong et al. (2021))

La computación cuántica (Gyongyosi and Imre (2019)) es una tecnología muy prometedora a la hora de desarrollar modelos de machine learning, gracias a sus propiedades cuánticas como lo son en entrelazamiento, existen diversos métodos de implementar Quantum Machine Learning (Zhang and Ni (2020)) pero no funcionan de la misma manera que el aprendizaje automático clásico.

### *2.1. Quantum Neural Networks*

Las redes neuronales cuánticas (Massoli et al. (2022)) funcionan de manera muy similar a las redes neuronales artificiales clásicas, ambas redes ajustan parámetros, en el caso de las redes clásicas son los pesos de conexión de las neuronas y minimizan o maximizan una función objetivo al modificar estos parámetros, la diferencia radica en que se utiliza un circuito cuántico parametrizado, este circuito cuántico aplica una serie de compuertas lógicas cuánticas de rotación en función de unos parámetros $\theta$, por lo que los qubits irán rotando según los valores de estos parámetros $\theta$, los cuales el circuito cuántico me-

diante un optimizador ira ajustando para obtener los mejores resultados, maximizando o minimizando la función objetivo.

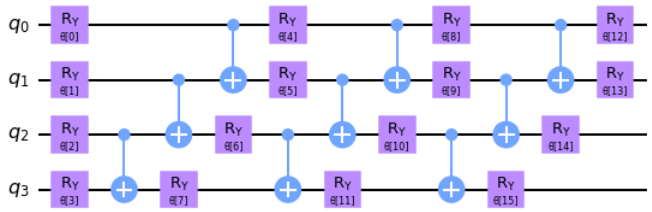


Figura 1: Circuito Cuántico Parametrizado a través de parámetros $\theta$

Para poder implementar una Red neuronal necesitamos poder transformar nuestros datos clásicos en datos cuánticos para así aprovechar las ventajas de la computación cuántica, para esto utilizaremos un **Feature Map**, luego necesitaremos una **Ansatz** que contiene nuestras rotaciones dependientes de los parámetros $\theta$ entrenables, por ultimo necesitaremos medir nuestros resultados para obtener las probabilidades de nuestro circuito, con estos 3 elementos podemos construir una red neuronal cuántica genérica.

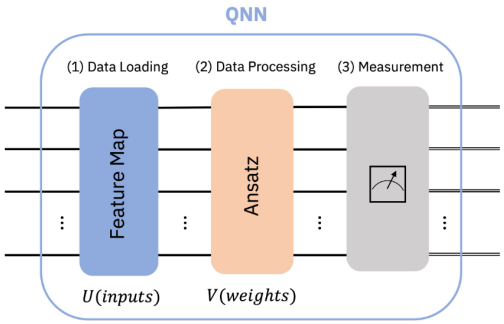


Figura 2: Quantum Neural Network Estándar

### 2.1.1. *Feature Map*

Nuestros datos son clásicos, lo que significa que consisten en un conjunto de bits, no de qubits. Necesitamos una forma de codificar los datos como qubits. Este proceso es crucial si queremos obtener un modelo cuántico efectivo. Por lo general, nos referimos a este mapeo como data encoding/embedding, y esta es la función que cumple el feature map Existen varios formas de crear un Feature Map, pero **Qiskit** provee modelos pre-construidos para llegar y utilizar como lo son *PauliFeatureMap*, *ZZFeatureMap*, entre otros. Estos modelos se basan en aplicar compuertas de fase, CNOTs y compuertas **hadamard**, para aplicar superposición de estados.

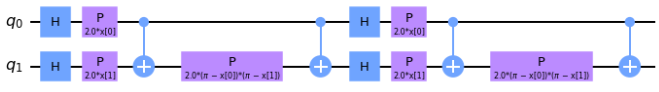


Figura 3: Feature Map para 2 qubits

### 2.1.2. *Ansatz*

Una vez cargados los datos cuánticos, debemos aplicar un circuito cuántico parametrizado. Este circuito es un análogo directo a las capas en las redes neuronales clásicas.

Tiene un conjunto de parámetros o pesos ajustables. Los pesos se optimizan de manera que minimicen una función objetivo, en este caso en vez de pesos tenemos compuertas CNOTs y distintas rotaciones que se aplican a los qubits y estas rotaciones dependen de parámetros $\theta$ que con un optimizador se van a ir entrenando para encontrar la solución óptima.

Esta función objetivo caracteriza la distancia entre las predicciones y los datos etiquetados conocidos. Un circuito cuántico parametrizado también se denomina estado de prueba parametrizado, forma variacional o ansatz.

El Ansatz recibe como input el numero de qubits a utilizar y las repeticiones de las capas de rotaciones, a mayor cantidad de rotaciones/qubits mas complejo sera el circuito y mas parámetros $\theta$ tendremos para entrenar.

Al igual que con el feature map, existen varias formas de crear ansatz, pero **Qiskit** provee modelos como *EfficientSU2*, *RealAmplitudes*, etc.

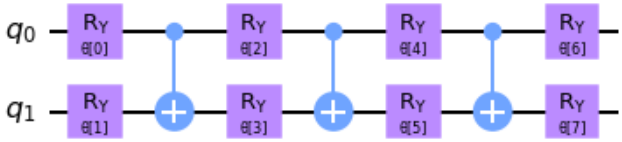


Figura 4: Ansatz para 2 qubits

## 2.2. *Clasificación Cuántica*

En este trabajo exploraremos distintos métodos para resolver problema de clasificación mediante el uso de Quantum Machine Learning, primero la clasificación se realizara sobre un dataset sintético, luego utilizaremos un dataset real y por ultimo veremos un enfoque híbrido al utilizar **Qiskit** junto con **Pytorch**.

Para esto utilizaremos y compararemos 3 métodos de Quantum Neural Networks que provee Qiskit, como lo son:

- **Estimator QNN:** Estimator QNN es una clase que implementa una red neuronal cuántica que se basa en la estimación de un circuito cuántico. El EstimatorQNN toma un circuito cuántico parametrizado como entrada, así como un observable cuántico opcional, y calcula los valores de esperanza para el Forward Pass de la red.

- **Sampler QNN:** Sampler QNN es una red neuronal cuántica que se basa en las muestras resultantes al medir un circuito cuántico. El SamplerQNN se instancia de manera similar al EstimatorQNN, pero debido a que consume muestras directamente al medir el circuito cuántico, no requiere un observable cuántico.

- **Variational Quantum Classifier (VQC):** El VQC es una variante especial del NeuralNetworkClassifier con un

SamplerQNN. Aplica un mapeo de paridad, lo que da como resultado un vector de probabilidad (One-Hot). Aplica la función CrossEntropyLoss que espera etiquetas dadas en formato codificado one-hot y también devolverá predicciones en ese formato.

### 2.2.1. *Dataset Sintético*

Para este dataset generamos un plano en dos dimensiones, donde los puntos sobre la diagonal tienen clase 1 (Puntos Azules) y los puntos bajo la diagonal tienen clase 0 (Puntos Verdes), este es un problema bastante sencillo y nos servirá como baseline de nuestros modelos cuánticos

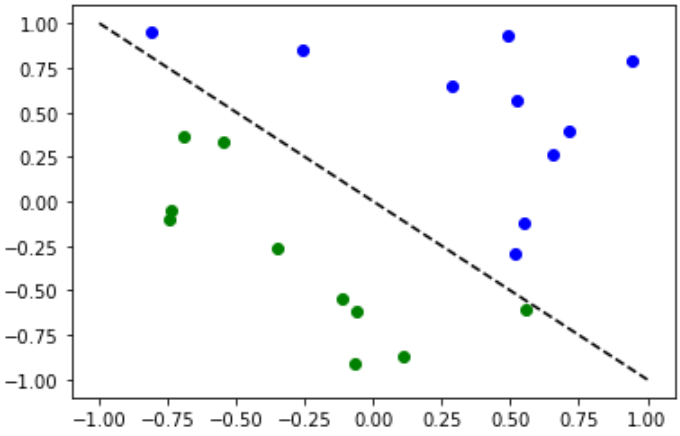


Figura 5: Dataset Sintético

Cada Modelo fue entrenado con 2 Qubits, el mismo Feature Map (ZZFeatureMap) y Ansatz (RealAmplitudes) y se utilizo el optimizador **COBYLA** (Constrained Optimization By Linear Approximation optimizer) con 60 iteraciones para todos los modelos, bajo estos hiperparametros todos los modelos estarían en las mismas condiciones siendo una evaluación justa del desempeño.

| Model | Accuracy | Puntos Mal Clasificados |
|---|---|---|
| Estimator QNN | 0.8 | 4 |
| Sampler QNN | 0.75 | 5 |
| VQC | 0.8 | 4 |

Cuadro 1: Tabla de Resultados Dataset Sintético

### 2.2.2. *Iris Dataset*

Para esta sección utilizaremos el dataset de Iris, el cual contiene 3 clases y 4 features, la tarea es clasificar cada dato en las distintas clases de flores. Para esto utilizaremos un circuito cuántico que nos permita clasificar los puntos en cada clase.

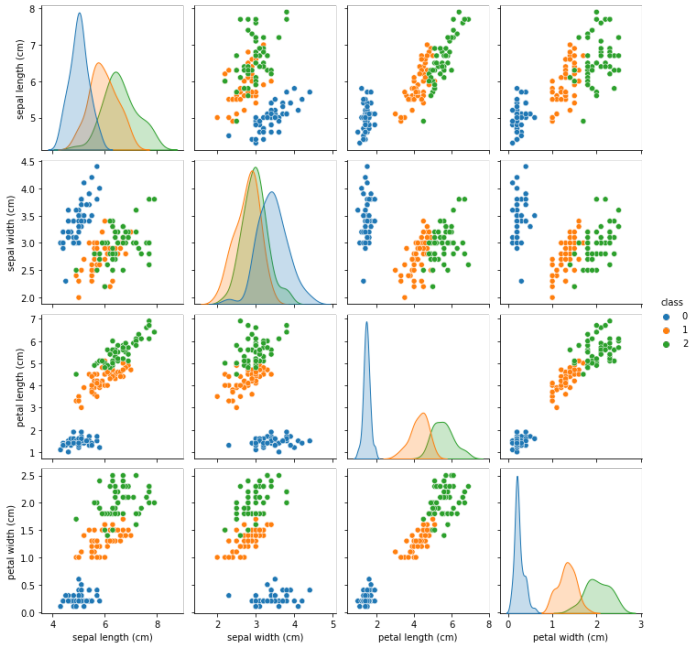


Figura 6: Dataset Iris

Vamos a proceder a escalar las features para que todas estén en el mismo rango con un **MinMaxScaler** y separaremos la data de entrenamiento y la de test en un 80 % y 20 % respectivamente, en esta ocasión entrenaremos un modelo de Support Vector Machine (SVM) clásico para comparar el desempeño con el VQC, entrenaremos 2 instancias del VQC variando el Ansatz para ver como se desempeña la red.

| Model | Test Score | Train Score |
|---|---|---|
| SVM | 0.99 | 0.97 |
| VQC-RealAmplitudes | 0.85 | 0.87 |
| VQC-EfficientSU2 | 0.9 | 0.8 |

Cuadro 2: Tabla de Resultados Iris Dataset

## 2.3. *Hybrid Quantum Networks*

En esta sección utilizaremos un enfoque híbrido para resolver el dataset MNIST.

El dataset MNIST es un dataset de Computer Vision de imágenes de 128x128 píxeles con 1 canal en blanco y negro de números del 0-9 escritos a mano y digitalizados, donde la tarea es poder predecir que numero aparece en la imagen.

El enfoque híbrido se basa en utilizar una arquitectura convencional, en este caso utilizaremos capas de convolución 2d, dropout, Relu y por ultimo utilizaremos como una capa mas del modelo, un EstimatorQNN a través del modulo **TorchConnector**

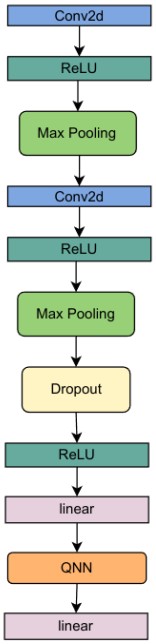


Figura 7: Red Convolucional Híbrida

En total la red tiene 17,470 parámetros entrenables, se utilizo el optimizador **Adam** y la función de perdida utilizada fue **NLLLoss** (Negative Log Likelihood Loss), se realizaron 10 epochs con un tiempo de entrenamiento total aproximado de 3 horas. Como se puede observar es muy costoso implementar una red híbrida y seria mucho mas eficiente dejar el modelo clásico solamente.

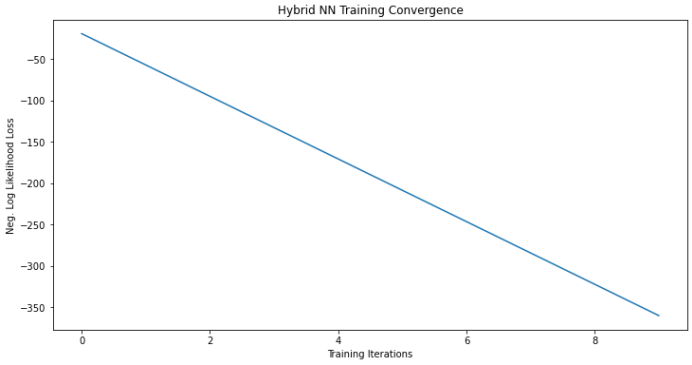


Figura 8: Entrenamiento de la red

| Model | Loss | Accuracy |
|---|---|---|
| Hybrid-CNN | -378.9845 | 0.113 |

Cuadro 3: Tabla de Resultados Red Híbrida

Como se puede observar la función de perdida va disminuyendo pero aun así por la poca cantidad de epochs, la red sigue sin aprender correctamente, ya que solo consigue un accuracy del 11.3 %, debido a que la red se demora demasiado al entrenar, no pudimos aumentar el numero de epochs pero para trabajos futuros seria interesante poder entrenar esta red híbrida sobre muchas mas epochs.

## 3. Quantum Key Distribution

El principio de Distribución de Llaves Cuánticas consiste en encontrar una clave segura para que pueda ser utilizada en el cifrado de la comunicación entre dos interlocutores de manera segura (Uttam Ghosh (2023)). Se implementa este principio según el procedimiento descrito por (Qiskit).

La clave segura se obtiene mediante la preparación de un Bit en Qubit utilizando uno de los cuatro posibles siguientes estados cuánticos: $|0>$, $|1>$ en el eje Z de la Esfera de Bloch, $|->$ y $|+>$ en el eje X, de ahora en adelante llamados **Base X** y **Base Z**, como es representado en la Figura 9.

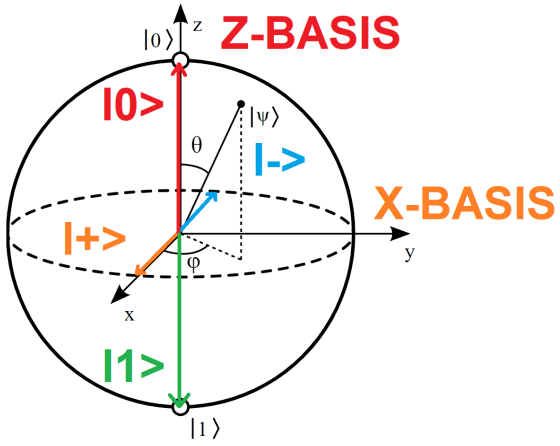


Figura 9: Esfera de Bloch

El principio es simple, Alice transmite un mensaje de prueba usando su propia codificación (Base propia), Bob lo recibe y lo decodifica según la codificación que determina. Luego intercambian los Bits que llegaron a destino para determinar una misma base común. Finalmente se concreta una Shared Key para ambos, que permite luego establecer una comunicación segura usando la llave obtenida con un algoritmo de encriptación.

Se utiliza Qiskit para manejar los circuitos cuánticos de cada Qubit. La regla general es la siguiente:

Se tiene el siguiente mensaje y base:

| Tipo | String |
|---|---|
| Mensaje_Test | 0011010111110101 |
| Base_Alice | 1000110010010000 |

Cuadro 4: Mensaje de inicio y Base de Alice

Esto genera un circuito cuántico, donde el primer elemento esta compuesto por una compuerta lógica de Hadamard, al presentar una codificación en base-**X** que supone una superposición de estados.

En general, para armar el circuito cuántico, para cada elemento se cumple la siguiente regla:

1. **Base 0**: Prepara la codificación en la Base-Z.

*a*) **Bit 0**: No requiere compuertas adicionales.

*b*) **Bit 1**: Requiere compuerta lógica **NOT**

1) Estados cuánticos en Qiskit comienzan en cero.

2. **Base 1**: Prepara la codificación en la Base-X.

*a*) **Bit 0**: Requiere compuerta de Hadamard

*b*) **Bit 1**: Requiere compuerta de Hadamard y además compuerta lógica **NOT**.

Como ejemplo, el primer item del circuito queda de la siguiente manera:

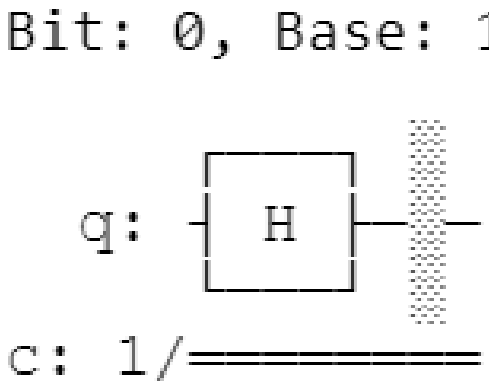


Figura 10: Extracto primer elemento Circuito Cuántico de la función **encode message()**

Bob obtiene el mensaje decodificado, sin ser la llave aún, utilizando la función **measure message**, que realiza la medición de cada elemento del circuito cuántico enviado por Alicia, considerando la base proporcionada(correspondiente a la de Bob para este caso). Si la base es **1**, entonces añade una compuerta lógica de Hadamard extra y luego realiza la medición, en caso contrario realiza la medición directamente.

Finalmente, se utiliza la función **remove no concidences(**Alicia_Base,Bob_Base,mensaje **)**, donde se almacena cada elemento del mensaje sólo si las bases de ambos coinciden. Alicia utiliza el mensaje que envía y Bob el que recibe, que puede o no ser el mismo dependiendo si fue interceptada la comunicación por un espía.

El resultado de esta función es la llave obtenida y ocurre lo siguiente:

- Si la llave es la misma para ambos interlocutores, es válida y funciona para establecer la comunicación de manera segura utilizandola en un algoritmo de encriptación.

- Cuando las llaves no coinciden, existe la posibilidad que exista un espía en la comunicación y se deben tomar acciones, utilizar un canal más seguro ó cortar la comunicación.

A continuación se analizan ambos escenarios con el objetivo de verificar que ocurra la no-coincidencia de Llaves cuando existe un espía en la comunicación.

### *3.1. Distribución de llave sin presencia de Eavesdropper*

En este escenario se analiza la distribución de llave cuántica cuando no existe la presencia de un Espía en la comunicación, similar al escenario propuesto por (GmbH (2021), pp. 4) en la sección de Quantum Key Distribution.

Alice manda un mensaje de inicio cuyo contenido son bits de prueba hacia Bob. Para mandar este mensaje lo hace codificando con una base propia y los transmite en el canal, Bob a su vez tambien tiene su base. Ambas bases son generadas aleatoriamente para este ejercicio con un largo de 16 bits.

| **Tipo** | **String** |
|---|---|
| Mensaje_Test | 01101111111001000010110011110 |
| Base_Alice | 10101101100101111101011101001 |
| Base_Bob | 10101000001100011010010111110 |

Cuadro 5: Mensaje de inicio y Bases, sin Intercepción.

Bob recibe este mensaje y lo decodifica utilizando su base propia, obteniendo una pre-Key asociada. La pre-Key de Alicia es **Mensaje_Test** enviado a Bob.

| **Pre-Key** | **String** |
|---|---|
| Alicia | 01101111111001000010110011110 |
| Bob | 01101111111000000000100001111 |

Cuadro 6: Pre-Key de Alicia y Bob, sin Intercepción.

Es posible notar que ambas Pre-Keys son distintas. El paso que sigue es escoger un subarreglo de ambos strings(con largo 16 bits para ser utilizado **AES** posteriormente), el cual almacena los valores binarios sólo si las bases asociadas de Alicia y Bob por cada elemento coinciden.

| **Key** | **String** |
|---|---|
| Alicia | 0110111000001001 |
| Bob | 0110111000001001 |

Cuadro 7: Llaves de Alicia y Bob con largo 16 bits, sin Intercepción.

De esta manera se consigue una Shared Key(común) entre ambos, observable en la figura anterior, cuando no existe un espía en la comunicación.

Con esta llave obtenida, Alice encripta el mensaje con el algoritmo **AES**. y lo manda a Bob. Bob a su vez utiliza su propia llave para desencriptar el mensaje, obteniendose los siguientes resultados:

| **Tipo** | **String** |
|---|---|
| Key_Alicia | 0110111000001001 |
| Key_Bob | 0110111000001001 |
| Mensaje a enviar | 1 2 3 2 2 1 |
| Mensaje codificado en Binario | 011011101001 |
| Mensaje AES(Alicia_Key) | 51fdcd90cfb6c91285d01... |
| Mensaje Recibido por Bob | Mensaje AES(Alicia_Key) |
| Mensaje Desencriptado(Bob_Key) | 011011101001 |
| Mensaje Obtenido( Texto) | 1 2 3 2 2 1 |

Cuadro 8: Mensaje enviado encriptado y Desencriptado, no intercepción.

Podemos notar que se transmitió el mensaje satisfactoriamente, de manera segura, entre Alice y Bob utilizando la llave obtenida por la método de distribución de llaves cuánticas.

Analizaremos en el siguiente escenario qué es lo que sucede cuando existe un espía en la comunicación.

### *3.2. Distribución de llave con presencia de Eavesdropper*

En este Escenario continua la comunicación entre ambos interlocutores, esta vez considerando un tercer personaje quien intercepta la comunicación, sin que lo sepa Alice ni Bob en principio.

Intuitivamente, en la computación cuántica re

| **Tipo** | **String** |
|---|---|
| Mensaje_Test | 01101111111001000001011001111 0 |
| Base_Alice | 00100110110101001000000111 1000 |
| Base_Eve | 11101110010011101111011111 1000 |
| Base_Bob | 01101010110101010001010001 1010 |

Cuadro 9: Mensaje de inicio y Bases, con Intercepción.

Eve intercepta este mensaje(y por tanto realizando la acción de medir cada Qubit), utilizando su propia base, y lo vuelve a enviar a Bob. Intuitivamente sabemos que el contenido enviado a Bob será distinto al que envió Alice debido a que Eve realiza una medición del mensaje, alterando el interior de este.

| **Tipo** | **String** |
|---|---|
| Key_Alicia | 11101001010001101110 |
| Key_Bob | 11101011111100111110 |
| Key_Eve | 110010100111011110 |
| Mensaje a enviar | 1 2 3 2 2 1 |
| Mensaje codificado en Binario | 011011101001 |
| Mensaje AES(Alicia_Key) | 977ac9c28d5ce23075770... |
| Mensaje Recibido por Bob | Mensaje AES(Alicia_Key) |
| Mensaje Desencriptado(Bob_Key) | ERROR |
| Mensaje Obtenido( Texto) | ERROR |

Cuadro 10: Mensaje enviado encriptado y Desencriptado, con Intercepción de Eve.

Dado que Alicia y Bob tienen llaves distintas, en el contexto que hay un espía en la comunicación(Eve), no se podrá llevar a cabo el cifrado y descifrado de la información, causando error en el mensaje desencriptado debido a que la Key no corresponde.

Alicia y Bob al notar esta diferencia en la distribución de llaves, deberían sospechar que la comunicación ha sido intervenida y por tanto cerrar la comunicación. En general, al utilizar la distribución de llaves cuánticas, se espera una llave simétrica para ambos en este escenario, sin considerar ruido en el canal. Cuando esto no ocurre es válido y recomendable sospechar la presencia de un espía y cerrar la comunicación. En síntesis, se verifica para este escenario que la intercepción de un espía afecta la distribución de llaves cuánticas causando un error al momento de desencriptar el mensaje enviado con la llave establecida. Cuanto más larga sea la longitud de la llave, aumenta la posibilidad de detectar a un espía en la comunicación si es que corresponde.

## 4. Conclusiones

En la sección de Quantum Machine Learning se pudo observar el desempeño de los distintos modelos y enfoques para resolver problemas de clasificación, hay que resaltar que gracias al uso de la librería Qiskit en python, se pueden crear modelos de forma muy fácil y simple, obteniendo resultados decentes que cada vez se acercan mas al desempeño de los algoritmos clásicos. También resaltar la integración con otras librerías como Pytorch para reutilizar conceptos y modelos de redes neuronales que son el estándar hoy en día, permitiendo poder desarrollar Quantum Machine Learning sin la necesidad de crear todo desde 0.

Respecto a la distribución de llaves cuánticas, se simulan dos escenarios utilizando Qiskit. EL primero corresponde a la determinación de una llave común entre dos interlocutores(Alice y Bob) haciendo uso de compuertas lógicas cuánticas como la de **Hadamard** y la compuerta **Not**. Con éxito se comprueba que Alice y Bob puedan generar una llave común que les permita luego cifrar y descrifrar la comunicación de manera segura. Concretamente, el procedimiento descrito debiese generar una llave común para Alice y Bob que pueda ser utilizada posteriormente como clave en un algoritmo de Cifrado, en este fue usado el algoritmo **AES-128** y se comprueba que la llave obtenida para ambos es idéntica, y por tanto Alice puede enviar los mensajes cifrados y Bob los descifra obteniendo el mensaje que fue enviado, sin errores. Si la generación de llaves utilizando el procedimiento descrito no resulta en una llave común para ambos (Alicia y Bob en este caso), entonces se debe sospechar que existe la presencia de un Espía en la comunicación. En estos casos se sugiere cortar la comunicación inmediatamente ya que no es segura. Lo anterior está intuitivamente relacionado con la superposición de estados cuánticos, cuando uno realiza la medición se obtiene un estado que puede ser distinto cada vez que se realiza la acción. Mientras mayor longitud tenga la llave, más probabilidades hay de detectar un espía en la comunicación cuando corresponde.

El segundo escenario es similar al primero con la inclusión de un espía en la comunicación. Basado intuitivamente en que cuando una realiza la medición en el mensaje preparado, deba cambiar el contenido, se comprueba la ocurrencia del cambio en el contenido del mensaje si una tercera persona(Eve) intercepta la comunicación leyendo el mensaje. En este caso la espía Eve cambia la llave común generada entre Alice y Bob y como consecuencia no pueden cifrar y descifrar la comunicación. Cuando ocurren este tipo de casos puede ser señal que la comunicación está siendo interceptada y es una de las ventajas del paradigma cuántico de la distribución de llaves, detección adecuada de Eavesdropper en la comunicación. Se comprueba este fenómeno en el escenario descrito anteriormente.

## Referencias


Dong, S., Wang, P., Abbas, K., 2021. A survey on deep learning and its applications. Computer Science Review 40, 100379.

GmbH, K., 2021. Quantum key distribution technology white paper. KEEQuant GmbH URL: https://shorturl.at/yHP59.

Gyongyosi, L., Imre, S., 2019. A survey on quantum computing technology. Computer Science Review 31, 51–71.

Massoli, F.V., Vadicamo, L., Amato, G., Falchi, F., 2022. A leap among quantum computing and quantum neural networks: A survey. arXiv:2107.03313.

Qiskit, . Quantum key distribution .

Uttam Ghosh, Debashis Das, P.C., 2023. A comprenhensive tutorial on cybersecurity in quantum computing paradigm. TechRxiv URL: https://doi.org/10.36227/techrxiv.22277251.v1.

Zhang, Y., Ni, Q., 2020. Recent advances in quantum machine learning. Quantum Engineering 2, e34.