COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Exploring Quantum Computing applications in Machine Learning: Variational Classifiers & Quantum Convolution

Author: Fernando Santana Garcia

Director: Alba Aparicio Pérez

Co-director: Alejandro Abol

Madrid, 2023

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Exploring Quantum Computing applications in Machine Learning: Variational Classifiers & Quantum Convolution

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Fernando Santana García Fecha: 05/07/2023

[illegible]

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Alba Aparicio Pérez Fecha: 05/07/2023

[illegible]

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Exploring Quantum Computing applications in Machine Learning: Variational Classifiers & Quantum Convolution

Author: Fernando Santana Garcia

Director: Alba Aparicio Pérez

Co-director: Alejandro Abol

Madrid, 2023

EXPLORING QUANTUM COMPUTING APPLICATIONS IN MACHINE LEARNING: VARIATIONAL CLASSIFIERS & QUANTUM CONVOLUTION

Author: Fernando Santana Garcia
Director, Co-director: Alba Aparicio Pérez, Alejandro Abol
Collaborating Entity: Accenture

RESUMEN

Este proyecto explora las aplicaciones de la computación cuántica (QC) en el ámbito del aprendizaje automático (ML). Comienza con un análisis de la mecánica cuántica, la computación cuántica, los algoritmos cuánticos y el aprendizaje automático cuántico. Se centra en el Clasificador Cuántico Variacional (QVC) y la Convolución Cuántica (QCNN), dos modelos híbridos cuántico-clásicos, destacando las sinergias entre la computación cuántica y el aprendizaje automático.

Se analiza el estado actual y el progreso en cuanto a hardware y algoritmos cuánticos, revisando las tecnologías más populares para construir qubits físicos y la literatura más reciente sobre aprendizaje automático cuántico. El trabajo resalta la clara brecha entre los algoritmos cuánticos en el marco teórico y su aplicabilidad práctica, evidenciando la necesidad de arquitecturas híbridas para acortarla.

Después, se implementan los modelos híbridos previamente discutidos, QVC y QCNN, usando la librería PennyLane con el plugin de Qiskit, y la plataforma de computación cuántica de IBM. En los resultados se muestran métricas de rendimiento para nuestros modelos en simuladores cuánticos y ordenadores cuánticos, evidenciando las limitaciones del hardware cuántico actual para las aplicaciones de aprendizaje automático. Los experimentos revelan el largo tiempo de ejecución para este tipo de algoritmos en los ordenadores cuánticos de acceso gratuito, como los disponibles en la plataforma IBMQ.

La tesis concluye con una reflexión sobre el potencial del aprendizaje automático cuántico, centrándose en los desafíos actuales y la escalabilidad futura.

Palabras Clave: Computación Cuántica, Aprendizaje Automático, Aprendizaje, Automático Cuántico, Clasificador Variacional Cuántico, Convolución cuántica, NISQ Pennylane

1. Introducción

Este proyecto explora la intersección de la computación cuántica y el aprendizaje automático, dos tecnologías con un gran potencial. Con los modelos de aprendizaje automático actuales que requieren cada vez más recursos, la computación cuántica podría ofrecer nuevas vías y posibilidades para su ejecución. En este proyecto, nos centramos en algoritmos híbridos cuántico-clásicos como el Clasificador Cuántico Variacional y la Convolución Cuántica (Quanvolution), que jugarán roles fundamentales para cortar la brecha cuántico-clásica.

El clasificador cuántico variacional ejemplifica este comportamiento híbrido al aprovechar un circuito cuántico parametrizado y un optimizador clásico para tareas de clasificación.

La red neuronal de convolución cuántica, otro modelo híbrido, busca mejorar las redes neuronales convolucionales, utilizadas especialmente para el procesamiento de imágenes, con el potencial de la computación cuántica. Ambos modelos representan una mezcla de técnicas clásicas de aprendizaje automático y principios cuánticos.

En el proyecto se realiza inicialmente un análisis de los conceptos teóricos, profundizando en la mecánica cuántica, la computación cuántica y luego sus aplicaciones en el aprendizaje automático. Se detalla el proceso de diseño y ejecución de estos algoritmos híbridos en la plataforma cuántica de IBM utilizando la biblioteca de Python PennyLane con populares conjuntos de datos. Destaca cómo la combinación de la computación cuántica y clásica puede abrir nuevas posibilidades para el aprendizaje automático.

2. Conceptos Teóricos

Mecánica Cuántica

La mecánica cuántica, principios que rigen el comportamiento de las partículas subatómicas, introduce conceptos fascinantes y cruciales para la computación cuántica. La superposición cuántica permite que las propiedades observables de una partícula existan simultáneamente en múltiples estados. Este principio cuántico fundamental se incorpora en la unidad básica de información cuántica - el bit cuántico o 'qubit'. A diferencia de los bits clásicos, que existen en estados definidos 0 o 1, los qubits pueden existir en una superposición de ambos estados, permitiendo el procesamiento paralelo, una de las claves de del potencial de la computación cuántica.

El entrelazamiento, otro fenómeno cuántico esencial, permite que las partículas interactúen instantáneamente, independientemente de la distancia espacial. En el ámbito de la computación cuántica, el entrelazamiento se utiliza para vincular qubits, generando una superposición de estados que permite el procesamiento cuántico. Esta misteriosa sincronización entre partículas es una propiedad fundamental utilizada en la teleportación cuántica, la criptografía cuántica y múltiples algoritmos cuánticos.

Notablemente, el problema de la medición en la mecánica cuántica resalta la naturaleza probabilística de la computación cuántica. En los sistemas cuánticos, las partículas permanecen en una superposición hasta que una medición las colapsa en un posible estado. Este proceso, crucial para la computación cuántica, introduce aleatoriedad, ya que los qubits colapsan en un estado específico (0/1) al medirse. Por tanto, para extraer datos significativos de los cálculos cuánticos, se requieren algoritmos avanzados capaces de abordar de manera efectiva el problema de medición.

(Feynman, 1963); (Susskind & Friedman, 2014)

Computación Cuántica

La computación cuántica aprovecha los principios anteriores para realizar operaciones que podrían ser exponencialmente más rápidas que los ordenadores clásicos. Los sistemas cuánticos utilizan qubits que existen en una superposición de estados representados por una combinación lineal de números complejos de estados de la base. Estos estados, así como las transformaciones, pueden visualizarse usando la esfera de Bloch (Quantiki, 2023) y a menudo se representan como matrices unitarias.

Los qubits se manipulan mediante puertas cuánticas, que son los elementos fundamentales de los circuitos cuánticos. Las puertas de un solo qubit incluyen la Identidad (I), Pauli (X, Y, Z), Fase (S, T), Hadamard (H) y las puertas de rotación (Rx, Ry, Rz). Estas tres últimas, en particular, juegan un papel clave en la creación de superposiciones y en la alteración de la fase de los qubits- Las puertas de múltiples qubits, como las puertas CNOT, CZ, SWAP y TOFFOLI, se utilizan para entrelazar qubits y realizar operaciones más complejas. Estas puertas se combinan en circuitos cuánticos, que luego se ejecutan en procesadores cuánticos.

El estado de un sistema cuántico se obtiene mediante realizando una medición. Sin embargo, debido a la naturaleza probabilística de la mecánica cuántica, las mediciones producen estados definitivos a partir de superposiciones, con las probabilidades dictadas por las amplitudes del vector de estado. Por lo tanto, la comprensión y manipulación de las puertas cuánticas, junto con una cuidadosa selección de la base para las mediciones, son fundamentales en la computación cuántica.

Aprendizaje Automático Cuántico

Para entender las aplicaciones de la computación cuántica en el aprendizaje automático, necesitamos comprender este concepto primero.

El aprendizaje automático (ML) es un campo que utiliza modelos matemáticos para identificar patrones y realizar predicciones a partir de los datos. Existen diferentes tipos de algoritmos de ML, como el aprendizaje supervisado (utilizando datos etiquetados), el aprendizaje no supervisado (encontrando patrones en datos no etiquetados) y el aprendizaje por refuerzo (adaptándose a través de retroalimentación). El proceso de implementar un algoritmo de ML implica la recopilación de datos, el preprocesamiento, la selección del modelo, el entrenamiento, las pruebas y la implementación. El éxito de un sistema de ML depende en gran medida de la calidad de los datos de entrenamiento y la elección adecuada del modelo.

El aprendizaje automático cuántico (QML) utiliza circuitos cuánticos para procesar información en tareas de aprendizaje automático, como clasificación y regresión. Uno de los desafíos principales es codificar datos clásicos en estados cuánticos. Sin embargo, los algoritmos híbridos cuántico-clásicos ofrecen soluciones al utilizar sistemas cuánticos para tareas computacionales complejas y sistemas clásicos para el preprocesamiento, el post-procesamiento y la gestión de elementos cuánticos.

El Clasificador Cuántico Variacional (VQC) (Schuld et al., 2018) es un algoritmo de aprendizaje automático híbrido cuántico-clásico que aprovecha el la computación cuántica y clásica para realizar tareas de clasificación de datos. El VQC emplea un sistema cuántico para la preparación de estados y circuitos variacionales, seguido de un bucle de optimización clásico que encuentra los parámetros óptimos para minimizar una función de coste. El algoritmo demuestra robustez frente al ruido y resulta beneficioso para su implementación en dispositivos cuánticos a corto plazo.

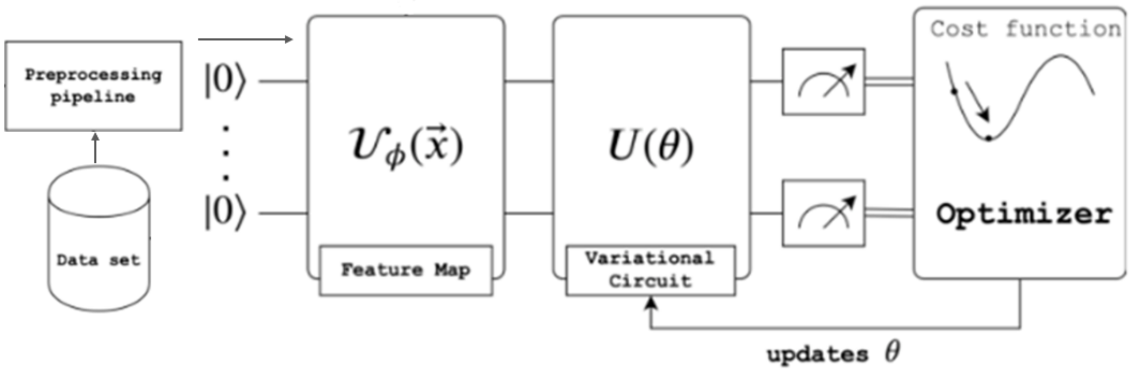


Figura 1 - Diagrama de Bloques del Clasificador Variacional. Fuente Q-munity: Building a VQC

La convolución cuántica, también conocida como "quanvolución" (Henderson et al., 2020), es un área innovadora de la computación cuántica inspirada en los procesos de convolución clásicos, fundamentales en las redes neuronales convolucionales (CNN). La quanvolución extiende el concepto de convolución a los sistemas cuánticos, utilizando circuitos cuánticos para transformar los datos de entrada, de forma similar a los filtros de las CNN clásicas.

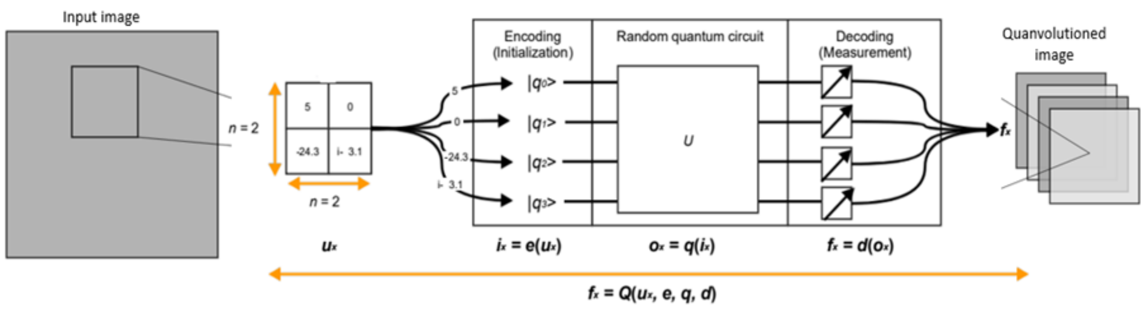


Figura 2 - Capa convolucional cuántica. Fuente: Elaborado a partir de (Henderson et al., 2020)

3. Tecnologías empleadas

Este proyecto utiliza principalmente el lenguaje de programación Python en un entorno de Visual Studio Code con Jupyter Notebooks, debido a la popularidad de Python en el aprendizaje automático y la ciencia de datos. La biblioteca de programación cuántica elegida es PennyLane (Bergholm et al., 2018), un software de código abierto desarrollado por Xanadu Quantum Technologies que facilita el aprendizaje automático cuántico. PennyLane es único por su uso de la programación cuántica diferenciable y su compatibilidad con varias plataformas de computación cuántica. Se utiliza el complemento PennyLane-Qiskit para integrar PennyLane con Qiskit de IBM, lo que permite la compilación y ejecución de circuitos cuánticos en el hardware cuántico de IBM.

IBM Quantum es la plataforma elegida para la simulación y computación cuántica. Esta plataforma ofrece servicios de computación cuántica basados en la nube, otorgando a los usuarios la capacidad de ejecutar programas cuánticos en ordenadores cuánticos reales y simuladores potentes ubicados en las instalaciones de IBM. IBM Quantum proporciona una variedad de ordenadores y simuladores cuánticos, cada uno con un número variable de qubits. Aunque los ordenadores más potentes solo están disponibles a nivel empresarial, algunas son de libre acceso para uso académico. Sin embargo, debido a la alta demanda, los tiempos de espera para la ejecución de circuitos cuánticos pueden ser largos. (IBMQ, 2023)

La ejecución de circuitos cuánticos, o 'trabajos', en la plataforma de IBM opera bajo un sistema basado en colas, lo que lleva a tiempos de espera variables que suelen oscilar entre una hora y varias horas. El tiempo de espera tota puede llegar a ser extremadamente largo dada la cantidad de trabajos necesarios para entrenar nuestros algoritmos híbridos. Los factores que afectan estos tiempos son la estructura del modelo, las características del conjunto de datos y las limitaciones del hardware y software cuántico actual.

4. Estado de la cuestión

La computación cuántica es un campo en gran medida teórico que recientemente ha experimentado algunos avances notables gracias a los nuevos desarrollos en tecnologías de hardware cuántico. Actualmente nos encontramos en la era cuántica de escala intermedia ruidosa (Noisy Intermediate-Scale Quantum, NISQ), que representa los ordenadores cuánticos actuales y del futuro próximo, con un bajo recuento de qubits, ruido y propensos a errores. A pesar de retos como mantener la coherencia cuántica, garantizar bajas temperaturas de funcionamiento y escalar a un gran número de qubits, los avances en hardware de computación cuántica son alentadores, y los investigadores se esfuerzan por aumentar los qubits, mejorar la calidad y realizar la corrección de errores. Se están desarrollando varios tipos de hardware cuántico, y actualmente la mayor inversión se destina a procesadores superconductores basados en compuertas.

El desarrollo de ordenadores cuánticos fiables no puede evaluarse únicamente por el número de qubits. Por ello, IBM ha introducido una métrica de rendimiento más holística denominada "volumen cuántico" (Baldwin & Mayer, 2022), que tiene en cuenta también otros factores como la calidad de los qubits, las tasas de error de puerta, la conectividad de los qubits y la eficiencia del compilador. El mayor volumen cuántico notificado hasta la fecha (06/2023) lo alcanzó el Model H1-1 de Quantinuum (Quantinuum News, 2023), siendo un gran avance en hardware de computación cuántica.

Los simuladores cuánticos, que imitan el comportamiento de un sistema cuántico utilizando computación clásica, desempeñan un papel vital en la computación cuántica, ya que ayudan a explorar nuevos algoritmos, aprender y experimentar sin altos costes. IBM lidera en este ámbito con algunos de los simuladores cuánticos gratuitos más potentes.

Muchos de los nuevos algoritmos cuánticos están diseñados para ordenadores cuánticos perfectos y, por lo tanto, aún no son factibles debido al número limitado de qubits lógicos prácticos. Están surgiendo resultados prometedores de algoritmos de computación cuántica variacional o métodos híbridos cuántico-clásicos, con los avances más notables en clustering, SVM y redes neuronales. Los circuitos cuánticos parametrizados, también llamados circuitos cuánticos variacionales, han sido ampliamente probados, mientras que la convolución cuántica es un concepto más reciente en experimentación.

5. Definición del proyecto

El objetivo es explorar y adaptar algoritmos novedosos de aprendizaje automático para hardware cuántico de escala intermedia ruidosa (NISQ), centrándose especialmente en los clasificadores variacionales y la convolución cuántica. Dado el interés activo de gigantes tecnológicos como IBM y Google, y la creciente demanda de los clientes de Accenture por soluciones computacionales cuánticas, este proyecto tiene como objetivo explorar las aplicaciones prácticas de QML, abordando tanto sus beneficios como sus desafíos.

Objetivos

- Explicar claramente la teoría detrás de la computación cuántica y sus aplicaciones en ciertos aspectos del aprendizaje automático.
- Programar y probar diferentes algoritmos.
- Obtener una precisión decente y mostrar cómo los modelos podrían ser escalables en el futuro con ordenadores cuánticos de mayor número de qubits.
- El objetivo no es superar a los ordenadores clásicas en velocidad o precisión, sino experimentar con la ejecución de algoritmos en ordenadores cuánticos reales y mostrar sinergias entre QC y ML.

Metodología

1. Investigación: Investigar algoritmos cuánticos existentes y bibliotecas de computación cuántica existentes para identificar los algoritmos más adecuados para la tarea de aprendizaje automático.

2. Explicación teórica de la física cuántica, las matemáticas detrás de la computación cuántica, las compuertas y circuitos cuánticos, cómo se puede aplicar la computación cuántica a los algoritmos de aprendizaje automático y cómo funcionan estos algoritmos en realidad.

3. Diseño: Codificar utilizando bibliotecas de python como Qiskit, PennyLane y TensorFlow, diferentes algoritmos cuánticos y variaciones.

4. Implementación: Implementar el algoritmo cuántico en los ordenadores cuánticos de IBM accediendo a sus máquinas de forma remota.

5. Evaluación: Evaluar los resultados en métricas como precisión, velocidad y escalabilidad, y si es necesario, refinar los algoritmos cuánticos.

6. Desarrollo

Clasificador Cuántico Variacional (VQC)

Comenzamos implementando un clasificador cuántico variacional, aprovechando tanto los algoritmos cuánticos para el procesamiento como los algoritmos clásicos para la optimización. El modelo se basa en circuitos cuánticos variacionales, compuestos de tres elementos principales: el feature map (o mapa de características), la capa variacional y la medición. (Schuld et al. 2018)

Inicialmente, definimos dos dispositivos cuánticos diferentes, uno para la ejecución en simuladores de IBM u ordenadores cuánticos y otro para la visualización de circuitos cuánticos. Vamos a usar el conjunto de datos Iris para la tarea de clasificación.

Primero, el mapa de características codifica datos clásicos en estados cuánticos, listos para ser manipulados en el ámbito cuántico. Usamos la incrustación de ángulo que emplea compuertas de rotación para codificar las entradas en estados cuánticos. La función luego devuelve los valores de expectación de las mediciones de Pauli-Z para cada qubit en el circuito.

Luego definimos la capa variacional que utiliza operaciones cuánticas parametrizadas que se pueden ajustar finamente para optimizar una función objetivo para la clasificación. La capa cuántica se define con una capa de compuertas de entrelazamiento con parámetros especificados en todos los cables del circuito (qubits). Luego, la etapa de medición convierte la información cuántica de nuevo en clásica.

Para unir todos estos componentes, integramos las arquitecturas cuánticas de Pennylane dentro del marco de Keras de TensorFlow para construir un modelo secuencial. Este modelo incorpora la capa cuántica variacional, convertida en una capa de Keras para la parte cuántica del modelo y una capa de activación regular de Keras con la función softmax al final para la clasificación.

Como extensión al modelo implementado, también diseñamos variaciones del modelo agregando capas densas clásicas antes o después de la capa cuántica. Esto no se hizo con la intención de mejorar el rendimiento, sino para demostrar la integración de capas cuánticas y clásicas en circuitos variacionales. La figura 3 muestra nuestro modelo base.

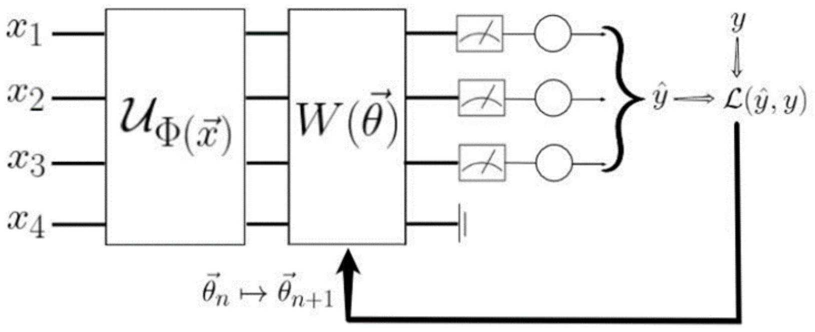

x_1
x_2
x_3
x_4
$\mathcal{U}_{\Phi(\vec{x})}$
$W(\vec{\theta})$
y
$\hat{y} \longrightarrow \mathcal{L}(\hat{y}, y)$
$\vec{\theta}_n \mapsto \vec{\theta}_{n+1}$

Figura 3 - Esquema del modelo de Clasificador Variacional. Fuente: realizado con github draw.io

<u>Convolución cuántica</u>

En esta sección, implementamos un modelo que consta de convolución cuántica y una red neuronal clásica (QCNN). La base de nuestro modelo radica en las convoluciones cuánticas que utilizan rotaciones unitarias parametrizadas realizadas en parejas vecinas de qubits. Estas convoluciones cuánticas son similares a sus contrapartes clásicas en las CNN tradicionales. Curiosamente, en lugar de seguir las capas convolucionales con capas de agrupación para la reducción de la dimensionalidad, logramos esto al medir un subconjunto de qubits en nuestra QCNN. (Henderson et al., 2020)

Nuestro estudio utiliza el conjunto de datos Fashion-MNIST, una colección de imágenes en escala de grises que representan diez categorías de prendas de vestir. Debido a la complejidad computacional y las restricciones de tiempo de los cálculos cuánticos, trabajamos con un conjunto de entrenamiento reducido de 200 imágenes y un conjunto de prueba de 60. Estas imágenes fueron normalizadas y remodeladas para ser compatibles con la operación de convolución cuántica.

El circuito cuántico para nuestra QCNN fue diseñado para manejar entradas de píxeles 2x2 de las imágenes, con cada píxel representado por un qubit. Se utilizaron compuertas de rotación RY para la codificación de cada píxel en estados cuánticos. A continuación, se aplicó un circuito cuántico aleatorio a los qubits, añadiendo un grado de aleatoriedad a las operaciones para la identificación de patrones complejos de los datos. El circuito se concluyó con una etapa de medición, proporcionando una forma de extracción de características.

Creamos una función para aplicar el circuito cuántico a las imágenes, dividiendo cada imagen en cuadrados de 2x2 píxeles, que fueron procesados por el circuito. La salida del circuito sirvió como una forma de extracción de características, reemplazando cada cuadrado de píxeles 2x2 con un solo píxel de salida que tenía cuatro canales o 'características'. La operación de convolución cuántica se aplicó entonces a todas las imágenes en el conjunto de datos como un paso de preprocesamiento. Esto nos permitió evitar la necesidad de ejecutar el circuito cuántico durante cada época del proceso de entrenamiento del modelo, ahorrando así recursos computacionales.

Finalmente, creamos un modelo simple de red neuronal clásica utilizando Keras para procesar los datos de imagen con convolución cuántica, con una función de activación softmax en su capa de salida para la clasificación multiclase. El modelo entrenado demostró cómo la QCNN puede integrar enfoques cuánticos y clásicos para abordar tareas de clasificación de imágenes. Las posibles modificaciones a esta arquitectura pueden incluir el aumento del número de filtros, la modificación del diseño del circuito cuántico, el cambio del tamaño del parche o la adición de más capas de quanvolución, ofreciendo amplias posibilidades para trabajos futuros en esta área. La figura 4 muestra la arquitectura.

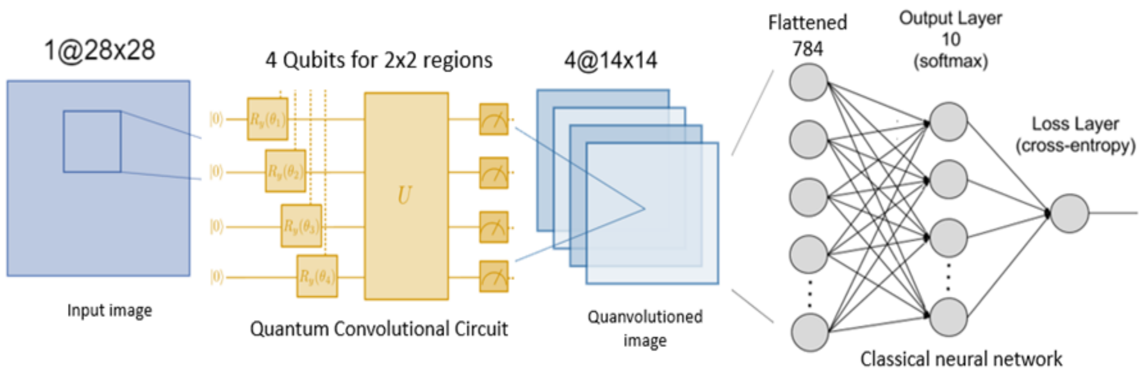


Figura 4 - QCNN: Arquitectura completa: red neuronal convolucional cuántica. Elaboración propia

7. Análisis de Resultados

<u>Clasificador Cuántico Variacional (VQC)</u>

Con el simulador cuántico, el modelo demuestra consistentemente una disminución de la pérdida y un aumento de la precisión a lo largo de diez épocas para ambos conjuntos de datos de entrenamiento y validación, indicando un aprendizaje exitoso y convergencia. La pequeña brecha entre los conjuntos de entrenamiento y validación sugiere la ausencia de un sobreajuste significativo. Además, la meseta observada en las gráficas de pérdida y precisión, después de alrededor de la quinta época, sugiere que el modelo ha logrado la convergencia. Posteriormente, se calcularon las métricas de rendimiento, revelando una precisión total del modelo del 83%. El rendimiento del modelo también fue probado con diferentes variaciones arquitectónicas, destacando la compatibilidad de la capa cuántica con marcos tradicionales de aprendizaje automático como TensorFlow Keras.

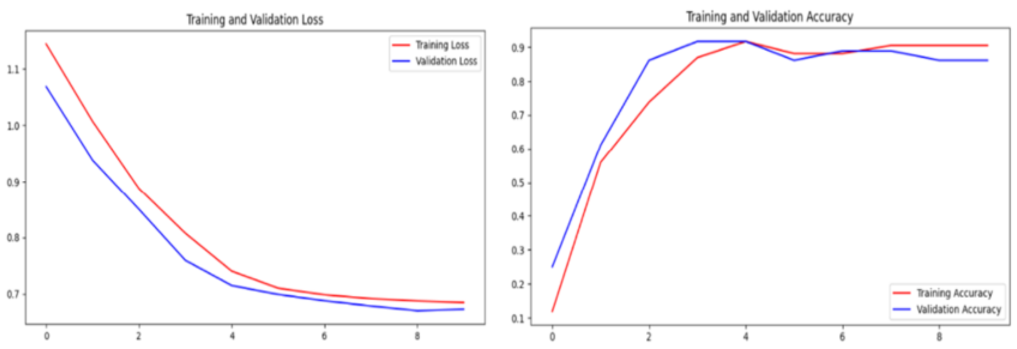


Figura 5 - entrenamiento y validación para el clasificador variacional en el simulador cuántico

Especie	precisión	recall	métrica-f1
0	1	1	1
1	1	0.5	0.67
2	0.67	1	0.8

Tabla 1- Métricas de rendimiento para el clasificador variacional ejecutado en el simulador cuántico

Accuracy	83%

Al pasar al ordenador cuántico real, se seleccionó el dispositivo "quito" de IBM debido a su capacidad para manejar el conjunto de datos Iris (4 características) con sus 5 qubits. Sin embargo, el entrenamiento del VQC en el hardware cuántico real presentó varios desafíos que llevaron a la ejecución de un gran número de trabajos. En particular, la integración de Pennylane y Keras con Qiskit resultó en un proceso de agrupación no soportado, lo que requiere la ejecución de un trabajo por punto de datos. Después de más de 40 horas, solo se completó la primera época. Esta limitación impidió una comparación directa de las métricas de rendimiento con el simulador cuántico.

Época 1			
Perdida - entrenamiento	Pérdida validación	Accuracy - entrenamiento	Accuracy - validación
1.28	1.32	0.18	0.27

Tabla 2 - Métricas de entrenamiento parciales: clasificador variacional en el ordenador cuántico real

Convolución Cuántica

Para el simulador cuántico, el modelo fue entrenado en un conjunto de datos relativamente pequeño, dada la naturaleza intensiva en tiempo del procesamiento convolucional cuántico. La pérdida y precisión del entrenamiento del modelo mejoró significativamente con cada época, sugiriendo un aprendizaje efectivo. Sin embargo, la pérdida de validación y la precisión de validación se estabilizaron mientras las métricas de entrenamiento continuaron mejorando, indicando un posible sobreajuste. Esta interpretación finalmente se confirma con la diferencia en la precisión final del entrenamiento y la prueba. Este sobreajuste no es inesperado debido al tamaño limitado del conjunto de datos de entrenamiento.

Accuracy en Entrenamiento	95%
Accuracy en Validación	62%

Tabla 3 - Precisión de entrenamiento y validación para la QCNN

Cuando la QCNN se comparó con una CNN clásica, el modelo cuántico rindió un poco mejor, pero con una diferencia insignificante que cae dentro de la variabilidad del entrenamiento de la red neuronal clásica. En resumen, con este conjunto de datos, modelo y procesadores cuánticos disponibles, no hay una ventaja clara para la convolución cuántica.

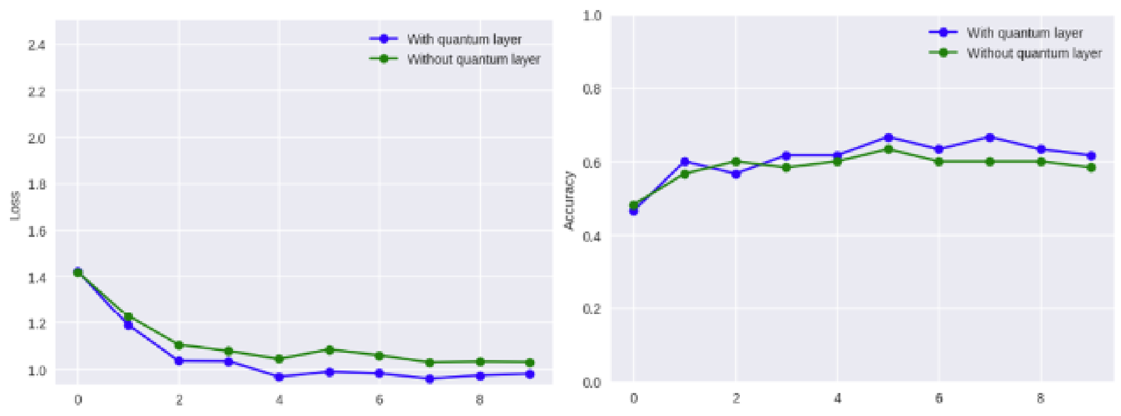


Figura 6 - Gráficos de entrenamiento y validación comparando la red neuronal con y sin la capa de convolución cuántica. El azul muestra con capa cuántica, el verde sin capa cuántica.
Fuente: Gráficos realizados con matplotlib

Al ejecutar el modelo en el ordenador cuántico real, se experimentaron problemas similares a los encontrados durante el experimento de VQC. La operación de convolución cuántica requirió descomponer una imagen estándar en ventanas de píxeles más pequeñas, lo que resultó en un número sustancial de ejecuciones de circuitos para cada imagen. Con el agrupamiento no totalmente soportado, el número de trabajos necesarios aumentó significativamente. En consecuencia, solo se procesaron cuatro imágenes dentro de un marco de tiempo razonable, subrayando los desafíos actuales de la computación cuántica para las tareas de aprendizaje automático. La ejecución de la convolución cuántica en un ordenador cuántico real se demostró con éxito en la siguiente imagen: Figura 7. Sin embargo, debido a que solo se pudieron procesar 4 imágenes, no tenía sentido entrenar luego la red neuronal clásica.

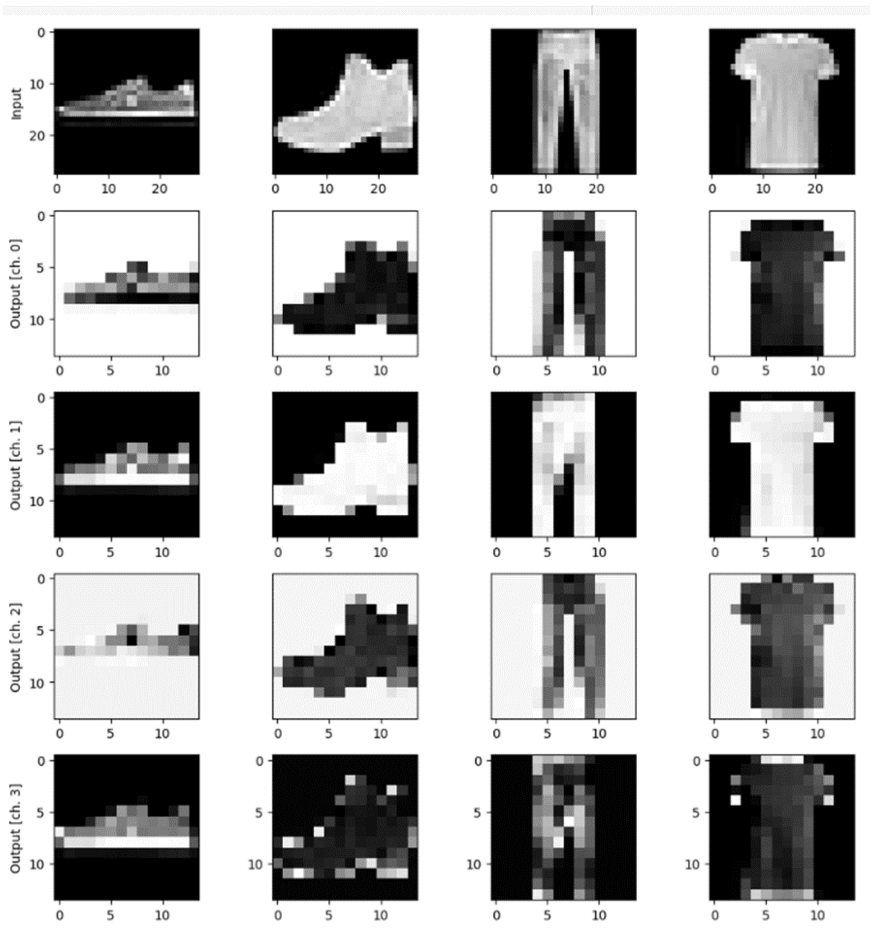


Figura 7 - Imágenes convolucionadas cuánticamente en un ordenador cuántico real.

8. Conclusiones y Trabajos Futuros

Nuestra exploración sobre la aplicabilidad de la computación cuántica dentro del aprendizaje automático ha proporcionado un análisis exhaustivo del estado actual del campo, así como de los desafíos persistentes. Evaluamos la integración de algoritmos cuánticos en los marcos de aprendizaje automático clásicos e identificamos la disparidad entre los algoritmos en la teoría y su implementación práctica en el hardware cuántico actual. Estos desafíos, que incluyen la limitación de qubits, los errores del sistema cuántico y la decoherencia limitan actualmente la aplicabilidad y adopción más amplias de los algoritmos cuánticos en el aprendizaje automático.

El trabajo futuro para este proyecto podría centrarse en explorar otras técnicas de aprendizaje automático, investigar una gama más amplia de modelos de aprendizaje automático cuántico, experimentar con variaciones de arquitectura de capas cuánticas dentro de modelos clásicos y evaluar el rendimiento en ordenadores cuánticos más potentes. A medida que avanza el campo de la computación cuántica, esto permitirá una comprensión más completa de los beneficios y limitaciones de los algoritmos cuánticos, contribuyendo al desarrollo de aplicaciones de aprendizaje automático cuántico más eficientes y prácticas.

9. Referencias

1. Feynman. (1963). Quantum Mechanics Lectures. Retrieved from The Feynman Lectures on Physics, Volume III: https://www.feynmanlectures.caltech.edu/III_toc.html

2. Susskind, L., & Friedman, A. (2014). Quantum mechanics: the theoretical minimum. Basic Books. Ch 1-8.

3. Quantiki Blosch Sphere. (2023). Retrieved from Quantiki: https://www.quantiki.org/wiki/bloch-sphere

4. Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Blank, C., McKiernan, K., & Killoran., N. (2018). PennyLane: Automatic differentiation of hybrid quantum-classical computations. arXiv:1811.04968.

5. IBMQ. (2023). Retrieved from IBM: https://quantum-computing.ibm.com/

6. Baldwin, C., & Mayer, K. (2022). Re-examining the quantum volume test: Ideal distributions, compiler optimizations, confidence intervals, and scalable resource estimations. Quantum. 6: 707. arXiv:2110.14808.

7. Quantinuum News. (2023). Retrieved from Quantinuum: https://www.quantinuum.com/news/quantinuum-h-series-quantum-computer-accelerates-through-3-more-performance-records-for-quantum-volume-217-218-and-219

8. Schuld, M., Bocharov, A., Svore, K., & Wiebe, N. (2018). Circuit-centric quantum classifiers. arXiv preprint arXiv:1804.00633.

9. Henderson, M., Shakya, S., Pradhan, S., & Cook, T. (2020). Quanvolutional neural networks: powering image recognition with quantum circuits. Quantum Machine Intelligence, 2(1), 2.

EXPLORING QUANTUM COMPUTING APPLICATIONS IN MACHINE LEARNING: VARIATIONAL CLASSIFIERS & QUANTUM CONVOLUTION

Author: Fernando Santana Garcia
Director, Co-director: Alba Aparicio Pérez, Alejandro Abol
Collaborating Entity: Accenture

ABSTRACT

This thesis explores quantum computing (QC) applications in machine learning (ML). It begins with an analysis of quantum mechanics, quantum computing, quantum algorithms and quantum machine learning. It focuses on the Variational Quantum Classifier (VQC) and Quanvolution (Quantum Convolution), two hybrid quantum-classical models, highlighting the synergies between QC and ML.

The current state of the field and the progress in quantum hardware and quantum algorithms are examined by reviewing popular technologies to build physical qubits and the latest literature on quantum machine learning. The work shows the clear gap between theoretical quantum algorithms and their practical applicability, evidencing the need for quantum-classical hybrid architectures to bridge the divide.

Then, the hybrid models previously discussed, VQC and QCNN, are implemented using technologies such as PennyLane, Qiskit, and IBM's Quantum Computing platform. Results provide performance metrics for our models on quantum simulators and quantum computers, acknowledging the limitations of current quantum hardware for ML applications. The experiments reveal how these types of algorithms take an extremely long time to run on freely available cloud quantum computers, such as the IBMQ platform.

The thesis concludes with a reflection on the potential for quantum machine learning, focusing on the current challenges and future scalability.

Keywords: Quantum Computing, Machine Learning, Quantum Machine Learning, Variational Quantum Classifier, Quantum Convolution, Pennylane, NISQ

1. Introduction

This project explores the intersection of quantum computing and machine learning, two powerful technologies with transformative potential. With state-of-the-art large machine learning models needing increasing resources, quantum computing could offer new ways and possibilities for the field. In this project, we focus on hybrid quantum-classical algorithms such as the Variational Quantum Classifier and Quantum Convolution (Quanvolution), that will be playing pivotal roles in bridging the quantum-classical divide.

The variational quantum classifier exemplifies this hybrid behavior by leveraging a parameterized quantum circuit and a classical optimizer for data classification tasks.

The quantum convolution neural network, another hybrid model, seeks to enhance convolutional neural networks, particularly used for visual data processing, with the potential of quantum computing. Both models represent an innovative mix of classical machine learning techniques and quantum principles.

This project unfolds an exhaustive examination of theoretical concepts, delving into quantum mechanics, quantum computing, and then their applications in machine learning. It details the process of designing and testing these hybrid algorithms on the IBM Quantum platform using the PennyLane python library with popular datasets. It highlights how the mix of quantum and classical computing can open new possibilities for machine learning.

2. Theoretical concepts

Quantum Mechanics

Quantum mechanics, the principles governing the behavior of subatomic particles, introduces fascinating concepts crucial to quantum computing. At its core, quantum superposition allows a particle's observable properties to exist simultaneously in multiple states. This fundamental quantum principle is incorporated into the basic unit of quantum information - the quantum bit or ‘qubit.’ Unlike classical bits, which exist in definite 0 or 1 states, qubits can exist in a superposition of both states, enabling the parallel processing of complex computations, a cornerstone for quantum computing's potential efficiency.

Quantum entanglement, another quantum phenomenon, allows particles to interact instantaneously, regardless of spatial distance. In the quantum computing realm, entanglement is utilized to link qubits, generating a superposition of states that enables quantum computation. This mysterious synchronization between particles is a fundamental property used in quantum teleportation, quantum cryptography, and multiple quantum algorithms.

Notably, the measurement problem in quantum mechanics underscores the probabilistic nature of quantum computing. In quantum systems, particles remain in a superposition until a measurement collapses them into one possible state. This process, pivotal to quantum computation, introduces randomness, as qubits collapse into a specific state (0/1) upon measurement. Consequently, extracting meaningful data from quantum computations needs advanced algorithms that can effectively navigate this measurement issue.

(Feynman, 1963); (Susskind & Friedman, 2014)

Quantum Computing

Quantum computing leverages the previous principles to perform operations that could be exponentially faster than classical computers. Quantum systems utilize qubits which exist in a superposition of states represented by a complex linear combination of basis states. These states, as well as transformations, can be visualized using the Bloch sphere (Quantiki, 2023) and are often represented as unitary matrices.

Qubits are manipulated using quantum gates, which form the building blocks for quantum circuits. Single qubit gates include the Identity (I), Pauli (X, Y, Z), Phase (S, T), Hadamard (H), and rotational (Rx, Ry, Rz) gates. The latter three, in particular, play a key role in creating superpositions and altering the phase of qubits. Multiple qubit gates, used to entangle qubits and perform more complex operations, include the CNOT, CZ, SWAP, and TOFFOLI gates. The quantum circuits formed by combining these gates are then executed on quantum processors.

The state of a quantum system is obtained via measurement. However, due to the probabilistic nature of quantum mechanics, measurements yield definitive states out of superpositions, with the probabilities dictated by the amplitudes of the state vector. The selection of the basis for these measurements significantly impacts the results obtained. Therefore, the understanding and manipulation of quantum gates, alongside careful basis selection for measurements, forms the core of quantum computation.

Quantum Machine Learning

To understand quantum computing applications on machine learning, we need to grasp this concept first.

Machine learning (ML) is a field that uses mathematical models to discern patterns and make predictions from data. There are different types of ML algorithms, such as supervised learning (using labeled data), unsupervised learning (finding patterns in unlabeled data), and reinforcement learning (adapting via feedback). The process of implementing an ML algorithm involves data collection, pre-processing, model selection, training, testing, and deployment. The success of an ML system depends mainly on the quality of training data and the chosen model's suitability.

Quantum machine learning (QML) uses quantum circuits for information processing in machine learning (e.g.: classification, regression) tasks. One of the main challenges is translating or encoding classical data into quantum states. However, hybrid quantum-classical machine learning algorithms offer some solutions by utilizing quantum systems for intricate computational tasks and classical systems for data pre-processing, post-processing, and managing the quantum element.

The Variational Quantum Classifier (VQC) (Schuld et al., 2018) is a promising hybrid quantum-classical machine learning algorithm that leverages the power of quantum and classical computation to perform data classification tasks. VQC employs a quantum system for state preparation and variational circuits, followed by a classical optimization loop that finds optimal parameters to minimize a cost function. The algorithm exhibits robustness against noise and proves beneficial for implementation on near-term quantum devices.

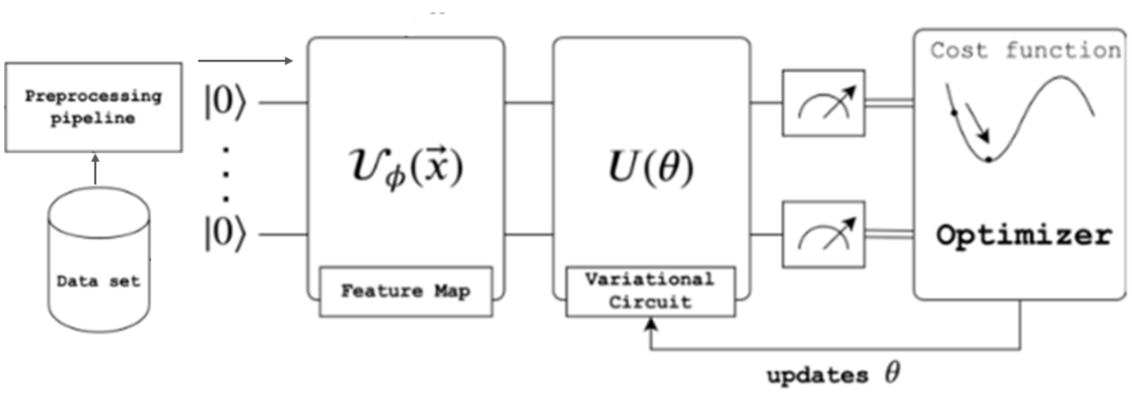


Figure 1 - Variational Classifier Block Diagram. Source Q-munity: Building a VQC

Quantum convolution, also known as 'Quanvolution' (Henderson et al., 2020), is a groundbreaking area of quantum computing inspired by the classical convolution processes fundamental to Convolutional Neural Networks (CNNs). Quanvolution extends the concept of convolution to quantum systems, using quantum circuits to transform input data, akin to filters in classical CNNs.

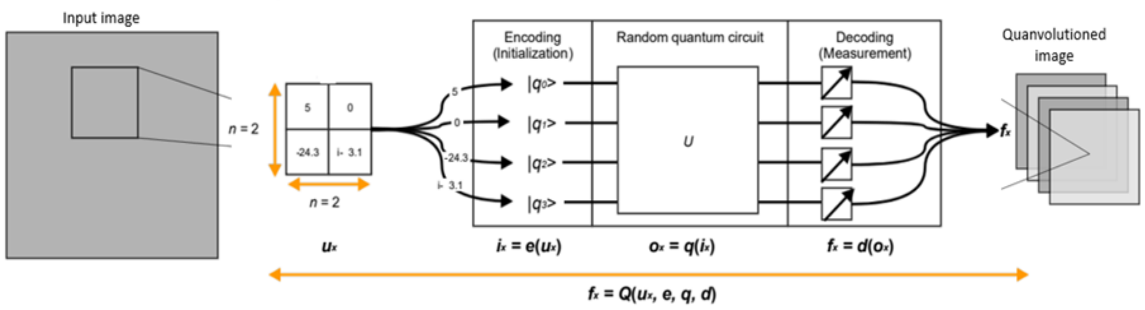


Figure 2 - Quantum Convolutional Layer. Source: Made from (Henderson, Shakya, et al. 2020)

3. Technologies used

This project primarily utilizes Python programming language in a Visual Studio Code environment with Jupyter Notebooks, due to Python's popularity in machine learning and data science. The quantum programming library chosen is PennyLane (Bergholm et al., 2018), an open-source software developed by Xanadu Quantum Technologies that facilitates quantum machine learning. PennyLane is unique for its use of differentiable quantum programming and its compatibility with various quantum computing platforms. PennyLane-Qiskit plugin is used to integrate PennyLane with IBM's Qiskit, enabling the compilation and execution of quantum circuits on IBM’s quantum hardware.

IBM Quantum is the chosen platform for quantum simulation and computation. This platform offers cloud-based quantum computing services, granting users the ability to run quantum programs on actual quantum computers and powerful simulators located in IBM's facilities. IBM Quantum provides a range of quantum computers and simulators, each with varying qubit counts. Although the most powerful computers are only available at an enterprise level, a few are freely accessible for academic use. However, due to the high demand, the waiting times for executing quantum circuits can be lengthy. (IBMQ, 2023)

The execution of quantum circuits, or 'jobs', on IBM’s platform operate under a queue-based system, leading to variable waiting times typically ranging from an hour to several hours. The cumulative waiting time can become extremely long given the numerous jobs required to train our hybrid algorithms. Factors affecting affect the waiting times include the model structure, dataset characteristics, and the limitations of the current quantum hardware and software.

4. State of the art

Quantum computing is a largely theoretical field that has recently made some remarkable advances owing to new developments in quantum hardware technologies. We are currently in the Noisy Intermediate-Scale Quantum (NISQ) era, representing the quantum computers of today and the near future, with a low qubit count, noisy and error-prone. Despite challenges such as maintaining quantum coherence, ensuring low operating temperatures, and scaling to large numbers of qubits, progress in quantum computing hardware is encouraging, with researchers striving to increase qubits, enhance quality, and perform error correction. Various types of quantum hardware are being developed, with the most investment currently in gate-based superconducting processors.

The development of reliable quantum computers cannot be assessed solely by qubit count. In response to this, IBM introduced a more holistic performance metric called Quantum Volume (Baldwin & Mayer, 2022), which considers also other factors such qubit quality, gate error rates, qubit connectivity and compiler efficiency. The highest reported quantum volume so far (06/2023) was achieved by Quantinuum's System Model H1-1 (Quantinuum News, 2023), indicating progress in quantum computing hardware.

Quantum simulators, which mimic the behavior of a quantum system using classical computation, play a vital role in quantum computing, aiding in exploring new applications, refining techniques, and teaching new generations of quantum programmers. IBM leads the way with some of the most powerful free-to-access quantum simulators.

Many of the new quantum algorithms are designed for perfect quantum computers and thus are not yet feasible due to limited numbers of practical logical qubits. Promising results are emerging from variational quantum computing algorithms or hybrid quantum-classical methods, with the most notable breakthroughs in quantum enhanced machine learning for clustering, SVM, and neural networks. Parametrized quantum circuits, also named variational quantum circuits, have been widely tested, while quantum convolution is a more recent concept under experimentation.

5. Scope of the Project

The goal is to explore and adapt novel machine learning algorithms for Noisy Intermediate Scale Quantum (NISQ) hardware, especially focusing on variational classifiers and quantum convolution. Given the active interest from tech giants like IBM and Google, and the increasing client demand at Accenture for quantum computational solutions, this project aims to explore practical applications of QML, addressing both its benefits and challenges.

Objectives

- Clearly explain the theory behind quantum computing and its applications in certain aspects of machine learning.
- Program and test different algorithms.

- Get decent accuracy and show how models could be scalable in the future with higher qubit quantum computers.
- The objective is not surpassing classical computers in speed or accuracy but experimenting with executing algorithms in real quantum computers and show synergies between QC and ML.

Methodology

1. Research: Research existing quantum algorithms and existing quantum computing libraries to identify the most suitable algorithms for the machine learning task.

2. Theoretical explanation of quantum physics, the math behind quantum computing, quantum gates and circuits, how quantum computing can be applied to machine learning algorithms and how these algorithms actually work.

3. Design: Code using python's libraries like Qiskit, PennyLane and TensorFlow, different quantum algorithms and variations.

4. Implementation: Implement the quantum algorithm on IBM quantum computers by accessing their machines remotely.

5. Evaluation: Evaluate the results on metrics like accuracy, speed, and scalability and refine the quantum algorithms if needed.

6. Development

Variational Quantum Classifier

We start by implementing a variational quantum classifier, leveraging both quantum algorithms for processing and classical algorithms for optimization. The model is based on variational quantum circuits, comprised of three main components: the feature map, the variational layer, and measurement. (Schuld et al. 2018)

Initially, we define two different quantum device backends, one for execution on IBM's simulators or quantum computers and another for the visualization of quantum circuits. We are going to use Iris dataset for the classification task.

First, the feature map encodes classical data into quantum states, ready to be manipulated in the quantum realm. We use angle embedding that employs rotation gates to encode inputs into quantum states. The function then returns the expectation values of Pauli-Z measurements for each qubit in the circuit.

Then we define the variational layer that utilizes parameterized quantum operations that can be fine-tuned to optimize a target function for classification. The quantum layer is defined with a layer of entangling gates with specified parameters on all the circuit wires (qubits). Then the measurement stage converts quantum information back into classical.

In order to bring all these components together, we integrate Pennylane's quantum architectures within TensorFlow's Keras framework to build a sequential model. This model incorporates the variational quantum layer, converted into a Keras layer for the quantum part of the model and a regular Keras activation layer with softmax function at the end for classification.

As an extension to the implemented model, we also design variations of the model by adding classical dense layers either before or after the quantum layer. This was not done with the intent of improving performance, but to demonstrate the integration of quantum and classical layers in variational circuits. Figure 3 shows our base model.

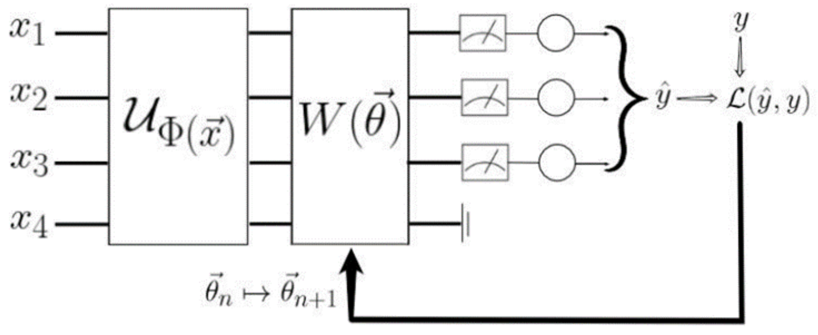

x_1
x_2
x_3
x_4
$\mathcal{U}_{\Phi(\vec{x})}$
$W(\vec{\theta})$
$\hat{y} \longrightarrow \mathcal{L}(\hat{y}, y)$
y
$\vec{\theta}_n \mapsto \vec{\theta}_{n+1}$

Figure 3 - Variational Classifier model scheme. Source: made using github draw.io

Quantum Convolution

In this section, we implement a model that consists of quantum convolution and a classical neural network (QCNN). The basis of our model lies in quantum convolutions, also termed as 'quanvolutions,' that utilize parameterized unitary rotations performed on neighboring pairs of qubits. These quantum convolutions are similar to their classical counterparts in traditional CNNs. Interestingly, instead of following convolutional layers with pooling layers for dimensionality reduction, we achieve this by measuring a subset of qubits in our QCNN. (Henderson et al., 2020)

Our study utilizes the Fashion-MNIST dataset, a collection of grayscale images representing ten categories of clothing items. Due to the computational complexity and time constraints of quantum computations, we worked with a reduced training set of 200 images and a test set of 60. These images were normalized and reshaped to be compatible with the quantum convolution operation.

The quantum circuit for our QCNN was designed to handle 2x2 pixel inputs from the images, with each pixel represented by a qubit. RY rotation gates were used for the encoding of each pixel into quantum states. Following this, a random quantum circuit was applied to the qubits, adding a degree of randomness to the operations, and facilitating the extraction of complex patterns from the data. The circuit was concluded with a measurement stage, providing a form of feature extraction.

We created a function to apply the quantum circuit to the images, dividing each image into squares of 2x2 pixels, which were each processed by the circuit. The output of the circuit served as a form of feature extraction, replacing each 2x2 pixel square with a single output pixel that had four channels or 'features.' The quantum convolution operation was then applied to all images in the dataset as a preprocessing step. This allowed us to bypass the necessity of running the quantum circuit during each epoch of the model's training process, thereby saving on computational resources.

Finally, we created a simple classical neural network model using Keras to process the quantum-convolved image data. with a softmax activation function in its output layer for multi-class classification. The trained model demonstrated how the QCNN can integrate quantum and classical approaches to tackle image classification tasks. Potential modifications to this architecture can include increasing the number of filters, modifying the quantum circuit design, changing the patch size, or adding more quanvolutional layers, offering extensive possibilities for future work in this area. Figure 4 shows the architecture.

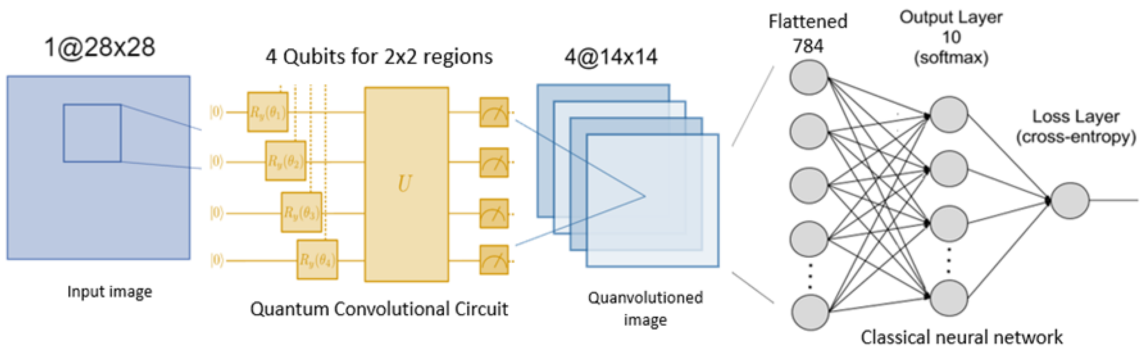


Figure 4 - QCNN: Quantum convolutional neural network full architecture. Source: self-made

7. Results Analysis

Variational Quantum Classifier

With the quantum simulator, the model consistently demonstrated decreased loss and increased accuracy across ten epochs for both the training and validation datasets, indicating successful learning and convergence. The small gap between the training and validation sets suggests the absence of any significant overfitting. Additionally, the plateau observed in the loss and accuracy graphs, after around the fifth epoch suggests that the model has achieved convergence. Performance metrics were subsequently calculated, revealing an overall model accuracy of 0.83. The model's performance was also tested with different architectural variations, highlighting the quantum layer's compatibility with traditional machine learning frameworks such as TensorFlow Keras.

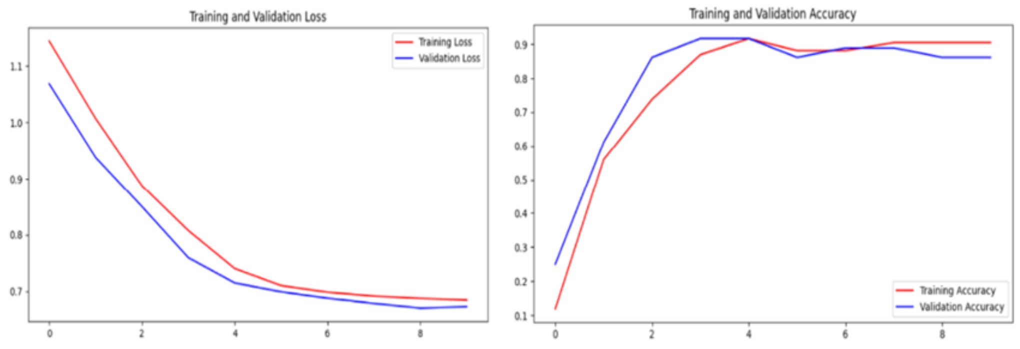


Figure 5 -Training & validation graphs for the variational classifier in quantum simulator

Feature	precision	recall	f1-score	support
0	1	1	1	10
1	1	0.5	0.67	10
2	0.67	1	0.8	10

Table 1- Full performance metrics for the variational classifier run in quantum simulator.

Accuracy	83%

When turning to the real quantum computer, the IBM's device "quito" was selected due to its capability to handle the Iris dataset (4 features) with its 5 qubits. However, the training of the VQC on the real quantum hardware presented several challenges that led to the execution of a large number of jobs. In particular, the integration of Pennylane and Keras with Qiskit resulted in an unsupported batching process, necessitating the execution of one job per data point. After more than 40 hours, only the first epoch was completed. This limitation impeded a direct comparison of performance metrics with the quantum simulator.

Epoch 1			
Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
1.28	1.32	0.18	0.27

Table 2 - *Partial training metrics for variational classifier in real quantum computer*

<u>Quantum Convolution</u>

For the quantum simulator, the model was trained on a relatively small dataset, given the time-intensive nature of quantum convolutional processing. The training loss and accuracy of the model significantly improved with each epoch, suggesting effective learning. However, the validation loss and validation accuracy plateaued while training metrics continued improving, indicating possible overfitting. This interpretation is finally confirmed with the difference in final training and test accuracy. This overfitting is not unexpected due to the limited size of the training dataset.

Training Accuracy	95%
Validation Accuracy	62%

Table 3 - *Training & Validation Accuracy for the QCNN*

When the QCNN was compared with a classical CNN, the quantum model performed slightly better but with an insignificant difference that falls within the variability of the classical neural network training. In summary, with this dataset, model and available quantum processors, there is no clear advantage for quantum convolution.

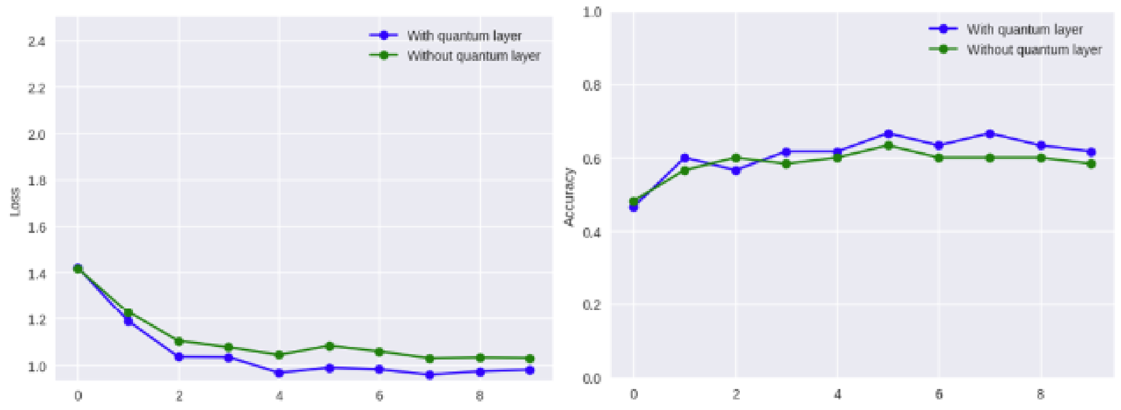


Figure 6 - T*raining & Validation Graphs comparing Neural network with & without Quantum convolution layer. Blue Shows With Quantum layer, Green without Quantum later.*
Source: Plotted with matplotlib

When executing the model on the real quantum computer, the issues encountered during the VQC experiment were similarly experienced. The quantum convolution operation required breaking down a standard image into smaller pixel windows, resulting in a substantial number of circuit runs for each image. With batching not fully supported, the number of necessary jobs increased significantly. Consequently, only four images were processed within a reasonable timeframe, underscoring the current challenges of quantum computing for machine learning tasks. The execution of quantum convolution on a real quantum computer was successfully demonstrated in the following image. However, due to only being able to process 4 images, it did not make sense to then train the classical neural network.

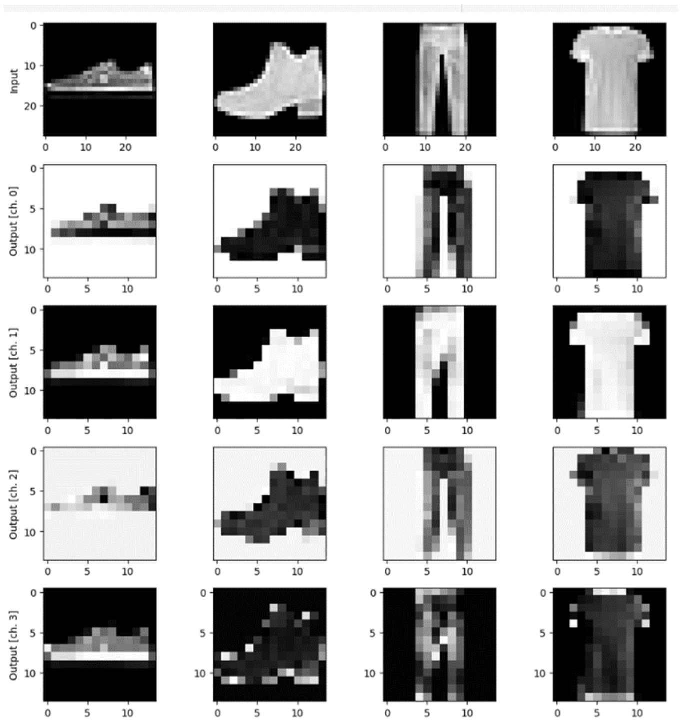


Figure 7 - Quantum convolved images on real Quantum Computer. Source: plotted matplotlib

8. Conclusions and Future Work

Our exploration into quantum computing applications within machine learning has provided significant insight into the current state of the field, as well as the challenges that persist. We evaluated the integration of quantum algorithms into classical machine learning frameworks and identified the disparity between theoretical quantum machine learning algorithms and their practical implementation on current quantum hardware. These challenges, including limited qubits, quantum system errors, and decoherence, currently limit the wider applicability and adoption of quantum algorithms in machine learning.

Future work for this project could be focused on exploring other machine learning techniques, investigating a broader array of quantum machine learning models, experimenting with architecture variations of quantum layers within classical models, and assessing the performance on more powerful quantum computers. As the field of quantum computing advances, this will enable a more thorough understanding of the benefits and limitations of quantum algorithms, contributing to the development of more efficient and practical quantum machine learning applications.

9. References

10. Feynman. (1963). Quantum Mechanics Lectures. Retrieved from The Feynman Lectures on Physics, Volume III: https://www.feynmanlectures.caltech.edu/III_toc.html

11. Susskind, L., & Friedman, A. (2014). Quantum mechanics: the theoretical minimum. Basic Books. Ch 1-8.

12. Quantiki Blosch Sphere. (2023). Retrieved from Quantiki: https://www.quantiki.org/wiki/bloch-sphere

13. Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Blank, C., McKiernan, K., & Killoran., N. (2018). PennyLane: Automatic differentiation of hybrid quantum-classical computations. arXiv:1811.04968.

14. IBMQ. (2023). Retrieved from IBM: https://quantum-computing.ibm.com/

15. Baldwin, C., & Mayer, K. (2022). Re-examining the quantum volume test: Ideal distributions, compiler optimizations, confidence intervals, and scalable resource estimations. Quantum. 6: 707. arXiv:2110.14808.

16. Quantinuum News. (2023). Retrieved from Quantinuum: https://www.quantinuum.com/news/quantinuum-h-series-quantum-computer-accelerates-through-3-more-performance-records-for-quantum-volume-217-218-and-219

17. Schuld, M., Bocharov, A., Svore, K., & Wiebe, N. (2018). Circuit-centric quantum classifiers. arXiv preprint arXiv:1804.00633.

18. Henderson, M., Shakya, S., Pradhan, S., & Cook, T. (2020). Quanvolutional neural networks: powering image recognition with quantum circuits. Quantum Machine Intelligence, 2(1), 2.

Table of Contents

Figure ToC

Table ToC

Section 1. INTRODUCTION

In this era of continuous technological evolution, one area that stands out for its potential to redefine our computational capabilities is Quantum Computing. As we persistently seek computational power beyond classical computing boundaries, Quantum Computing provides a ray of hope. It is in this captivating space where Quantum Computing intersects with Machine Learning, a well-established area that has been transforming sectors from healthcare to e-commerce, where the focus of this project lies.

Within Machine Learning, there has been a relentless endeavor to design algorithms that enable computers to learn from and predict data patterns effectively. However, the computational resources needed for state of the art, large models can be massive. Quantum computing could offer new ways to manage this load but to tap into this potential; it is important to bridge the quantum-classical divide. Hybrid quantum-classical algorithms, such as the Variational Quantum Classifier and Quantum Convolution (or Quanvolution), form the core of this bridge and are the central focus of our project.

The Variational Quantum Classifier is an interesting blend of classical and quantum computing principles, harnessing the best of both worlds. By employing a parameterized quantum circuit to prepare a quantum state and a classical optimizer to fine-tune the parameters, it represents a potent tool for complex data classification tasks.

Quantum Convolution, another hybrid approach, aims to augment Convolutional Neural Networks (CNNs), a class of deep learning models fundamental in processing visual data. The concept of Quantum Convolution, often called Quanvolution, seeks to bring the power of quantum computing into the realm of image processing. Both hybrid models are potential game-changers, offering an interesting fusion of classical machine learning techniques and quantum computing principles.

The project ahead will provide an in-depth understanding of these key concepts, navigating the intricacies of Quantum Mechanics, Quantum Computing, and Machine Learning. We will delve into how Quantum Computing has enhanced Machine Learning, illustrating this with the Variational Quantum Classifier and Quanvolution models. We will use the IBM Quantum platform and the PennyLane library, key technologies enabling our experiments in Quantum Machine Learning. We will design and implement variations of these two hybrid algorithms and test their performance, benefits and drawbacks with popular datasets using IBM's quantum simulators and quantum computers.

In the relentless quest for advancing our computational capacities, this project explores how quantum and classical computing merge, bringing new possibilities for machine learning.

Section 2. THEORETICAL CONCEPTS

2.1 *QUANTUM MECHANICS*

Quantum computing is a complex field that first requires familiarization with the basic principles of quantum mechanics as they serve as its foundation. This understanding must be paired with a proficient ability to operate within its rigorous mathematical framework, essential for the design and experimentation with quantum computing algorithms. (Feynman, 1963), (Susskind, 2008), (Susskind & Friedman, 2014)

2.1.1 MATHEMATICAL FRAMEWORK FOR QUANTUM MECHANICS

2.1.1.1 Linear Algebra

The behavior of subatomic particles, like electrons, cannot be explained using the same physical rules applied to larger objects in the classical world. Classical physics describes objects through continuous functions capable of taking any value within a specific range. However, in the quantum realm, particles act like waves and are defined by discrete energy levels. Hence, the rules for the macroscopic world fall short when explaining the behavior of particles.

A system's physical state is represented using a vector in a complex vector space. These vectors, known as state vectors or wavefunctions, bear the symbol Ψ. The wavefunction's magnitude squared reveals the probability of the system being in a particular state. This probabilistic aspect of quantum mechanics implies that, unlike classical physics, we cannot predict an exact measurement outcome; we can only estimate the probability of different outcomes.

The question is: why does linear algebra serve as the ideal mathematical tool for representing quantum mechanics? The answer lies in the fact that quantum systems' properties can be described by linear operators, which act on the system's state vectors. Linear algebra depicts these operators as matrices, and their properties can be analyzed using techniques like eigenvalues and eigenvectors. For instance, the Schrödinger equation, quantum mechanics' fundamental equation, is a linear partial differential equation. Linear algebra offers an intuitive framework for comprehending the mathematical foundation of quantum mechanics.

2.1.1.2 Vector Space and Wave function

In quantum mechanics, particle states are represented by vectors of complex numbers, with the evolution of the quantum system represented through operations with matrices of complex numbers. We use the Dirac Notation, a ket (a column vector), to represent a quantum state. Contrary to the macroscopic state, a ket represents a quantum state that provides information about the properties of a physical system at a particular moment. However, each time these properties are measured, different values may be obtained due to the probabilistic nature of quantum mechanics.

2.1.1.2.1 Complex Numbers and basis

In quantum mechanics we are especially interested in space C^n being to the set of n-tuples containing complex numbers. Complex numbers are numbers that include a real part and an imaginary part, often denoted as z=a + bi, where z is the complex number, a and b are real numbers and i is the imaginary unit, with the property that i^2 = -1 or i = $\sqrt{-1}$. The structure and dimension of a vector space is described by a basis. A basis is a set of linearly independent vectors that can be combined to represent any vector in the space.

2.1.1.2.2 The wave function

The **wavefunction** is a mathematical tool that outlines a particle's quantum state concerning its position. Often represented by the symbol Psi (Ψ), it is a complexvalued function of position that meets certain mathematical criteria. The wavefunction helps calculate the probabilities of different physical observables like position, momentum, and energy, along with their uncertainties. In quantum mechanics, the wavefunction is vital for predicting particle behavior and interactions.

2.1.1.3 Hilbert Space

In quantum mechanics, the space of quantum states can have infinitely many dimensions. This is because a quantum state can be a linear combination of an infinite number of possible outcomes, and we use these outcomes as a basis for our calculations. However, working with infinity in mathematics can be tricky because it can lead to situations where objects fall outside our defined space. Therefore, we need to be careful when dealing with infinite dimensions in quantum mechanics.

To avoid this problem in quantum mechanics, we add an extra rule to our vector space: **every convergent sum of vectors must converge to an element inside our vector space**. This rule creates a **Hilbert space**: a vector space with a defined inner product that is Cauchy complete. Cauchy complete means that every convergent sequence of vectors converges to an element inside the vector space. By defining the Hilbert space with these properties, we ensure that every quantum state is inside our vector space, and we can use linear algebra to describe our particles with confidence.

The Hilbert Space is the mathematical space that describes the possible states of a quantum system. It is a complex vector space, meaning that its elements are complex numbers that can be added and multiplied. (Quantiki Hilbert Space, 2023)

The Hilbert Space is used to represent the states of particles, which can be described as a superposition of multiple energy levels. For example, an electron in an atom can be in a superposition of different energy states. The Hilbert Space provides a framework for describing the probability of a particle being in a particular state, as well as for calculating the evolution of the system over time.

2.1.1.4 Inner Product

In the context of quantum mechanics, the concept of inner products is crucial for understanding the behavior of quantum states. Inner products, an abstraction of the dot product, are central to defining angle and orthogonality in Hilbert spaces.

One of the essential applications of inner products in quantum mechanics is the ability to work with orthogonal basis vectors. Orthogonal basis vectors allow us to expand quantum states in an orthonormal basis, simplifying the calculations and interpretation of quantum systems.
In quantum mechanics, we use the **Dirac or bra-ket notation** when representing vectors. Likewise, the inner product is denoted using this notation as the following:

$$\text{Inner product of (v,w)} = \langle v|\text{w}$$

Kets $|\mathbf{w}\rangle$ can be identified as column vectors, and **bras $\langle \boldsymbol{v}|$** as row vectors.
This notation makes it easier to manipulate and understand mathematical expressions in quantum mechanics.

2.1.1.5 Dirac's delta

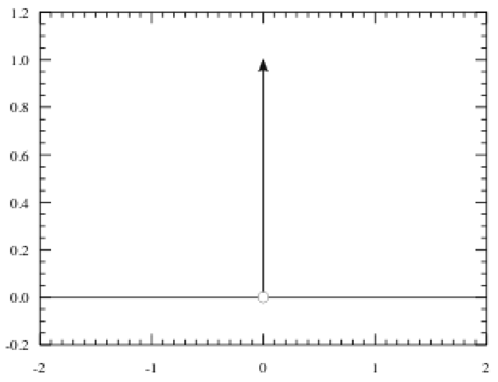


Figure 1 - Representation of the Dirac delta function. Source: Wolfram Alpha

The Dirac delta is a generalized function whose value is zero everywhere except at zero, and whose integral over the entire real line is equal to one. The Dirac delta function serves a significant role in mathematical and physical applications by 'selecting' a specific value from a continuous function when it is integrated with it. This function behaves like a 'pronounced peak' at the desired value while retaining zero value everywhere else.

The Dirac delta function plays a crucial role in quantum mechanics by providing a mathematical tool for working with continuous systems, particularly in the context of wavefunction inner products. It allows us to extract specific coefficients from continuous functions and enables us to work with continuous orthonormal inner products, which are essential for understanding and analyzing quantum states.

The Dirac delta is essential when working with continuous orthonormal inner products of qubit states, a key element in the understanding and operation of quantum computers.

2.1.1.6 Observables operators

Observables are any measurable physical quantity from a particle, such as: position, momentum, energy, and angular momentum. These observables are represented by linear operators on the Hilbert space of quantum states, or kets.

To find the possible values of an observable that can be measured, we look at the eigenvalues of the linear operator representing that observable. The eigenvectors corresponding to these eigenvalues are called eigenstates and represent the definite states of a particle with 100% certainty of having a specific value for the observable.

A quantum state can be expressed as a superposition of all potential measurement outcomes by forming a linear combination of the observable's eigenstates. Physical observables have several properties:

- Observables must have real eigenvalues, as physical quantities are real.
- Eigenstates of observables must encompass the entire vector space. This ensures any quantum state can be written as a linear combination of eigenstates.

- Eigenstates must be mutually orthogonal, ensuring that the definite states are clearly defined.

Taking these properties into account, it is safe to infer that the eigenstates of an observable constitute an orthonormal eigenbasis. This understanding holds crucial relevance in quantum computing. Observables serve as indispensable tools in quantifying attributes such as position, momentum, and energy of qubits, thereby playing a key role in the realm of quantum computing.

2.1.1.7 Probability, amplitude, and the Born Rule

To predict the likelihood of a specific outcome, like measuring an energy value, we use probabilities. Based on the Born rule, the observation probability of a particular outcome is determined by squaring the absolute value of the corresponding eigenstate's amplitude. In other words, if a system is in a superposition of several states, the probability of measuring a particular state is proportional to the square of the magnitude of the corresponding coefficient in the superposition.

Mathematically, if we represent the wave function of a quantum system as ψ, and we measure an observable with eigenstates $|a\rangle$, $|b\rangle$, $|c\rangle$: the probability of measuring the system in state $|a\rangle$ is given by $|\langle a|\psi\rangle|^2$. Similarly, the probability of measuring the system in state $|b\rangle$ is given by $|\langle b|\psi\rangle|^2$, and so on.

When dealing with quantum computing, if a quantum system (qubits) is in a superposition of states, the Born rule provides the means to calculate the probability of measuring a specific state, which is crucial to quantum algorithm execution.

2.1.1.8 Unitary operators

Unitary operators in quantum mechanics are a special class of linear operators that play a crucial role in maintaining the conservation of probability. They preserve the inner product of two vectors when both vectors are transformed by the operator.

This property ensures that the lengths of the vectors and the angles between them remain unchanged, which makes unitary operators resemble generalized rotations.

A unitary operator, denoted by U, is defined as an operator that satisfies the condition U $* \; U^{\dagger} =$ I, where $U^{\dagger}$ is the Hermitian conjugate of the operator U, and I is the identity operator. This means that the Hermitian conjugate of a unitary operator is equal to its inverse.

One essential property of unitary operators is that their eigenvalues must have a magnitude of one. In other words, the eigenvalues are unit complex numbers. This property is consistent with the idea that unitary operators, as generalized rotations, should not change the lengths of their eigenvectors.

In quantum mechanics, unitary operators are important because they conserve probability. Inner products are primarily used to calculate probabilities in quantum mechanics, and since unitary operators preserve inner products, they ensure that the probability of obtaining a specific measurement remains unchanged when a unitary operator acts on every vector in the space. Additionally, the total probability of a state remains equal to one.

In quantum mechanics, processes such as rotation, translation, and time evolution should conserve total probability. Therefore, the corresponding operators for these transformations need to be unitary. For example, in the context of the Schrödinger equation, time evolution is governed by a unitary operator to maintain the conservation of probability.

Unitary operators, crucial in quantum mechanics for probability conservation, have a vital role in quantum computing as they represent the quantum gates in a quantum circuit.

2.1.2 Quantum Physics

After going over the mathematical intuition and tools behind quantum mechanics, it is time to understand some of the physical principles of Quantum mechanics and how they relate to quantum computing.

Quantum mechanics is the set of rules for the smallest entities in the universe: atoms and subatomic particles. It fills in where classical physics, which works well for larger, macroscopic phenomena, falls short.

Quantum physics history starts in the late 1800s and early 1900s from experimental atom observations that defied classical physics' intuition. A fundamental principle is wave-particle duality which originated from experiments showing that light and matter exhibited both particle and wave properties.

2.1.2.1 Double Slit Experiment

There are multiple variations of the double-slit experiment, with Young's early 1800s version laying the groundwork (Young, 1804). This experiment introduced the wave theory of light, challenging Newton's corpuscular theory of light, and a century later, Einstein's publications about the photoelectric effect revealed light's particle-like behavior. This contradiction could only be solved with a quantum understanding of light.

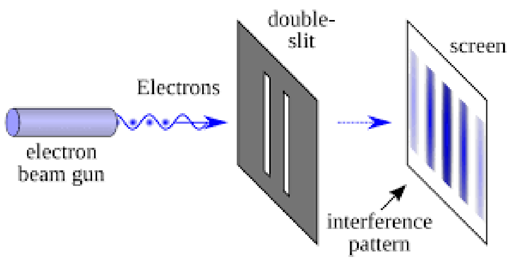


Figure 2 Double-slit experiment diagram. Source: Wikipedia

The experiment involves sending a beam of particles, such as electrons or photons, toward a barrier with two slits and a detection screen behind it. According to classical mechanics, we would expect the particles to pass through either slit, forming two separate patterns on the screen. However, the actual results reveal a surprising interference pattern, like the overlapping ripples seen in wave interference.

Things get even more interesting when we try to measure which slit each particle goes through. When we observe the process, the interference pattern vanishes, and the expected two separate patterns appear again. This strange behavior demonstrates the wave-particle duality, where particles exhibit both wave and particle properties.

In April 2023, a team led by physicists from Imperial College London carried out a novel variation of the double-slit experiment, using 'slits' in time rather than space (Tirole, et al., 2023). By firing light through a swiftly changing material, they permitted light to pass only at specific times. This experiment shows how we can transfer concepts such as interference from the domain of space to the domain of time and this could potentially lead to the development of ultrafast optical switches.

2.1.2.2 Quantum Superposition

Unlike classical mechanics, where properties like position or momentum are always well-defined, quantum mechanics allows that particle's observable properties like position or energy, simultaneously possess more than one possible value. This is not just a gap in our understanding of the system but an inherent property of quantum systems.

An essential element of quantum superposition is its linear nature, allowing quantum states to be added together (superposed) to form another valid quantum state. The Schrödinger equation, fundamental in quantum mechanics, underlines this by asserting that any linear combination of solutions will also be a solution. This principle is manifested in experiments like double-slit interference, where superposed wave functions produce an interference pattern.

This principle translates to quantum computing, where the fundamental unit of information is the quantum bit or "qubit" which, unlike classical bits, can exist simultaneously in both 0 and 1 states due to superposition. This enables quantum computers to process complex computations in different, sometimes more efficient ways.

2.1.2.3 Heisenberg Uncertainty Principle

The uncertainty principle (formulated by the German physicist Werner Heisenberg in 1927) states that we cannot know specific pairs of a particle's properties with complete precision at the same time. The most famous example involves the uncertainty between position and momentum. Simply put, the more we know about a particle's position, the less we can know about its momentum, and vice versa.

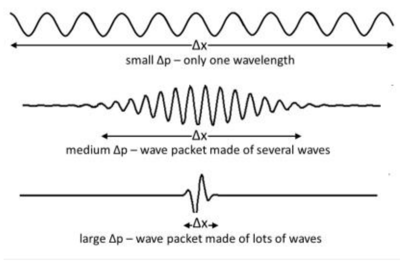

Δx
small Δp – only one wavelength
Δx
medium Δp – wave packet made of several waves
Δx
large Δp – wave packet made of lots of waves

Figure 3 - Wavelength comparison. Source: Feynman PH300 Modern Physics SP11 Slides

- The first wave, with a consistent wavelength and amplitude, represents a particle with a well-defined momentum (related to wavelength) but uncertain position (spread out amplitude).
- In the second image, we see the wavelength varying, implying that the momentum is not well-defined. However, we can infer a probable position where the amplitude peaks (in the middle)
- Finally, the third wave is essentially a spike, with its amplitude peaking sharply at one point. Here, the position is highly certain (at the peak), but the momentum is highly uncertain due to the lack of a consistent wavelength.

This graph encapsulates the uncertainty principle: the more precisely the position of a particle is determined, the less precisely its momentum can be known, and vice versa.

Regarding quantum computing, the uncertainty principle supports the concept of superposition and is the foundation of the probabilistic nature of quantum computing.

2.1.2.4 Measurement Problem

The measurement problem in quantum mechanics refers to how quantum systems remain in a superposition state until measured when it collapses to one of the possible states instantly.

Quantum mechanics principles state that the wave function (Ψ mathematical encapsulation of quantum system) develops deterministically and linearly as per the Schrödinger equation. However, when we measure it, the wave function seems to collapse instantly and randomly into one of its possible eigenstates. This non-deterministic collapse contradicts the deterministic evolution dictated by the Schrödinger equation.

This inconsistency creates a challenge in interpreting quantum mechanics because it is unclear why the wave function collapses upon measurement or what defines a measurement. Several interpretations of quantum mechanics have been proposed to tackle the measurement problem, such as the Copenhagen interpretation, Many Worlds Interpretation, and de Broglie-Bohm pilot-wave theory. Each interpretation offers a unique perspective on the wave function's nature and the role of measurement. The Copenhagen interpretation, being one of the most accepted, states that "the past is determined, the future is uncertain.

The measurement problem is relevant to quantum computing because the process of reading out the result of a quantum computation involves measuring the state of the qubits. Since qubits can exist in a superposition of two states (0 and 1), measuring them causes their wave function to collapse into one specific state, which is then read as a classical bit (0 or 1). This means that the outcome of quantum computation is inherently probabilistic.

Moreover, the measurement problem highlights the limitations of quantum computers. For instance, although quantum computers can perform many calculations simultaneously due to the superposition of qubits, extracting useful information from the results requires sophisticated algorithms and techniques that can work around the measurement problem.

The development of quantum error correction methods and fault-tolerant quantum computing is also closely related to the measurement problem. These methods aim to mitigate the effects of decoherence and measurement-induced errors to achieve more reliable and accurate quantum computers. By better understanding the measurement problem and its implications, researchers can develop new strategies to harness the full potential of quantum computing.

2.1.2.5 Entanglement

In the realm of quantum mechanics, one of the most intriguing phenomena is quantum entanglement. Here, two particles become intertwined, their states deeply connected, regardless of the vast distances that may separate them. This synchrony between particles is instantaneous, implying that when the wave function of one entangled particle collapses, its entangled partner's wave function also collapses instantly.

This peculiar phenomenon, often perceived as 'spooky action at a distance' has been experimentally confirmed and defies our conventional understanding of space and time. However, it is important to note that entanglement does not allow for the transfer of information faster than light, aligning with Einstein's theory of relativity.

In quantum computing, entanglement is harnessed to link qubits, creating a superposition of states that allows for the simultaneous processing of computations. It is one of the most important quantum properties for quantum computing, being key to quantum teleportation, quantum cryptography and several quantum algorithms.

2.1.2.6 Physics Nobel Prize 2022

The Nobel Prize in Physics 2022 was awarded to Alain Aspect, John F. Clauser, and Anton Zeilinger "for experiments with entangled photons, establishing the violation of Bell inequalities and pioneering quantum information science." (Sinha, 2023)

The laureates' collective work has made significant strides in exploring the phenomenon of quantum entanglement, confirming the counterintuitive predictions of quantum mechanics. Their research addressed critical aspects like quantum teleportation, loophole closure in entanglement measurements, and empirical violations of Bell's inequality. This work has reaffirmed the principles of quantum superposition and instant communication between entangled particles, thus paving the way for revolutionary breakthroughs in quantum information technology.

2.2 QUANTUM COMPUTING

2.2.1 WHY QUANTUM

Quantum computing's appeal is its capability to execute some operations that require exponential time on classical computers in a linear time frame. Exclusive quantum mechanics properties, such as superposition and entanglement, enable quantum computers to perform calculations in unattainable ways by classical computers. For instance, a quantum computer can leverage superposition to encode multiple problem solutions concurrently, which facilitates more efficient and quicker search processes through various possibilities compared to a classical computer.

However, the solution to the problem is still a superposition of many different answers where many are incorrect, and when measuring, we will only get one of the answers, that probabilistically is very rare to be the correct one. Quantum algorithms are all about shuffling around this exponential collection of terms in the superposition so that, when we observe the system, our random snapshot has a high probability of showing us the thing we are looking for.

Quantum computers could potentially revolutionize fields like drug discovery, material discovery, chemical simulation, machine learning and many others.

2.2.2 QUBITS

As we have mentioned before, while classical computers use bit to represent information, quantum computers use qubits (quantum bits) (Schumacher, 1995). We will explore the hardware regarding qubits in section 4.1.1.

A qubit is a two-state quantum mechanical system. While a classical bit would have to be in one state or the other, a qubit can be in a superposition of both states. (Basics of quantum information, n.d.)

The two possible states in a qubit are represented using the bra-ket notation: $|0\rangle, |1\rangle$. As we explained in section 2.1.1.4, kets are column vectors in vector form, so the representation would be:

$$|0\rangle = \begin{bmatrix}1\\0\end{bmatrix}, \quad |1\rangle = \begin{bmatrix}0\\1\end{bmatrix}$$

These two states are particularly important because they form a basis. In this case the basis is formed by orthogonal vectors where the inner product between the two vectors (as defined in section 2.1.1.4) is 0.

This is a special basis called the computational basis, and it is the most commonly used basis to express quantum states. The computational basis is not only orthogonal but also orthonormal meaning that its states are normalized to have length 1 (length is calculated by taking the square root of the inner product with itself).

The distinctiveness of qubits is that they can exist in a superposition of $|0\rangle\ and\ |1\rangle$. Mathematically, the state of a qubit in superposition is a linear combination of the basis states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Here, α and β are complex numbers representing the probability amplitudes of the qubit so $|\alpha|^2 + |\beta|^2 = 1$ to ensure the total probability adds up to 1.

2.2.2.1 Bloch sphere representation

The Bloch sphere is a very intuitive and useful visual representation of a qubit state. It is a three-dimensional ball with a radius of one. The points within and on the surface of this sphere represent all the possible states of a qubit. The state of a qubit can be represented as a vector that starts at the center of the sphere and points to a location on or within the sphere (Quantiki Blosch Sphere, 2023).

At first, it might seem that there should be four degrees of freedom (α, β are complex numbers with two degrees of freedom each) but the previous constraint of total probability equals 1 removes one degree, making spherical coordinates suitable.

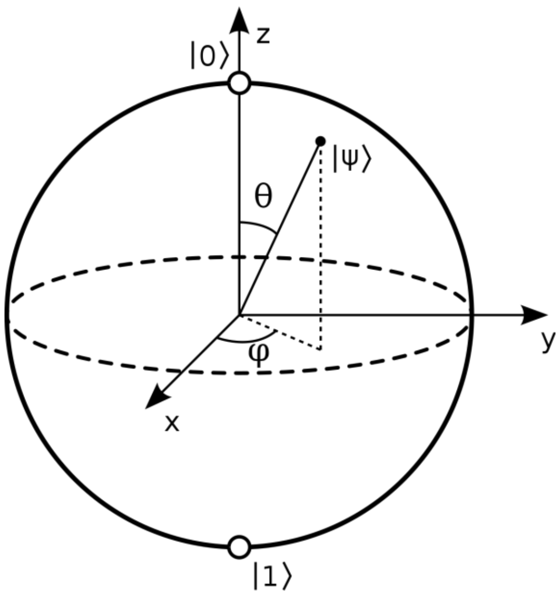


Figure 4 - Bloch Sphere Representation. Source Wikipedia

The north and south poles of the Bloch sphere represent the two basis states:

- A qubit in state $|0\rangle$ is represented as a vector pointing to the north pole
- A qubit in state $|1\rangle$ is represented as a vector pointing to the south pole.

Superposition states, where a qubit can be in a state both $|0\rangle$ and $|1\rangle$ at the same time, are represented as points on the sphere's surface between the poles. The exact location on the surface depends on the relative weights (probability amplitudes) of the states $|0\rangle$ and $|1\rangle$ in the superposition. The latitude of the point gives the relative phases between the states, another important concept in quantum mechanics.

It is important to note that the points inside the sphere represent mixed states, which are statistical mixtures of basis states, not superpositions. These are often the result of partial information or interactions with the environment, also known as quantum decoherence.

2.2.3 Quantum Circuits

Quantum circuits consist of quantum gates and qubits forming the building blocks for quantum computing and information processing.

A quantum circuit begins with a series of wires that signify our qubits. These qubits are arranged from top to bottom, a collective group of qubits is referred to as a quantum register. At the start of a computation, these qubits are initialized to a certain state, often the state $|0\rangle$. In circuit diagrams, It is common to explicitly state the initial states of the qubits.

Quantum operations, or gates, manipulate the qubits in a variety of ways. These gates are represented by different shapes on the wires in the circuit. The shape on a wire shows a gate acting on a particular qubit at a specific moment.

We measure the efficiency of quantum circuits by their depth, which is the minimum number of non-overlapping layers of gates, with fewer layers often being better. Lastly, quantum computation typically concludes with the measurement of one or more qubits. This is how we extract the result of our computation. In circuit diagrams, we represent measurements with a box containing a dial.

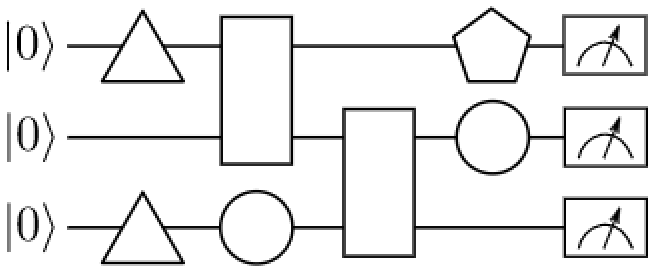


Figure 5 - Representation of quantum circuit with generic gates. Source: Pennylane

2.2.4 Unitary Matrices

We talked about unitary operators before in section 2.1.1.8, which is essentially what unitary matrixes are within the context of quantum mechanics. When we talk about qubits and their state transformations, we represent these transformations with square matrices known as unitary matrices. They maintain the all-important property of normalization, meaning they keep the total probability of all outcomes at 100% when applied to a qubit is state.

An invertible complex square matrix U is unitary if its conjugate transpose U^* is also its inverse. In quantum mechanics the conjugate transpose is referred to as the Hermitian adjoint denoted by (†).

$$\mathrm{U}^*U = U\mathrm{U}^* = U\mathrm{U}^{-1} = I, \qquad \mathrm{U}^\dagger U = U\mathrm{U}^\dagger = I$$

At first it might seem that 9 real numbers (four complex entries, two real numbers each) are needed to specify a 2x2 matrix. But the structure and constraints of unitary matrices allow us to describe them with just three real parameters.

2.2.5 Single Qubit Gates

All of the following gates are detailed in Figure 7 where we can see the corresponding matrix, circuit representation and how it affects qubits.

- **I gate**: Like in classical linear algebra, the Identity gate does not modify the qubit state.
- **X gate**: It is equivalent to the NOT gate, flipping amplitudes of $|0\rangle$ and $|1\rangle$
- **H gate**: The Hadamard gate is used to create a uniform superposition of states $|0\rangle$ and $|1\rangle$. If applied twice the state remains the same HH=I. This gate represents a 90º rotation on the y axis followed by 180º on x axis.

- **Z gate**: Represents a 180º rotation on the Z axis. As seen on the table, when applied |0⟩ it remains unchanged but it converts |1⟩ into -|1⟩
- **S gate**: Also known as the phase gate, it represents a 90º rotation on z axis.
- **T gate**: Known as the π/8 gate, it represents a 45ª rotation on z axis
- **Y gate**: This gate represents a 180º or π/2 rotation on the y axis
- **Rx, Ry, Rz gates**: These gates rotate the qubits is state vector about the appropriate axis by angle φ. X, Y, Z gates are known as the Pauli gates, represented by Pauli matrices. These gates are a generalization of Rx, Ry, Rz.

We can implement any single-qubit operation by using these Rx, Ry, Rz gates. Any two rotations of {Rx, Ry, Rz} are a universal set for single-qubit operations. A gate set is universal if combinations of the gates can be used to approximate any unitary matrix up to arbitrary precision.

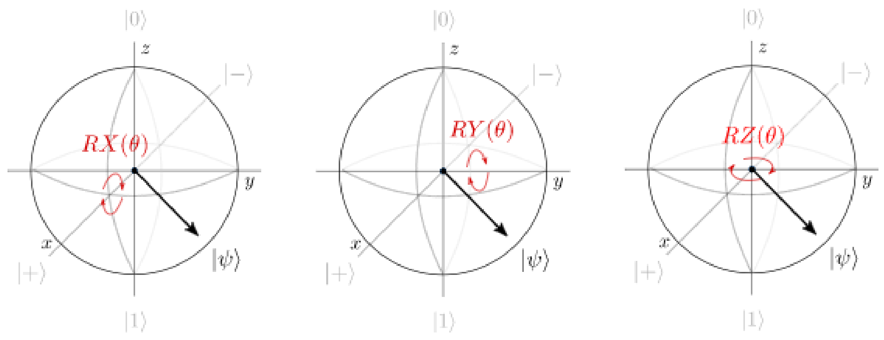
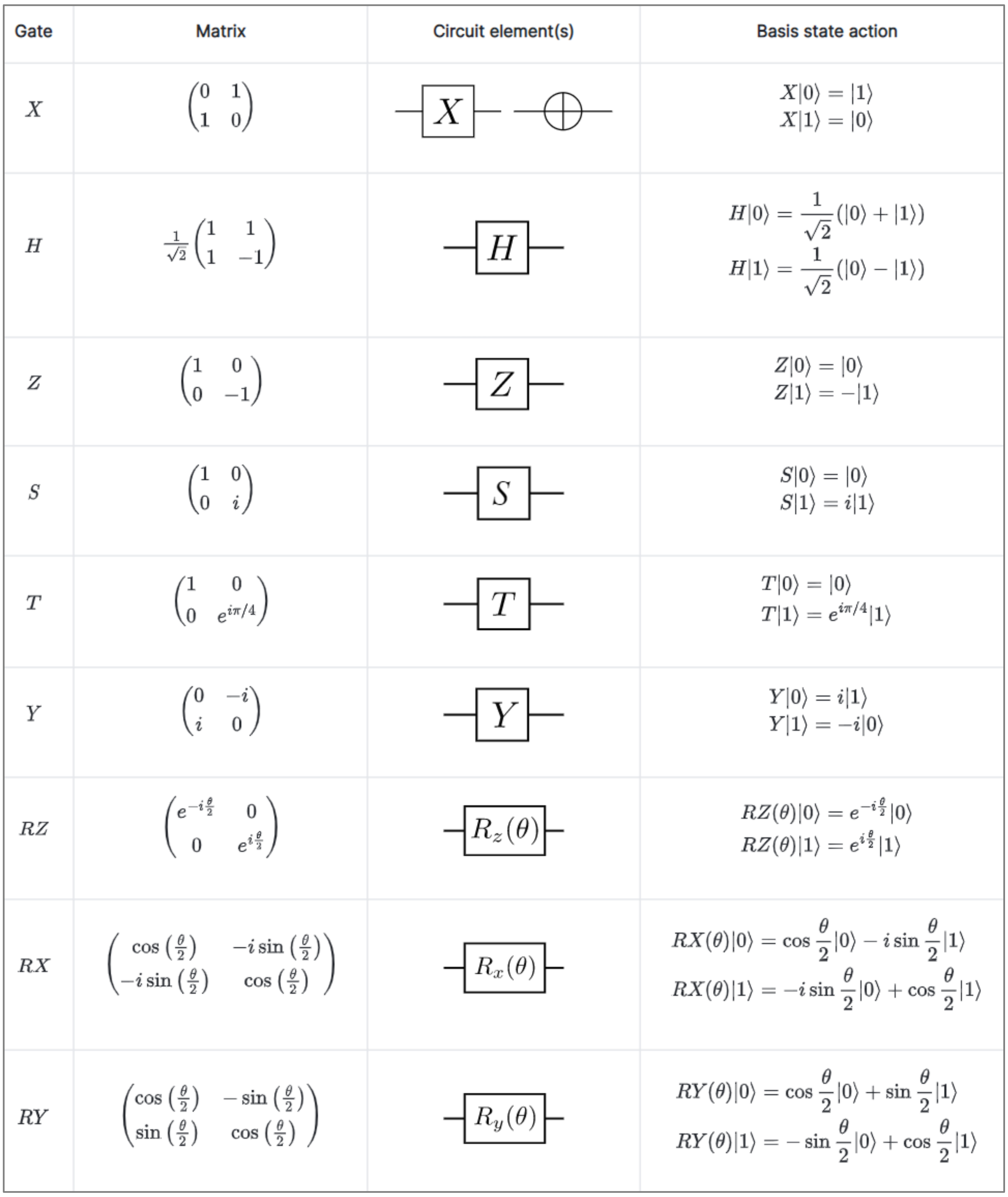


Figure 6 - Rotations Represented on the Bloch Sphere. Source: Pennylane codebook



Gate	Matrix	Circuit element(s)	Basis state action
X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\boxed{X}$ $\oplus$	$X\|0\rangle = \|1\rangle$ $X\|1\rangle = \|0\rangle$
H	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$\boxed{H}$	$H\|0\rangle = \frac{1}{\sqrt{2}}(\|0\rangle + \|1\rangle)$ $H\|1\rangle = \frac{1}{\sqrt{2}}(\|0\rangle - \|1\rangle)$
Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$\boxed{Z}$	$Z\|0\rangle = \|0\rangle$ $Z\|1\rangle = -\|1\rangle$
S	$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$	$\boxed{S}$	$S\|0\rangle = \|0\rangle$ $S\|1\rangle = i\|1\rangle$
T	$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$	$\boxed{T}$	$T\|0\rangle = \|0\rangle$ $T\|1\rangle = e^{i\pi/4}\|1\rangle$
Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\boxed{Y}$	$Y\|0\rangle = i\|1\rangle$ $Y\|1\rangle = -i\|0\rangle$
RZ	$\begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$	$\boxed{R_z(\theta)}$	$RZ(\theta)\|0\rangle = e^{-i\frac{\theta}{2}}\|0\rangle$ $RZ(\theta)\|1\rangle = e^{i\frac{\theta}{2}}\|1\rangle$
RX	$\begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$	$\boxed{R_x(\theta)}$	$RX(\theta)\|0\rangle = \cos\frac{\theta}{2}\|0\rangle - i\sin\frac{\theta}{2}\|1\rangle$ $RX(\theta)\|1\rangle = -i\sin\frac{\theta}{2}\|0\rangle + \cos\frac{\theta}{2}\|1\rangle$
RY	$\begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$	$\boxed{R_y(\theta)}$	$RY(\theta)\|0\rangle = \cos\frac{\theta}{2}\|0\rangle + \sin\frac{\theta}{2}\|1\rangle$ $RY(\theta)\|1\rangle = -\sin\frac{\theta}{2}\|0\rangle + \cos\frac{\theta}{2}\|1\rangle$

Figure 7 - Quantum Single Qubit Gates (codebook.xanadu.ai). Source: Pennylane codebook

2.2.6 Multiple Qubit Gates

In the context of quantum computing, single-qubit states reside in a 2-dimensional vector space, defined by the basis vectors $|0\rangle$ and $|1\rangle$. However, when dealing with multiple qubits, we have to understand how these vector spaces interact, which brings us to the tensor product.

2.2.6.1 Tensor Product

The tensor product allows us to combine Hilbert spaces. Suppose we have a pair of two-dimensional vectors (two single-qubit states). The tensor product is computed:

$$\begin{pmatrix} a \\ b \end{pmatrix} \otimes \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} a\begin{pmatrix} c \\ d \end{pmatrix} \\ b\begin{pmatrix} c \\ d \end{pmatrix} \end{pmatrix} = \begin{pmatrix} ac \\ ad \\ bc \\ bd \end{pmatrix}$$

The tensor product also applies to the unitary operations that act on qubits:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \otimes \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} a\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} & b\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \\ c\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} & d\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \end{pmatrix} = \begin{pmatrix} a\alpha & a\beta & b\alpha & b\beta \\ a\gamma & a\delta & b\gamma & b\delta \\ c\alpha & c\beta & d\alpha & d\beta \\ c\gamma & c\delta & d\gamma & d\delta \end{pmatrix}$$

When considering the bases of multi-qubit systems, things become a bit more interesting. A two-qubit computational basis includes four vectors, denoting every possible pair of two qubits. However, when we move up to a three-qubit computational basis, the number of vectors increases significantly. In fact, for an n-qubit system, the count and size of these vectors underline the necessity for actual quantum computers, as simulating such large systems on classical computers becomes highly impractical.

It is also worth noting that for multi-qubit systems, for example a 4-qubit state $|0\rangle \otimes |1\rangle \otimes |0\rangle \otimes |1\rangle$ can be concisely written as $|0101\rangle$.

2.2.6.1 Entangled States

Entanglement, alongside superposition, serves as one of the defining features of quantum computing. Consider two single-qubit states and let's take their tensor product to form a combined state. By definition, a state is entangled if it cannot be expressed as a tensor product of individual qubit states. On the flip side, if it can be, It is termed as separable. An entangled state must be described entirely, as opposed to specifying individual qubits. Entanglement is not limited to two qubits either, it extends to larger systems too.

Entanglement in quantum computing is achieved through various quantum gates, including the Hadamard (H) gate and the CNOT gate. These gates can be used together to create entangled states, such as the Bell state, which is a simple and commonly used method for entangling two qubits.

2.2.6.2 Gates

The following are the main multiple qubit gates, also represented in Figure 8:

- **CNOT**: Also know as controlled-NOT acts on a pair of qubits. It performs an action on one qubit (the target) based on the state of another (the control). The control qubit remains unchanged, while a NOT gate is applied to the target qubit if the control qubit is in the $|1\rangle$. If the control qubit is in superposition and the target qubit is either in $|0\rangle$ or $|1\rangle$, then the CNOT gate creates entanglement between both qubits.

Earlier, we saw universal gate sets for single-qubit operations. For multi-qubit operations, we just need one more gate, the CNOT. The set {CNOT, Ry, Rz} would be a universal gate set for multi-qubit computation.

- **CZ**: The Controlled-Z gate is similar to the CNOT but instead of applying a NOT gate (or X gate), it applies a Z gate if the control qubit is on state $|1\rangle$.
- **SWAP**: This gate exchanges the state of two qubits.
- **TOFFOLI**: Similar to the CNOT gate but having two control and one target qubit. It now applies the NOT gate to the target qubit only if both control qubits are on state $|1\rangle$.

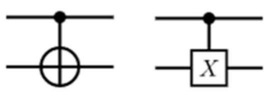
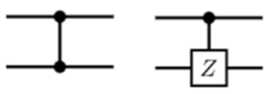
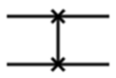
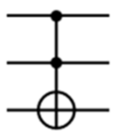
Gate	Matrix	Circuit element(s)	Basis state action
$CNOT$	$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}$		$CNOT\|00\rangle = \|00\rangle$ $CNOT\|01\rangle = \|01\rangle$ $CNOT\|10\rangle = \|11\rangle$ $CNOT\|11\rangle = \|10\rangle$
CZ	$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{pmatrix}$		$CZ\|00\rangle = \|00\rangle$ $CZ\|01\rangle = \|01\rangle$ $CZ\|10\rangle = \|10\rangle$ $CZ\|11\rangle = -\|11\rangle$
$SWAP$	$\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{pmatrix}$		$SWAP\|00\rangle = \|00\rangle$ $SWAP\|01\rangle = \|10\rangle$ $SWAP\|10\rangle = \|01\rangle$ $SWAP\|11\rangle = \|11\rangle$
TOF	$\begin{pmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{pmatrix}$		$TOF\|000\rangle = \|000\rangle$ $TOF\|001\rangle = \|001\rangle$ $\vdots = \vdots$ $TOF\|101\rangle = \|101\rangle$ $TOF\|110\rangle = \|111\rangle$ $TOF\|111\rangle = \|110\rangle$

Figure 8 - Multi-qubit gates. Source: codebook.xanadu.ai

2.2.7 MEASUREMENTS

In quantum computing, measurement is not a straightforward action, but a probabilistic one. When measuring, we cannot see the superposition of a qubit, but instead observe it in a definite state, either $|0\rangle$ or $|1\rangle$. The probabilities of these outcomes are encoded in the amplitudes of the state vector. After measurement, the qubit remains in the observed state, thereby limiting our immediate knowledge about its initial state. Due to this probabilistic nature, we would need multiple measurements to support the result obtained.

As explained earlier if we measure a qubit in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we will observe it in state $|0\rangle$ with probability $|\alpha|^2$ and state $|1\rangle$ with probability $|\beta|^2$.

A more formal way of expressing the measurement outcome probabilities is using the inner product. The probability that we observe the qubit in the state $|\varphi\rangle$ when we measure it with respect to a basis that includes $|\varphi\rangle$ is equal to

$$\Pr(\varphi) = |\langle\varphi|\psi\rangle|^2$$

This is called projective measurement, which asks how much each basis vector contributes to a given state, thus determining the probabilities of potential outcomes. (Albornoz, et al., 2021)

The choice of basis for measurement in quantum computing is very important. The selection of the basis can significantly impact the measurement results. Most commonly, measurements are made in the computational basis, consisting of $|0\rangle$ and $|1\rangle$ states, or the Hadamard basis. It is crucial to remember that a change in the basis, also known as a basis rotation, can allow different measurements.

While obtaining measurement outcome probabilities gives us useful information about a qubit is state, we are usually interested in other measurable quantities that correspond to something physical like energy. These are called observables and where explained in detail in section 2.1.1.6. To get an idea of the value of an observable we measure its expectation value, essentially the weighted average of what we would see over many experiments.

2.3 MACHINE LEARNING

In a nutshell, machine learning is the science of deriving insights, patterns and predictions from data. It is about constructing and training mathematical models that can infer or predict from given data points. Data would be numeric values harvested from measurements, surveys, or interactions with technology, anything that can be processed by a computer.

We usually consider each individual piece of data as a tuple of 'm' real numbers, data points, represented as x ∈ Rm. A collection D = $\{x_1, \dots, x_n\}$ of 'n' such data points form a dataset.

Machine learning leverages datasets to draw insights about specific scenarios. The fundamental premise is that data obtained from a particular context or domain is not random but follows certain patterns that may be too complex for a human to analyze and discover. Essentially, a machine learning solution assumes the existence of an underlying mechanism that can account for variations in the data. This is mathematically expressed by the function $f \in$ F with parameters $\theta \in \Theta$, modeling characteristics of a dataset, with the noise term ϵ_j accounting for possible inaccuracies. F represents the space of possible functions that can be used to model the dataset, and Θ represents the space of possible parameter values that can be assigned to the function. For a labeled dataset, the equation would be:

$$y_j = f(x_j|\theta) + \epsilon_j$$

The key paradigms in machine learning are supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning is about training a model using labeled data to generate the right output y for a given input x. Semi-supervised learning is a variant where a small set of labeled data helps to assign labels to unlabeled data.

In contrast, unsupervised learning deals with unlabeled data, intending to discover hidden structures within the dataset. These could be clusters, correlations, or relational dependencies.

Reinforcement learning, a type of supervised learning, receives feedback instead of labels. This feedback helps tune models for decision-making scenarios where a current output could influence the next input, typically framed as a partially observable Markov decision process (POMDP).

2.3.1 THE MACHINE LEARNING PROCESS

The application of machine learning in practical scenarios involves multiple steps:

Step 1: Data Collection: This is the starting point where we collect and possibly annotate examples relevant to the problem at hand. It is crucial that the data collected should represent all possible situations the system could encounter when It is deployed. In an ideal scenario, the data would be balanced, showing all possible use cases in approximately equal proportions, but this might not be feasible all the time.

Step 2: Data Pre-processing: Here, the gathered data is converted into a format that is easier to process, and any faulty data points are removed. For instance, we might convert text data into a suitable numeric form or smooth out any noise in sensor data.

Step 3: Data Splitting and Model Selection: Next, we divide the data into two separate sets: one for training the model and another for testing it. The type of model to be used will depend on the data and the use case, requiring some domain expertise. As we dive deeper into model selection, there are many more specific choices to make, like deciding the number of states in a Markov chain or the depth of a decision tree. However, It is becoming common to use flexible models, such as deep neural networks, which are applicable to a wide range of situations.

Step 4: Training Phase: Here, we adjust the parameters of the chosen model to best match the training data. This is done automatically via learning algorithms, often using optimization techniques. For this kind of mathematical learning, we need to measure how well the model and the current parameters match the training data. We do this through an error or loss function, which we seek to minimize during the training process. It is worth noting that the choice of loss function should be made considering the data and model.

Step 5: Testing Phase: The model, once trained, is evaluated on the test data set. The performance measure used for this evaluation will again depend on the specific application. In a classification task, for instance, we often check the accuracy, which is the percentage of correct class predictions, while for regression we check the error distance from predicted to actual points. Remember that the training and testing data sets must be independent, to ensure we are assessing the model's ability to generalize to new data.

Step 6: Deployment: After the model has been thoroughly tested and shown to perform reliably and accurately, it can be launched in the real world.

Overall, the described process is most common in situations where systems need to make data-driven predictions, suggest actions or decisions based on data, or classify new observations or measurements. The capabilities of a machine learning system largely depend on the nature of the available training data and the chosen model.

2.4 QUANTUM ALGORITHMS

A quantum algorithm is a procedural set of directions that directs the actions of a quantum computer. The promise of quantum algorithms lies in their potential to solve certain computational problems faster than classical algorithms, usually due to quantum superposition, allowing a quantum computer to examine many possible solutions simultaneously, and quantum interference, guiding the quantum computer towards the correct answer.

Navigating this landscape, one inevitably wonders: Are there problems that can be solved efficiently on a quantum computer? To illustrate this point, Deutsch, in his exploration of a universal quantum computer, demonstrated quantum parallelism by devising a method to compute the parity $f(0) \oplus f(1)$ of a one-bit function.

Building upon this groundwork, Deutsch and Jozsa advanced a quantum algorithm that exceeded the efficiency of classical counterparts for a specific problem (Deutsch & Jozsa, 1992). This significant step motivated the quest for algorithms capable of tackling “real” problems.

In this chapter, we will examine the Deutsch-Jozsa algorithm, which serves as a starting point of our understanding of quantum efficiency. Following that, we will explore other notable quantum algorithms, such as Shor’s algorithm, celebrated for its potential to crack cryptographic codes, and Grover’s algorithm, known for its prowess in searching unsorted databases.

2.4.1 FAMOUS QUANTUM ALGORITHMS EXPLAINED

2.4.1.1 Deutsch-Jozsa Algorithm

The problem itself, called Deutsch’s problem (Deutsch & Jozsa, 1992), revolves around determining whether a given binary function is constant (having the same output for all inputs) or balanced (having an equal number of zeros and ones as outputs).

In the classical computing paradigm, we would need to evaluate the function for half the inputs plus one to conclusively establish whether the function is balanced or constant. For a function with a large number of inputs, this process can quickly become computationally expensive ($O(n) = 2^{n-1} + 1$).

In contrast, the Deutsch-Jozsa algorithm solves this problem using a quantum computer with only a single query, regardless of the size of the input. This performance leap is due to superposition, allowing to examine possible input states simultaneously.

To accomplish this, the algorithm prepares a quantum system in a superposition of all possible input states and then applies the quantum version of the function (called an oracle). This operation transforms the quantum states in a way that depends on the function values. Finally, a measurement strategy reveals whether the function is constant or balanced without the need to inspect each state individually.

The Deutsch-Jozsa algorithm, while not practical for real-world applications due to the artificial nature of the problem, represented a milestone in quantum computing. It provides a clear demonstration of how quantum computers can outperform classical computers under the right circumstances.

2.4.1.2 Grover's Search Algorithm

Grover's algorithm, (Grover, 1996) is a quantum algorithm for unstructured search problems. Simply put, it is a way to speed up searching in an unsorted database.

Imagine we have a huge list of items, and only one item is marked as the "target." In classical computing, on average, we would need to check half of the items in the list before finding the target. In the worst case scenario, we might even have to check the entire list. This is what is called a linear search, a classical algorithm.

This is where Grover's algorithm steps in, being able to find the target item in roughly square root of the number of items in the list, significantly faster than any classical algorithm. This is a quadratically faster search, which in the world of computation, is a substantial speed boost, especially for large lists.

Now, let's break down how Grover's algorithm works:

- Initialization: All quantum states are prepared in a superposition (equal probability of being any item in the list).
- Oracle Application: An oracle function is applied to the quantum states, which marks the target item by flipping its phase (in other words, multiplying it by -1).
- Amplitude Amplification (Diffusion Operator): This step amplifies the probability of the marked target state and decreases the probabilities of the non-marked states. This process is repeated several times.
- Measurement: After approximately the square root of the total number of items iterations, a measurement is performed, revealing the target item with high probability.

So, Grover's algorithm leverages superposition and interference, to significantly speed up the process of searching in an unsorted list or database. It is important to note, however, that while Grover's algorithm accelerates unstructured searches, it does not apply to structured problems like factoring, where other quantum algorithms, like Shor's Algorithm outperform it.

2.4.1.3 Shor's Algorithm

This algorithm is in essence a factorization algorithm for finding the prime factors of an integer (Shor, 1994). To understand the importance of this algorithm we are going to first contextualize its implications in cryptography.

Public key cryptography is a widely used method of encryption that involves a pair of encrypt-decrypt keys, ideal for nowadays multi-user environments, being RSA the most popular algorithm. The security of RSA is based on the difficulty to find the factors of large integers that are the product of two large prime numbers.

The key parameter of RSA is $N = p \cdot q$, where p and q are big prime numbers. If we could find p and q, we could break RSA scheme, come up with the private key and be able to decrypt ciphertexts which are encrypted messages.

Peter Shor developed in 1994 an algorithm that is able to find these factors in polynomial time when run in a quantum computer, instead of the usual exponential time it takes when run on a classical computer. However, when run in a classical computer it would take even longer thank just factorizing N in other ways. For RSA-2048 it would take a classical computer around 300 trillion years to break the key, while a perfect Quantum Computer can do it in 10 seconds. However, this perfect computer does not exist yet, since practical quantum computers do not have the necessary number of error free qubits. Runtime complexities are

- Best classical factorization algorithm on classical computer:
    - $O\left[e^{\left(1.9\,(LogN)^{1/3}(loglogN)^{2/3}\right)}\right]$ "Number Field Sieve" (wolfram.com, 2015)
- Shor's algorithm on perfect quantum computer
    - $O[(logN)^2\,(loglogN)(logloglogN)] \approx O[(logN)^3]$

Although we are far from achieving this perfect quantum computer, there are papers that describe how a quantum computer with noisy qubits could break RSA (Gidney & Ekerå, 2021).As for now the largest number factored successfully using Shor's original algorithm in a quantum computer is 21 (Martín-López, et al., 2012) (35 failed), which is not very hopeful.

Shor's algorithm can be split up into three main parts:

1) Converting the factoring problem into a period finding problem using the modular exponentiation function. Dividing our number by a guest number $'a'$ and computing the remainder. For good guesses of $'a'$ the function is periodic as we increase the power of $'a'$.

2) Using the Quantum Fourier transform to find the period of the modular exponentiation function. The key is to send a quantum superposition of numbers to avoid exponential time and use the QFT as a computational interferometer so the outputs interfere and we only get the answer we want

3) The period is used to efficiently compute the factors of the original number. We will first obtain two numbers that are co-factors of N, $(b \cdot p_{factor})(c \cdot q_{factor})$. Then by using the Euclidean Greatest Common Divisor algorithm we can obtain p and q original factors efficiently.

2.5 QUANTUM MACHINE LEARNING

Modern machine learning is extremely successful but also a very resource intensive endeavor. Very powerful hardware, typically located in data centers, normally accessed using cloud platforms is currently needed in order to train complex state-of-the-art machine learning models. Access to numerous CPUs and GPUs that forms clusters are usually offered to users at different prices by large cloud companies such as Google Cloud, AWS, or Microsoft Azure.

Since this trend is likely to continue, it is no surprise that an increasing number of machine learning researchers are starting to look at the potential benefits of quantum computing. We are going to focus on machine learning quantum-classical hybrid algorithms with classical data, however different options include:

- Quantum algorithms for classical data.

- Quantum – classical hybrid algorithms classical data.
- Classical algorithms for quantum data.
- Quantum algorithms for quantum data.

Due to currently low numbers of qubits and high error rates there is a gap between the theoretical designs of quantum algorithms and their practical applications.

2.5.1.1 Data encoding

One of the challenges for many quantum algorithms is the ability to effectively translate classical data into quantum states. Methods for such translations typically require a number of gates proportional to $O(2^n)$ for a precise representation of standard data into an n-qubit state. As a result, scaling concerns must be considered as they can easily increase quantum algorithm's complexity and render any quantum speedups useless. (Bauckhage, et al., 2022)

The issue lies at the heart of state preparation. Consider an n-qubit state:

$$|\psi\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_j}|j$$

where |j⟩ symbolizes the computational basis state that corresponds with the binary encoding of integer j, and p_j is an arbitrary, fixed, classical probability mass function over $\{0,1\}^n$. Several types of data can be encoded into this representation:

- Intensity values of pixels in an $2^{n/2} \times 2^{n/2}$ image
- Term frequencies over some text corpus with a vocabulary of size 2^n
- Probability of observing a traffic jam on a specific set of streets

In that representation, a greyscale image in 4K resolution (3840 x 2160 pixel) would occupy $[log_2(3840 \times 2160)] = 23$ qubits.

However, preparing the state |ψ⟩ demands resources proportional to the dimension of the underlying qubit register's Hilbert space. If n is large, we cannot prepare |ψ⟩. Even if a quantum algorithm promises exponential improvement over the best classical algorithm, any advantage is neutralized due to the complexity of preparing the initial state. Grover and Rudolph propose a method to generate the quantum state efficiently, but their assumptions are often unfulfilled.

Similar problems arise when we want to interpret the result of a quantum computation. An exponential number of results must be measured if the full Hilbert space representation of the state encodes our desired outcome. Therefore, any quantum speedup could be nullified when all probability amplitudes of the final state need to be estimated.

Finally, several QML assumes state preparations to be given and the existence of Quantum Random Access Memories (QRAMs). However, current technologies do not allow for the creation of such devices. The quantum mechanical principle, the no-cloning theorem, states that it is impossible to create independent identical copies of arbitrary quantum states, and this could limit the repeated access to quantum states for processing in a QRAM. Even if a quantum algorithm can produce a quantum state representation for a solution much faster than classically possible, the effort for measuring and reading the resulting state into classical memory could still be substantial enough to cancel out any advantage. (Bauckhage, et al., 2022)

In conclusion, before asserting the superiority of quantum machine learning algorithms, it is crucial to consider the efforts for state preparation or measurements and compare these against pre and post-processing efforts of efficient classical algorithms.

2.5.1.2 Hybrid algorithms

Hybrid quantum-classical machine learning algorithms play a crucial role in bridging the gap between classical and quantum computing realms. These approaches leverage the strengths of both computational paradigms to solve complex problems.

In essence, hybrid quantum-classical machine learning algorithms entail the utilization of quantum systems to execute certain computationally expensive parts of a machine learning task, while classical systems take on the rest of the procedure, such as data pre-processing, post-processing, and sometimes even steering the quantum component of the algorithm. This composition provides a powerful approach to overcome the restrictions of current Noisy Intermediate-Scale Quantum (NISQ) devices and harness their potential.

A prime example of such hybrid models is the Variational Quantum algorithm. This algorithm operates by defining a parameterized quantum circuit, the parameters of which are updated iteratively by a classical optimization routine to minimize a certain cost function. The optimization loop is repeated until an optimal solution is reached or a stopping condition is met. The advantage lies in the possibility of employing the quantum system's ability to traverse the complex solution space efficiently and the classical system's capacity for reliable optimization and processing.

Another example is the quantum support vector machine, where the kernel of the support vector machine is computed using a quantum device. The quantum device offers an efficient way of computing the kernel for higher-dimensional data, after which a classical machine learning algorithm can utilize the calculated kernel for subsequent steps, such as classification.

Hybrid quantum-classical machine learning algorithms show great promise (Bonet-Monroig, et al., 2023), pointing towards a future where classical and quantum systems co-exist and work together to solve complex machine learning problems.

2.5.2 Variational Quantum Classifier

The Variational Quantum Classifier (VQC) is a hybrid quantum-classical machine learning algorithm (Schuld, Bocharov, Svore, & Wiebe, 2018). It is one of the algorithms that is getting better performance with current quantum hardware

(Bonet-Monroig, et al., 2023), characterized by their limited number of qubits and susceptibility to noise. The VQC is designed to be robust against noise in the inputs and parameters, therefore, suitable for implementation on near-term devices.

2.5.2.1 Loading the Data into the Quantum System: Feature Map

The first step in the VQC algorithm is to encode the classical data into a quantum system, necessary because quantum computers operate on quantum states, which are fundamentally different from classical bits. This is achieved through a process known as a feature map. The feature map transforms the classical data into a quantum state, effectively mapping an input vector from an N-dimensional real space to a 2^n dimensional amplitude vector that describes the initial quantum state. The total number of qubits used to represent the features corresponds to the letter n . This process also referred to as state preparation, creates an encoded feature vector which is now a ket vector in the Hilbert space of a n qubit system.

Depending on our encoding method, a different number of qubits would be needed. One common encoding scheme is amplitude encoding, which is highly efficient and allows the encoding of N classical features into $log_2(N)$. Therefore, a quantum state of n qubits can represent 2^n complex amplitudes, which can be used to encode 2^n real features. However, it is important to note that while amplitude encoding is efficient, it also requires more complex quantum operations to implement, which might not be feasible on near-term quantum devices (Weigold, Barzen, Leymann, & Salm, 2020).

In the case of angle embedding, another popular method, each feature in the dataset corresponds to one qubit in the quantum system. This is because each feature controls the rotation of a qubit around a certain axis. This is a simpler and more direct encoding scheme compared to amplitude encoding, but it also requires more qubits for a given number of features.

2.5.2.2 Variational Circuit

Once the data is encoded into the quantum system, the next step is to apply a variational circuit to the prepared quantum state. The variational circuit is a sequence of parameterized quantum gates represented by a unitary operation U_θ where θ is a set of adjustable parameters. The purpose of the variational circuit is to explore the space of quantum states and find the one that minimizes a given cost function. This is achieved by adjusting the parameters θ of the unitary operation.

2.5.2.3 Classical Optimization Loop

The classical optimization loop is an iterative process that aims to find the optimal parameters θ that minimize the cost function. This is typically achieved using classical optimization algorithms such as gradient descent. The optimization loop involves a feedback mechanism between the quantum and classical components of the algorithm: the quantum component generates a candidate solution (i.e., a quantum state), the classical component evaluates the cost function for this solution and updates the parameters θ based on the result, and the updated parameters are then fed back into the quantum component for the next iteration.

On the following diagram we see the different parts of the VQC:

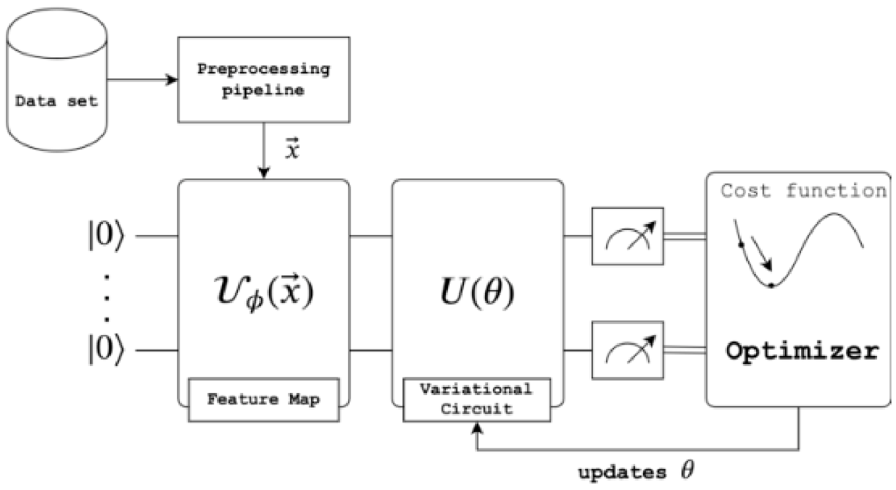


Figure 9 - Variational Classifier Block Diagram. Source Q-munity: Building a VQC

2.5.2.4 Measurement and Assigning of Label

After the optimal parameters θ have been found, the final quantum state is measured. In the case of the VQC, the measurement is based on the probability of measuring the first qubit in state 1. This probability is estimated by repeating the entire algorithm multiple times and taking samples from the resulting Bernoulli distribution. The final prediction of the model is then obtained by thresholding the continuous output of the model, which is the sum of the probability and a learnable bias term. If the result is greater than 0.5, the output is 1; otherwise, it is 0.

2.5.2.5 Conclusion

The Variational Quantum Classifier is a promising approach for leveraging the computational capabilities of quantum computers in combination with classical ones for machine learning tasks. By combining quantum state preparation, variational circuits, classical optimization, and quantum measurement, the VQC provides a robust and flexible framework for data classification.

The performance and effectiveness of the VQC depend on the choice of feature map, the structure of the variational circuit, and the optimization strategy. Therefore, further research and experimentation are needed to further advance this quantum-classical hybrid algorithm to its full potential.

2.5.3 QUANVOLUTION – QUANTUM CONVOLUTION

Quantum convolution is an exciting field of research in quantum computing, inspired by classical convolution principles, an essential component in machine learning. (Cong, Choi, & Lukin, 2019) (Henderson, Shakya, Pradhan, & Cook, 2020)

2.5.3.1 Classic Convolution in CNNs

The convolution operation is critical in image processing and is a fundamental part of Convolutional Neural Networks (CNNs). These networks, which represent a specific architecture of deep learning models, have demonstrated their proficiency in image, video, and audio machine learning tasks.

In a CNN, the input data, typically an image, is subdivided into smaller, localized chunks. The processing of these regions occurs through the use of a filter, also called kernel, which is a matrix of weights applied consistently to each chunk of the image.

The application of the filter generates specific results for each chunk, which are typically linked to different channels of a single output pixel. The concept of a channel refers to the individual components of a pixel, such as the red, green, and blue channels in a color image.

The cumulative output pixels from this process construct a new, image-like structure. Convolutional layers are usually followed by pooling layers that reduce dimensionality. Then, they are further processed through additional layers, if required, each of which employs the same convolution process with different filters.

A key feature of CNNs is their capability to learn spatial hierarchies and patterns in the data adaptively. As a result, the weights in these filters are not static but get updated and optimized during the learning process.

2.5.3.2 Quantum Convolution

Quantum convolution extends the idea of classical convolution to quantum variational (Henderson, Shakya, Pradhan, & Cook, 2020):

1. **Input Image Embedding**: The first step involves embedding a small region of the input image into a quantum circuit. This image region is typically a grid of pixels, with each pixel's color intensity value corresponding to a particular state of a qubit. It is important to note that qubits are initialized in the ground state, the lowest energy state, so it is a uniform starting point that allows superposition and entanglement.

2. **Parameterized Rotations**: Once the qubits have been initialized, parameterized rotations are applied to them. These rotations, change the state of the qubits based on specific parameters such as the intensity values of the pixels in the image region.
3. **Quantum Computation**: Following the application of the parameterized rotations, a quantum computation is executed. This computation is represented by a unitary transformation U, which can be considered the "quantum equivalent" of a filter in classical convolution. The transformation could be generated by a variational quantum circuit or a simpler random circuit.
4. **Measurement**: After the quantum computation, the quantum system undergoes measurement. In quantum mechanics, the measurement process forces the qubit into one of its possible states, yielding classical information. Each measurement provides a specific outcome for each qubit, and after several measurements, we have frequencies for classical expectation values, essentially probabilities associated with the states of the qubits.
5. **Output Mapping**: The classical expectation values derived from the measurement step are then mapped to different channels of a single output pixel. This process mirrors the operation in a classical convolution layer, where the result of the kernel operation on an image region is mapped to an output pixel.
6. **Iteration**: Steps 1-5 are then repeated for each region of the input image, allowing the entire image to be scanned. The output is a multi-channel image similar to the output of a traditional CNN layer.
7. **Further Processing**: The output image from the quantum convolution layer can then be passed on to further layers for additional processing. These could be quantum layers, using the same process as above, or classical layers, utilizing traditional CNN techniques.

The following diagram Figure 10 represents a ‘quanvolutional’ layer

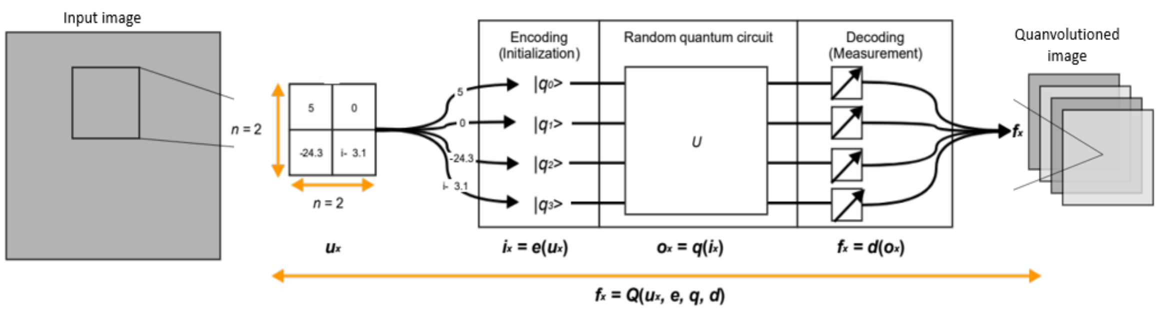


Figure 10 - Quantum Convolutional Layer. Source: Made from (Henderson, Shakya, Pradhan, & Cook, 2020)

2.5.3.3 Quanvolution filters and motivation

Unlike classical convolutional filters that apply element-wise matrix multiplication, quanvolutional filters use quantum circuits to transform input data. Quantum circuits are able to generate sophisticated kernels capable of tasks beyond classical methods. This unique feature of quantum convolution holds promising potential to revolutionize image processing and related fields.

Strengths include its hybrid nature, no QRAM requirements, and potential resiliency to consistent error models, making it ideal for the NISQ computing era. Limitations involve determining optimal interfacing with classical data, a potentially large number of quantum circuit executions, and a definitive demonstration of quantum advantage. Despite its potential benefits, proving the usefulness of the QNN approach over classical methods is essential for its wider acceptance.

Section 3. TECHNOLOGIES INVOLVED

The following section details the specific libraries and quantum computing frameworks used during the project.

3.1 IDE AND PROGRAMMING LANGUAGE: PYTHON

This project's primary Integrated Development Environment (IDE) was Visual Studio Code with Jupyter Notebooks. Due to its suitability for experimental and exploratory projects, the notebook format was chosen rather than a regular scripting interface.

As a programming language, the clear choice was python. Due to its wide use in machine learning and data science, developers and companies are now creating high-level quantum libraries for python, which made it ideal for our use case.

To ensure smooth execution and prevent potential conflicts among libraries, the project was run on virtual environment with just the required packages installed.

3.2 QUANTUM PROGRAMMING LIBRARY: PENNYLANE

There are several Python libraries designed to build quantum algorithms and interact with quantum computer. The following are the most popular:

- Qiskit: An open-source library developed by IBM, Qiskit provides tools for creating and manipulating quantum programs. (Qiskit, 2023)
- Cirq: An open-source library developed by Google, Cirq is specifically tailored for running on Noisy Intermediate Scale Quantum (NISQ) circuits. (Cirq, 2023)

- PennyLane: An open-source library developed by Xanadu Quantum Technologies that integrates quantum computers into existing machine learning frameworks, providing tools for quantum machine Learning. (Bergholm, et al., 2018)

From these libraries, we chose Pennylane due to its focus on machine learning.

PennyLane allows programmers to interface with quantum computing hardware and simulators across multiple platforms. Its distinctive feature is its use of differentiable quantum programming, which facilitates the optimization of quantum and hybrid quantum-classical computations. This feature sets PennyLane apart as it makes it possible to perform computations on a quantum device and compute their gradients, providing key information for optimizing quantum circuits and algorithms.

In essence, PennyLane functions as a quantum-computing-compatible extension to popular machine learning libraries such as TensorFlow or PyTorch. It essentially "quantum-enables" these libraries, allowing for the development of quantum machine learning models alongside classical models.

A core design principle of PennyLane is its multi-platform nature. It allows quantum circuits to be run on various types of quantum simulators or hardware devices with minimal adjustments, simplifying the task of optimizing communication with devices, compiling circuits to suit the backend, and choosing the best gradient strategies.

While it comes with built-in simulator devices, PennyLane is designed to work seamlessly with external quantum computing platforms such as IBM’s Qiskit, Google’s Cirq, or Rigetti’s Forest. This flexibility allows programmers to experiment with different quantum hardware and simulators while using a unified interface.

In an academic sense, PennyLane represents an important advance in the field of quantum computing, bridging the gap between quantum and classical computation and offering new possibilities for research and application in quantum machine learning. (Bergholm, et al., 2018)

3.2.1 Pennylane-Qiskit Plugin

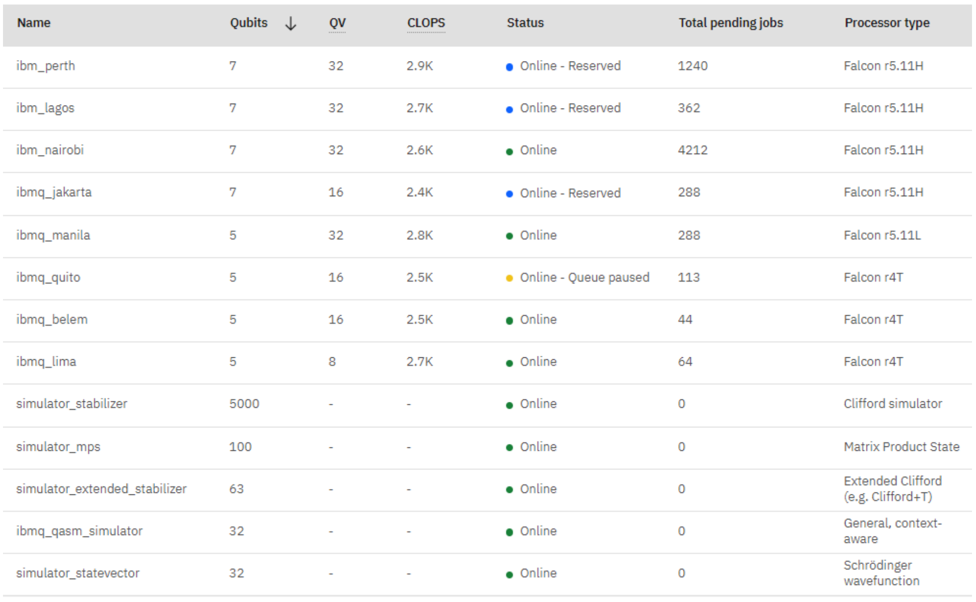
This plugin integrates PennyLane capabilities with IBM Qiskit, which is IBM framework for quantum computing. This allows us to compile and run our pennylane defined quantum circuits on IBM's quantum simulators and quantum computers.

3.3 Quantum Simulation & Computing Platform: IBMQ

IBM Quantum is a platform developed by IBM that provides access to quantum computers, simulators, and a comprehensive set of tools for developing quantum applications. The platform provides a cloud-based quantum computing service that allows users to run quantum programs on actual quantum computers and powerful simulators located at IBM's facilities. (IBM Quantum, 2023)

Qiskit is IBM's open-source quantum software development kit (SDK) which allows users to create quantum algorithms and circuits. As described previously, we are using pennylane-qiskit plugin to integrate both of them and be able to code using the pennylane library while running the circuits on IBM's hardware,

IBM has several quantum simulators and computers of varying qubit counts. Some are free to use for academic use while higher qubit systems are only available with premium enterprise-grade access. However, due to the popularity of IBM Quantum as the main platform to access quantum computing, the wait times and queues are very long, even for the least powerful of their quantum computers. The following shows the available quantum computers first, followed by the simulators, each with their respective qubits, quantum volume and other features.



Name	Qubits	QV	CLOPS	Status	Total pending jobs	Processor type
ibm_perth	7	32	2.9K	● Online - Reserved	1240	Falcon r5.11H
ibm_lagos	7	32	2.7K	● Online - Reserved	362	Falcon r5.11H
ibm_nairobi	7	32	2.6K	● Online	4212	Falcon r5.11H
ibmq_jakarta	7	16	2.4K	● Online - Reserved	288	Falcon r5.11H
ibmq_manila	5	32	2.8K	● Online	288	Falcon r5.11L
ibmq_quito	5	16	2.5K	● Online - Queue paused	113	Falcon r4T
ibmq_belem	5	16	2.5K	● Online	44	Falcon r4T
ibmq_lima	5	8	2.7K	● Online	64	Falcon r4T
simulator_stabilizer	5000	-	-	● Online	0	Clifford simulator
simulator_mps	100	-	-	● Online	0	Matrix Product State
simulator_extended_stabilizer	63	-	-	● Online	0	Extended Clifford (e.g. Clifford+T)
ibmq_qasm_simulator	32	-	-	● Online	0	General, context-aware
simulator_statevector	32	-	-	● Online	0	Schrödinger wavefunction

Figure 11 - Free to use available quantum computers & simulators. Source: Screenshot IBMQ

As we see, IBM's most powerful free to use quantum computer: ibm_perth has only 7 qubits, in contrast with the 127 qubit Eagle Quantum computer available at enterprise level. We also see the high number of jobs being launched by people across the world, at a specific point in time, which causes our algorithms to take hours or days to run.

3.3.1 IBM Quantum Jobs

Jobs are executions of the quantum circuit on IBM's platform. Key factors range from the structure of the model and the characteristics of the dataset to the limitations of current quantum hardware and software.

The IBM quantum computer operates under a queue-based system. Each job submitted is placed in a queue and must wait its turn to be executed. The waiting time can vary significantly, but it typically ranges from an hour to several hours. Given the large number of jobs required to train the variational classifier, the cumulative waiting time can be exceedingly long.

Section 4. State of the Art

4.1 Quantum Computing

Quantum computing, a domain once reserved to the world of theoretical physics, has rapidly transformed into an exciting field of practical research and development. This shift has been primarily fueled by advancements in quantum hardware technologies. However, the road to building reliable quantum computers is filled with unique challenges.

A critical term in contemporary discussions around quantum computing hardware is NISQ, which stands for Noisy Intermediate-Scale Quantum. This refers to quantum devices that are available today and in the near future. These devices typically have a few to a few hundred qubits and are subject to noise, meaning that errors can and do occur during their operation. Despite this, NISQ devices can still perform some tasks that surpass the capabilities of classical computers, thus proving their potential.

There's a vast spectrum of technologies employed for creating qubits - the basic units of quantum information, and It is still unclear which one, if any, will triumph in the long run. The main challenge lies in maintaining quantum coherence or keeping a qubit in its quantum state long enough to perform calculations.

Temperature is another substantial challenge in quantum computing. Quantum processors require very low temperatures, often close to absolute zero, to function. This is because any thermal energy in the environment can cause qubits to change state or lose coherence. Cooling systems for quantum computers are complex and contribute significantly to the size and energy cost of these machines.

Despite these challenges, the pace of progress in quantum computing hardware is promising. Researchers around the globe are continually innovating and developing techniques to increase qubit counts, improve qubit quality, and reduce or correct errors.

4.1.1 Quantum Hardware

Many companies are currently developing quantum hardware and there are several different methods to implement qubits. The following are the main ones:

- **Gate-Based Superconducting Processors**: These quantum computers rely on superconducting circuits operating at very low temperatures. The property of superconductors is that their electrical resistance drops to zero at certain temperatures (close to absolute zero), allowing electric currents to persist indefinitely without a power source. Superconducting qubits are created within these circuits using capacitors and Josephson junctions. They help to isolate two energy levels to form a qubit - the '0' and '1' states. By employing microwave pulses, the state of the superconducting qubit can be manipulated, leading to superposition of states and entanglement between qubits.
- **Gate-Based Ion Trap Processors:** Here, qubits are implemented using the electronic states of ions, which are charged atoms. These ions are held above a trap through electromagnetic fields, and quantum gates are applied using lasers that alter the ions' electronic states. A significant advantage is that these qubits use atoms from nature, eliminating the need for synthetic manufacturing.
- **Photonic Processors**: These quantum computers use special light sources that emit specific light pulses. The qubit counterparts correspond to modes of continuous operators such as position or momentum.
- **Neutral Atom Processors**: Similar to trapped ion technology, neutral atom processors use light instead of electromagnetic forces to trap the qubits. The atoms are neutral, i.e., not charged, and the circuits can work at room temperatures, making them more practical.

- **Rydberg Atom Processors**: These involve excited atoms, known as Rydberg atoms, with electrons that are positioned further from the nucleus. These atoms have peculiar properties, such as exaggerated responses to electric and magnetic fields, and longer lifespans. As qubits, they offer strong and tunable atomic interactions.
- **Quantum Annealers**: This hardware places the system's qubits in an absolute energy minimum through a physical process, then alters the system's configuration to reflect the problem to be solved. Although they can have a larger number of qubits than gate-based systems, their use is limited to specific cases. D-wave, one of the most experienced quantum computing companies, developed a qubit quantum annealing system.

It is worth noting that there is no consensus on the "best" qubit technology yet. However, given its promising potential, it is in gate-based superconducting processors where most major tech corporations such as Google, IBM, Intel, and Rigetti are invested.

IBM was one of the first major corporations to bet on quantum computing and start developing their own hardware. Since we chose IBM quantum platform to run our algorithms, we will focus on their recent advancements. Below we see IBM's roadmap when it comes to quantum computing. On November 2022, IBM unveiled its new 433 qubit 'Osprey' processor triplicating its predecessor Eagle which had 127 qubits.

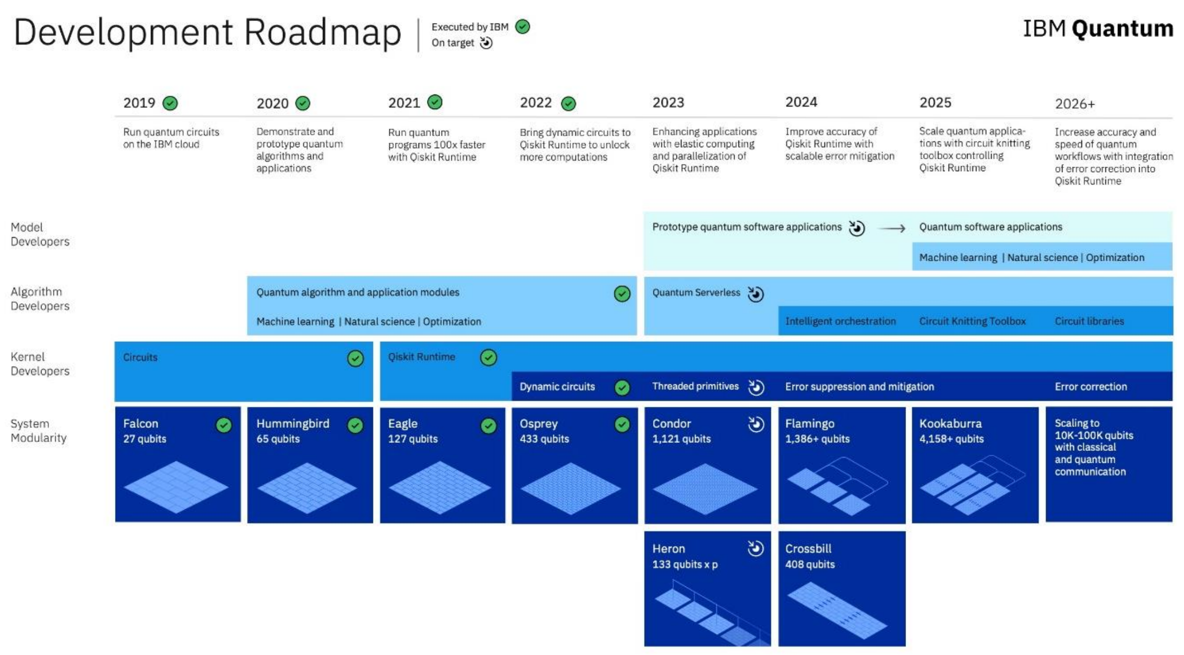

Development Roadmap
Executed by IBM
On target
IBM Quantum
2019
Run quantum circuits on the IBM cloud
2020
Demonstrate and prototype quantum algorithms and applications
2021
Run quantum programs 100x faster with Qiskit Runtime
2022
Bring dynamic circuits to Qiskit Runtime to unlock more computations
2023
Enhancing applications with elastic computing and parallelization of Qiskit Runtime
2024
Improve accuracy of Qiskit Runtime with scalable error mitigation
2025
Scale quantum applications with circuit knitting toolbox controlling Qiskit Runtime
2026+
Increase accuracy and speed of quantum workflows with integration of error correction into Qiskit Runtime
Model Developers
Prototype quantum software applications
Quantum software applications
Machine learning | Natural science | Optimization
Algorithm Developers
Quantum algorithm and application modules
Machine learning | Natural science | Optimization
Quantum Serverless
Intelligent orchestration
Circuit Knitting Toolbox
Circuit libraries
Kernel Developers
Circuits
Qiskit Runtime
Dynamic circuits
Threaded primitives
Error suppression and mitigation
Error correction
System Modularity
Falcon 27 qubits
Hummingbird 65 qubits
Eagle 127 qubits
Osprey 433 qubits
Condor 1,121 qubits
Flamingo 1,386+ qubits
Kookaburra 4,158+ qubits
Scaling to 10K-100K qubits with classical and quantum communication
Heron 133 qubits x p
Crossbill 408 qubits

Figure 12 - IBM Quantum Computing Roadmap. Source: ibm.com

It is important to note that the qubit number is not the only metric of quantum computer performance.

4.1.2 QUANTUM VOLUME

While the initial stages of quantum computing's development were often benchmarked by the total number of qubits, it was soon evident that just the qubit count was not a comprehensive measure of a quantum computer's performance. In response, IBM introduced Quantum Volume (Baldwin & Mayer, 2022), a more holistic performance metric. Quantum volume considers various factors, including:

- Number of qubits: The basic building blocks of quantum information.
- Quality of qubits: How long qubits can maintain their state (coherence time).
- Gate error rates: The accuracy of operations (gates) performed on qubits.
- Connectivity between qubits: How well qubits can interact with each other.
- Software and circuit compiler efficiency: The ability to translate complex algorithms into executable instructions.

By assessing these elements, quantum volume provides a more rounded measure of a quantum computer's overall power and potential problem-solving ability. As a single-number metric, it aids in tracking the advancement and performance of quantum computing hardware more effectively than a simple qubit count.

On June 30, 2023, a significant milestone was achieved in the field of quantum computing by Quantinuum's System Model H1-1, powered by Honeywell (Quantinuum news, 2023). This system demonstrated a quantum volume of 524,288, marking a substantial leap in performance improvement, being currently the highest reported, impressively 1000x higher than the next best.

They use trapped-ion computing which allow for flexibility in algorithmic design and shows a clear pathway to scaling. The H1 system operates with a 20-qubit universal quantum computer and achieves two-qubit gate fidelities of 99.87% (Quantinuum Hardware, 2023), mirroring the best fidelities observed in leading two-qubit experimental setups.

4.1.3 QUANTUM SIMULATORS

Quantum simulators are powerful tools that play a crucial role in quantum computing and its applications, including machine learning. In essence, these simulators mimic the behavior of a quantum system using classical computation. This allows researchers to study and explore quantum systems without needing access to a full-scale quantum computer, which is presently a resource with limited availability.

The use of quantum simulators is crucial for exploring new applications of quantum computing, refining computational techniques, and experimenting with error correction methods. Additionally, they offer an invaluable platform for education and training, fostering the development of a new generation of quantum programmers and researchers.

In the realm of quantum simulators, IBM has established itself as a leading player. The company provides some of the most powerful free-to-access quantum simulators, enabling a wider audience to delve into the world of quantum computing and its potential applications, including machine learning.

4.1.4 THE FUTURE

Despite the promise, quantum computing still grapples with significant technological challenges. Currently, qubits' physical implementation struggles with stability, decoherence, error tolerance, and scalability. These issues necessitate multiple physical qubits for error correction to perform useful computations. Thus, there is a gap between the theory and practice of quantum algorithm design, with many theoretically valid algorithms not yet feasible due to limited numbers of practical logical qubits.

Furthermore, today's quantum computing is still rooted at the bit level, meaning it lacks the abstract data structures we're used to in higher-level programming. . Even basic programming patterns like variable assignments face challenges due to quantum principles like the no-cloning and no-broadcast theorems, which prevent creating independent identical copies of arbitrary quantum states.

As we look forward to the future of quantum computing, one of the crucial aspects that emerges is the advancement in hardware. The development of reliable quantum processors capable of mitigating noise and decoherence effects poses a formidable challenge. However, researchers are ceaselessly striving to improve error correction techniques and design better quantum processors (Quantinuum news, 2023).

Quantum computing also offers immense potential in the realm of optimization problems and machine learning. Complex problems, such as those found in logistics and supply chain management, are often difficult to solve using classical computers due to their inherent limitations.

The promise of quantum computing extends to chemistry and materials science as well. By enabling the simulation of complex chemical reactions, it could fast-track the creation of new materials and drugs, pushing advancements in medicine and science. Additionally, the implications for cryptography are significant, where it could crack some of the current encryption methods, but also help develop new, stronger encryption resistant to quantum attacks.

In short, despite the challenges, the future of quantum computing is promising,

4.2 QUANTUM MACHINE LEARNING

There has recently been a "quantum machine learning mini-revolution," and the number of scientific reports on quantum circuits for certain general problems in machine learning has grown considerably over the past decades.

However, many of these papers explore quantum algorithms designed for perfect quantum computers with many logical qubits instead of real physical qubits. Some of the most promising algorithms require encoding and decoding classical data into quantum data, which is not yet clear how fast it can be done. In addition, these authors hypothesize using Quantum Memory or QRAM, which has not yet been achieved, and it is questionable whether it is possible in its true sense.

Therefore, in the last years, the most promising results are from variational quantum computing algorithms or hybrid quantum-classical methods. The classical computer is used to adjust the parameters of quantum gates in a parametrized quantum circuit to optimize its performance. The quantum computer is then used to perform the actual computation, and the measurement results are fed back to the classical computer in order to further refine the parameters.

Overall, there have been many recent breakthroughs in Quantum Enhanced Machine Learning by effectively using quantum algorithms for clustering, SVM, Boosting and Neural networks or, as they are more commonly called: parametrized quantum circuits. Particularly variational quantum circuits have been extensively tested and implemented over the last few years using different architectures and libraries (Bauckhage, et al., 2022). On the other hand, quantum convolution is a more recent concept that has not yet been experimented with as widely as other quantum algorithms. Some researches (Henderson, Shakya, Pradhan, & Cook, 2020) have explored their potential utility and implemented it with varying success.

4.2.1 Hybrid Quantum-Classical Machine Learning

On this subsection, we discuss some of the most recent literature regarding hybrid quantum-classical machine learning publications.

A key breakthrough discussed by (Abbas, et al., 2021) is the Quantum Neural Networks (QNN). QNNs are shown to train faster on noisy quantum devices. They introduced a new concept, 'effective dimension', which measures a model's capacity to predict unseen data accurately, crucial for the machine learning model's generalization capability.

In another innovative approach, (Chen & Yoo, 2021) combined quantum and classical machine learning methods. Their model, called federated QML, tackles the rising privacy concerns while utilizing the limited quantum hardware efficiently. Essentially, this approach allows local quantum devices to work as clients, handling both classical and quantum data.

Quantum algorithms for image classification are also being developed. Notably, (Dang, Jiang, Hu, Ji, & Zhang, 2018) presented the Quantum K-Nearest Neighbors (QKNN) algorithm, a quantum version of a popular classical machine learning model. Another important contribution (Adhikary, Dangwal, & Bhowmik, 2020) introduced a quantum classifier that encodes N-dimensional data using a single quantum system.

(Havlíček, et al., 2019) proposed two quantum techniques in supervised machine learning, the Variational Quantum Classifier (VQC) and Quantum Kernel Estimator (QKE). These methods utilize the quantum state space for feature extraction and data classification.

These are just a few of the most promising publications in the area over the last few years. Due to increasing interest and widely available usage of quantum computers through cloud platforms, more researchers are entering and experimenting on this relatively new field.

Section 5. SCOPE OF THE PROJECT

5.1 JUSTIFICATION

In the current era of data-driven decision-making, Machine Learning (ML) has emerged as the foundation of numerous industry domains. Simultaneously, Quantum Computing (QC) is making advancements with its potential to resolve issues beyond the scope of classical computing. A unique fusion of these two technologies—Quantum Machine Learning (QML)—promises to unlock new paradigms in data processing and knowledge discovery.

The motivation for this project springs from the understanding that QML is an new but rapidly evolving field. Novel models and algorithms are being introduced frequently, reflecting the active research interest and the potential for groundbreaking advancements. Within the significant and constant progress, there are possibilities for new adaptations of machine learning algorithms for NISQ hardware. This project aims to bridge this gap, exploring the practical implementation variational classifiers and quantum convolution.

Notably, leading tech corporations are showing a keen interest in the quantum space. Tech giants like IBM and Google are not only developing advanced quantum hardware but also spearheading the integration of QC into areas like machine learning and artificial intelligence by developing their own libraries.

Moreover, at Accenture, there is already an increasing demand from clients to understand the impact of quantum computation on their operations and many are eager to explore potential projects that leverage quantum computing. Thus, there is a clear market need for practical exploration of QC applications in ML. This project will serve as a valuable guide to demonstrate the practical benefits and challenges of implementing QML algorithms. It will also help in preparing businesses for the quantum future by providing them with a roadmap for leveraging QC in their machine learning applications.

In summary, this project is an exploration of the intersection of QC and ML. It aims to provide a practical perspective on its implementation, and inform how quantum computation can revolutionize traditional machine learning techniques.

5.2 Objectives

- Clearly explain the theory behind quantum computing and its applications in certain aspects of machine learning.
- Program and test different algorithms.
- Get decent accuracy and show how models could be scalable in the future with higher qubit quantum computers.
- The objective is not surpassing classical computers in speed or accuracy but experimenting with executing algorithms in real quantum computers and show synergies between QC and ML.

5.3 Methodology

For this project, a combination of theoretical and practical methods will be used to explore the potential of quantum computing to enhance machine learning. The theoretical approach will involve research into the theoretical foundations of quantum computing and its potential applications to machine learning algorithms.

The practical approach will involve utilizing IBM quantum computers and libraries such as Qiskit and PennyLane to implement quantum computing algorithms and analyze their performance compared to classical machine learning algorithms. Different data sets will be used to evaluate the performance of the algorithms. Experiments will be conducted on the IBM quantum platform and the performance of the algorithms will be measured. The results of these experiments will be analyzed and discussed to draw conclusions on the effectiveness and applicability of quantum computing for machine learning tasks.

1. **Research**: Research existing quantum algorithms and existing quantum computing libraries to identify the most suitable algorithms for the machine learning task.
2. **Theoretical explanation** of Quantum Physics, the math behind Quantum Computing, Quantum Gates, and Circuits, how Quantum Computing can be applied to Machine Learning algorithms and how these algorithms actually work.
3. **Design**: Code using python's libraries like Qiskit, PennyLane and TensorFlow, different quantum algorithms and variations.
4. **Implementation**: Implement the quantum algorithm on IBM quantum computers by accessing their machines remotely
5. **Evaluation**: Evaluate the results on metrics like accuracy, speed, and scalability and refine the quantum algorithms if needed

Section 6. DEVELOPMENT

6.1 VARIATIONAL CLASSIFIER

This section covers the software implementation of variational quantum classifiers with Pennylane, pennylane-qiskit and Keras python libraries. As mentioned in section 2.5.2, variational quantum circuits are also referred to as quantum neural networks in some publications (Farhi & Neven, 2018).

We start by defining a seed so that the algorithms that we use throughout the development of this section behave in a deterministic way.

```
def set_seeds(seed=10):
    np.random.seed(seed)
    tf.random.set_seed(seed)
    random.seed(seed)
    tf.keras.utils.set_random_seed(seed)
    tf.config.experimental.enable_op_determinism()
```

6.1.1 A HYBRID APPROACH

As described in section 2.5.2, the variational classifier quantum circuit is formed by three main components:

1. Feature map $\mathcal{U}_{\Phi(\vec{x})}$: responsible for encoding classical data $\vec{x}$ into quantum states that will be processed in the quantum circuit by the algorithm.
2. Variational layer $U(\vec{\theta})$ This part of the circuit is parameterized by $\vec{\theta}$. These are the parameters adjusted during the training process.
3. Measurement: The final step of the circuit involves measuring the qubits, which produces classical information

The VQC is a hybrid algorithm that combines this quantum circuit with classical optimization as represented in Figure 13.

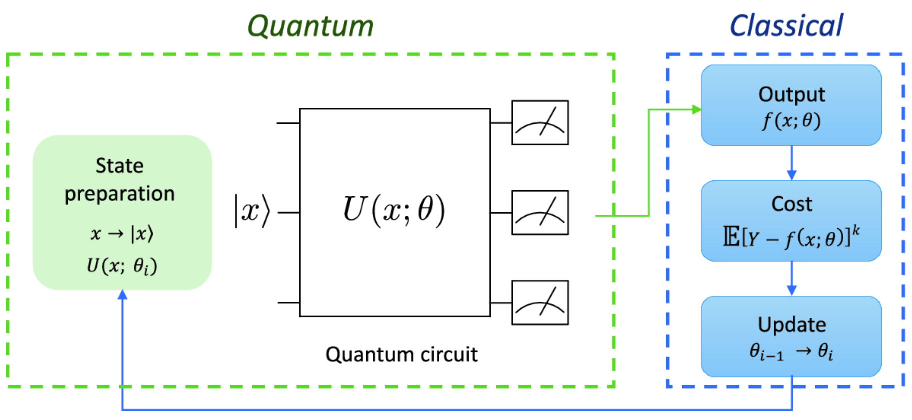


Figure 13 - Scheme of a hybrid quantum-classical algorithm (VQC).

Source: (Macaluso, Clissa, Lodi, & Sartori, 2020)

The key features of our model architecture are:

- The forward propagation is quantum in nature.
- With measurement, we retrieve classical information and build the loss/cost function classically.
- The parameters $\vec{\theta}$ are updated to optimize the cost function classically.
- In this sense, the backpropagation is classical.

6.1.2 DATA

For the experiments, we have chosen the Iris dataset (Fisher, 1936), widely used as a starting point for classification tasks. The dataset contains 150 rows of data, divided evenly among three distinct iris flower species: setosa, versicolor, and virginica. Each flower has four numerical features: sepal length, sepal width, petal length, and petal width, all measured in centimeters.

In the following plot we see an overview of the distribution and correlation of the 4 different features for the three different flower species.

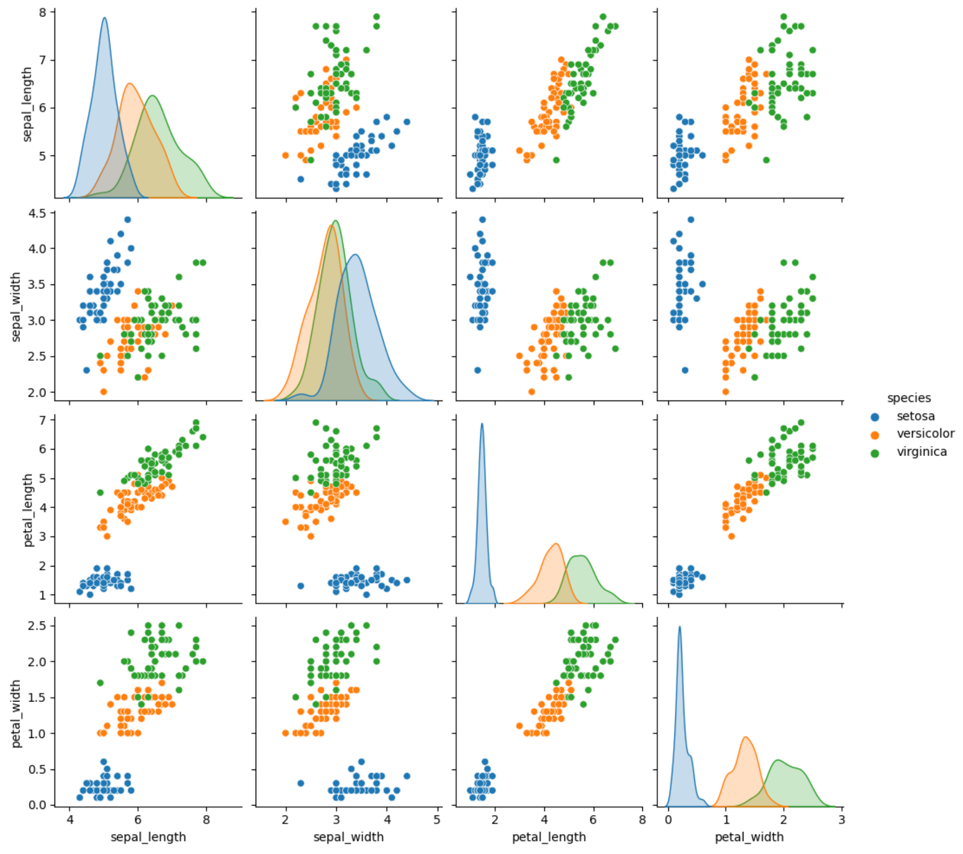


Figure 14 - Iris data distributions pairplot - Source: Self-made with seaborn

We start by separating the data into a training, validation, and test set with proportions of 60%, 20%, 20% where stratification is applied to ensure proportional class distribution. The target classes are represented numerically using categorical codes to convert them into numeric labels.

For the neural network, the target classes are further transformed into binary vectors which is important for the neural network to perform multi-class classification.

6.1.3 ARCHITECTURE: VARIATIONAL QUANTUM LAYER

We start by defining an execution backend which describes what hardware, either a quantum simulator or a quantum computer, will be used for the different computations.

```
dev = qml.device("default.qubit", wires=n_qubits)
dev_qiskit = qml.device("qiskit.aer", wires=n_qubits, seed_simulator=42)
```

We define two different backends: a default backend for circuit execution and a backend with visualization capabilities through the qiskit framework to represent quantum circuits.

6.1.3.1 Feature map

```
@qml.qnode(dev_qiskit)
def feature_map(inputs):
    qml.AngleEmbedding(inputs, wires=range(n_qubits))
    return [qml.expval(qml.PauliZ(i)) for i in range(n_qubits)]
dev_qiskit._circuit.draw("mpl")
```

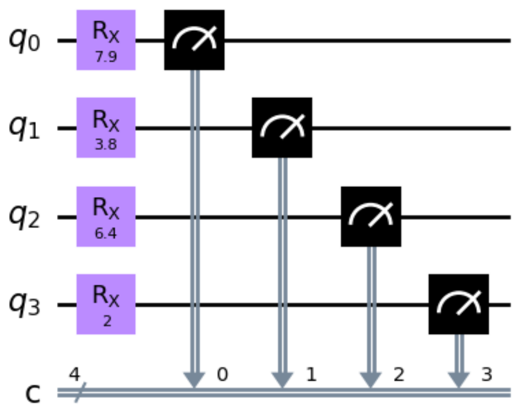


Figure 15 - Feature map qiskit compiled quantum circuit. Source: run on IBMQ

Angle Embedding operation is used to encode the inputs into the quantum circuit by applying rotation gates with angles specified by the inputs. The function then returns the expectation values of Pauli-Z measurements for each wire (qubit) in the circuit. The feature map is then visualized:

On Figure 15 we see IBM's low-level compilation of the circuit we defined using high-level penny lane code. We can see four wires (qubits), each representing one of 4 features in the dataset, followed by the Rx rotation gates and the measurements.

6.1.3.2 Variational Layer

We now build a Variational Quantum layer which involves parameterized quantum operations that can be adjusted to optimize a target function for classification.

```
@qml.qnode(dev_qiskit)
def var_layer(parameters):
    qml.StronglyEntanglingLayers(parameters, wires=range(n_qubits))
    return [qml.expval(qml.PauliZ(i)) for i in range(n_qubits)]
```

This function takes parameters as an argument, which represents the parameters of the variational layer. The qml.StronglyEntanglingLayers operation is used to apply a layer of entangling gates with the specified parameters on all the wires (qubits) of the circuit. The function then returns the expectation values of Pauli-Z measurements for each wire.

We are now going to see how 1 and 2 variational layers are represented:

```
shape = qml.StronglyEntanglingLayers.shape(n_layers=1, n_wires=4)
parameters = np.random.random(size=shape)
dev_qiskit._circuit.draw("mpl")
```

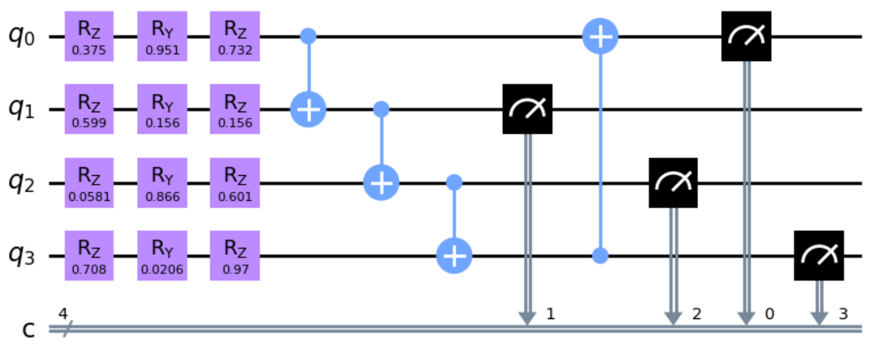


Figure 16 - 1 Entangling layer qiskit compiled quantum circuit. Source: run on IBMQ

```
shape = qml.StronglyEntanglingLayers.shape(n_layers=2, n_wires=4)
parameters = np.random.random(size=shape)
dev_qiskit._circuit.draw("mpl")
```

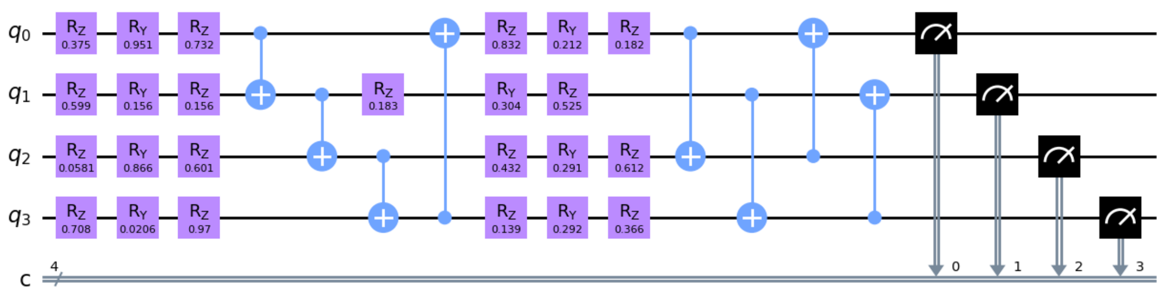


Figure 17 - 2 Entangling layer qiskit compiled quantum circuit. Source: run on IBMQ

We generate initial parameters, to feed into the Variational Quantum layer, with the appropriate shape determined by the number of layers in the strongly entangling layer and the number of qubits involved.

On the visualizations we can see the Rz rotation gates and the CNOT gates resulting from our variational circuit compiled. We see that the only difference when using 2 strongly entangling layers instead of 1 is the duplication of gates before the final measurements.

6.1.4 MODEL

```
set_seeds(10)
n_qubits = X_train.shape[1]
n_classes = y_train.nunique()
n_var_layers = 2
weight_shapes = {"parameters" : (n_var_layers, n_qubits, 3)}

# ================ Quantum layer definition ================
@qml.qnode(dev)
def vqc_layer(inputs, parameters):
    qml.AngleEmbedding(inputs, wires=range(n_qubits))
    qml.StronglyEntanglingLayers(parameters, wires=range(n_qubits))
    return [qml.expval(qml.PauliZ(i)) for i in range(n_classes)]

# ============= Keras model with quantum layer =============
model = tf.keras.models.Sequential()
model.add(qml.qnn.KerasLayer(vqc_layer, weight_shapes,output_dim=n_classes))
model.add(tf.keras.layers.Activation("softmax"))
```

```
# ========== Classical optimization & compilation ==========
opt = tf.keras.optimizers.Adam(learning_rate=0.05)
model.compile(loss="categorical_crossentropy",optimizer=opt,metrics=["accuracy"])
```

Firstly, the random seed is set for reproducibility, and the number of qubits, classes, and variational layers are defined according to the dataset. A specific weight shape for the quantum node is set, reflecting the structure of the strongly entangling layers in the variational circuit.

Then the Variational Quantum layer is defined according to the explanation in section 6.1.3.2. The Pauli-Z expectation values are returned for each class, providing quantum state representations for each possible classification outcome.

Following the definition of the quantum function, a Sequential model is initiated using TensorFlow's Keras. The quantum node is converted into a Keras layer establishing the quantum layer of the model. An activation layer using the softmax function is added, ensuring the output of the model can be interpreted as probabilities for each class classification.

In the final section, the model is compiled for classical optimization. An Adam optimizer with a specified learning rate is chosen as the optimization algorithm, and the loss function is set to be the categorical cross-entropy, which is suitable for multi-class classification tasks. The metric used to evaluate the model's performance is accuracy.

The following image represents the created model:

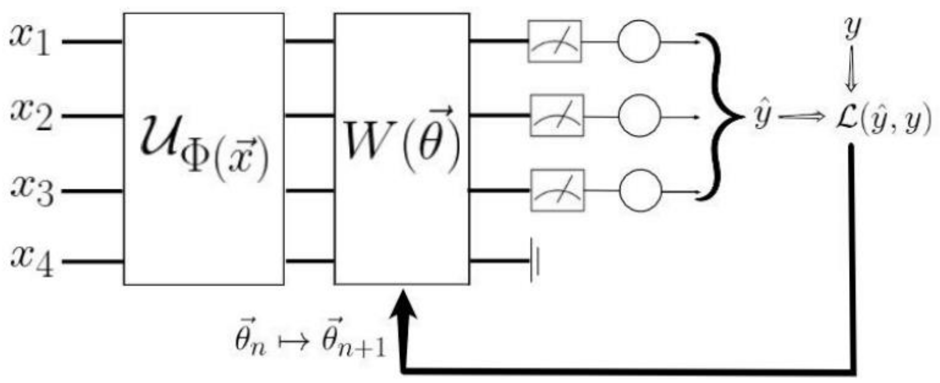

x_1
x_2
x_3
x_4
$\mathcal{U}_{\Phi(\vec{x})}$
$W(\vec{\theta})$
y
$\hat{y} \longrightarrow \mathcal{L}(\hat{y}, y)$
$\vec{\theta}_n \mapsto \vec{\theta}_{n+1}$

Figure 18 - Variational Classifier model scheme. Source: made using github draw.io

6.1.5 Model variation 1: One Classical Layer

We now add a classical dense layer after the quantum layer. This makes the model hybrid also in the forward propagation.

```
model = tf.keras.models.Sequential()
model.add(qml.qnn.KerasLayer(vqc_layer, weight_shapes, output_dim=n_qubits))
model.add(tf.keras.layers.Dense(n_classes, activation="softmax"))
```

This last line of code adds a fully-connected classical layer of neurons with softmax activation function included. The circuit is represented on Figure 19:

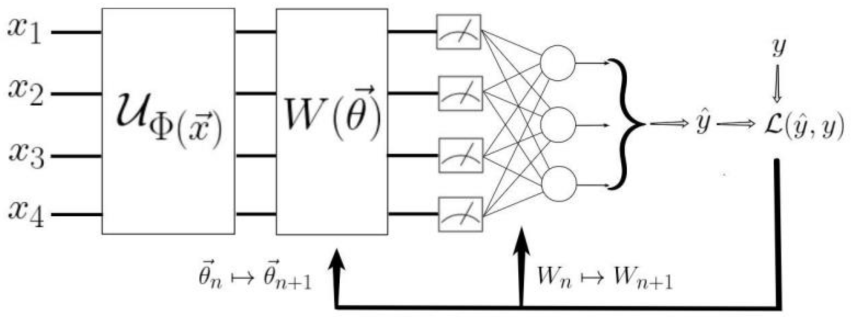

x_1
x_2
x_3
x_4
$\mathcal{U}_{\Phi(\vec{x})}$
$W(\vec{\theta})$
y
$\hat{y} \longrightarrow \mathcal{L}(\hat{y}, y)$
$\vec{\theta}_n \mapsto \vec{\theta}_{n+1}$
$W_n \mapsto W_{n+1}$

Figure 19 - Variational Classifier model variation 1. Source: made using github draw.io

6.1.6 Model variation 2: Two classical layers

In this case, we add another classical fully connected layer before the quantum layer, resulting in a hybrid model with two classical layers in the forward propagation.

```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(n_qubits,activation="relu",input_dim=X_train.shape[1]))
model.add(qml.qnn.KerasLayer(vqc_layer, weight_shapes, output_dim=n_qubits))
model.add(tf.keras.layers.Dense(n_classes, activation="softmax"))
```

As seen on the second line, we now start the sequential model with a fully connected layer of 4 neurons (one for each feature) and a ReLU activation function that applies a non-linear transformation where any input value below zero is set to zero and any value above zero remains unchanged, enabling the model to learn complex patterns during training.

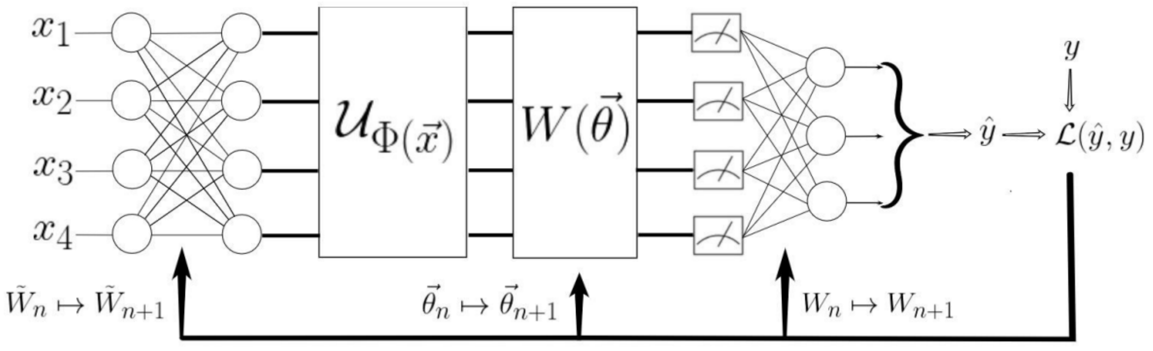

x_1
x_2
x_3
x_4
$\mathcal{U}_{\Phi(\vec{x})}$
$W(\vec{\theta})$
y
$\hat{y}$
$\mathcal{L}(\hat{y}, y)$
$\tilde{W}_n \mapsto \tilde{W}_{n+1}$
$\vec{\theta}_n \mapsto \vec{\theta}_{n+1}$
$W_n \mapsto W_{n+1}$

Figure 20 - Variational Classifier model variation 2. Source: made using github draw.io

The objective of these other models is not to demonstrate a performance improvement when using more classical layers but to show the easy and seamless integration of quantum and classical layers on variational circuits.

6.2 QUANVOLUTIONAL NEURAL NETWORKS

This section covers the software implementation of a quanvolutional neural network with Pennylane (Henderson, Shakya, Pradhan, & Cook, 2020), and pennylane-qiskit python libraries. We are going to first apply quantum convolution to a set of images and then use this new set of 'quanvolved' images to train a classical neural network. An overview architecture of our model is represented in the following Figure 21

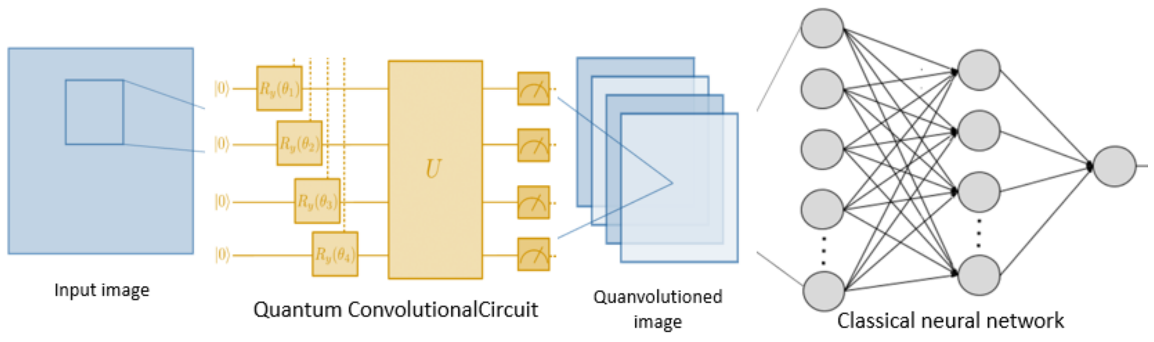


Figure 21 - QCNN: Quantum convolutional neural network full architecture. Source: self-made

Quanvolution and quantum convolution is used interchangeably. As described in section 2.5.3.2 quantum convolutions are just parameterized unitary rotations like those of regular variational circuit, performed on neighboring pairs of qubits. In classical CNNS convolutional layers are followed by pooling layers, in QCNNS this dimensionality reduction occurs when by measuring a subset of the qubits.

6.2.1 DATA

Convolution is generally used when dealing with image related tasks so we chose one of the most widely used datasets for testing image classification: the Fashion-MNIST dataset (Xiao, Rasul, & Vollgraf, 2017). The Fashion-MNIST dataset comprises of 70,000 grayscale, Zalando's article images, each of 28x28 pixels in size. These images represent ten categories of clothing items: T-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. It is worth noting that, while the Fashion-MNIST dataset provides a more challenging problem space than the original MNIST, it is still relatively simple for image classification.

After loading the dataset, we reduce the training set to 200 images and the test set to 60 due to the computational complexity and time constraints associated with quantum computations. Next, we normalize the images in the dataset to have pixel values between 0 and 1, which facilitates the learning process and improves the performance of the model. Finally, we reshape the images by adding an extra dimension to accommodate the requirement of convolution channels in the quantum convolution operation.

6.2.2 Quantum Convolution

6.2.2.1 Quanvolution circuit

We start by defining the quantum circuit:

```
dev_qiskit = qml.device("qiskit.aer", wires=4, seed_simulator=10)
rand_params = np.random.uniform(high=2 * np.pi, size=(n_layers, 4))

@qml.qnode(dev_qiskit, interface="autograd")
def circuit(phi):
        for j in range(4):
        qml.RY(np.pi * phi[j], wires=j)
        RandomLayers(rand_params, wires=list(range(4)))
        return [qml.expval(qml.PauliZ(j)) for j in range(4)]
```

The quantum circuit is designed to handle a 2x2 pixel input from the images, which is why there are four qubits (one for each pixel in a 2x2 pixel square). The circuit function takes a 4-element input (phi), where each element represents a pixel. These values are used to perform an RY rotation on each of the 4 qubits. Here, RY is a type of rotation gate in the Bloch sphere's y-axis used to encode the pixel values into quantum states.

Following the encoding layer, the random quantum circuit is applied. This circuit adds a certain degree of randomness in the operations applied to the qubits, which can help in extracting complex patterns from the data.

The quantum circuit concludes with a measurement stage, where the expectation values of each qubit in the Z-basis (the computational basis) are calculated. These expectation values are real numbers that can be seen as the output of the quantum circuit, providing a form of feature extraction.

We now select from the first image, a region approximately in the center of the image to see the compiled circuit:

```
print(circuit(selected_region))
dev_qiskit._circuit.draw("mpl")
```

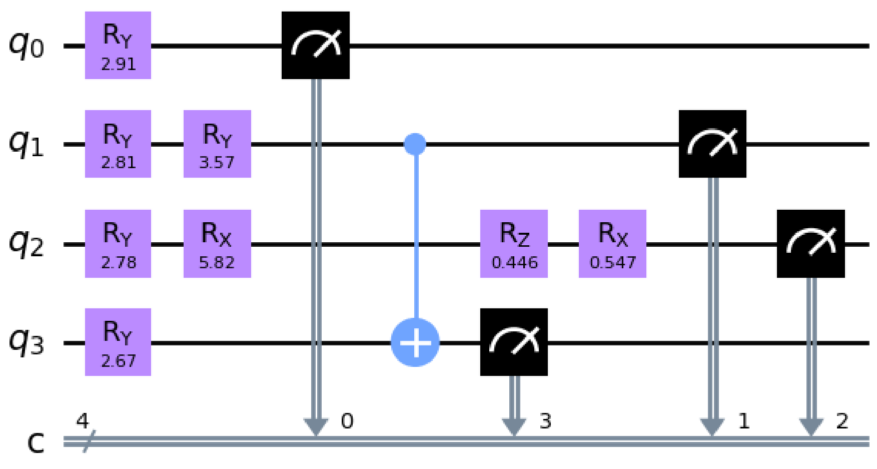


Figure 22 - Feature map + random layer qiskit compiled quantum circuit. Source: run on IBMQ

As described, on the left we see how a Ry rotation gate is applied to each qubit. We selected a region of 4 pixels in the middle of the image because when selecting for example the first region (top left 4 pixels), since they are mostly all black for many images, we always observed 0 values for the angles in the Ry gates represented.

Following the Ry gates there is a random combination of gates including: Rx, Ry, Rz, CNOT to create a more complex entanglement and superposition among the qubits. Finally, each qubit is measured.

6.2.2.2 Applying the Quanvolution circuit

After creating the quanvolution function, we create a function to apply it to the

```
def quanv(image):
    out = np.zeros((14, 14, 4))
    for j in range(0, 28, 2):
        for k in range(0, 28, 2):
            q_results = circuit(
                [image[j, k, 0],image[j, k + 1, 0],
                    image[j + 1, k, 0], image[j + 1, k + 1, 0]])
            for c in range(4):
                out[j // 2, k // 2, c] = q_results[c]
    return out
```

Here, the image is divided into squares of 2x2 pixels, and each square is processed by the previously defined quantum circuit. The output is four expectation values (the result of the final measurement of the quantum circuit), which are assigned to four different channels of a single output pixel. It is important to note that this process halves the resolution of the input image, as each 2x2 square is replaced by a single pixel in the output which is why the output is 14x14. However, the output contains four channels that can be viewed as features that the quantum circuit has extracted from the 2x2 pixel patch.

Then the Quantum Convolution is applied to all images in the Fashion MNIST training and test datasets as a preprocessing step. The benefit of this approach is that the computationally heavy quantum circuit does not have to be run for each epoch during the model's training process, it is done once, and the outputs are saved for later use.

6.2.2.3 Visualization

The following image Figure 23 shows 4 images in our training set and the respective ‘quanvolved’ objects. As explained, after our quantum convolution, images have halved in resolution but now have 4 channels that can be represented as different images.

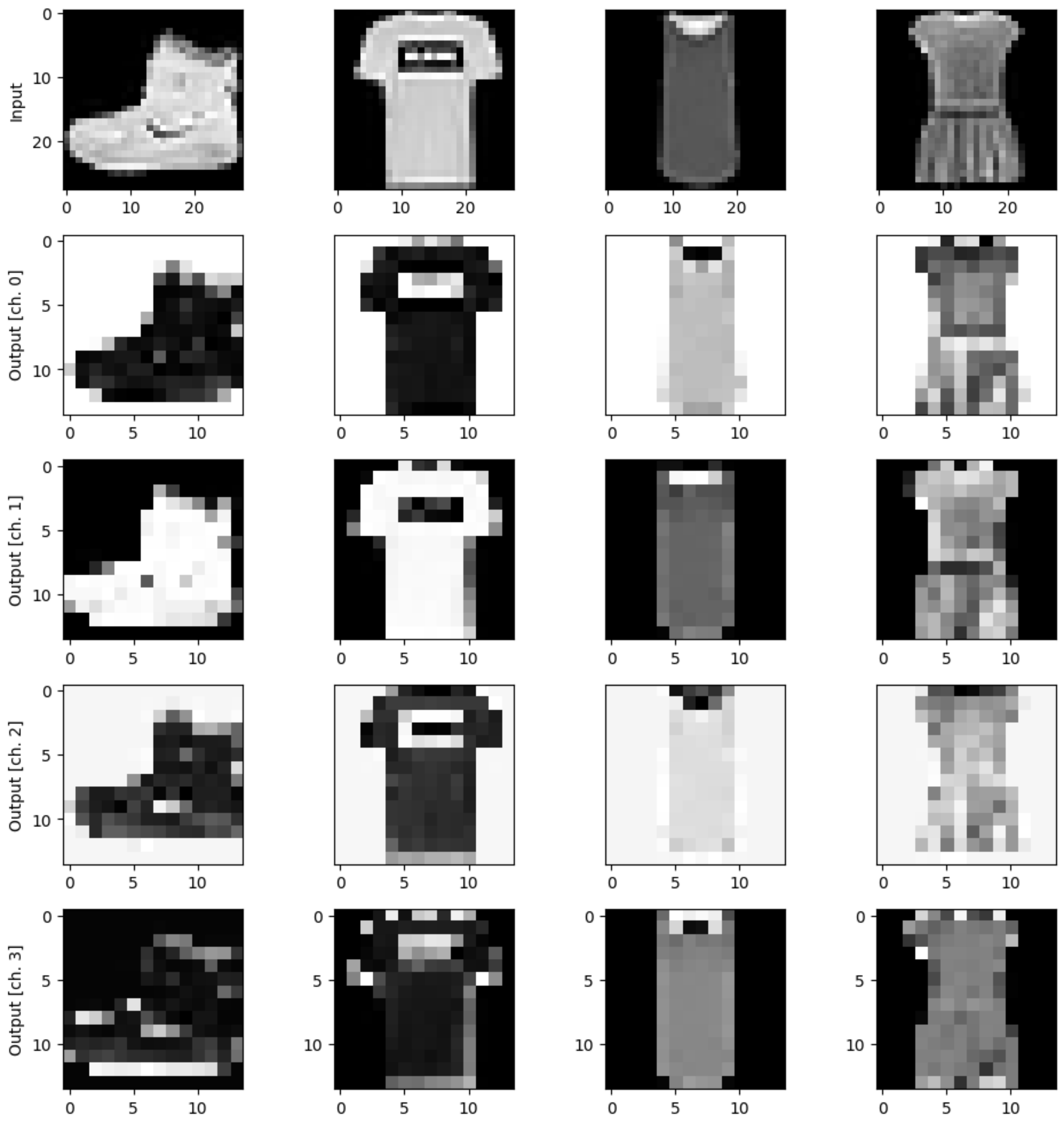


Figure 23 - Quantum convolved images on simulator. Source: plotted with matplotlib

6.2.2.4 Architecture Variations

We can modify the architecture of our quantum convolution layer in several ways:

- **Number of Filters**: Just like in a classical CNN, you could use multiple quantum circuits, each acting as a separate filter. The outputs from each circuit would form different channels in the output image, increasing its depth.
- **Quantum Circuit Design**: the design of the quantum circuit that is used to process each patch of the image can be modified. This might involve using different gates or different methods to encode the input data into the quantum state.
- **Patch Size**: Currently, we are using a 2x2 patch size so 4 qubits. We could do larger patch sizes, which would result in larger quantum circuits, therefore needing more qubits.
- **Add more quanvolutional layers**: Simply add more layers one after the other

6.2.3 QCNN

In the previous section we processed the images to obtain a new set of quantum convolved images. Now we are going to model a simple neural network that receives them as input to the first layer.

We use a basic Keras Sequential model with a flattening layer to convert multidimensional input into a one-dimensional array, allowing for straightforward processing by subsequent layers. Following the Flatten layer is a Dense layer with 10 units, using a softmax activation function for multi-class classification. The model is compiled with the Adam optimizer and the sparse categorical cross-entropy loss function, suitable for integer labels. The model is trained using the quantum-processed image data, with validation performed on a separate quantum-processed test set.

The following shows the full model that combines quantum convolution with a classical feedforward neural network for classification:

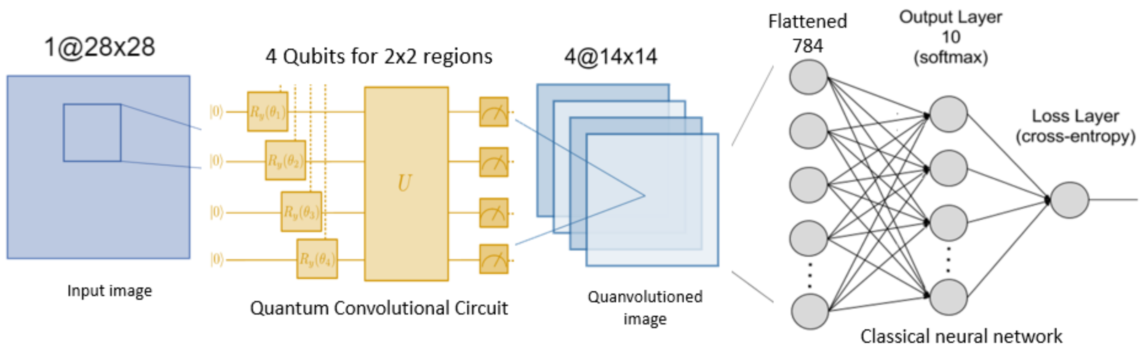


Figure 24 - QCNN: Quantum convolutional neural network full architecture. Source: self-made

Section 7. RESULTS ANALYSIS

This section includes training and performance metrics for the two completely different models we have experimented within 0

7.1 VARIATIONAL CLASSIFIER

7.1.1 QUANTUM SIMULATOR

The following show the results for the model built in section 6.1. The algorithm was run on ibmq_qasm_simulator by connecting to the IBM quantum platform.

This is "A general purpose simulator for simulating quantum circuits both ideally and subject to noise modeling. The simulation method is automatically selected based on the input circuits and parameters." (IBM Quantum, 2023) with 32 qubits.

7.1.1.1 Training

The fit function from the Keras API is used to initiate the training process. The model is trained 10 epochs (number of complete passes through the entire training dataset). The batch size 10 denotes the number of training data in one iteration.

Furthermore, validation data is provided to the fit function allowing the model's performance to be assessed on an unseen dataset during training, which can help in monitoring the model for overfitting. Model described in section 6.1.4.

Epoch	1	2	3	4	5	6	7	8	9	10
Train Loss	1.14	1.01	0.89	0.81	0.74	0.71	0.70	0.69	0.69	0.68
Valid Loss	1.07	0.94	0.85	0.76	0.71	0.70	0.69	0.68	0.67	0.67
Train Accuracy	0.12	0.56	0.74	0.87	0.92	0.88	0.88	0.90	0.90	0.90
Test	0.25	0.61	0.86	0.92	0.92	0.86	0.89	0.89	0.86	0.86

Accuracy										

Table 7.1:1 - Training & validation metrics for the variational classifier in quantum simulator

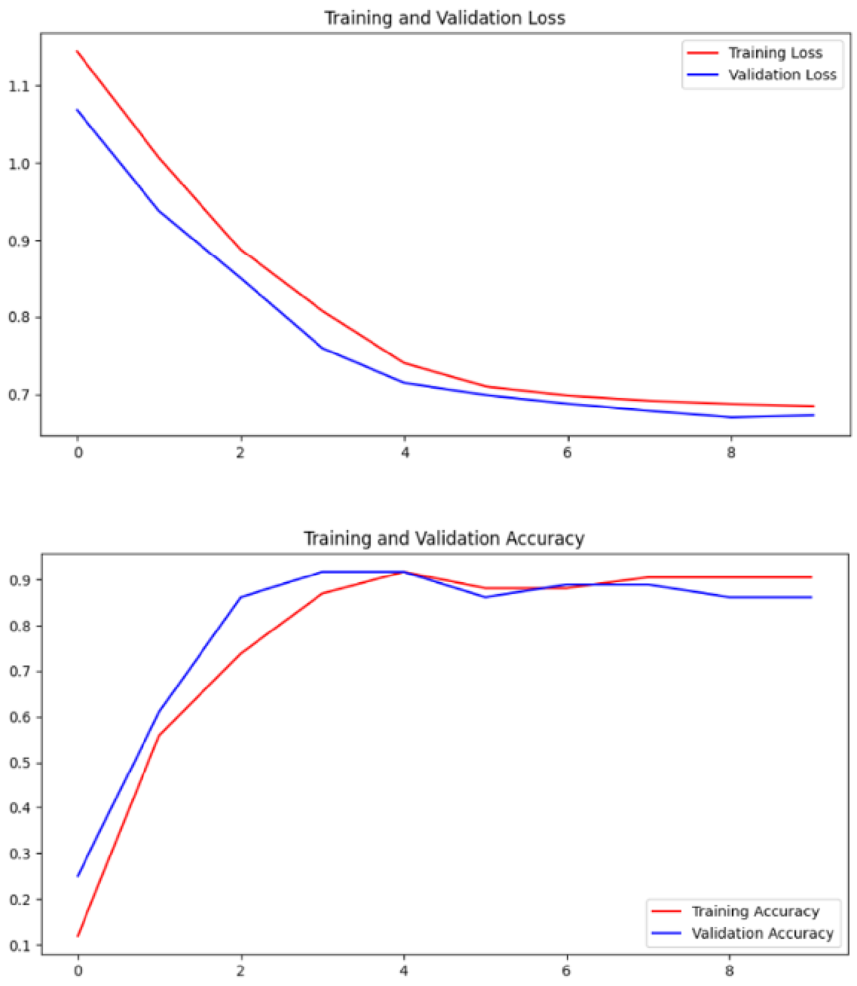


Figure 25 - Training & validation graphs for the variational classifier in quantum simulator

From the training logs, we observe that the model demonstrates a consistent decrease in loss and an increase in accuracy over the ten epochs for both the training and validation sets. This behavior indicates successful learning and convergence.

Moreover, the small gap between training and validation sets suggest no visible overfitting. If a gap between training and validation lines where widening significantly as epochs increase, it could be a sign of overfitting, where the model performs well on the training data but fails to generalize on unseen data.

Lastly, we notice the lines for loss and accuracy start to flatten after around the 5th epoch. This plateau suggests that the model has achieved convergence, with no substantial improvement in performance despite further training. This convergence, added to the absence of visible overfitting, indicates a well-trained model.

7.1.1.2 Performance metrics

Once the model is trained, we use the sklearn.metrics module to produce a classification report and a confusion matrix for the validation set:

Species	precision	recall	f1-score	support
0	1	1	1	10
1	1	0.5	0.67	10
2	0.67	1	0.8	10

Table 7.1:2 – Full performance metrics for the variational classifier run in quantum simulator

Accuracy	83%

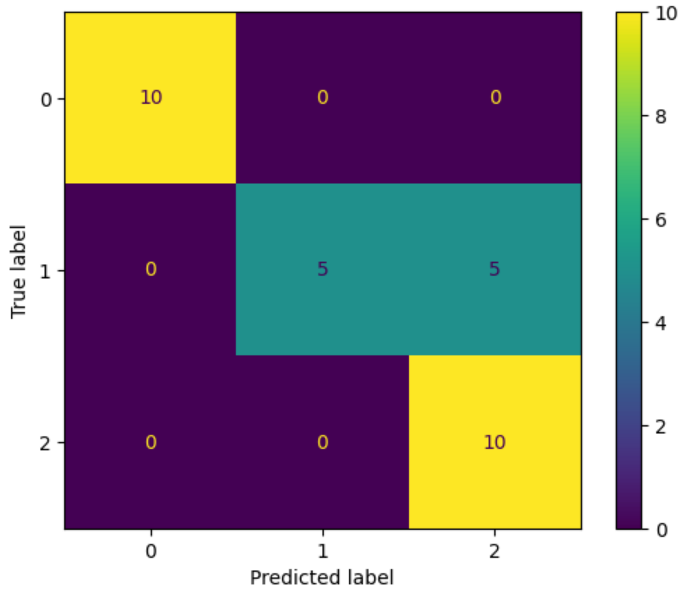


Figure 26 – Confusion matrix for validation set - variational classifier in quantum simulator

From the precision metrics as well as from the confusion matrix, we see perfect precision and recall for class 0-setosa, and some errors in classification between classes 1-versicolor and 2-virginica.

For class 1 while the model achieved perfect precision (meaning there were no false positives), its recall is 0.5, suggesting that it could only correctly identify half of the actual instances of this class. This is reflected in the f1-score, a harmonic mean of precision and recall, which is at 0.67.

Class 2 was identified with a precision of 0.67, implying some false positives, while it had a perfect recall, indicating no false negatives. The f1-score for this class stands at 0.8.

Overall, the model achieved an accuracy of 0.83 on the validation set, which indicates it correctly classified 83% of the instances. The macro and weighted averages for precision, recall, and f1-score are all around 0.8, indicating a relatively balanced performance across all classes.

7.1.1.3 Model Variations

As described on section 6.1.5 and section 6.1.6 we then experiment by adding more classical layers. A quick overview of the results for these variations is shown on the next table.

Validation set metrics	Accurcay	Precision (weighted)	Recall (weighted)	F1-Score (weighted)
Base Model	0.83	0.89	0.83	0.82
Model 1	0.93	0.93	0.93	0.93
Model 2	0.78	0.82	0.77	0.76

Table 7.1:3 - Architecture variations performance metrics - variational classifier in quantum simulator

We see some improvement in model 1 while worsening in model 2. However, the objective of experimenting with architecture variations by adding classical layers after and before the variational quantum layer is not to improve performance but to show the versatility and compatibility of our quantum later. By successfully executing these models, we illustrate the potential for smooth integration of quantum computing techniques within traditional machine learning frameworks such as TensorFlow Keras.

7.1.2 Real Quantum Computer

As explained on section 3.3.1 we have several quantum computers to choose from. We chose IBM quito quantum computer (IBM Quantum, 2023) that has 5 qubits and a quantum volume of 16. We did not choose the most powerful Quantum computer freely available since 5 qubits are enough with our dataset that contains 4 features and the 7 qubits computers were much more saturated.

7.1.2.1 Training

Training a variational classifier on IBM's quantum computer introduces several challenges that can lead to the necessity of executing a large number of jobs.

A variational classifier is trained iteratively over multiple epochs, where an epoch is a complete pass through the entire dataset. The Iris dataset, although relatively small by classical standards, contains 150 data points. For each epoch, each of these data points needs to be processed, meaning the quantum circuit has to be executed 150 times per epoch, reaching hundreds for a standard training time of 10+ epochs.

Another challenge arises from the batching process. Batching refers to the practice of feeding multiple data points through the model at once, which can significantly speed up the training process. Unfortunately, when integrating pennylane and Keras with Qiskit, the batching of jobs is not fully supported when executing on quantum hardware. This limitation necessitates the execution of one job per data point, further amplifying the number of jobs that need to be run.

In our case, we were only able to complete the first epoch after more than 40 hours.

Epoch 1			
Training Loss	Validation Loss	Training Accuracy	Validation Accuracy
1.28	1.32	0.18	0.27

Table 7.1:4 – Partial training metrics for variational classifier in real quantum computer

These results are not representative of performance since it is only the first epoch where the model is starting to train. We would need about 10 epochs in order to compare performance with the quantum simulator, but as we explained this was not possible due to the queuing system of IBM's platform and the hybrid nature of the algorithm.

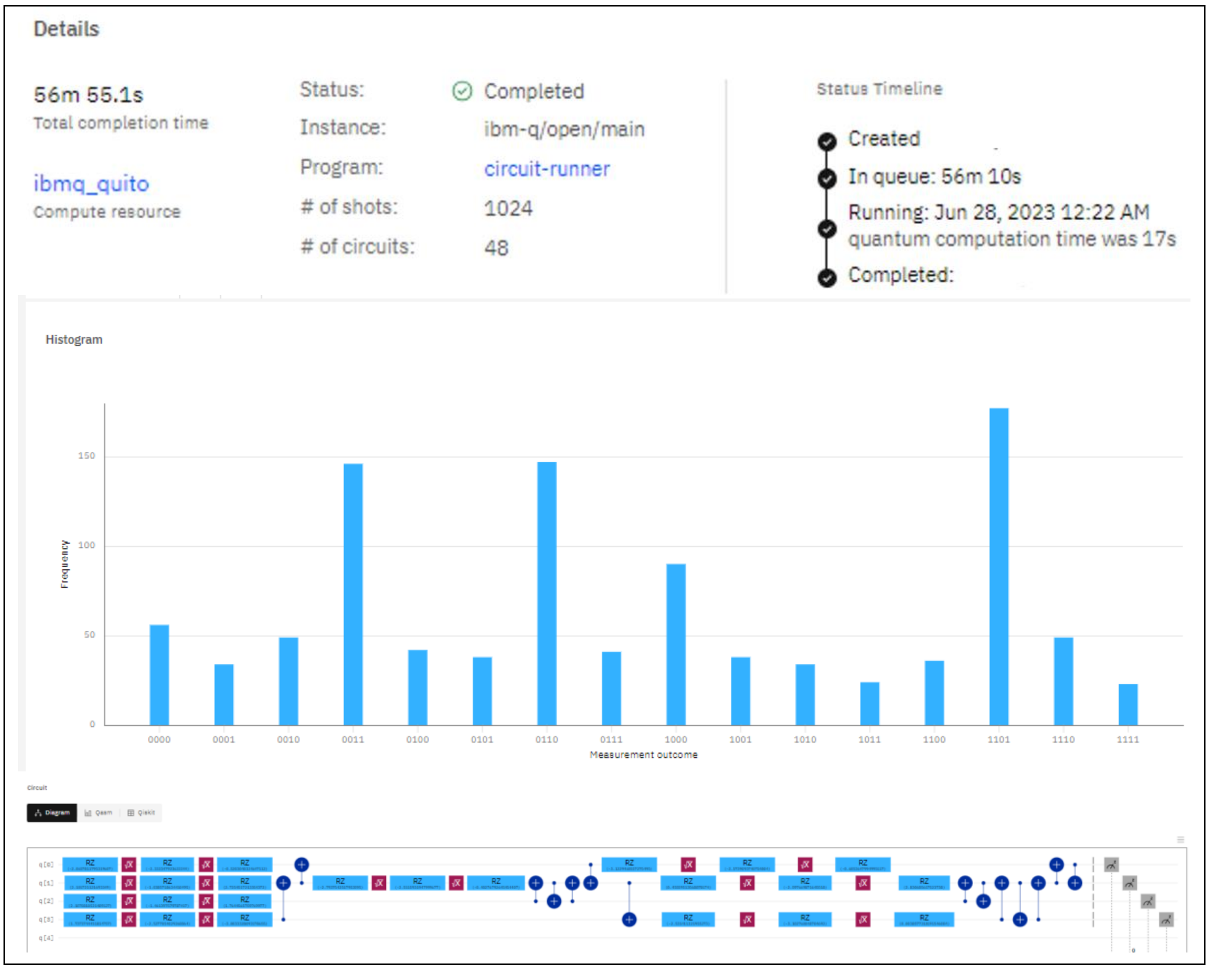
7.1.2.2 IBM Quantum jobs execution

Figure 27 shows a screenshot of IBMQ during the execution of one of the jobs to train the variational classifier. We observe the completion times as well as the number of shots and circuits.

Shots are the number of times the given quantum circuit is executed. Because of the probabilistic nature of quantum mechanics, the result of executing a quantum circuit once may not provide sufficient information. Therefore, quantum circuits are typically executed multiple times, with the number of executions often referred to as "shots". The bar chart you shows the frequency of different measurement outcomes for our 4 qubits over a certain number of shots. These frequencies can provide useful information about the probabilities associated with different states of the quantum system.

The number of circuits refers to the number of individual quantum circuits that are packaged together to be executed as a single job on a quantum computer. When running a quantum algorithm, especially in the context of quantum machine learning or optimization, we need to execute a variety of different quantum circuits. For instance, in variational quantum algorithms, you might need to run different circuits corresponding to different parameter settings of your variational form.

As for the visualization of the compiled quantum circuit, this provides a schematic representation of the quantum operations that will be performed during the execution of each job. Each operation corresponds to a quantum gate, and the overall structure of the circuit reflects the sequence and combination of gates that implement the quantum algorithm.



Details

56m 55.1s
Total completion time

ibmq_quito
Compute resource

Status: Completed
Instance: ibm-q/open/main
Program: circuit-runner
# of shots: 1024
# of circuits: 48

Status Timeline
Created
In queue: 56m 10s
Running: Jun 28, 2023 12:22 AM
quantum computation time was 17s
Completed:

Figure 27 – IBM Quantum Platform screenshot showing for a random job: main characteristics: time, number of circuits & jobs; bar chart of frequency of measurements for shots & compiled circuit

7.2 QUANVOLUTION

7.2.1 QUANTUM SIMULATOR

The following show the results for the model built in 6.2.3. The algorithm was also run on ibmq_qasm_simulator by connecting to the IBM quantum platform.

7.2.1.1 Training

The model was trained on a small dataset with a total of 200 training samples and validated with 60 test samples. This is because of the time-consuming process of quantum convolutional processing for quantum circuits. Images were batched in groups of 4 and the neural network was trained for 10 epochs.

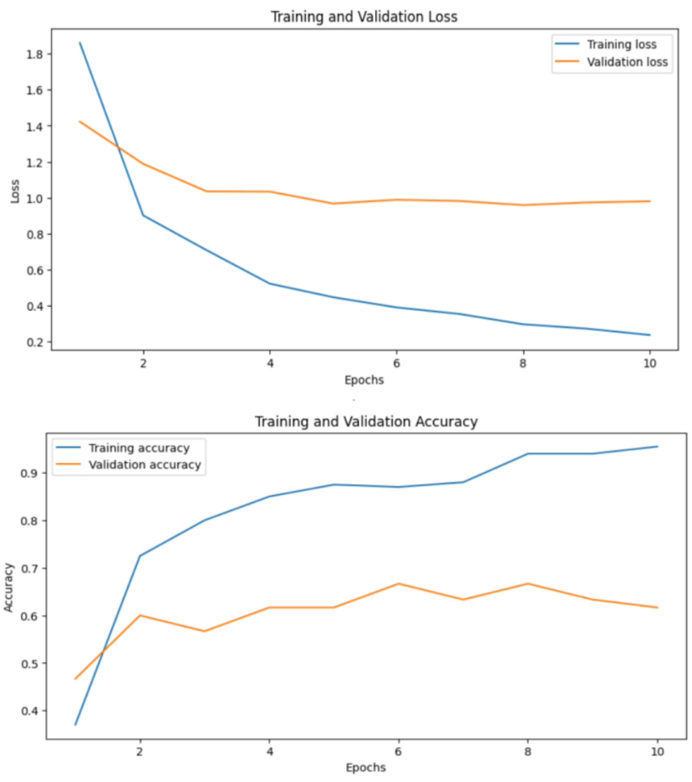


Figure 28 - Training & validation metrics for the Quantum Convolutional Neural Network.

Source: Plotted with matplotlib

From the generated plots, we can observe that both the training loss and accuracy improve considerably with each epoch. This indicates that the model is learning well from the training data.

However, although the validation loss initially decreases, it then flattens after the 5th epoch, while the validation accuracy also seems to flatten and even decrease slightly after reaching a peak around the 6th epoch. This may suggest that the model begins to overfit to the training data after about 5-6 epochs, as it performs well on the training data but not as well on the unseen validation data.

The phenomenon of overfitting is not surprising given the small dataset size. We only have 200 samples for training, which is quite small, especially for image classification.

7.2.1.2 Performance Metrics

On the images we see that the neural network with the quantum layer performs slightly better. However, the difference is not significantly big and it is accounted for due to the variability of the classical neural network training.

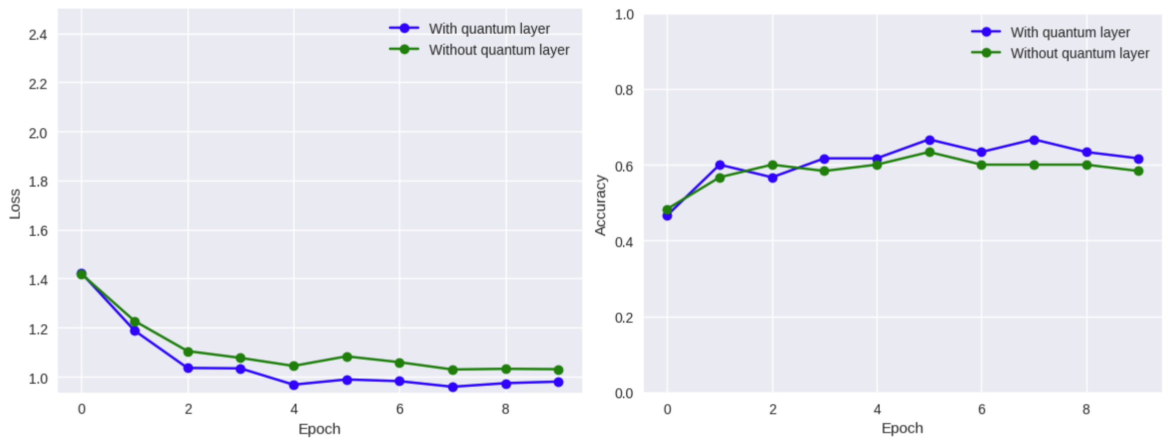


Figure 29 - Training & Validation Graphs comparing Neural network with & without Quantum convolution layer. Blue Shows With Quantum layer, Green without Quantum later

Source: Plotted with matplotlib

Training Accuracy	95%
Validation Accuracy	62%

Table 7.2:1 - Training & Validation Accuracy for the QCNN

As we already saw during the training, due to the small size of the dataset, our model is not able to generalize well with unseen data. This is clear from the high training accuracy and much lower validation accuracy. Also, in the plots we do see some improvement, but it clearly stops after only about 5 epochs.

7.2.2 Real Quantum Computer

In section 7.1.2.2, we highlighted the high number of jobs needed to run the variational classifier. This also applies when processing images via quantum convolution on the IBM quantum computer-

Quantum convolution necessitates breaking down a standard 28x28 pixel image into 2x2 pixel windows, resulting in 196 circuit runs for each image. Again, due to the unsupported batching, the number of jobs required increases to several hundred.

Therefore, due to these limitations, our experiment could only process four images within a feasible timeframe. These results underline the current challenges of quantum computing, especially for machine learning tasks. Due to only being able to perform the quantum convolution in four of the images, applying the classical CNN model with such small amount of data did not make sense.

However, we do show how it is possible to execute quantum convolution on a Real Quantum computer as results for 4 images are shown on Figure 30:

Our quantum convolution operation resulted in four distinct images for each input image, corresponding to four channels. 2x2 windows reduced resolution from 28x28 to 14x14 pixels. Each channel aims to identify different features within the image, just like filters in convolutional neural networks operate. These channels can detect diverse attributes like edges, textures, or other patterns, providing a multilayered, comprehensive view of the initial image's key features.

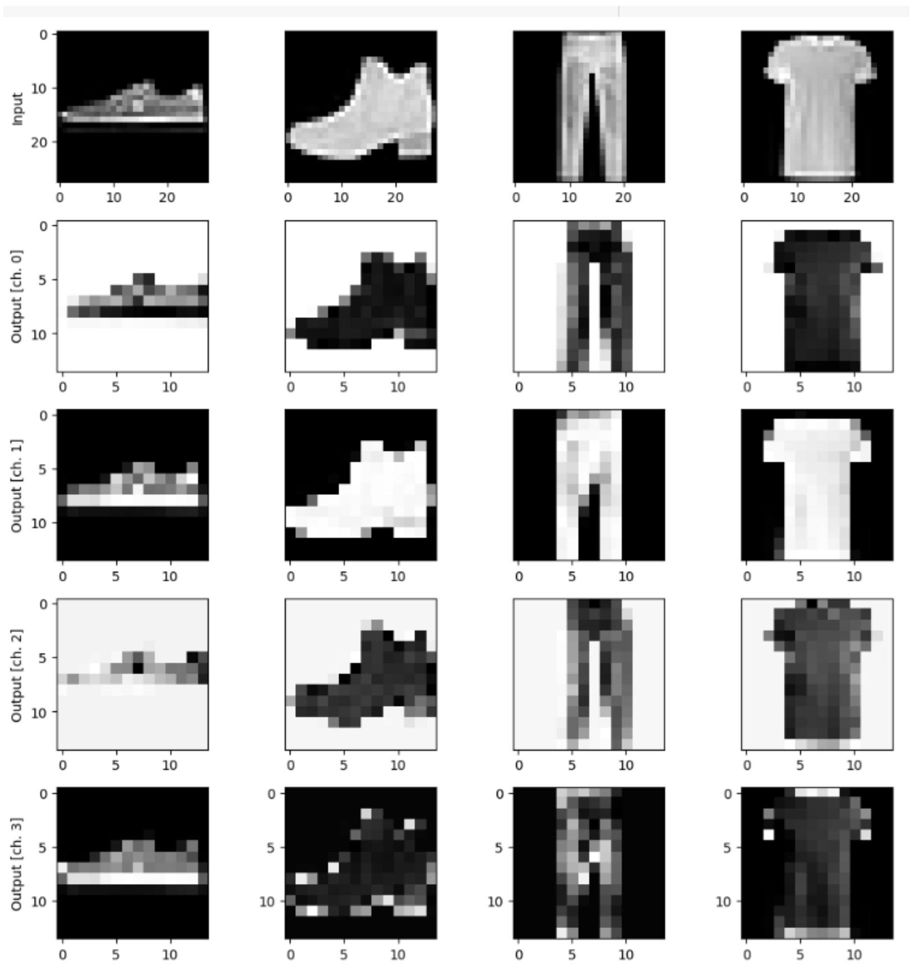


Figure 30 - Quantum convolved images on real Quantum Computer. Source: plotted matplotlib

Section 8. CONCLUSIONS & FUTURE WORK

8.1 CONCLUSIONS

Our exploration of quantum computing (QC) applications in machine learning (ML) has shown both exciting advancements and considerable challenges. Despite the field's growth, we are still in the early stages, with obstacles such as the limited number of qubits, quantum systems' errors, and quantum decoherence inhibiting the practical implementation of promising quantum algorithms.

In response, there has been a significant shift towards developing hybrid quantum-classical models and numerous research papers are emerging, exploring different architecture variations, and experimental configurations aimed at optimizing the use of our existing NISQ devices.

In the context of this project, which focused on the investigation of Variational Quantum Classifiers and Quantum Convolution, these challenges and their implications were acutely evident. In the realm of quantum machine learning, we faced a significant bottleneck when dealing with regular classical datasets. Current quantum systems, like IBM's freely available quantum computers, are still not capable of completely executing moderately complex ML algorithms, which is a significant barrier to wider applicability and adoption.

In the experiments carried out with the Variational Quantum Classifier, performance was promising when using IBM's quantum simulator. However, when the same algorithm was implemented on a real quantum computer, we encountered issues with extended execution times due to the high number of jobs, the characteristics of IBMQ's queue, and the algorithm's hybrid iterating nature. This illustrates the real-world constraints we are currently facing and underscores the need for advancements in quantum hardware.

Similarly, our work with Quantum Convolution, while promising in theory, revealed a set of distinct challenges in practice. The quantum simulator was able to handle the algorithm but was limited to a small number of images due to time and computational constraints. This limitation affected our ability to fully train and test the Quantum Convolutional Neural Network (QCNN), resulting in results that did not show an advantage of the quantum convolution layer. On real quantum hardware, these limitations were even more pronounced since processing single images took hours. Quantum convolution has not yet been proven useful over regular convolution, and our results evidenced this.

In summary, these experiments have served to show the integration of quantum algorithms within classical machine learning frameworks but also highlight the gap between the theoretical quantum learning algorithms and the practical realities of implementing these algorithms on current quantum hardware.

The field of Quantum Machine Learning is promising and offers compelling synergies between quantum and classical computing approaches. However, realizing this potential will require us to overcome significant challenges both technological and in terms of refining and adapting quantum algorithms.

8.2 Future Work

Based on the proyect's exploratory analysis and results, potential future directions I would like to explore include:

- Other types of Machine Learning: I have tried specifically classification and quantum convolution. Other areas could include regression, unsupervised algorithms or reinforcement learning.
- Quantum Model Exploration: Given the rapid development of the quantum machine learning field, investigating a broader range of quantum machine learning models would be insightful. There are many other hybrid algorithms and purely quantum machine learning algorithms to study.
- Architecture Variations: Exploring even more variations of quantum layers within classical models could also be a interesting. This could involve tweaking the structure and parameters of quantum layers, or experimenting with different configurations of hybrid quantum-classical architectures.
- Access to More Powerful Quantum Computers: As quantum hardware advances, accessing more powerful it would provide an opportunity to study the scalability and potential benefits of quantum algorithms in more depth. Larger qubit quantum computers would offer a richer platform for developing and testing quantum machine learning models.

Section 9. REFERENCES

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

Adhikary, S., Dangwal, S., & Bhowmik, D. (2020). Supervised learning with a quantum classifier using multi-level systems. *Quantum Information Processing, 19*, 1-12.

Albornoz, C., Alonso, G., Andrenkov, M., Angara, P., Asadi, A., Ballon, A., . . . Wakeham, D. (2021). *Xanadu Quantum Codebook.*

Baldwin, C., & Mayer, K. (2022). Re-examining the quantum volume test: Ideal distributions, compiler optimizations, confidence intervals, and scalable resource estimations. *Quantum. 6: 707. arXiv:2110.14808.*

Basics of quantum information. (n.d.). Retrieved from Qiskit learn: https://qiskit.org/learn/course/basics-quantum-information/

Bauckhage, Bye, Iftikhar, Knopf, Mustafic, Piatkowski, . . . Sultanow. (2022). *Quantum Machine Learning - State of the Art and Future Directions.* Federal Office for Information Security.

Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., Blank, C., McKiernan, K., & Killoran., N. (2018). PennyLane: Automatic differentiation of hybrid quantum-classical computations. *arXiv:1811.04968.*

Bonet-Monroig, X., Wang, H., Vermetten, D., Senjean, B., Moussa, C., Bäck, T., & O'Brien. (2023). Performance comparison of optimization methods on variational quantum algorithms. . *Physical Review A, 107(3), 032407.*

Chen, S. Y., & Yoo, S. (2021). Federated quantum machine learning. *Entropy, 23(4), 460.*

Cirq, c. (2023). Cirq, a python framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits.

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

Dang, Y., Jiang, N., Hu, H., Ji, Z., & Zhang, W. (2018). Image classification based on quantum K-Nearest-Neighbor algorithm. *Quantum Information Processing, 17,*, 1-18.

Deutsch, D., & Jozsa, R. (1992). Rapid solutions of problems by quantum computation. *Proceedings of the Royal Society of London A.*, 439 (1907): 553–558.

Farhi, E., & Neven, H. (2018). Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002.*

Feynman. (1963). *Quantum Mechanics Lectures.* Retrieved from The Feynman Lectures on Physics, Volume III: https://www.feynmanlectures.caltech.edu/III_toc.html

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics 7 (2)*, 179-188.

Gidney, C., & Ekerå, M. (2021). How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. . *Quantum, 5, 433.*

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, 212–219.

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

Henderson, M., Shakya, S., Pradhan, S., & Cook, T. (2020). Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence, 2(1), 2.*

IBM Quantum. (2023). Retrieved from IBM: https://quantum-computing.ibm.com/

Macaluso, A., Clissa, L., Lodi, S., & Sartori, C. (2020). A variational algorithm for quantum neural networks. In Computational Science–ICCS 2020: 20th International Conference. *In Computational Science–ICCS 2020: 20th International Conference* (pp. 591-604). Amsterdam, Netherlands: Springer International Publishing.

Martín-López, E., Martín-López, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X.-Q., & ., J. L. (2012). Experimental realization of Shor's quantum factoring algorithm using qubit recycling. *Nature Photonics.*, 6 (11).

Qiskit, c. (2023). Qiskit: An Open-source Framework for Quantum Computing. 10.5281/zenodo.2573505.

Quantiki Blosch Sphere. (2023). Retrieved from Quantiki: https://www.quantiki.org/wiki/bloch-sphere

Quantiki Hilbert Space. (2023). Retrieved from Quantiki: https://www.quantiki.org/wiki/hilbert-spaces

Quantinuum Hardware. (2023). Retrieved from Quantinuum: https://www.quantinuum.com/hardware

Quantinuum news. (2023, June 30). Retrieved from Quantinuum H-Series quantum computer accelerates through 3 more performance records for quantum volume : https://www.quantinuum.com/news/quantinuum-h-series-quantum-computer-accelerates-through-3-more-performance-records-for-quantum-volume-217-218-and-219

Schuld, M., Bocharov, A., Svore, K., & Wiebe, N. (2018). Circuit-centric quantum classifiers. *arXiv preprint arXiv:1804.00633.*

Schumacher, B. (1995). Quantum coding . *Physical Review A.*, 51 (4): 2738–2747.

Shor, P. (1994). Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science.*, 124–134.

Sinha, U. (2023). The Experiments That Led to the Nobel Prize in Physics 2022. *Resonance*, 28(1), 85-116.

Susskind. (2008). *Modern Physics Lectures: Quantum Mechanics.* Retrieved from Stanford: https://www.youtube.com/watch?v=JzhlfbWBuQ8&list=PL84C10A9CB1D13841&ab_channel=Stanford

Susskind, L., & Friedman, A. (2014). *Quantum mechanics: the theoretical minimum.* Basic Books. Ch 1-8.

Tirole, R., Vezzoli, S., Galiffi, E., Robertson, Maurice, & Tilmann. (2023). Double-slit time diffraction at optical frequencies. *Nat. Phys.*

Weigold, M., Barzen, J., Leymann, F., & Salm, M. (2020). Data encoding patterns for quantum computing. *In Proceedings of the 27th Conference on Pattern Languages of Programs*, 1-11.

wolfram.com. (2015, October 23). Retrieved from Number Field Sieve: https://mathworld.wolfram.com/NumberFieldSieve.html

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747.*

Young, T. (1804). The Bakerian lecture. Experiments and calculation relative to physical optics. *Philosophical Transactions of the Royal Society of London*, 94: 1–16.

APPENDIX I: SDGs Alignment

- Climate action (SDG 13): Quantum machine learning algorithms could be used to analyze large datasets related to climate change and to develop more accurate models for predicting and mitigating the impacts of climate change.
- Healthcare (SDG 3): Quantum machine learning algorithms could be used to analyze large datasets from healthcare systems and to develop more effective treatments for diseases. Quantum computing has been proven to have many applications in chemistry when modeling molecules and chemical reactions, which could be useful for the development of new, more effective drugs
- Agriculture (SDG 2) and clean water and sanitation (SDG2): related to chemical simulations, QC could also accelerate the use of heterogeneous catalysts for water treatment and develop more effective methods of fixing nitrogen in fertilizers.
- Affordable clean energy (SDG 7): Quantum gates are reversible, with the advantages of reduced power consumption and no heat generation. In theory, reversible gates take require no energy to run, but this is far in the future from our current Quantum computers that even need to be kept at near to absolute zero temperatures to work.