



भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

QUANTUM MACHINE LEARNING AND QUANTUM NEURAL
NETWORKS FOR BEGINNERS
B.Tech. Project Report

Raghav Juyal*

EP20BTECH11018
Engineering Physics
Indian Institute of Technology, Hyderabad

Supervisor

Dr. Nithyanandan Kanagaraj

Department of Physics
Indian Institute of Technology, Hyderabad

Friday 28th April, 2023

*ep20btech11018@iith.ac.in

Contents

1	Introduction	4
2	Machine Learning	5
2.1	Supervised Learning	5
2.2	Unsupervised Learning	6
2.3	Reinforcement Learning	6
2.4	Neural Networks	7
3	Quantum Computing	7
3.1	Basic Concepts	7
3.1.1	Qubits	7
3.1.2	Entanglement	8
3.1.3	Products	8
3.1.4	Measurements	9
3.1.5	Gates	9
3.1.6	Observables	10
3.1.7	Quantum Circuits	11
3.2	Quantum Algorithms	11
3.2.1	Grover's Search Algorithm	11
3.2.2	Quantum Minimization Algorithm (GroverOptim)	12
3.2.3	SwapTest algorithm	13
3.2.4	DistCalc algorithm	13
4	Quantum Machine Learning and Neural Networks	13
4.1	Variational quantum classifiers	15
4.1.1	Background: Classification	15
4.1.2	Background: Variational circuits	15
4.1.3	Approach: Variational circuit as a classifier	16
4.1.4	Summary	17
4.2	Quantum k-means clustering	17

4.2.1	Background: k-means clustering	17
4.2.2	Approach: Quantum k-means algorithm	17
4.2.3	Summary	18
4.3	Support vector machines	18
4.3.1	Background: Support vector machines	18
4.3.2	Approach: Quantum support vector machines	19
4.3.3	Summary	19
4.4	Neural networks	19
4.4.1	Background: Neural Networks	19
4.4.2	Approach: Quantum Perceptron	20
4.4.3	Approach: Quantum Neural Networks	20
4.4.4	Summary	21
4.5	Convolutional neural networks	21
4.5.1	Background: Convolutional neural networks	21
4.5.2	Approach: Direct approach	22
4.5.3	Approach: Quantvolutional neural networks	22
4.5.4	Approach: Quantum convolutional networks	22
4.5.5	Summary	22
5	Conclusions	23
6	Future Works	23
7	Acknowledgements	23

ABSTRACT

In this project we aim to compile and present information on some quantum machine learning and quantum neural network algorithms and compare them to their classical counterparts. The algorithms are presented not in a heavy mathematical fashion but a more intuitive nature. This is to help beginners in this field gain a basic understanding of how these algorithms work and provide them with tools to get a better understanding of the original papers and about the field in general and how to apply these concepts. To this end we also discuss some machine learning and quantum computing concepts so that those who do not know much about these fields can gain a basic understanding of them to use in quantum machine learning and quantum neural networks.

1 Introduction

Artificial Intelligence is the imitation of human intelligence by machines. This has been a long sought after goal by inventors for a very long time. One of the interesting sub-fields of artificial intelligence is machine learning. In machine learning we try to make the programs learn by experience(similar to humans) without direct programming. These algorithms will then develop by themselves when exposed to new data. Machine Learning has been very useful in a wide variety of fields in science and society, especially for tasks involving finding patterns and regularities in data-sets. Neural networks are particular models of machine learning which have become very useful in recent times due to the abundance of data and improvement of hardware, as their accuracy increases much more than other models with the amount of data we have nowadays. From face recognition to language translators, there are a lot of useful tools created by neural networks. However, the training of large neural networks requires lots of time and computational power. Thus a lot of research is being done in finding ways to improve how we implement these neural networks, both the algorithms as well as the hardware we run them on.

Quantum computing has been on the rise recently. Quantum computing is a type of computation where we harness the power of quantum mechanics phenomena like superposition, interference and entanglement. We have seen that quantum computers can solve a certain set of problems faster than classical computers. Even though the quantum computers are in the experimental stage, there is a focus on trying to find more quantum algorithms which can outperform classical computers. This leads us to the naturally occurring question: Can we use quantum computing to speed up our machine learning models, or more specifically, can we speed up neural networks? This gives rise to the subjects of quantum machine learning and quantum neural networks. While this field seems very interesting, most of the articles are math intensive and require physics knowledge which would make them beginner unfriendly. This project aims to allow the readers to gain a basic intuition and appreciation of these algorithms and hopefully prepare them to then tackle the original papers or learn how to apply these algorithms.

2 Machine Learning

Machine learning is a field where we provide machines with the ability to learn automatically from past data. They are very helpful in automating task by finding patterns and regularities in data. We generally consider that data follows an arbitrary function $g(x)$ which we normally do not know. To predict future data we require knowledge of $g(x)$. We assume a prediction function $f(x)$ and by looking at the existing data we modify the parameters of $f(x)$ to make it as close to $g(x)$ as possible. An intuitive way to understand how machine learning algorithms work is by breaking it into the following parts.

1. Choose a model/function to predict the data.
2. Have a cost function(error function) which evaluates how accurate the prediction of the model is.
3. Optimize the model by changing the value of the weights(parameters) to better fit the prediction.
4. Repeat steps 2 and 3 till you are satisfied with the error or it goes no lower.

Choosing the initial model/function to optimize depends heavily on the type of data that is given. For example let us assume our data follows $g(x)$ (not known to us) and the function we choose to predict $g(x)$ is $f(x) = ax + b$ where x is our input and a and b are parameters that we need to tune so that $f(x)$ will eventually become $g(x)$ or close enough depending on our requirements. By following steps 2 and 3 we change the values of a and b so that our prediction $f(x)$ becomes close to the actual distribution $g(x)$. If $g(x)$ is a linear function, then $f(x)$ can eventually become $g(x)$ by changing the values of a and b . However, if $g(x)$ is some other function, say quadratic, $f(x)$ won't be able to accurately predict $g(x)$. This is why we should choose the algorithms depending on the data we are given.

There are 3 major types of machine learning models:

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

In this section we will briefly go over some general details of machine learning. We will focus more on the algorithms when we encounter them in the quantum machine learning and quantum neural network section.

2.1 Supervised Learning

In supervised learning, we provide the model with input and the expected output. The model then tries to create a mapping equation based on them and will use that mapping equation to predict the outputs for future inputs. There are 2 types of supervised learning algorithms:

- **Regression:** Used for predicting continuous variables
- **Classification:** Used for predicting discrete(categorical) variables

Some supervised learning models are

- Linear Regression
- Logistic Regression
- Decision Trees
- Support Vector Machines
- K-nearest neighbours

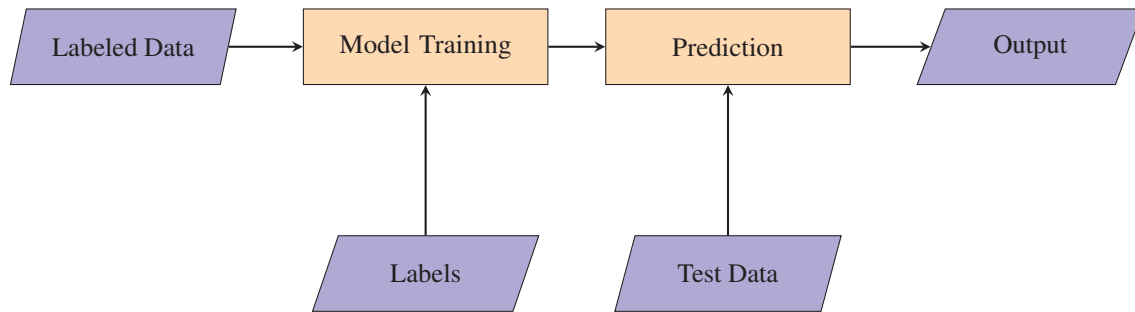


Figure 1: Working of supervised learning algorithms

2.2 Unsupervised Learning

In unsupervised learning, we provide the model with only the inputs. The model tries to classify the input data into classes with similar features and tries to classify future inputs into these classes based on similarity to the class members. There are 2 types of unsupervised learning algorithms:

- **Clustering:** Used to group objects into clusters such that objects with most similarities are in the same group.
- **Association:** Used for finding relationships between variables in the data-set

Some unsupervised learning models are

- K-means Clustering
- Hierarchical Clustering
- Principal Component Analysis

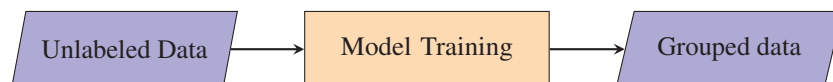


Figure 2: Working of unsupervised learning algorithms

2.3 Reinforcement Learning

In reinforcement learning, the model tries to take the best possible action in a given situation. It learns by getting feedback from the past outcomes. For example, if there were 2 paths A and B. The first time the model will choose anyone of the paths. The next time it gets into a similar situation, it will use the feedback to decide which path to go to.

In future sections of this text we will cover a few supervised and unsupervised learning algorithms.

2.4 Neural Networks

Neural networks are a specific type of machine learning algorithms that are defined by a set of individual processing units(neurons) and their interconnections. Each neuron receives a combined signal from the surrounding neurons, reacts to the signal non-linearly, and passes its own signal to subsequent neurons in the network. The interconnections transmit signals with different weights, and it is the choice of these weights that determine the network function. If these weights are engineered precisely enough, one can obtain powerful information processing systems for recognition and classification of patterns with imperfect, noisy or even incomplete information. Some neural network architecture are

- Convolutional Neural Networks
- Recurrent Neural Networks
- Transformers
- Autoencoders

3 Quantum Computing

In quantum computing we exploit quantum mechanical effect like superposition, interference and entanglement to perform our computation. We see that by harnessing the power of these effects we can solve some problems more efficiently than what can be done on classical computers. While it is still in it's experimental stage, there is a lot of research being done in this field.

3.1 Basic Concepts

In this section we will explain some of the basic concepts and common terms of quantum computing encountered by a typical programmer so that we can understand how the quantum machine learning and quantum neural network algorithms work. If one is interested in more details they can refer to these sources. [1, 2, 3].

3.1.1 Qubits

The qubit (quantum bit) is the fundamental unit of information that describes a two dimensional quantum system. It is represented as

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1)$$

Where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. In the ket notation, $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ so we can represent the state in (1) in vector notation as $|\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. On measuring a qubit, it collapses to a particular bit so the state of a qubit cannot be measured without changing it. On measuring the state in (1) we will either get the outcome as the classical value of zero ($|0\rangle$) with probability $|\alpha|^2$ or one ($|1\rangle$) with probability $|\beta|^2$.

Similarly to how this is done in two dimensions, the structure of a qubit can be generalized to higher dimensions of quantum states by representing them with a vector in complex vector space. Since the total probability of all outcomes of a measurement must sum to one, the state of a quantum system

is represented as a normalized vector in the complex vector space. When working with many qubits we should know how to find the combined state of a set of qubits. This is done by performing an operation called the tensor product \otimes which is mathematically equivalent to the Kronecker product of the two vectors. For $|\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ and $|\psi\rangle = \begin{pmatrix} \gamma \\ \delta \end{pmatrix}$ we have

$$|\phi\rangle \otimes |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta \end{pmatrix} \quad (2)$$

The tensor product can also be represented as $|\phi\rangle \otimes |\psi\rangle = |\phi\psi\rangle$. For higher dimension quantum systems it becomes easier to compute the product using the distributive property of the Kronecker product, i.e.,

$$|\phi\psi\rangle = |\phi\rangle \otimes |\psi\rangle = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle \quad (3)$$

We see that the measurement of both qubits can result in any of the four (2^2) possible bit-strings (associated with the four basis vectors). We can see that the dimension of the state space grows exponentially in the number of qubits n as (2^n).

Since quantum states follow superposition, the linear combination of any two quantum states, once normalized, is also a valid quantum state. This implies that we can write any n qubit system as the normalized linear combination of the 2^n bit-string states forming the computational basis.

3.1.2 Entanglement

One could see that in (3) we have described a quantum system which could be represented as the tensor product of other (in this case two) quantum states. However, there are states that cannot be represented as the tensor product of quantum states like

$$|\theta\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad (4)$$

Such states that cannot be represented as the tensor product of individual quantum states are called entangled states. This is a feature of quantum mechanics that cannot be seen in classical mechanics. By exploiting the existence of entangled states, we can create powerful algorithms that create a 2^n complex vector space to perform computations in using only n qubits.

3.1.3 Products

Some of the linear algebra concepts necessary for understanding these algorithms include inner and outer products. The overlap between two states is given by the inner product between the two vectors. For $|\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ and $|\psi\rangle = \begin{pmatrix} \gamma \\ \delta \end{pmatrix}$ we have the inner product given as

$$\langle\psi|\phi\rangle = \gamma^*\alpha + \delta^*\beta \quad (5)$$

where $*$ represents complex conjugate. One should note that $\langle\psi|\phi\rangle = \langle\phi|\psi\rangle^*$. The inner product of a state with a bit-string state gives the corresponding coefficient. Another way to look at the inner products is by considering the bra state. The bra state corresponding to $|\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ is $\langle\phi| = (\alpha^* \ \beta^*)$. Thus the inner product is simply the matrix product of the bra and ket.

The outer product can be defined as $|\psi\rangle\langle\phi|$. Using this notation we can write any matrix as a linear combination of outer products of bit-string states.

$$A = \begin{pmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{pmatrix} = A_{00} |0\rangle\langle 0| + A_{01} |0\rangle\langle 1| + A_{10} |1\rangle\langle 0| + A_{11} |1\rangle\langle 1| \quad (6)$$

While this representation may seem tedious, it makes algebraic manipulation of quantum states much more easy to understand.

3.1.4 Measurements

Measurement is when we transform the quantum information stored in the quantum system into classical information like when we measure a qubit, we typically get a classical bit(0 or 1). In quantum mechanics, the outcomes of measurement are probabilistic. The probability of getting the bit-string $|x_1 \dots x_n\rangle$ as the outcome after measuring an n qubit state $(|\phi\rangle)$ is $|\langle x_1 \dots x_n | \phi \rangle|^2$.

3.1.5 Gates

A system of qubits changes its state by undergoing a series of unitary transformation. A matrix U is unitary if

$$UU^\dagger = U^\dagger U = I \quad (7)$$

where U^\dagger is the Hermitian conjugate of U (transposed complex conjugate) and I is the identity matrix. A qubit state evolves under U as

$$|\phi\rangle \rightarrow U|\phi\rangle = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} U_{00}\alpha + U_{01}\beta \\ U_{10}\alpha + U_{11}\beta \end{pmatrix} \quad (8)$$

Operators acting on different qubits can be combined using the Kronecker product, i.e., if U_1 and U_2 are acting on 2 different qubits, then the full operator is given by $U_1 \otimes U_2$. The transformations allowed on an n qubit system include all $2^n \times 2^n$ unitary matrices. While the size of a general transformation increases exponentially with the number of qubits, in practice a transformation on n qubits is usually a combination of unitary transformation which act on 1 or 2 qubits at a time.

Similar to how classically we build complex circuits using AND and NOT gates, here using basic unitary transformations we build more complicated transformations called gates. The gates are unitary transformations. The number of input and output qubits must be same. The gates must also be reversible (as for gate U there is a gate U^\dagger that undoes the transformation). So the classical NOT gate is an example of a reversible gate as it maps 0 to 1 and 1 to 0 while the classical AND gate is not reversible as we cannot reconstruct the inputs from the outputs. A set of gates that can execute all possible quantum computations is called a universal gate set.














Operator	Gate(s)	Matrix
Pauli-X (X)	 	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Figure 3: Commonly used quantum gates

3.1.6 Observables

The experiments in quantum mechanics are probabilistic however we often need to associate a real number with the measurement outcome so the quantities we measure will be statistical averages of these numbers. If we associate the value 1 when the measurement outcome is state $|0\rangle$ and the value -1 when the outcome is state $|1\rangle$, the average value (expectation value) for state $|\phi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ is $|\alpha|^2 - |\beta|^2$. Using certain operators there is a neat way to represent such experiments in quantum formalism. Here the associated operator is the Z gate

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (9)$$

We can then write the average outcome of the experiment as the overlap between $|\phi\rangle$ and $Z|\phi\rangle$,

$$\langle\phi|Z|\phi\rangle = \langle\phi|0\rangle\langle 0|\phi\rangle - \langle\phi|1\rangle\langle 1|\phi\rangle = |\alpha|^2 - |\beta|^2 \quad (10)$$

Here the operator Z is called the observable and the quantity $\langle Z \rangle = \langle\phi|Z|\phi\rangle$ is the expectation value. This can be extended to measurements that are not on the computational basis as well.

3.1.7 Quantum Circuits

Quantum algorithms are often represented as circuits. In the circuit representation, qubits are represented by horizontal lines. Gates are drawn on the qubits they act on. This is done in sequence from left to right. The initial state of the qubit is denoted at the beginning of each line. The measurement of a qubit is denoted by the gate with a meter on it. Below we can see an example of quantum circuit.

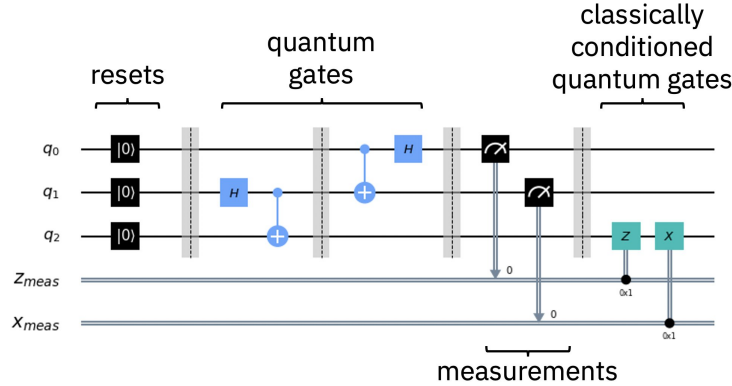


Figure 4: Quantum circuit

3.2 Quantum Algorithms

A quantum algorithm consists of three basic steps:

1. Encode the data (classical or quantum) into the state of a set of input qubits.
2. A sequence of quantum gates are applied to this set of input qubits.
3. Measure one or more of the qubits at the end to obtain a classically interpretable result.

In this section we will be discussing the basic ideas the quantum algorithms that will be used as subroutines in the quantum machine learning and quantum neural network algorithms. For more details on quantum algorithms and their implementation, refer to [1, 4]

3.2.1 Grover's Search Algorithm

Grover's algorithm is a quantum search algorithm which runs quadratically faster ($O(\sqrt{2^n})$) than any classical counterpart ($O(2^n)$). It's aim is to search through function inputs to check whether the function return true or false. This function is represented as a quantum oracle which has already been given the algorithm and acts as a black box.

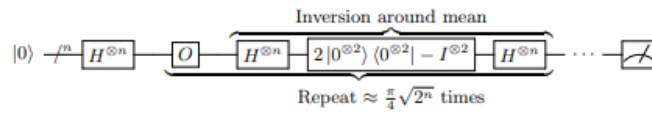


Figure 5: Circuit for Grover's algorithm [4]

We use the following steps to implement Grover's algorithm:

1. Initialize n qubits resulting in state $|\psi_0\rangle = |0 \dots 0\rangle$
2. Apply Hadamard gates to get state $|\psi\rangle = \frac{1}{\sqrt{2^n}}(|0 \dots 0\rangle + |0 \dots 1\rangle + \dots + |1 \dots 1\rangle)$
3. Apply the oracle which will flip the sign of the amplitude of the qubit which returns true to the function
4. Invert the amplitudes of all the qubits about the average of their amplitudes
5. Repeat steps 3 and 4 about $\frac{\pi}{4} \sqrt{2^n}$ times

The algorithm can be understood as amplitude amplification as that is what happens when we change the sign of the matching qubit and then invert about the mean.

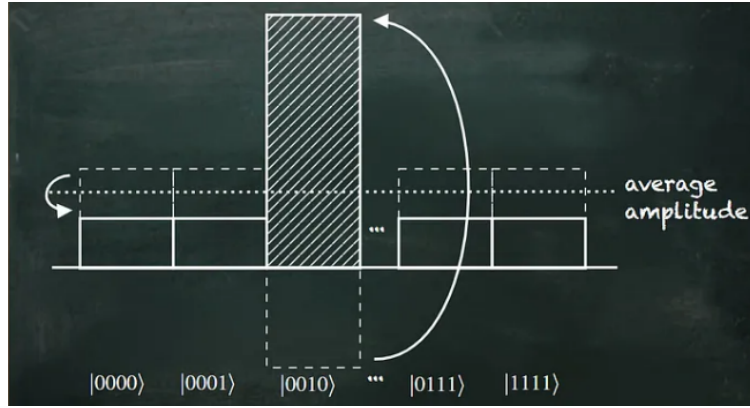


Figure 6: Amplitude amplification in Grover's algorithm [5]

3.2.2 Quantum Minimization Algorithm (GroverOptim)

Grover's algorithm can be extended to create the quantum minimization algorithm. Here we require the quantum oracle as well as the objective function $f(x)$ we require to minimize. We then do the following steps:

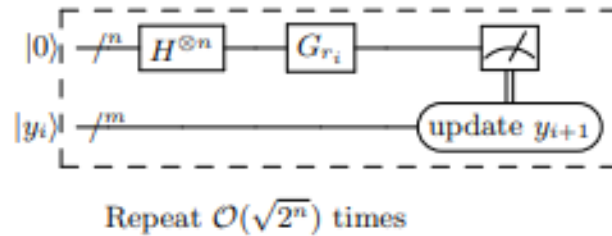


Figure 7: Circuit for Quantum minimization algorithm [4]

1. Randomly select input x_1 and set $y_1 = f(x_1)$
2. Initialize the first register with n inputs qubits and the second register with m qubits to store the threshold y_i
3. Apply Hadamard gates on the first register to represent all possible inputs of $f(x)$
4. Apply Grover's algorithm and using the output $|x\rangle$ of the algorithm calculate $y = f(x)$

5. Set new threshold by comparing the values of y and y_i . If $y < y_i$ then set $x_{i+1} = x$ and $y_{i+1} = y$ else set them to the previous values.
6. Repeat steps 2-5. Convergence is achieved in order $\sqrt{2^n}$ iterations.

3.2.3 SwapTest algorithm

SwapTest is a quantum algorithm which expresses the overlap of two states $\langle a|b \rangle$ in terms of the probability of the control qubit being in the state $|0\rangle$. The overlap can be used to in the calculation of the distance between classical vectors in DistCalc algorithm.

We use the following steps:

1. Initialize the two states $|a\rangle$ and $|b\rangle$ as well as the control bit $|0\rangle$
2. Apply Hadamard gate on the control qubit
3. Apply the controlled SWAP gate
4. Apply Hadamard gate on the control qubit
5. Measure the control qubit. We get the probability that the qubit is in state $|0\rangle = \frac{1}{2} + \frac{1}{2} |\langle a|b \rangle|^2$. If the probability that the qubit is in state $|0\rangle$ is 0.5 then the states $|a\rangle$ and $|b\rangle$ are orthogonal and if it is 1 then they are identical.

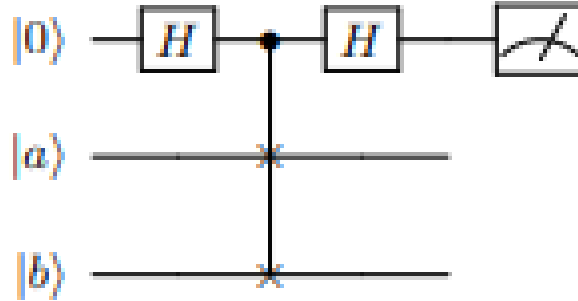


Figure 8: Circuit for SwapTest algorithm [4]

3.2.4 DistCalc algorithm

Here we calculate the Euclidean distance $|a - b|$ between the vectors a and b .

1. We first initialize the quantum states $|\psi\rangle = \frac{1}{\sqrt{2}}(|0, a\rangle + |1, b\rangle)$ and $|\phi\rangle = \frac{1}{\sqrt{|a|^2 + |b|^2}}(|a| |0\rangle + |b| |1\rangle)$
2. Use SwapTest to calculate the overlap $\langle \phi|\psi \rangle$
3. We can now calculate the distance using $\langle \phi|\psi \rangle = \frac{1}{\sqrt{2(|a|^2 + |b|^2)}}(a - b)$ which gives us $|\langle \phi|\psi \rangle|^2 = \frac{1}{2(|a|^2 + |b|^2)} |a - b|^2$

4 Quantum Machine Learning and Neural Networks

Quantum machine learning is the field where we try to use quantum algorithms to create machine learning models. We try this to find algorithms which can run faster than our current day machine

learning models while still retaining the accuracy. We can split machine learning based on the nature of the input and how we are performing the computation. This gives us:

- Classical data on Classical devices
- Classical data on Quantum devices
- Quantum data on Classical devices
- Quantum data on Quantum devices

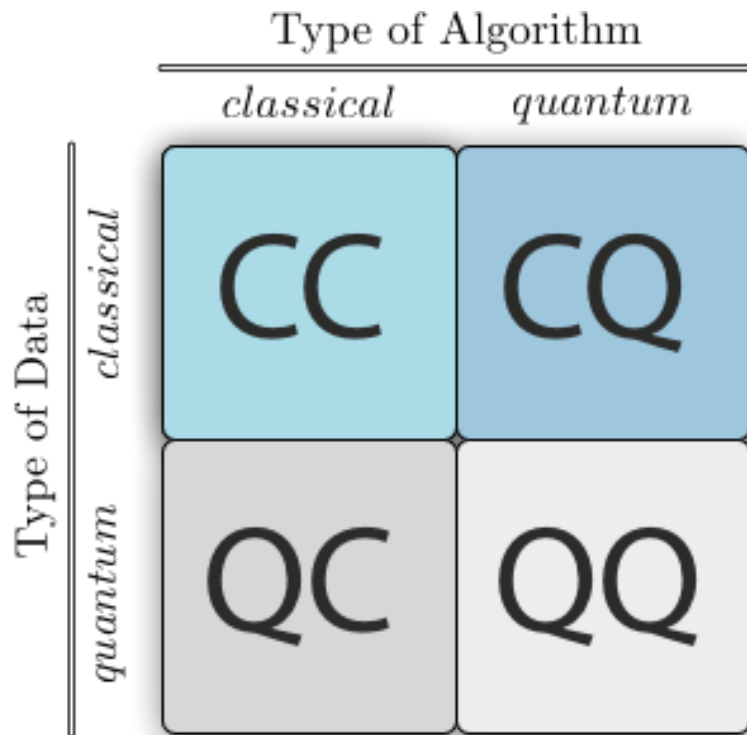


Figure 9: Types of algorithms [6]

Here we will be focusing on Classical data on Quantum devices. As mentioned in machine learning section, to train models we choose a model and train it on the data by checking the cost function and optimizing the parameters. Here we look for ways to use a quantum model or add some quantum parts to a classical model (making a hybrid model) to see if that can be advantageous in any way. In general we can split the quantum machine learning process into the following parts:

1. Encode the classical data into a quantum state.
2. Apply the model (series of gates)
3. Measure the circuit to get the labels
4. Optimize the model by updating the parameters.

In this section we cover some quantum machine learning and neural network algorithms and compare them with their classical counterparts.

4.1 Variational quantum classifiers

4.1.1 Background: Classification

Classification problems are those where we have to identify which of k classes or categories (labels) does some input belong to. These fall under supervised machine learning. Here we are given data with their corresponding labels. We then perform the following steps:

1. We take the data and pass it into our model $f(x)$.
2. We get the prediction of our model and see how accurate it is via the cost function.
3. We optimize our model by changing the parameters such that the cost function is minimized (using techniques like gradient descent)
4. We repeat steps 2 and 3 till we are satisfied with the results.

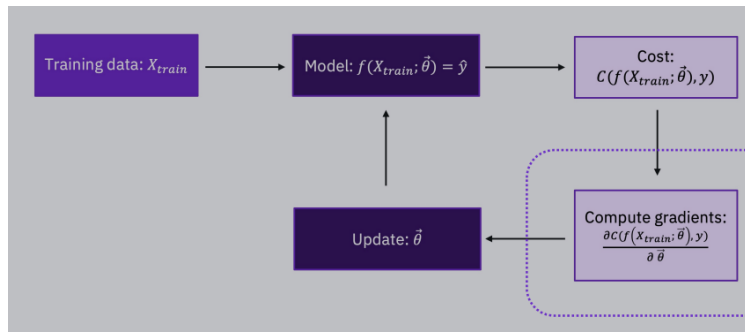


Figure 10: Procedure [7]

Examples of classification tasks include object detection in images, classifying whether a patient has a particular disease based on medical data, and any other categorical tasks. When choosing a model we want to make sure that it performs good on the training data. However, this may lead to the model 'over-fitting' to our training data and not being able to generalize well. So we should make sure that our model does not over fit the data and maybe consider splitting the data into three sections: train, validation and test sets. After training on the train set we can see how it performs on the validation set and make adjustments to our model. We should only check how our model performs on the test set at the end and not make any adjustments, to make sure our model is not biased towards the test set.

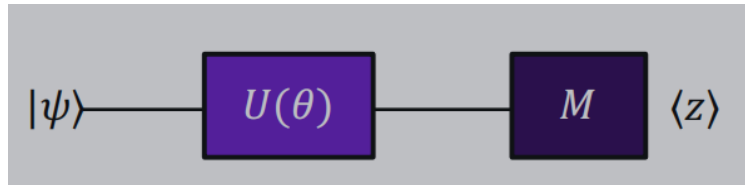


Figure 11: Variational circuit [7]

4.1.2 Background: Variational circuits

In variational circuits we have the normal quantum circuit structure, but the gates are now parameterized. This implies that we can now change the gates by changing the value of the parameters. They are also called parameterized quantum circuits or ansatz.

4.1.3 Approach: Variational circuit as a classifier

Here we will be following the general classification steps and just use our variational circuit as our model. We then perform the following steps:

1. Data encoding

Since we have classical data we need to encode it into the quantum state so that we can perform quantum gates on them. The way you encode depends on the data you have. Here are some ways one can encode classical data to a quantum state.

- Basis Encoding

Here we represent the classical data as the computational basis states, i.e., $|00\rangle$, $|111\rangle$, etc.

- Amplitude Encoding

Apply some rotations on the qubits such that the probability amplitudes will be the same as the input data

- Angle Encoding

We take as many qubits as there are features and apply rotations on the qubits which will depend on the value of the feature.

- Higher order Encoding

Here we again take as many qubits as there are features, apply hadamard gates and apply rotations on the qubits similarly to angle encoding. We then apply some entanglement gates, and now perform more rotations which depend on the product of the feature values.

2. Apply the model

Here we will apply our parameterized gate. How to design such a gate is currently an open question. One can consider using gates such that we can access more of the Hilbert space so that we can have more generalization and sample more from the space. But it is not necessary that this is always more beneficial. Below you can see a few variational circuits.

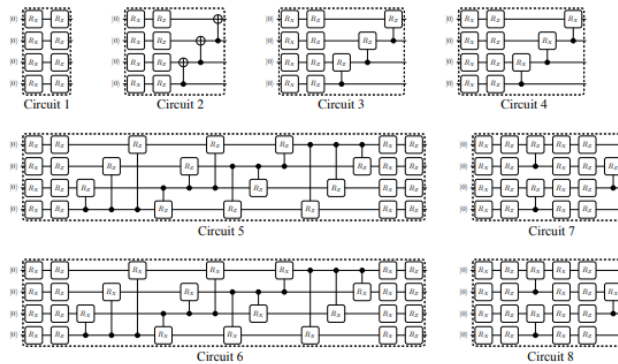


Figure 12: Variational circuit examples [8]

Generally, the structure of the variational circuit involves rotations on the qubits which depend on trainable parameters followed by entanglement. To increase number of parameters we repeat this block.

3. Measure the circuit to get labels

After the model we need to then measure the circuit to get labels. One can do this in a few ways like seeing the probability distribution of the outputs and assigning even and odd parity as 2 separate classes (considering binary classification here) or more complicated measurement schemes. We could also just measure the first qubit. While it may seem that we would be losing some information by doing this, by looking at the complex measurement scheme as a set of rotations, we can consider that as already being part of our parameterized circuit allowing us to simply measure the first qubit.

4. Use optimization techniques

One way to optimize is to find gradient of the model. We can calculate it using the parameter shift rule. We shift the parameters up and down by a small amount and take the difference to get the gradient.

$$\text{Gradient} = \langle 0 \rangle^{\otimes n} \xrightarrow{U(\theta + s)} \text{Measurement} = \hat{y}_{\theta+s} - \langle 0 \rangle^{\otimes n} \xrightarrow{U(\theta - s)} \text{Measurement} = \hat{y}_{\theta-s}$$

Figure 13: Parameter shift rule [7]

4.1.4 Summary

Here we see how a variational circuit can be used as a classifier. Doing this does not provide any known advantage over classical machine learning as they are simply performing linear classification after acting as a feature map when converting the classical data to quantum state. So currently more focus is on the feature maps and how to encode the classical data into quantum states so that it is not simply linear classification.

4.2 Quantum k-means clustering

4.2.1 Background: k-means clustering

The k-means algorithm belongs to the unsupervised machine learning algorithm family. Here we group our unlabeled data into k clusters based on the data-set. Here, k is the number of clusters that we are dividing the data into. Each cluster is associated with a centroid and the aim of this algorithm is to minimize the distance between the data point and the cluster it will belong to. These are the steps to perform k-means clustering:

1. Select the number of clusters to divide the data into, k
2. Randomly select k centroids (points)
3. Assign each data point to their closest centroid, thus forming the clusters.
4. Reassign the centroid for each cluster by finding the point with least distance from all other points in same cluster.
5. Repeat steps 3 and 4 till there is no more reassignment.

4.2.2 Approach: Quantum k-means algorithm

To perform the quantum algorithm we will be using the GroverOptim, SwapTest and DistCalc algorithms mentioned in the quantum computing section. Procedure:

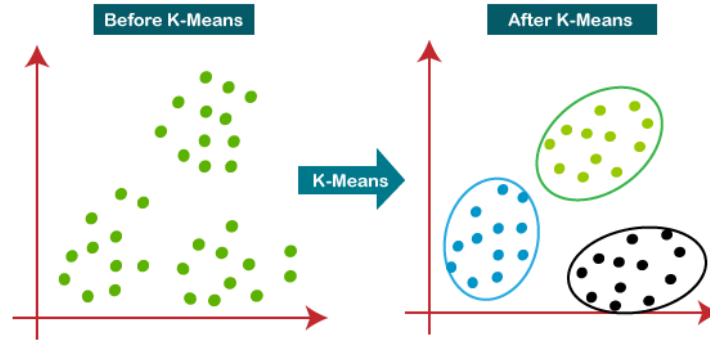


Figure 14: Before and after applying k-means clustering

1. Choose number of clusters k
2. Randomly initialize k cluster centroids
3. Loop over the data points and
 - (a) use DistCalc to find distance between each point and the centroids
 - (b) use GroverOptim to choose the centroid which minimizes the distance between the point and centroid
4. Loop over the clusters and reassign the centroid to be the mean of all the points in the cluster.
5. Repeat steps 3 and 4 till there is no more reassignment.

4.2.3 Summary

Here we have seen how to perform the quantum k-means clustering algorithm. The classical k-means algorithm has a time complexity of $O(MNk)$ (Lloyd's version of k-means) while the quantum algorithm has a time complexity of $O(Mk \log(N))$. This is an exponential speed up and does not consider the speed up due to Grover's algorithm in the minimization process. One should keep in mind that this is limited in the presence of noise.

4.3 Support vector machines

4.3.1 Background: Support vector machines

Support vector machines are supervised machine learning algorithms. The algorithm aims to find a hyperplane which separates the two classes which is then later used as the decision boundary for future points. We consider the points closest to our decision boundary as the support vectors and the minimum distance between the points and the boundary is the margin. Here we try to find the boundary which maximizes the margin length.

When we have data that is not linearly separable, we can make it linearly separable by adding new features (taking it into higher dimensions). This transformation is called a feature map.

We then try to find the decision boundary. This can be done by maximizing the distance between the points and the hyperplane such that the points lie on the correct side. We should note that knowing the support vectors is equivalent to knowing the hyperplane that is the decision boundary since they constrain the decision boundary.

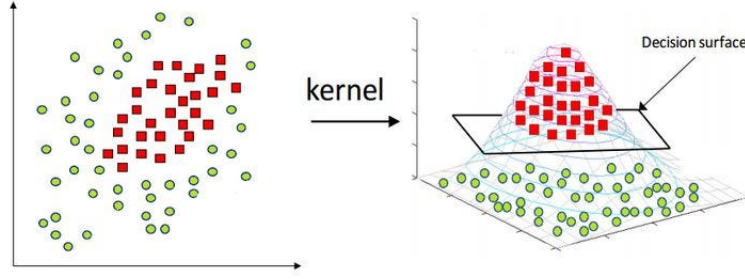


Figure 15: Applying the feature map to make the data linearly separable

This allows us to consider it in another form (dual form) which is independent of the hyperplane and depends on the inner product of the data points. We also see that we only care about the support vectors instead of all the data points. This dual form is useful since the dot products can be replaced by a kernel function that will implicitly encode the feature map. This is called the kernel trick. Here, the inner products can be organized into a matrix called the kernel matrix which is full of scalars.

This allows us to calculate the hyperplane without actually computing the feature vectors. This also allows us to have feature maps go to higher dimensions. Thus if our kernel function can be computed efficiently, we can efficiently solve our problem. For more detail refer to [9]

4.3.2 Approach: Quantum support vector machines

To implement the quantum support vector machine algorithm we use the following steps:

1. Initialize the free parameters and calculate the kernel matrix
2. Encode the classical data into quantum states
3. Use the GroverOptim algorithm to find the optimal parameters.

4.3.3 Summary

Here we see how to perform the quantum support vector machine algorithm. The classical version has a time complexity of $O(M^2 k N)$ while the quantum version has a complexity of $O(M^2 \sqrt{N})$. The main problem comes in computing the kernel function. There are attempts to use the quantum version to use kernel functions which are hard to implement classically (like the DLOG kernel). The quantum kernel can be used in other algorithms like spectral clustering, principal component analysis and kernel k-means algorithm.

4.4 Neural networks

4.4.1 Background: Neural Networks

Neural networks are mathematical representation of biological neurons. They are composed of a series of layers which contain a number of fundamental units which we call nodes (also called perceptrons or artificial neurons).

The nodes are essentially simple non-linear functions. They accept a particular number of inputs (which will be affected by trainable weights), compute a non-linear activation signal and pass on the output to further layers. By adjusting the weights of the neural networks we can make the whole neural network represent some arbitrary function. These are mostly used in supervised learning.

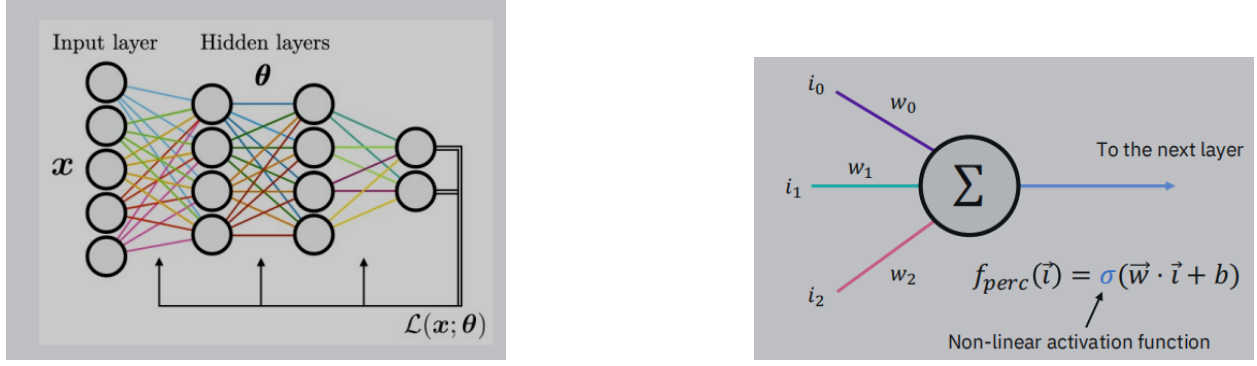


Figure 16: Neural network on the left and perceptron on the right [7]

4.4.2 Approach: Quantum Perceptron

When starting to develop quantum versions of classical neural networks, one of the first ideas is the quantum perceptron. The key feature of the classical perceptron, its non-linearity, is not immediately easy to implement in a quantum model since quantum computing and quantum mechanics in general are linear in a mathematical point of view (unitary transformations are linear on their inputs). There have been many various methods to realise quantum perceptron models over the years including Quantum Fourier Transform based perceptron, a repeat until success scheme to achieve non-linearity or using measurement to achieve non-linearity. The perceptron in quantum computing is essentially a parameter controlled unitary operation.

4.4.3 Approach: Quantum Neural Networks

We see that similar to the classical version, in the quantum neural network also has layers of perceptrons and have adjustable parameters. In the image, we see 2 kinds of layers in the QNN circuit:

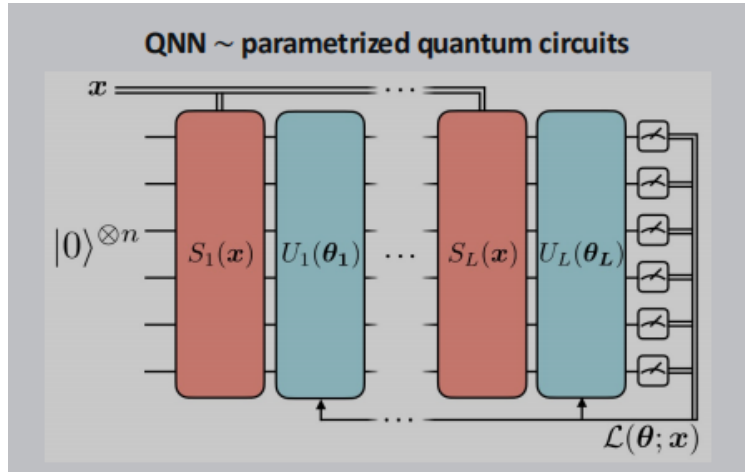


Figure 17: Circuit for quantum neural network [10]

- One which is parameterized by the input variables
- One which depends on the trainable parameters

We must note that the layers in the QNN are unitary operations and thus are linear. To use these models we need to keep having the layers which depend on the input.

4.4.4 Summary

Here we have looked at how to implement a quantum neural network. Below we have circuits of some quantum neural network architectures.

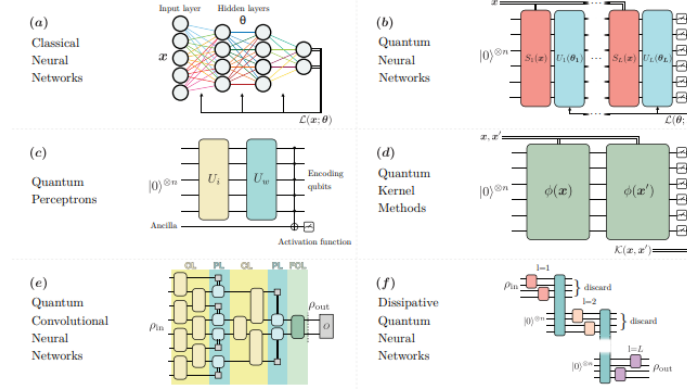


Figure 18: Circuits for some quantum neural network architectures [10]

4.5 Convolutional neural networks

4.5.1 Background: Convolutional neural networks

Convolutional neural networks are a very popular neural network architecture. They are very useful in the analysis and classification of images where typically the input is too large for the normal neural networks to run efficiently on. They do this by alternating some specialized layers called convolutional layers and pooling layers.

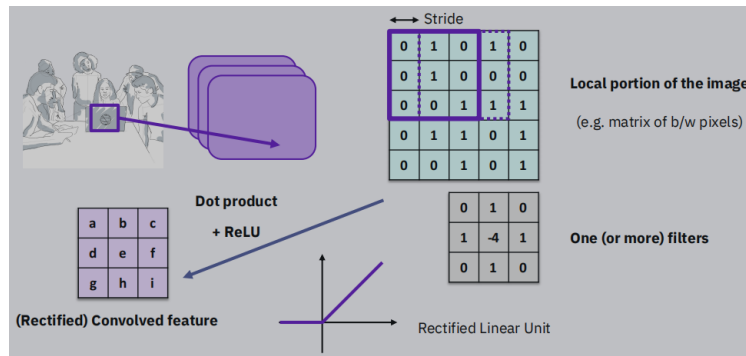


Figure 19: Convolutional Layer [7]

Convolutional layers will analyze local areas of the image. We first convert every pixel to a number and create the mathematical representation of the image. The convolutional layer applies one or more filters which are shifted over the image. We apply a dot product followed possibly by a non-linear combination of the product values and we receive a transformed image. The filters can be specialised to detect specific patterns or features in the image.

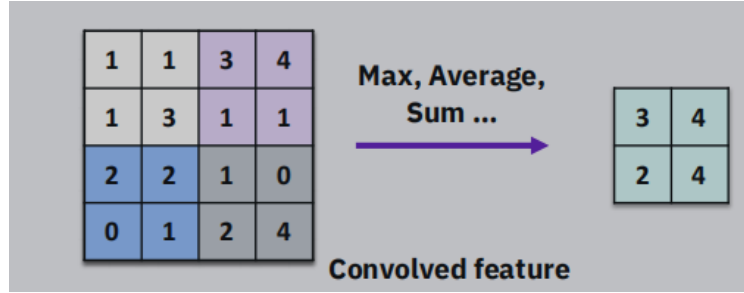


Figure 20: Pooling Layer [7]

These transformed features then undergo dimension reduction in the pooling layer. We divide the transformed image into regions and condense each region to a single value, say taking the maximum value or average in the region. This allows the input to be more manageable for further layers and makes the network tolerant to small changes. We repeat these layers till we apply a fully connected layer to perform a final classification.

4.5.2 Approach: Direct approach

Here the aim is to directly implement the whole convolutional network on quantum hardware. With some assumptions, provable speedups can be achieved based on faster linearized manipulations achieved by leveraging quantum superposition effects. However, encoding classical data causes a bottleneck and the overall algorithm may require non trivial subroutines. Some implementations include [11, 12]

4.5.3 Approach: Quconvolutional neural networks

Here the aim is to extend the capabilities of classical convolutional neural networks by using quantum kernels as a transformation or filter in classical layers. We can do this by taking the inputs of the classical filter and use them as parameters in a quantum feature map. It is compatible and integrable with classical models and can be used in near term implementations. Implementation includes [13]

4.5.4 Approach: Quantum convolutional networks

Here the aim is to build quantum machine learning models inspired by convolutional neural networks for the analysis of classical and quantum data. Here we have a set of convolutional layers where we apply transformation to subsets of qubits. We then have pooling layers which reduce the dimensionality of the system by measuring some of the qubits and using these measurements to control gates on the remaining qubits. We repeat these layers and then use the quantum version of a fully connected layer which is essentially a unitary operation on the remaining qubits. Here there are very few trainable parameters. Works include [14, 15, 16]

4.5.5 Summary

Here we have looked at some implementations of quantum convolutional neural network.

5 Conclusions

Here we have looked at how some of the quantum machine learning and quantum neural network models work. We have covered the basic intuition of these algorithms but have skipped over the mathematics which are required to show that these algorithms work. After reading this you are encouraged to go through the sources mentioned in the references to gain deeper knowledge in this subject. Some code implementations can be found here [17].

6 Future Works

Some future works include:

- Making a more comprehensive guide by including the mathematics behind these algorithms.
- Implementing new algorithms like a quantum transformer network.
- Trying to find different ways to encode classical data into quantum states.

7 Acknowledgements

I am extremely grateful to Dr. Nithyanandan Kanagaraj for giving me the opportunity to do this project and for guiding me through the entire process.

I am also thankful to the Department of Physics, IIT-H for giving me the wonderful opportunity to credit the project work.

REFERENCES

- [1] Adetokunbo Adedoyin, John Ambrosiano, Petr Anisimov, Andreas Bärtzchi, William Casper, Gopinath Chennupati, Carleton Coffrin, Hristo Djidjev, David Gunter, Satish Karra, et al. Quantum algorithm implementations for beginners. *arXiv preprint arXiv:1804.03719*, 2018.
- [2] Noson S. Yanofsky and Mirco A. Mannucci. *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008. doi: 10.1017/CBO9780511813887.
- [3] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
- [4] Dawid Kopczyk. Quantum machine learning for data scientists. *arXiv preprint arXiv:1804.10068*, 2018.
- [5] Frank Zickert. Towards Understanding Grover’s Search Algorithm. <https://towardsdatascience.com/towards-understanding-grovers-search-algorithm-2cdc4e885660> 2021.
- [6] Maria Schuld and Francesco Petruccione. *Supervised learning with quantum computers*, volume 17. Springer, 2018.
- [7] Qiskit contributors. Qiskit: An open-source framework for quantum computing, 2023.
- [8] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.
- [9] Andrew Ng. CS229 Lecture Notes Part V: Support Vector Machines (SVM). <https://see.stanford.edu/materials/aimlcs229/cs229-notes3.pdf>.
- [10] Stefano Mangini, Francesco Tacchino, Dario Gerace, Daniele Bajoni, and Chiara Macchiavello. Quantum computing models for artificial neural networks. *Europhysics Letters*, 134(1):10002, 2021.
- [11] YaoChong Li, Ri-Gui Zhou, RuQing Xu, Jia Luo, and WenWen Hu. A quantum deep convolutional neural network for image recognition. *Quantum Science and Technology*, 5(4): 044003, 2020.
- [12] Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. *arXiv preprint arXiv:1911.01117*, 2019.
- [13] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quantvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):2, 2020.
- [14] Edward Grant, Marcello Benedetti, Shuxiang Cao, Andrew Hallam, Joshua Lockhart, Vid Stojevic, Andrew G Green, and Simone Severini. Hierarchical quantum classifiers. *npj Quantum Information*, 4(1):65, 2018.
- [15] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.
- [16] Arthur Pesah, Marco Cerezo, Samson Wang, Tyler Volkoff, Andrew T Sornborger, and Patrick J Coles. Absence of barren plateaus in quantum convolutional neural networks. *Physical Review X*, 11(4):041011, 2021.
- [17] Raghav Juyal. Quantum machine learning and neural networks implementations. <https://github.com/RaghavJuyal/Quantum-Neural-Networks>, 2023.