UNIVERSITÀ DEGLI STUDI FIRENZE

Scuola di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Tesi di Laurea

RETI NEURALI QUANTISTICHE CONVOLUZIONALI PER LA RILEVAZIONE DI TUMORI CEREBRALI UMANI DALLE IMMAGINI MRI

QUANTUM CONVOLUTIONAL NEURAL NETWORKS FOR HUMAN BRAIN TUMOR DETECTION FROM MRI IMAGES

JACOPO DARDINI

Relatore: *Prof. Filippo Caruso*

Correlatore: *Prof. Rosario Pugliese*

Anno Accademico 2023-2024

CONTENTS

List of Figures 3

1 Introduction 7

- 1.1 Machine Learning and Deep Learning 7
- 1.2 Introduction to Quantum Computing 9
- 1.3 Fundamentals of Quantum Machine Learning 11

2 Machine Learning for MRI Image Classification 15

- 2.1 Introduction to Convolutional Neural Networks (CNNs) . 15
  - 2.1.1 Overall Architecture 15
  - 2.1.2 Convolutional Layer 16
  - 2.1.3 Pooling Layer 17
  - 2.1.4 Fully-connected Layer 18
- 2.2 Application of CNNs in MRI Image Classification 18

3 Implementation of an Hybrid Quantum-Classical ML Model for Image Classification 21

- 3.1 Description of QCNN for MRI Image Classification 21
  - 3.1.1 Introduction to QCNN 21
  - 3.1.2 Overall Architecture 21
- 3.2 Convolution using a Quantum Circuit 22

4 Experimental Results 25

- 4.1 Comparison of QCNN Performance with Different Circuit Designs 25
  - 4.1.1 Random Layers 26
  - 4.1.2 Strongly Entangling Layers 28
  - 4.1.3 Quantum Fourier Transform Inspired Circuit 29

5 Conclusions and Future Developments 31

- 5.1 Summary of main conclusion 31
- 5.2 Final considerations of the importance and prospects of QML in Medical Image Analysis 32

Bibliography 33

LIST OF FIGURES

Figure 1 Architecture of a simple deep neural network [16]. 8
Figure 2 Bloch sphere representation of possible states of a single qubit [4]. 10
Figure 3 Schematic diagram of a parametrized quantum circuit [8]. 12
Figure 4 Example of a parametrized quantum circuit with 6 qubits [8]. 13
Figure 5 A simple CNN architecture [15]. 16
Figure 6 Example of convolution with kernel 3x3 [12]. . . . 17
Figure 7 Architecture of the CNN model used for benchmarking 20
Figure 8 Example of the Quantum Convolution [10]. 22
Figure 9 2x2 Kernel and 1 Layer quantum circuit. 23
Figure 10 Example of feature extraction using quanvolution. 24
Figure 11 Illustration of a 3 layers circuit for convolution with a 3x3 kernel on original images. 26
Figure 12 These images represent the before and after of the application of the quantum circuit shown in Figure 11. The image in the top left is the original one, and the other 9 are the output channels. 27
Figure 13 A 2x2 Kernel and single Layer circuit. 28
Figure 14 From the left: the original input image and its four convolved images. 28
Figure 15 3x3 Kernel and 3 Layer Circuit. 28
Figure 16 2x2 Kernel and 2 Layer Circuit, yielding the images in Figure 17. 29
Figure 17 From the left: the original input image and its four convolved images. 29

"I know of no better life purpose than to perish in the attempt of the great and the impossible."
Friedrich Nietzsche

1

INTRODUCTION

1.1 MACHINE LEARNING AND DEEP LEARNING

Machine Learning (ML) is a subset of Artificial Intelligence (AI) aimed at learning specific tasks, such as classification, by identifying hidden patterns in collected data [6]. The typical steps involved in creating a Machine Learning model are data collection, data processing, model training, and evaluating the model's performance on unseen data subsets. A model developed through this process can perform a given task on new data based on its training and validation.

There are three main types of Machine Learning: Supervised Learning, Unsupervised Learning, and Reinforcement Learning.

- Supervised Learning: In this type, algorithms are provided with labeled data during training. For example, in image classification tasks like distinguishing between dogs and cats, the training data consists of images labeled with specific tags, such as 0 or 1, indicating whether the image depicts a dog or a cat. While these algorithms are commonly used, organizing such labeled data can be challenging.

- Unsupervised Learning: Algorithms in unsupervised learning work with unlabeled data during training. The model's task is to discover patterns in the data without any guidance. Clustering is an example of unsupervised learning, where data is classified by grouping similar items and separating those with distinct features.

- Reinforcement Learning involves an agent interacting with an environment, learning to make decisions through rewards or penalties, aiming to maximize cumulative rewards over time. This type of learning is widely used in fields like robotics and gaming.

Another crucial concept in machine learning models is the loss function. The loss function measures the error between the actual values the model was trained on and its predictions. Having a function that quantifies the accuracy of our model allows us to optimize the model's parameters to minimize the error as much as possible.

There are many examples of loss functions, some of the most common being:

- Mean Squared Error: $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$.

- Mean Absolute Error: $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$.

- Cross-Entropy: $-\sum_{i=1}^{n} y_i \log(\hat{y}_i)$.

where n is the number of samples, y_i is the actual label and $\hat{y}_i$ is the predicted one.
A notable subset of machine learning algorithms is deep learning. These algorithms simulate human learning by forming models composed of many layers of neurons that succeed each other to create deep neural networks. These models excel at extracting features hierarchically from datasets, thus enabling them to identify highly complex patterns. An example of such structure is shown in Figure 1.

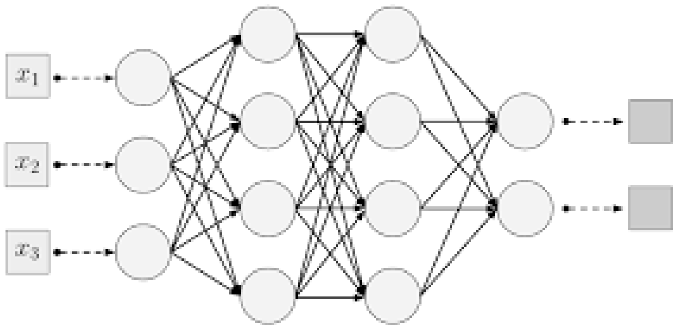


Figure 1: Architecture of a simple deep neural network [16].

We can identify three kinds of layers: the input layer which contains the data provided to the model. For instance, when working with images, each neuron represent the value of one pixel of the original image. The hidden layers, which can be hundreds or thousands, perform complex computations by combining the inputs from the previous layer linearly followed by a non-linear activation function. Every connection between neurons of different layers has a weight that gets updated during the training process, adjusting these parameters improves model's performance. Finally, we have an output layer, usually implemented using a fully connected layer, where the real decisions of the model are made, for example, in image classification, there will be an output neuron for each possible output class.

1.2 INTRODUCTION TO QUANTUM COMPUTING

Despite its power and widespread use, classical computing has several limitations that have motivated the exploration of alternative computing paradigms. These are some of the most evident limitations of classical computing:
-Exponential growth in complexity: Some intricate problems show exponentially growth in complexity as the input increase. This pushes classical computers to require an exponential increase in computing resources.
-Intractability of certain problems: Even the most advanced algorithms and hardware find difficult to solve some inheritable complicated problems such as: factoring large numbers, solving complex optimization problems and simulating quantum systems.
-Limitations to system's state representation: In classical computing, system states are represented using classical bits, which can exist in one of two states, 0 or 1. Classical memory systems, such as RAM and hard disk drives, store and manipulate these bits using classical logic gates and circuits. However, the representation of complex states or superposition is not possible within the classical computing framework, restricting the range of problems that can be efficiently solved.
Quantum Computing is an exciting and promising field at the intersection between mathematics, computer science and physics. It harnesses quantum mechanics properties to improve the efficiency of computation [14]. The quantum bit or qubit is the simplest quantum mechanical system and it generalizes the classical bit. Considering a two-level system, that is, a system described by two possible values. We notate the two states as follows, in the space of complex squares $\mathbb{C}^2$:
$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$
An important new feature absent in classical computing is the possibility to have a superposition of the two basis states $|0\rangle$ and $|1\rangle$. This means that a qubit can represent both 0 and 1 simultaneously, along with all possible combinations of 0 and 1 with varying probabilities. The ability of qubits to exist in superpositions enables quantum computers to perform computations in parallel. This corresponds to a 2-dimensional vector with

complex coefficients, which is an element of the vector space $\mathbb{C}^2$. The state of such superposition can be described as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$, where α and β are complex amplitudes, meaning that the state is in $|0\rangle$ with probability $|\alpha|^2$ or in $|1\rangle$ with probability $|\beta|^2$, this implies that $|\alpha|^2 + |\beta|^2$ must be equal to 1.
The quantum state of a qubit can be visualized as a vector of length 1 on the Bloch sphere, as shown in Figure 2. The angles θ and γ represent the angles of spherical coordinates. The two basis states $|0\rangle$ and $|1\rangle$ are represented by $\theta = 0$ and $\theta = \pi$, respectively, pointing in +z and -z direction.

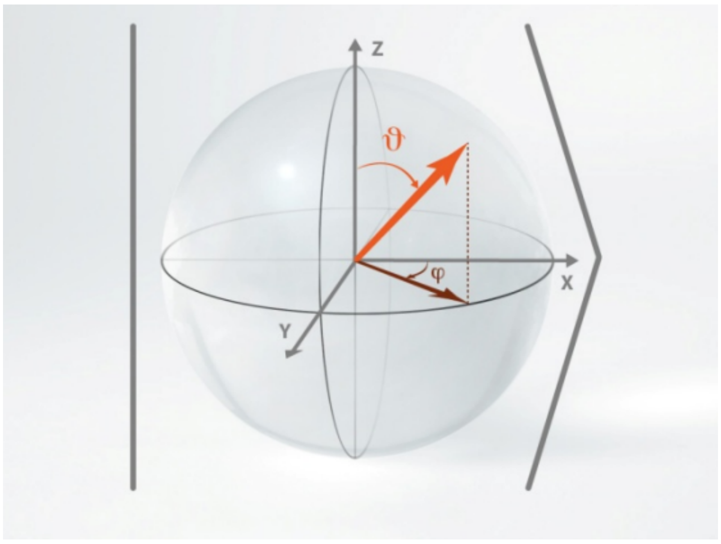


Figure 2: Bloch sphere representation of possible states of a single qubit [4].

The quantum state of a qubit can be manipulated and it evolves in time. An operation on a single qubit can be represented as a rotation on the Bloch sphere and mathematically is determined by a unitary operation, a 2x2 matrix U, with $U^\dagger U = UU^\dagger = 1$.
Similarly to the classical case, quantum gates are the building blocks of quantum circuits. They are used to manipulates qubits and perform specific computational tasks. These transformations can rotate the state of a qubit in the complex vector space $\mathbb{C}^2$, change the probability amplitudes of the states, or entangle multiple qubits together.
Quantum entanglement is a phenomenon in which two or more particles become correlated in such a way that the state of one particle cannot be

described independently of the state of the others, even if the particles are separated by large distances. This implies that a measurement performed on one of the particles will instantaneously influence the state of the other, violating classical intuition about spatial separation. Entanglement is a fundamental element of quantum mechanics and has been widely studied for its implications in quantum information theory and quantum computation [14]. Usually, at the end of quantum circuit, you can measure the qubits. Measurement of qubits is a fundamental operation in quantum computing, playing a crucial role in extracting information from a quantum system. When a qubit is measured, its quantum superposition collapses into a definite state, yielding a specific outcome: 0 or 1. This collaps process enables us to go from quantum information to classical data. There are examples of quantum superiority over classical counterpart for specific tasks. One of the most famous of these algorithms is Grover's algorithm, also known as the quantum search algorithm. This algorithm for unstructured search finds with high probability the unique input to a black box function that produces a particular output value, using just $O(\sqrt{N})$ evaluations of the function, where N is the size of the function's domain. It was conceived by Lov Grover in 1996.

1.3 FUNDAMENTALS OF QUANTUM MACHINE LEARNING

In recent years, researchers have been exploring the use of quantum computing routines to enhance the performance of classical machine learning algorithms. The ability of quantum systems to exist in superposition allows operations to be sped up, as they can be executed in parallel.
In quantum machine learning, quantum algorithms are developed to address classical machine learning problems by leveraging the efficiency of quantum computation.
Machine learning can be divided into four paradigms: Classical-Classical (CC), Classical-Quantum (CQ), Quantum-Classical (QC), and Quantum-Quantum (QQ). The majority of research in quantum machine learning is focused on CQ, where we use classical data and a quantum model.
Models of particular interest for the purpose of this thesis are variational quantum classifiers. Variational or parametrized quantum circuits are quantum algorithms that depend on trainable parameters, as illustrated in Figure 3. Like standard quantum circuits, they consist in 4 steps:

1. Encoding of classical data into a quantum state.

2. Apply the parametrized model.

3. Measure the circuit to extract labels.
4. Use optimization techniques (like gradient descent) to update model parameters.

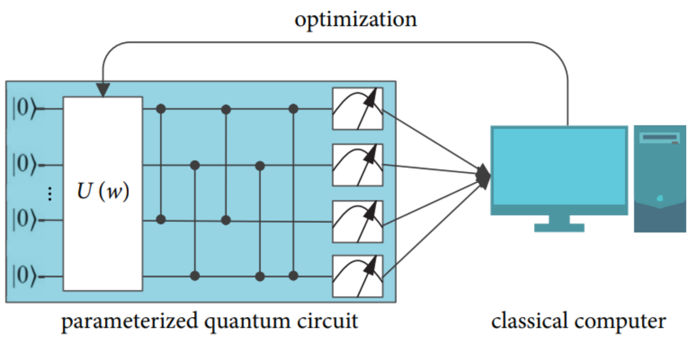


Figure 3: Schematic diagram of a parametrized quantum circuit [8].

A particular relevant challenge is how to represent classical data in quantum systems. One of the most common approaches is to represent classical data as binary strings $(x_1, \ldots, x_n)$ with $x_i \in \{0, 1\}$ for $i = 1, \ldots, n$, which are then directly translated into n-qubit systems $|x_1 \ldots x_n\rangle$ from a 2n-dimensional Hilbert space with basis $\{|0 \ldots 00\rangle, |0 \ldots 01\rangle, \ldots, |1 \ldots 11\rangle\}$, and to read information out through measurements. Another type of encoding is called amplitude encoding. In this paradigm the original value data is encoded directly in the qubits of the quantum circuit using rotation gates. For this type of encoding, the original data is normalized to be in the range $[0, \pi]$.
After data encoding, we apply the parametrized quantum circuit, as presented in Figure 4, that consists in a series of quantum entangling and rotational gates, to the qubits. We can represent the circuit as: The circuit described above can be referred to by different names. It can be known as a parametrized quantum circuit, a variational circuit, or an ansatz—a circuit serving as a template with parameters embedded within it. To increase the number of parameters, one approach is to repeat the circuit multiple times. After applying quantum operations, the final state of the quantum system is measured. This measurement causes the quantum state to collapse into one of its possible classical states, each outcome occurring with a probability determined by the quantum amplitudes of the state.

Typically, the expectation values $f(\theta) = \langle 0|U^\dagger(\theta)\hat{B}U(\theta)|0\rangle$ of one or more such circuits — possibly with some classical post-processing —

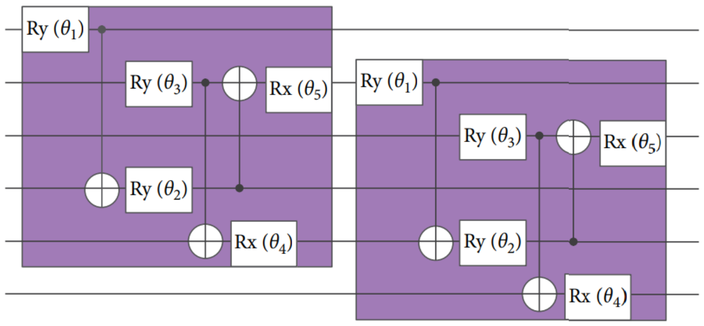


Figure 4: Example of a parametrized quantum circuit with 6 qubits [8].

define a scalar cost for a given task. The free parameters $\theta = (\theta_1, \theta_2, \ldots)$ of the circuit(s) are tuned to optimize this cost function.
Usually classical optimization algorithms are used to make queries to the quantum device. The optimization is typically an iterative scheme that searches for better candidates for the parameters θ with every step.

2 MACHINE LEARNING FOR MRI IMAGE CLASSIFICATION

2.1 INTRODUCTION TO CONVOLUTIONAL NEURAL NETWORKS (CNNS)

2.1.1 *Overall Architecture*

CNNs are based on four types of layers: input, convolutional, pooling and fully-connected [6]. When these layers are stacked, a CNN architecture has been formed.
1. The input layer holds the pixel values of the image, similar to other types of Artificial Neural Networks (ANN).
2. The convolutional layer processes the output of neurons connected to localized regions of the input by computing the scalar product between their weights and the corresponding region within the input volume. The rectified linear unit (ReLU) is often utilized as an 'elementwise' activation function applied to the output activation from the preceding layer.
3. The pooling layer performs downsampling along the spatial dimensionality of the input, further reducing the number of parameters within that activation.
4. The fully-connected layers conduct similar tasks as standard ANNs, aiming to produce class scores from the activations for classification purposes. ReLu activation may also be used between these layers to enhance performance.
In Figure 5, we observe how convolutional and downsampling techniques, applied layer by layer, transform the original input in CNNs, ultimately enabling the generation of class scores for classification and regression tasks.

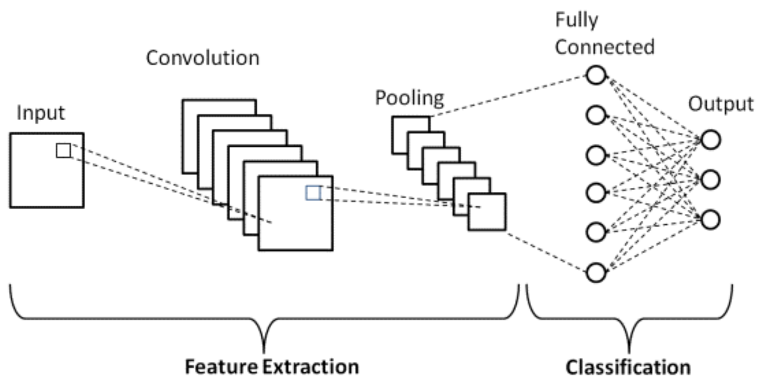


Figure 5: A simple CNN architecture [15].

2.1.2 *Convolutional Layer*

Convolution, as shown in Figure 6, harnesses three fundamental principles that can enrich a machine learning system: sparse interactions, parameter sharing, and equivariant representations. In convolutional networks, sparse interactions are typically achieved by employing kernels smaller than the input. For instance, when processing an image with thousands or even millions of pixels, kernels detecting small yet meaningful features such as edges occupy only tens or hundreds of pixels. Consequently, this approach requires fewer parameters, thereby reducing the model's memory demands and enhancing its statistical efficiency.
Parameter sharing involves utilizing the same parameter for multiple functions within a model, also known as tied weights. This implies that the weight value applied to one input is linked to a weight value applied elsewhere. In a convolutional neural network, every element of the kernel is utilized at each position of the input image.
This form of parameter sharing endows the convolutional layer with a property known as equivariance to translation. Equivariance implies that if the input changes, the output changes in a corresponding manner. For instance, when processing time-series data, convolution generates a timeline illustrating when different features appear in the input. Similarly, in images, convolution creates a 2D map depicting where specific features emerge in the input. If the object in the input is moved, its representation will shift by the same amount in the output.

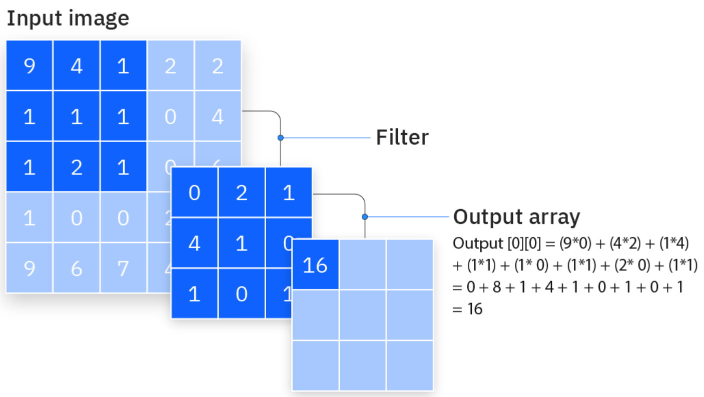


Figure 6: Example of convolution with kernel 3x3 [12].

2.1.3 *Pooling Layer*

The pooling layer plays a crucial role in convolutional neural networks by reducing the spatial dimensions of the input volume. Its main functions include:
1. Dimensionality Reduction: By downsampling the input, the pooling layer reduces the number of parameters and computations in the network. This helps in controlling overfitting and improving the computational efficiency of the model.
2. Translation Invariance: Pooling helps in creating features that are invariant to small translations in the input data. By aggregating nearby features, the network becomes less sensitive to small variations in the input, thus improving the generalization capability of the model.
3. Feature Learning: Pooling helps in capturing the most important features while discarding less relevant information. By selecting the maximum (max pooling) or average (average pooling) value within a certain region, the pooling layer retains the most salient features of the input.
4. Spatial Hierarchy: Pooling creates a spatial hierarchy in the network by progressively reducing the spatial dimensions of the input volume. This helps in learning hierarchical features at different levels of abstraction, leading to more discriminative representations.
Overall, the pooling layer plays a crucial role in reducing the spatial dimensions of the input, controlling overfitting, improving translation invariance, and learning hierarchical features, thereby contributing to the overall effectiveness of convolutional neural networks in various

computer vision tasks. One of the most common pooling operation is Max Pooling, we use it to extract maximum value from the Feature Map according to filter size and strides.

2.1.4 *Fully-connected Layer*

Fully connected layers, or dense layers, in CNNs aggregate features, introduce non-linearity with activation functions, and perform tasks like classification or regression. They contain trainable parameters adjusted during training to improve network performance. These layers are crucial for learning complex patterns and relationships in the input data, making them essential in CNN architectures for various machine learning tasks.

2.2 APPLICATION OF CNNS IN MRI IMAGE CLASSIFICATION

The dataset utilized in this study consists of 253 MRI images of human brains, specifically aimed at tumor detection [13]. Convolutional Neural Networks (CNNs) are particularly effective for classifying grid-like data such as MRI images. To benchmark the performance of the hybrid quantum-classical model discussed in the following sections, a straightforward CNN model was implemented. The architecture of this model is depicted in Figure 7 and comprises three convolutional layers, each followed by a pooling layer. These layers feed into a flatten layer that converts the input into a suitable format for the final dense layer, where the actual classification occurs. Training was conducted over 20 epochs to ensure adequate learning and performance evaluation. The evaluation of the performance of the model is based on one metric: the F1-score, that is the harmonic mean of precision and recall. Precision measures the accuracy of the positive predictions and recall measures the model's ability to correctly identify all relevant instances. The model employs the Adam optimizer with a learning rate of 0.001. After extensive trials with various kernel sizes, a kernel size of 5x5 achieved an F1-score of 0.866 and so was selected for its slightly superior performance, as displayed in Table 1.

Kernel size	**F1-score**
3x3	0.84140952
5x5	0.86615826
7x7	0.79696935
10x10	0.762767

Table 1: Average F1-score for different kernel sizes

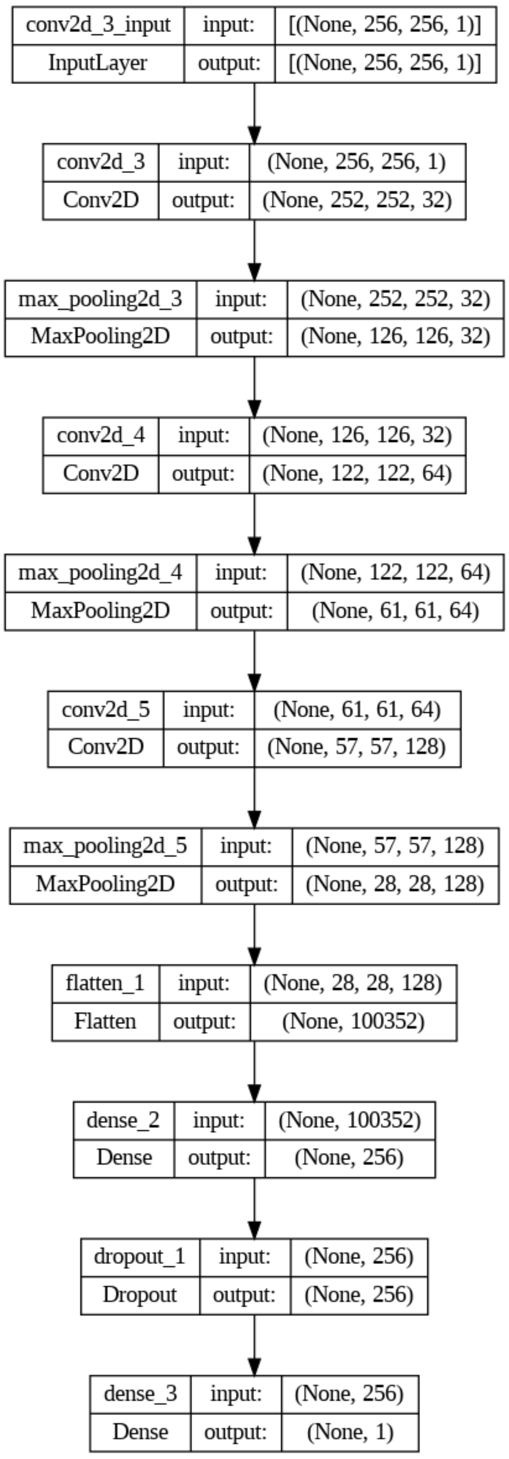

conv2d_3_input | input: | [(None, 256, 256, 1)]
InputLayer | output: | [(None, 256, 256, 1)]
conv2d_3 | input: | (None, 256, 256, 1)
Conv2D | output: | (None, 252, 252, 32)
max_pooling2d_3 | input: | (None, 252, 252, 32)
MaxPooling2D | output: | (None, 126, 126, 32)
conv2d_4 | input: | (None, 126, 126, 32)
Conv2D | output: | (None, 122, 122, 64)
max_pooling2d_4 | input: | (None, 122, 122, 64)
MaxPooling2D | output: | (None, 61, 61, 64)
conv2d_5 | input: | (None, 61, 61, 64)
Conv2D | output: | (None, 57, 57, 128)
max_pooling2d_5 | input: | (None, 57, 57, 128)
MaxPooling2D | output: | (None, 28, 28, 128)
flatten_1 | input: | (None, 28, 28, 128)
Flatten | output: | (None, 100352)
dense_2 | input: | (None, 100352)
Dense | output: | (None, 256)
dropout_1 | input: | (None, 256)
Dropout | output: | (None, 256)
dense_3 | input: | (None, 256)
Dense | output: | (None, 1)

Figure 7: **Architecture of the CNN model used for benchmarking.** The model consists of 3 convolutional layers, each followed by a pooling layer. At the end, there is a flatten layer to convert the 3D outputs to a 1D vector, followed by a dense layer, a dropout layer to reduce overfitting, and a final dense layer that produces the model's predictions.

3

IMPLEMENTATION OF AN HYBRID QUANTUM-CLASSICAL ML MODEL FOR IMAGE CLASSIFICATION

3.1 DESCRIPTION OF QCNN FOR MRI IMAGE CLASSIFICATION

3.1.1 *Introduction to QCNN*

Convolutional Neural Networks are extensively employed in computer vision tasks, yet they grapple with issues such as high computational complexity and susceptibility to overfitting. In pursuit of enhanced performance, researchers are delving into the realm of quantum computing to expedite both training and inference processes. However, the practical realization of Quantum Convolutional Neural Networks (QCNNs) demands substantial advancements in both hardware and software, given the early stage of quantum computing technology [21].

3.1.2 *Overall Architecture*

We can divide the model illustrated in Figure 8 in three parts: the encoding, the quantum circuit and the measurements. A localized portion of the input image, for instance, a 2x2 square, is embedded into a quantum circuit. This involves applying parametrized rotations to qubits initialized in the ground state. A quantum computation, represented by a unitary operator U, is executed on the system. This unitary could be a variational quantum circuit or a random circuit. The quantum system undergoes measurement, yielding a set of classical expectation values. These results will be subjected to a classical machine learning model. Mirroring a classical convolution layer, each expectation value corresponds to a distinct channel of a single output pixel. Repeating this process across various regions enables the scanning of the entire input image, generating an output structured as a multi-channel image. Following quantum convolution, classical layers will be applied. The primary distinction from a classical convolution lies in the potential of a quantum circuit to generate highly intricate kernels, the computational complexity of which could be, in principle, classically intractable.

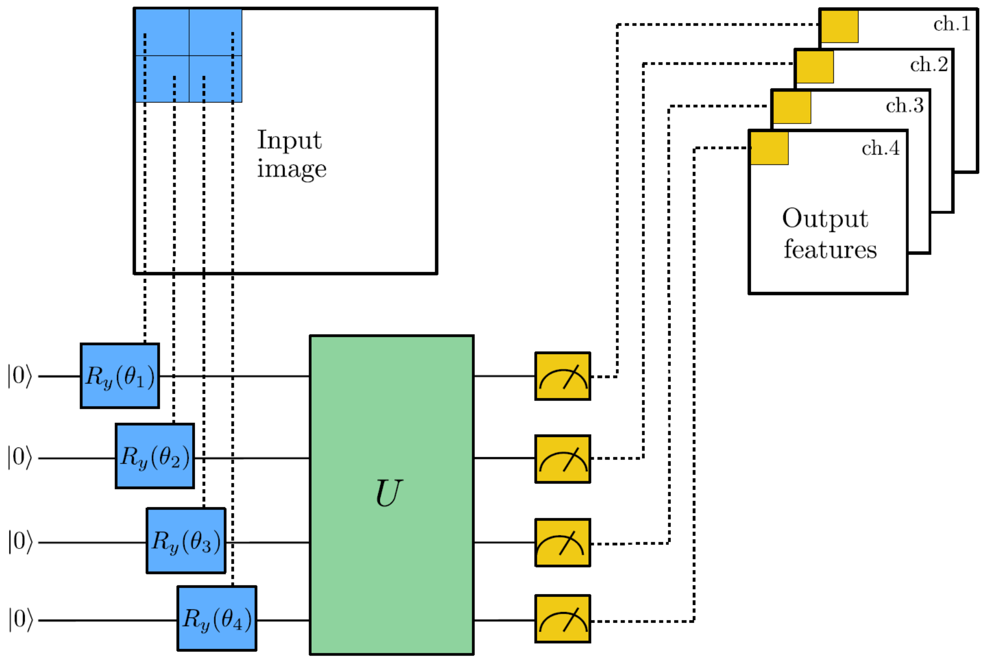


Figure 8: Example of the Quantum Convolution [10].

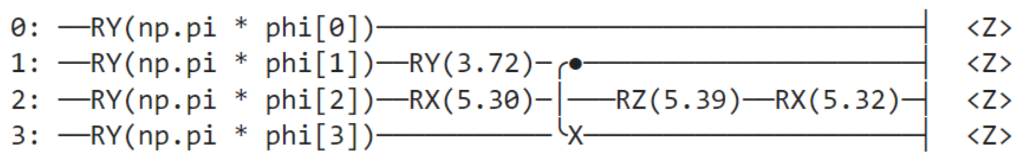
3.2 CONVOLUTION USING A QUANTUM CIRCUIT

The Python code I wrote, outlined in Listing 1, implements the quantum circuit shown in Figure 9 using the Pennylane library. It starts by defining parameters `kernel_size` and `n_layers`, representing the size of the kernel and the number of layers of random quantum gates applied to the qubits, respectively. A matrix of random parameters, `rand_params`, is generated. A quantum device is then set up using the "default.qubit" backend, with a number of qubits equal to `kernel_size`^{2}. The function `circuit`, decorated as a Pennylane qnode, takes an input `phi`, that contains the value of the 2x2 pixels, and encodes it into the qubits using RY gates. A layer of random quantum gates is applied using the `RandomLayers` template.

Finally, the expectation value of the Pauli-Z operator is measured for each qubit, returning an array of classical values.

Listing 1: Quantum Circuit using Pennylane

```
import numpy as np
import pennylane as qml
from pennylane.templates import RandomLayers

# Parameters
kernel_size = 2
n_layers = 1
rand_params = np.random.uniform(high=2 * np.pi, size=(n_layers,
    kernel_size**2))

# Device setup
dev = qml.device("default.qubit", wires=kernel_size**2)

@qml.qnode(dev)
def circuit(phi):
   # Encoding of kernel_size classical input values
   for j in range(kernel_size**2):
      qml.RY(np.pi * phi[j], wires=j)
   # Quantum circuit
   RandomLayers(rand_params, wires=list(range(kernel_size**2)))
   # Measurement producing kernel_size classical output values
   return [qml.expval(qml.PauliZ(j)) for j in range(kernel_size**2)]
```



```
0: ──RY(np.pi * phi[0])──────────────────────────────────┤  <Z>
1: ──RY(np.pi * phi[1])──RY(3.72)─╭●─────────────────────┤  <Z>
2: ──RY(np.pi * phi[2])──RX(5.30)─│──RZ(5.39)──RX(5.32)──┤  <Z>
3: ──RY(np.pi * phi[3])───────────╰X─────────────────────┤  <Z>
```

Figure 9: 2x2 Kernel and 1 Layer quantum circuit.

When the circuit is ready, I can perform 4 measurements on the qubits to generate 4 new processed images for each input image, as displayed in Figure 10. This process reduces the dimensions of the input image by half. I employ the function ‘quanv(image)‘ to iterate over the entire image, performing convolutions at each step, and returning an image of dimensions 128x128x4. The pre-processed images are then saved to avoid repeating this process again.

This process enhances anomalies in the images, such as tumors in this case, making it easier for the classical model following to train on these images.

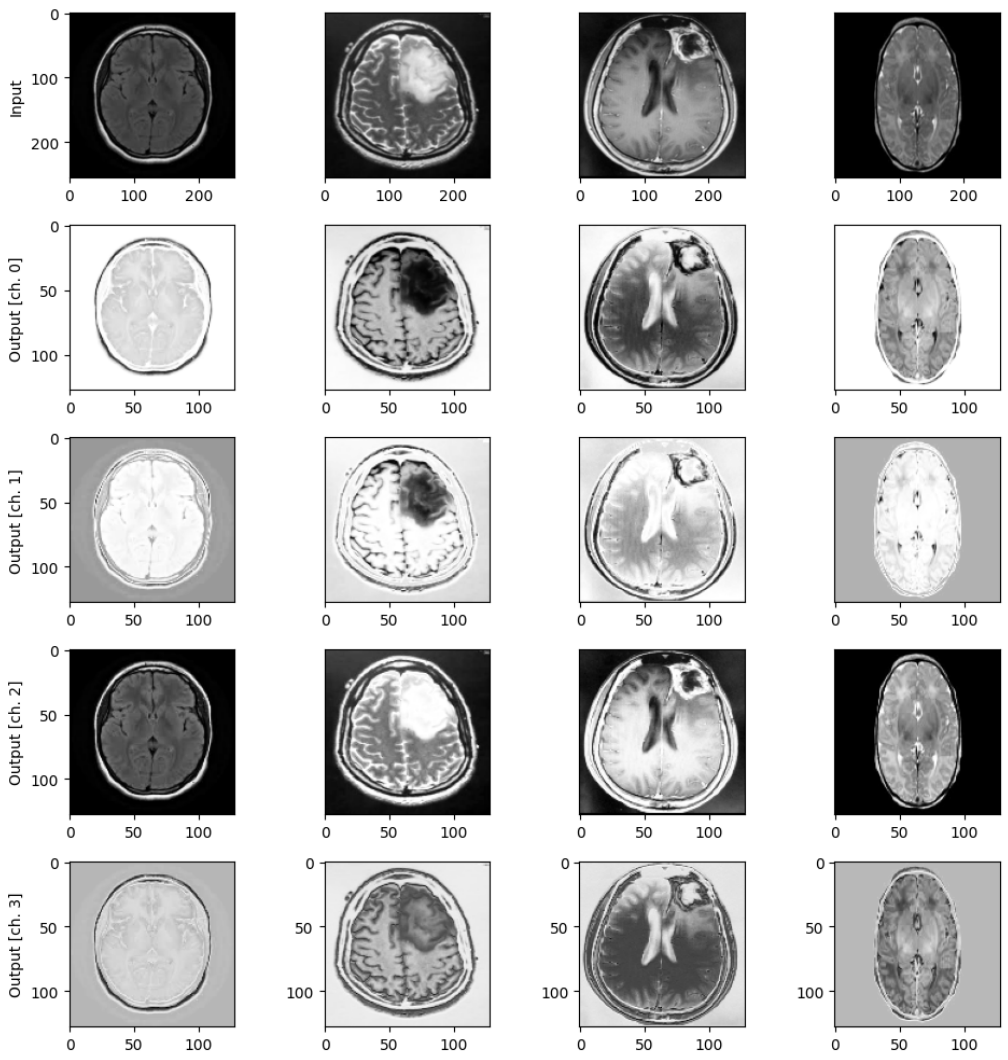


Figure 10: **Example of feature extraction using quanvolution.** The top row shows the original images, while the four rows below display the convolved images obtained from measurements of the quantum circuit.

4

EXPERIMENTAL RESULTS

4.1 COMPARISON OF QCNN PERFORMANCE WITH DIFFERENT CIRCUIT DESIGNS

To obtain an optimal model, I divided the dataset images, originally organized as shown in Table 2, into training, testing, and validation sets, consisting of 181, 26, and 46 images, respectively. Additionally, I converted them from being 3-channel RGB images to grayscale and then normalized them. Normalization involves dividing each pixel of every image by the maximum value of 255, thus ensuring that the values of the images are scaled between 0 and 1. Data normalization helps stabilizing the model training process by reducing sensitivity to variations in input data values. If not specified otherwise the preparation of the images before applying the quantum circuit is always the same.

	Number of samples	Number of classes
Brain MRI	253	2

Table 2: Dataset shape.

4.1.1 *Random Layers*

Random layers involve applying a series of randomly chosen quantum gates to the qubits. This method helps in exploring a wide range of quantum states, extracting a diverse set of features from the images.

The performance of the circuit described in the previous chapter and shown in Figure 9, composed of a single layer of randomly applied gates, is: F1-score = 0.8545.

This next circuit, displayed in Figure 11, executes convolution with a 3x3 kernel on original images, utilizing 9 qubits for operations at this kernel size. It comprises three main components: embedding gates, three layers of randomly distributed and parameterized rotational and entangling gates, followed by measurement. The performance of this circuit is F1-Score = 0.8816.

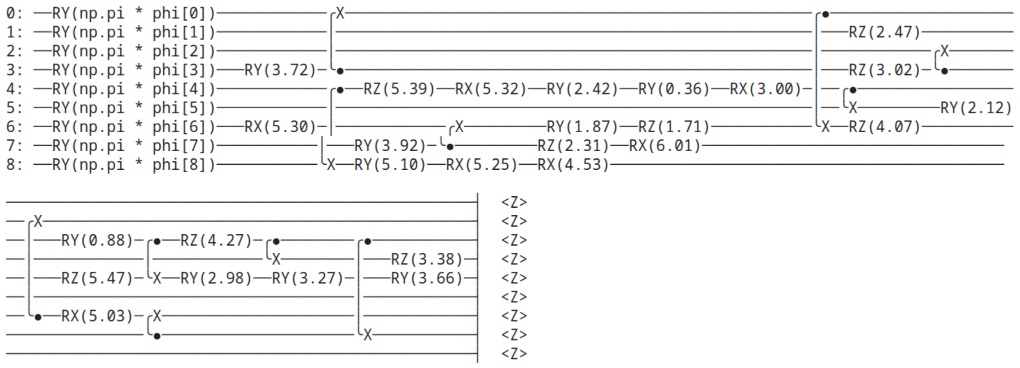

0: RY(np.pi * phi[0])
1: RY(np.pi * phi[1])
2: RY(np.pi * phi[2])
3: RY(np.pi * phi[3]) RY(3.72)
4: RY(np.pi * phi[4]) RZ(5.39) RX(5.32) RY(2.42) RY(0.36) RX(3.00)
5: RY(np.pi * phi[5])
6: RY(np.pi * phi[6]) RX(5.30) RY(1.87) RZ(1.71)
7: RY(np.pi * phi[7]) RY(3.92) RZ(2.31) RX(6.01)
8: RY(np.pi * phi[8]) RY(5.10) RX(5.25) RX(4.53)
RZ(2.47)
RZ(3.02)
RY(2.12)
RZ(4.07)
RY(0.88) RZ(4.27)
RZ(3.38)
RZ(5.47) RY(2.98) RY(3.27) RY(3.66)
RX(5.03)
<Z>

Figure 11: Illustration of a 3 layers circuit for convolution with a 3x3 kernel on original images.

Since this circuit uses 9 qubits, there will be 9 different output channels for every image, one for each qubit, as illustrated in Figure 12.

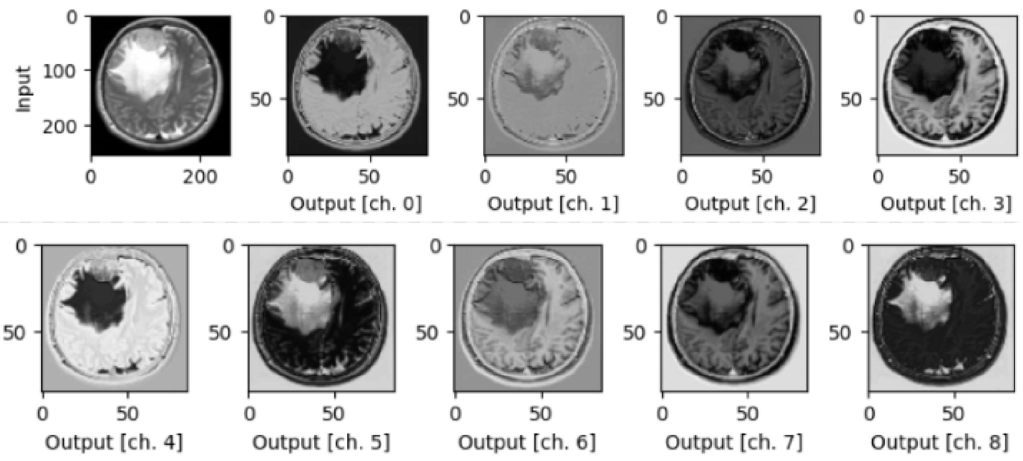
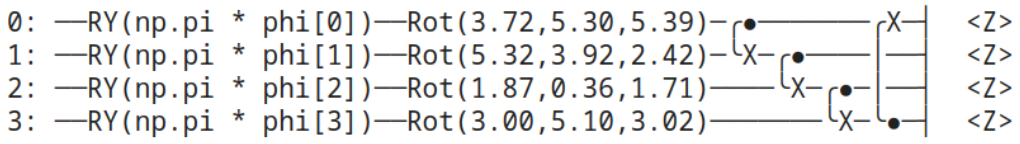


Figure 12: These images represent the before and after of the application of the quantum circuit shown in Figure 11. The image in the top left is the original one, and the other 9 are the output channels.

4.1.2 *Strongly Entangling Layers*

Strongly entangling layers are layers consisting of single qubit rotations and entanglers, inspired by the circuit-centric classifier design in [17]. The following circuit, illustrated in Figure 13, is composed of a single layer of randomly applied entangling gates. The use of 4 qubits simulates the convolution with a 2x2 kernel and generates the images in Figure 14.



```
0: ──RY(np.pi * phi[0])──Rot(3.72,5.30,5.39)─╭●───────╭X─┤  <Z>
1: ──RY(np.pi * phi[1])──Rot(5.32,3.92,2.42)─╰X─╭●────│──┤  <Z>
2: ──RY(np.pi * phi[2])──Rot(1.87,0.36,1.71)────╰X─╭●─│──┤  <Z>
3: ──RY(np.pi * phi[3])──Rot(3.00,5.10,3.02)───────╰X─╰●─┤  <Z>
```

Figure 13: A 2x2 Kernel and single Layer circuit.

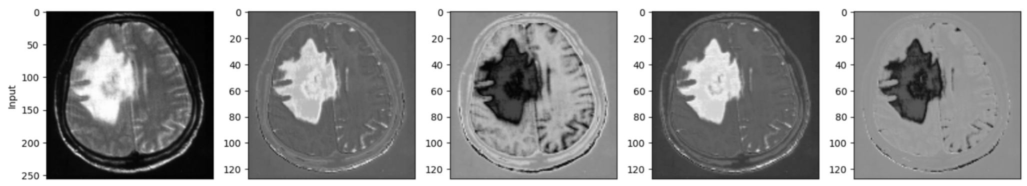
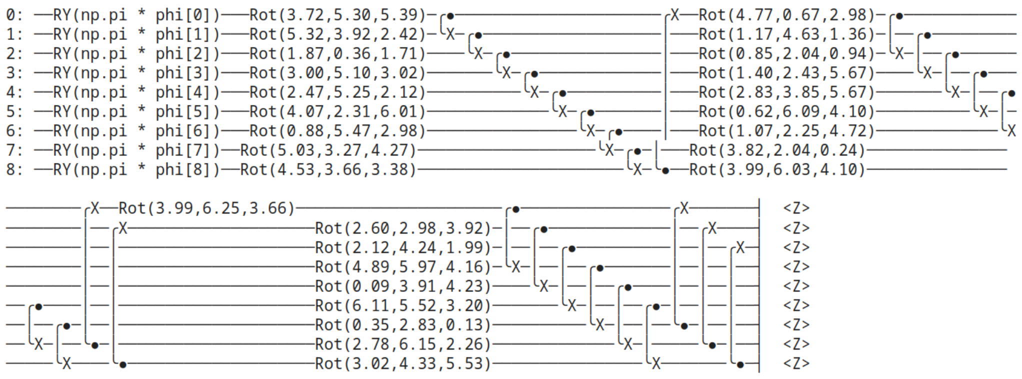


Figure 14: From the left: the original input image and its four convolved images.

The F1-score obtained using this quantum circuit for the first convolution is: 0.8358. The second circuit, depicted in Figure 15, is the most complex; it consists of 9 qubits, forming a 3x3 kernel, and 3 layers of strongly entangling gates. The idea is to make the output pixels more dependent on the adjacent ones.



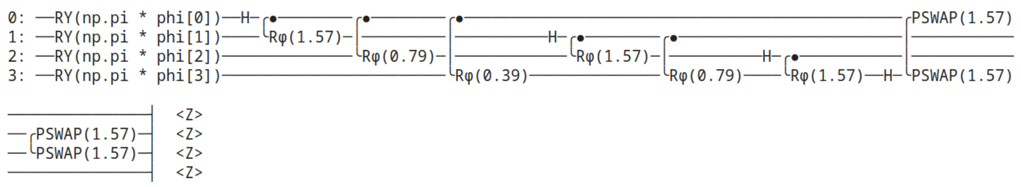
```
0: ──RY(np.pi * phi[0])──Rot(3.72,5.30,5.39)─╭●──────────────╭X──Rot(4.77,0.67,2.98)─╭●──────────
1: ──RY(np.pi * phi[1])──Rot(5.32,3.92,2.42)─╰X─╭●───────────│───Rot(1.17,4.63,1.36)─│──╭●───────
2: ──RY(np.pi * phi[2])──Rot(1.87,0.36,1.71)────╰X─╭●────────│───Rot(0.85,2.04,0.94)─╰X─│──╭●────
3: ──RY(np.pi * phi[3])──Rot(3.00,5.10,3.02)───────╰X─╭●─────│───Rot(1.40,2.43,5.67)────╰X─│──╭●─
4: ──RY(np.pi * phi[4])──Rot(2.47,5.25,2.12)──────────╰X─╭●──│───Rot(2.83,3.85,5.67)───────╰X─│─╭●
5: ──RY(np.pi * phi[5])──Rot(4.07,2.31,6.01)─────────────╰X─╭●│──Rot(0.62,6.09,4.10)──────────╰X│─
6: ──RY(np.pi * phi[6])──Rot(0.88,5.47,2.98)────────────────╰X╭●─Rot(1.07,2.25,4.72)────────────╰X
7: ──RY(np.pi * phi[7])─Rot(5.03,3.27,4.27)───────────────────╰X─╭●─│──Rot(3.82,2.04,0.24)────────
8: ──RY(np.pi * phi[8])─Rot(4.53,3.66,3.38)──────────────────────╰X─╰●─Rot(3.99,6.03,4.10)────────

───────╭X──Rot(3.99,6.25,3.66)─────────────────────╭●──────────────────╭X────────┤  <Z>
───────│──╭X───────────────Rot(2.60,2.98,3.92)─│──╭●───────────────│──╭X─────┤  <Z>
───────│──│────────────────Rot(2.12,4.24,1.99)─│──│──╭●────────────│──│──╭X──┤  <Z>
───────│──│────────────────Rot(4.89,5.97,4.16)─╰X─│──│──╭●─────────│──│──│───┤  <Z>
───────│──│────────────────Rot(0.09,3.91,4.23)────╰X─│──│──╭●──────│──│──│───┤  <Z>
─╭●────│──│────────────────Rot(6.11,5.52,3.20)───────╰X─│──│──╭●───│──│──│───┤  <Z>
─│──╭●─│──│────────────────Rot(0.35,2.83,0.13)──────────╰X─│──│───╰●─│──│───┤  <Z>
─╰X─│──╰●─│────────────────Rot(2.78,6.15,2.26)─────────────╰X─│──────╰●─│───┤  <Z>
────╰X────╰●───────────────Rot(3.02,4.33,5.53)────────────────╰X────────╰●──┤  <Z>
```

Figure 15: 3x3 Kernel and 3 Layer Circuit.

The F1-score obtained using this quantum circuit for the first convolution is: 0.8141.

4.1.3 *Quantum Fourier Transform Inspired Circuit*

As the last circuit design, I decided to use an architecture inspired by the application of the Quantum Fourier Transform (QFT), as outlined in Figure 16.



```
0: ──RY(np.pi * phi[0])──H─╭●─────────╭●─────────╭●──────────────────────────────────────────────╭PSWAP(1.57)
1: ──RY(np.pi * phi[1])────╰Rφ(1.57)─│──────────│─────────H─╭●─────────╭●──────────────────────│───────────
2: ──RY(np.pi * phi[2])──────────────╰Rφ(0.79)─│───────────╰Rφ(1.57)─│────────H─╭●───────────│───────────
3: ──RY(np.pi * phi[3])────────────────────────╰Rφ(0.39)────────────╰Rφ(0.79)──╰Rφ(1.57)──H─╰PSWAP(1.57)

───────────────┤  <Z>
──╭PSWAP(1.57)─┤  <Z>
──╰PSWAP(1.57)─┤  <Z>
───────────────┤  <Z>
```

Figure 16: 2x2 Kernel and 2 Layer Circuit, yielding the images in Figure 17.

The Quantum Fourier Transform is the quantum analogue of the discrete Fourier transform [14]. It is a key component of many quantum algorithms, including Shor's algorithm for integer factorization and algorithms for quantum search and simulation. The QFT takes a vector of quantum amplitudes and transforms it into another vector of amplitudes by applying quantum interference operations. The quantum circuit implementing the QFT for convolution uses Hadamard gates and controlled phase gates. Each qubit undergoes a Hadamard transformation, $H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, followed by controlled phase shifts, $R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{i2\pi/2^k} \end{pmatrix}$. The QFT circuit for 4 qubits consists of applying these gates in sequence, creating a superposition that represents the frequency components of the input image. After applying the QFT, convolution is performed by manipulating these frequency components using quantum gates. This method leverages the efficiency of the QFT in processing and manipulating frequency information.

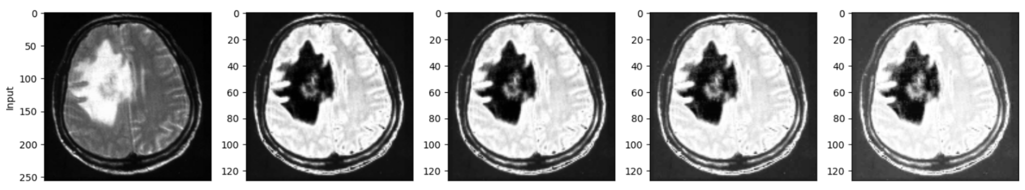


Figure 17: From the left: the original input image and its four convolved images.

The F1-score obtained using this quantum circuit as the first convolution is: 0.8325.

5 CONCLUSIONS AND FUTURE DEVELOPMENTS

5.1 SUMMARY OF MAIN CONCLUSION

In this thesis, we explored the application of Quantum Convolutional Neural Networks (QCNN) for MRI image classification to detect human brain tumors. By combining classical and quantum machine learning models, we demonstrated a potential application to enhance classification accuracy by leveraging the unique benefits of quantum computing. The final results are displayed in Table 3.

Circuit Design	**F1 Score**
2K1L RAN	0.8545
2k1L SEL	0.8358
3K3L RAN	0.8816
3K3L SEL	0.8141
2K2L QFT	0.8325

Table 3: Performance of the hybrid model with different circuits

Although the results didn't demonstrate a profound superiority of the hybrid model over the purely classical one, several peculiarities can be noted. Models employing numerous gates aimed at entangling qubits tend to produce convoluted images with less pronounced edges, which could explain why they don't seem to perform as well as models with circuits featuring a random distribution of gates.
Another interesting point is the training speed. Training the classical model with original-sized images takes approximately 2 minutes per epoch. In contrast, the same model, receiving input from images that have undergone quantum convolution, takes about 15 seconds per epoch. This comparison was naturally conducted using the same hardware.

5.2 FINAL CONSIDERATIONS OF THE IMPORTANCE AND PROSPECTS OF QML IN MEDICAL IMAGE ANALYSIS

Quantum Machine Learning (QML) holds significant promise for advancing medical image analysis. The ability of quantum systems to exist in superposition and perform parallel computations provides a substantial computational advantage, particularly for complex and high-dimensional data such as MRI images.
It would have been interesting to create a purely quantum model for medical image classification. The issue I encountered when attempting to implement it was the complexity of the images to be classified. The original images are 256x256, with numerous details, and the limitation of a quantum circuit's inputs, determined by the number of available qubits, made implementation unfeasible.
One approach I consider viable is as follows: applying filters to the original images to deal with smaller input dimensions. The filters would effectively crop the brain image, for instance, isolating just the two lobes, thus transforming it from a rectangular image to a more focused representation.
Then, performing PCA on these filtered data to transition from matrix-form inputs to a vectorized form, more easily manageable by a quantum circuit. Subsequently, training the circuit on these data.
In conclusion, while still in its early stages, QML presents a promising avenue for revolutionizing medical image analysis, offering the potential for more accurate and efficient diagnostic tools. Continued advancements in quantum computing and its applications in machine learning are expected to further unlock this potential, leading to significant improvements in healthcare outcomes.

BIBLIOGRAPHY

[1] Amira Abbas, David Sutter, Christa Zoufal, Aurelien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science*, 1:403–409, 2021.

[2] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549:195–202, 2017.

[3] Jianwei Chen, Weijian Yao, and Xuebin Zhang. Quantum convolutional neural networks for high energy physics data analysis. *Physical Review D*, 104(11):112001, 2021.

[4] W. Dür and Stefan Heusler. What we can learn about quantum physics from a single qubit. *Nature Physics*, 9:143–146, 2013.

[5] Ethan N. Evans, Dominic Byrne, and Matthew G. Cook. A quick introduction to quantum machine learning for non-practitioners, 2024.

[6] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[7] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional neural networks: Powering image recognition with quantum circuits, 2019.

[8] Wei Li, Peng-Cheng Chu, Guang-Zhe Liu, Yan-Bing Tian, Tian-Hui Qiu, and Shu-Mei Wang. An image classification algorithm based on hybrid quantum classical convolutional neural network. *Quantum Engineering*, 2022:5701479, 2022.

[9] Shuai Lu, He-Liang Huang, Shao-Feng Li, Dong Li, Yangsheng Zhang, and Zhen-Biao Zhang. Quantum machine learning on near-term quantum devices: A survey. *Science China Information Sciences*, 62(11):112201, 2019.

[10] Andrea Mari. Quanvolutional neural networks, 2020.

[11] MathWorks. Deep learning - mathworks italia. https://it.mathworks.com/discovery/deep-learning.html, s.d.

[12] Narmina. Convolutional neural networks, 2023.

[13] Navoneel. Brain mri images for brain tumor detection, 2023. Accessed: 2024-06-18.

[14] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*, volume 2. Cambridge University Press, 2001.

[15] Patole Gore Nikam, Survase. Lane detection and object detection, 2023.

[16] Josh Patterson and Adam Gibson. *Deep Learning: A Practitioner's Approach*. O'Reilly Media, Inc., 2017.

[17] Maria Schuld, Alex Bocharov, Krysta M. Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3), March 2020.

[18] Maria Schuld, Manuel Fingerhuth, and Francesco Petruccione. Introduction to quantum machine learning with python and qiskit. *Quantum Machine Intelligence*, 1(1-2):37–43, 2015.

[19] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, October 2014.

[20] UpGrad. Basic cnn architecture: Convolutional neural networks explained, 2024. Accessed: 2024-06-11.

[21] Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Academic Press, 2014.

[22] Peng Yin and Xing Chen. Quantum machine learning for medical image classification. *Quantum Information Processing*, 22(1):28, 2023.