# Machine Learning using Quantum Computing

*Implementing Machine Learning on Quantum Simulators

Prajwal A Nazre
*School of Computer Science and Engineering*
*KLE Technological University*
Hubli, Karnataka
prajwal.nazre777@gmail.com

Dr. Satyadhyan Chickerur
*School of Computer Science and Engineering*
*KLE Technological University*
Hubli, Karnataka

***Abstract*—Quantum Computing as a whole, is a very vast field in Computer Science. It is basically leveraging the concepts of Quantum Mechanics to perform the Computations. It is believed to solve a lot of complex problems with greater efficiency than traditional computing techniques.**

**Machine Learning using Quantum Computing is an interdisciplinary research field formed from the combination of Machine Learning and Quantum Computing. It can be a perfect example to leverage the power of computation provided by Quantum computing techniques. It is believed that many machine learning algorithms can be implemented on Quantum Computers and get better efficiency. My research is to mainly focused on this aspect of the vast field.**

## I. Introduction

Quantum machine learning (QML) has revolutionized performance and speed. Quan- tum computation refers to a computational paradigm that relies on quantum mechanics' laws. Quantum information technologies and intelligent learning applications are both hot areas and released technologies form the evolution of informatics. The quantum information is now used within machine learning (ML). Quantum Information (QI) is information that is stored in the state of a quantum system. Quantum information is known by the information is used by the quantum system. Machine learning refers to a data analysis method that can be constructing the analytical model automatically. ML is an essential branch of artificial intelligence that relies on the concept of learning from data; recognize patterns and decision making with minimum human interference. The construction of quantum computers is very difficult and requires complex programs. The quantum has a big impact on the interference or entanglement happen. So the computers relying on quantum propose more efficient solutions for the chosen challenges than the computers based on the classical method.

## II. Working of Quantum Computers

### A. Nature of Computation

On a basic level, Quantum Objects are nothing but Non-Classical objects with the de-Broglie wavelengths too significant to ignore. These objects have uncertain positions at a specific given time. To predict the end state of a quantum object, we need to take into account all the paths it could have taken to get there.

On doing a number of researches and experiments, it is concluded that directly applying the quantum objects to perform computation increases the speed when compared to the likes of traditional computing techniques. In fact, the results are much faster as the computation gets more complex. This is can be used greatly to our advantage. Quantum computers just follow the laws of physics and taking advantage of it highly depends on how we restate the computational problems in terms of equivalent quantum systems.

### B. Quantum Bits

Before we jumped into Quantum Bits or Q-Bits, we made sure to go through the Logical Gates and Circuits relating to the computers including the basic gates and universal gates which we felt necessary before jumping to Quantum. Knowing the way algorithms working in traditional computers can help understand the relatively new field of Quantum when compared to the mature field of Computation. Quantum computer isn't really computing at all. It's just allowing the laws of physics to act on a system designed for a very particular purpose. The challenge of building a quantum computer is designing components that behave according to the laws of quantum mechanics but can be adapted to solve many problems — not just simulating the path of photons using photons or simulating the electrons in a molecule by studying that molecule.

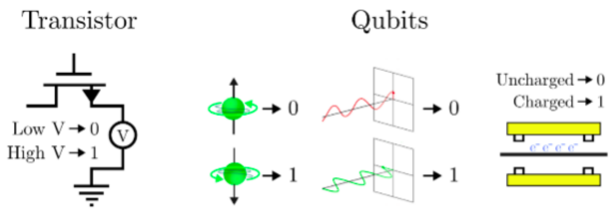


Fig. 1. Working of Qubits

Classical computers perform tasks by using two-state objects called bits as inputs, manipulating them according to

Fig. 1 Source : Brilliant

some algorithm, then yielding one or more bits as an output that can be measured. Much as classical computation uses bits, quantum computation uses qubits, isolated quantum objects that behave quantum mechanically, with two or more states that can be distinguished with a measurement.

The Bloch sphere is a geometrical representation of the pure state space of a two-level quantum mechanical system (qubit). This forms a fundamental building block of quantum computer.

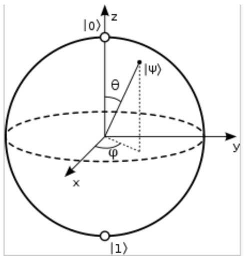


Fig. 2. Bloch Sphere

## C. *Quantum Circuits*

The current quantum model is defined in terms of Quantum Logic gates. This can be considered equivalent to linear-algebraic equations in mathematics. Quantum computers would be built using these logic gates. A Sequence of such gates form Quantum Circuits. A memory of n-bits of info can have around $2^n$ different states. Thus, a vector that represents the memory states has $2^n$ entries. This is a probability vector and it can represent one particular state out of all other possible states.

In a traditional computer, one bit can have only one value and it is a fixated one. In quantum mechanics, the vectors are represented using a density matrix. Let us consider a simple qubit. This memory may be found in one of two states: the zero state or the one state. We represent the state of this memory using Dirac notation

$$|0\rangle := \begin{pmatrix}1\\0\end{pmatrix}; \quad |1\rangle := \begin{pmatrix}0\\1\end{pmatrix}$$

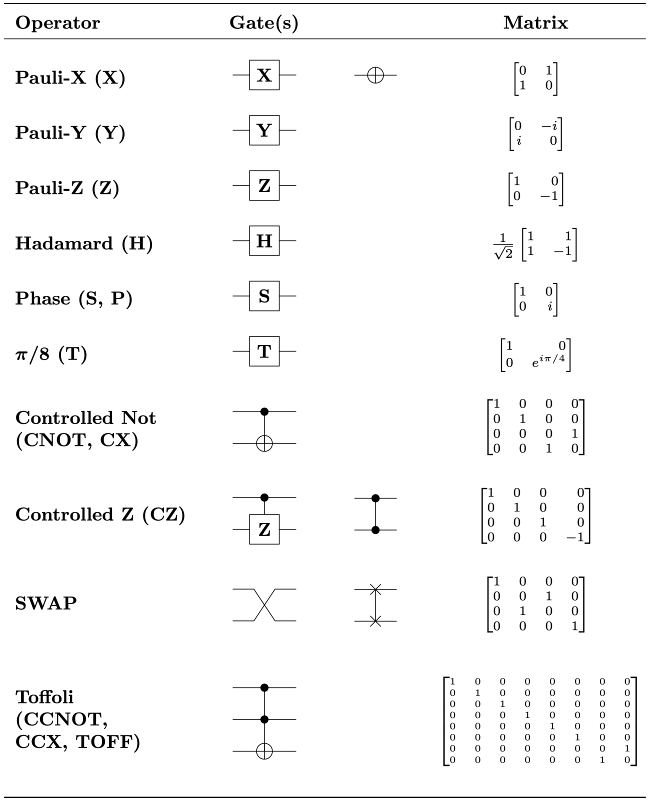
Among all the gates, we am going to use Pauli-X gate. This gate acts on a single bit. It is a quantum equivalent to a not gate. It indicates rotation of the bloch sphere in the x-axis direction by $\pi$ radians. Pauli-X matrix is represented as

$$X = \begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$$

## III. Simulating Quantum Computers

The task of simulating quantum computers is a real challenge. Thankfully, there are various platforms and frameworks to help us simulate quantum computations and we can further work on our problem statement of implementing Machine Learning. Some platforms are :

- PennyLane
- Qiskit
- Tensorflow



| Operator | Gate(s) | Matrix |
|---|---|---|
| **Pauli-X (X)** | $\boxed{X}$ | $\begin{bmatrix}0 & 1\\1 & 0\end{bmatrix}$ |
| **Pauli-Y (Y)** | $\boxed{Y}$ | $\begin{bmatrix}0 & -i\\i & 0\end{bmatrix}$ |
| **Pauli-Z (Z)** | $\boxed{Z}$ | $\begin{bmatrix}1 & 0\\0 & -1\end{bmatrix}$ |
| **Hadamard (H)** | $\boxed{H}$ | $\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\1 & -1\end{bmatrix}$ |
| **Phase (S, P)** | $\boxed{S}$ | $\begin{bmatrix}1 & 0\\0 & i\end{bmatrix}$ |
| **$\pi/8$ (T)** | $\boxed{T}$ | $\begin{bmatrix}1 & 0\\0 & e^{i\pi/4}\end{bmatrix}$ |
| **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)** | $\boxed{Z}$ | $\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&-1\end{bmatrix}$ |
| **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)** | | $\begin{bmatrix}1&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&0&0&1&0\end{bmatrix}$ |

Fig. 3. Quantum Logic Gates

## A. *PennyLane*

A cross-platform Python library for differentiable programming of quantum computers. We can train a quantum computer just like a neural network. Some great features are as follows:

- Automatic differentiation of Quantum Circuits
- Hybrid models, connecting Quantum hardware to PyTorch, Tensorflow and Numpy
- Open-Source Tools for Quantum Machine Learning
- Resources and documents to learn and get familiar with the environment

## B. *Qiskit by IBM*

Qiskit is an open-source SDK for working with quantum computers at the level of pulses, circuits and algorithms. Qiskit accelerates the development of quantum applications by providing the complete set of tools needed for interacting with quantum systems and simulators.

Fig. 2 Source : Wikipedia

Fig. 3 Source : Wikipedia

### C. *Tensorflow*

Tensorflow needs no introduction. This major machine learning open-source framework has been one of the best tools for programming models alongside keras. Recently, Tensorflow Quantum library has been added to its framework which has been phenomenal to all the Quantum enthusiasts. This library is used to prototype hybrid QML(Quantum Machine Learning) models. We have decided to go with this as Tensorflow has provided with a ton of instructions, references and resources to learn from. The fact that it is an open-source makes it free to use where anybody could try simulating Quantum Circuits.

## IV. Implementing ML on Quantum Environment

### A. *Initial Setup*

Before starting the implemetation, we must make sure that we have taken care of all the dependancies and frameworks. Install Tensorflow 2.3.1 along with Tensorflow Quantum (TFQ) library. TensorFlow Quantum is a quantum machine learning library for prototyping of hybrid quantum-classical ML models. Research in quantum algorithms and applications can leverage Google's quantum computing frameworks, all from within TensorFlow. Some necessary libraries like seaborn, numpy, sympy, matplotlib and most importantly cirq. Cirq is a Python library for writing, manipulating, and optimizing quantum circuits and running them against quantum computers and simulators.

### B. *Load Data*

Import the required datasets into the environment (Jupyter Notebook preferred). In my case, it is mnist fashion dataset. The raw dataset contains 60000 training set images and 10000 test set images. This dataset contains 10 labels. As we am doing a binary classification, we am filtering out datas labelled 3 (Dresses) and 6 (Shirts). Now there are 12000 training images with 2000 testing images.

All the images are of the size 28 x 28. The current quantum circuits are limited. Hence, this size is too high for the circuits due to which, downscaling of the images become necessary. I downscaled the image to 4 x 4. Since the image is downscaled to a large extent, there might be images that can be contradictory i.e a same image can be present in both the labels. To make our model stronger, we would need to remove the contradictory images.

If each pixel is to be considered as a single qubit then, an image will be converted to a circuit. If a pixel has a value greater than the threshold, it must be have the value opposite to that of the background (MNIST Dataset is grayscaled) which means that pixels that form image should have a corresponding circuit. This can be achieved using the cirq library by applying Pauli-X gate on the required pixels and couple them to form the circuit. And finally, convert the circuits to its equivalent tensors for tfq.

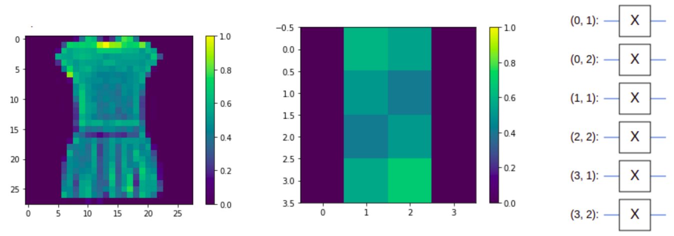


Fig. 4. Equivalent Quantum Circuits for Images

### C. *Quantum Neural Network*

We need to build a quantum circuit structure to classify the image. So, lets go with an approach of running a simple RNN across the pixels. We need to initialize 4 x 4 image grid as the data for our QNN. A special qubit known as readout qubit must be created which is used to read the values from the grid. This bit is going to be connected all other qubits. To prepare this bit, I am passing it through one Hadamard Gate(H) and one Pauli-X Gate(X). This makes it eligible for reading out the values of it's connected bit. For this dataset, a double layered QNN would be more than enough. Hence, we must add 2 layers of a double Pauli-X circuit and Pauli-Z circuit. The 'XX' denotes that it is accepting 2 different values. Since, our numpy array of the images are in this form, it would match the input.To end the ciruit, append a final Hadamard Gate for reading. The final circuit would look something like this. Use SVGCircuit to print your circuit.

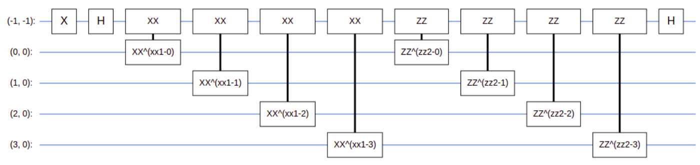


Fig. 5. Quantum Circuit Model

All the preparations are done now. We have to link the data and the Quantum Circuit Model we built just now. Keras has a special layer type for accepting quantum data and circuits. This special layer is known as Parametrized Quantum Circuit layer to train the model circuit, on the quantum data. This can be accessed using 'tfq.layers.PQC'. Our model and data, both needs to be fed into Keras model. To classify these images, we must take the expectation of a readout qubit in a parameterized circuit. The expectation returns a value between 1 and -1. Then, portray the preparation strategy to the model, utilizing the Compile Method. Since the normal readout is in the range [-1,1], optimizing the hinge loss is required. To utilize the hinge loss, we must make two little changes. First convert the data from boolean to [-1,1], true to form by the hinge loss.

Train the model (takes around 30 mins). On the off chance that you would prefer not to stand by that long, utilize a little subset of the dataset. This doesn't actually influence the model's advancement during training (it just has 32 parameters). Utilizing lesser data simply closes training early, yet runs adequately long to show that it is making progress in the validation logs.

### D. *Classical Neural Network*

For the sake of comparison, lets build a Classical Machine Learning model. Create a keras model for CNN, add required number of layers and compile the model. Fit the model with the original raw dataset and check for the results and parameters. The model would easily converge at 99.9 percent accuracy since it is a MNIST dataset.

## V. Results

### A. *Comparison*

The QNN produced a result of 69 percent while the CNN produced a whopping 99.9 percent. The QNN was built on the grounds of quantum circuits and considering each image as an indvidual circuit. The computations it undergoes are complex. The loss factors needs to be optimized even further to get better results.

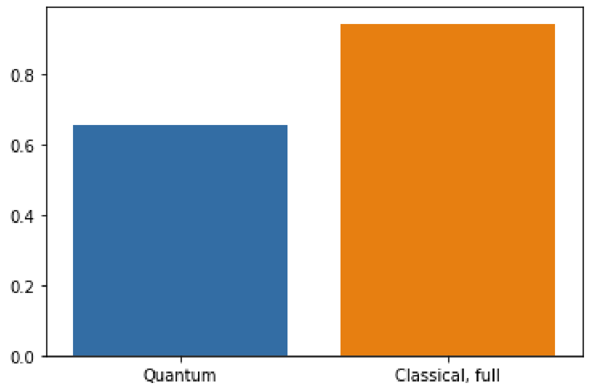


Fig. 6. QNN vs CNN with Full Data Feed

A question arises as to what would happen when the data fed to both the models would be reduced. To find out, the data was halved and fed to both QNN and CNN. Interestingly, the accuracy of the QNN increased to 73 percent while that of CNN decreased significantly to 86 percent. CNN behaves bad when data fed is low while QNN took a bit of an advantage in this case. So, it is quite probable that QNN might have an application when the initial data is relatively less.

### B. *Future Scope*

Quantum Computers today are still in their infancy. Many big names like IBM, Google, Amazon are trying to reach this domain and come up with methods to leverage the full potential of the Quantum Computers. This research of mine is just a demonstration that Machine Learning on a quantum environment is possible. But, it would take another decade or two to finally implement them on real systems and utilize the phenomenal speed and accuracy. Further research on this is possible as to how we can improve the accuracy of the QNN model. However, all the actual results would have to wait until we perform these on actual Quantum Computers.

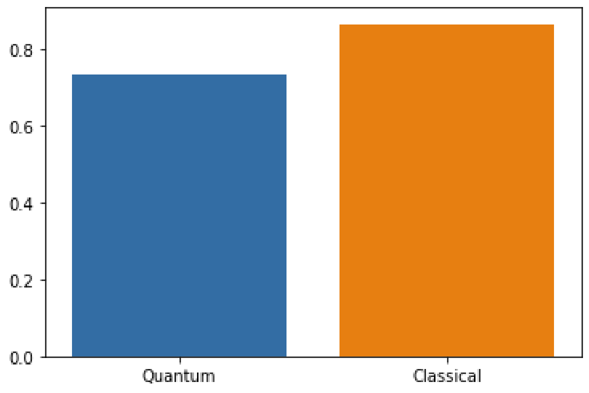


Fig. 7. QNN vs CNN with Half Data Feed

## Acknowledgment

This research was made possible from the guidence of my professor Dr. Satyadhyan Chickerur. I thank him as well as our University for providing me an opportunity to do research on Quantum Computing and Machine Learning aspect of it.

## References

[1] Google Quantum AI. Cirq framework. https://quantumai.google/cirq/. Usage of the cirq library.
[2] Brilliant. https://brilliant.org/practice/quantum-bits/?p=3. Figure 1.
[3] Brilliant. Quantum computing. https://brilliant.org/courses/quantum-computing/. Basics of Quantum Computers.
[4] Alessandro Davide Ialongo Massimiliano Pontil Andrea Rocchetto Simone Severini Carlo Ciliberto, Mark Herbster and Leonard Wossnig. Quantum machine learning: a classical perspective. 2018.
[5] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. 2018.
[6] IBM. Qiskit by ibm. https://qiskit.org/. Knowing the platform.
[7] Iordanis Kerenidis Jonathan Allcock, Chang-Yu Hsieh and Shengyu Zhang. Quantum algorithms for feedforward neural networks. 2019.
[8] Aaron Kalvani. Machine learning on quantum computing in 2020. https://aijourn.com/a-summary-of-machine-learning-with-quantum-computing-in-2020/. Machine Learning on Quantum Computing.
[9] Iordanis Kerenidis and Alessandro Luongo. Quantum classification of the mnist dataset via slow feature analysis. 2018.
[10] Ilya Sinayskiya Maria Schulda and Francesco Petruccionea. An introduction to quantum machine learning. 2014.
[11] Trevor McCourt Antonio J. Martinez Michael Broughton, Guillaume Verdon. Tensorflow quantum: A software framework for quantum machine learning. 2020.
[12] Pennylane. Pennylane by xanadu. https://pennylane.ai/. Knowing the platform.
[13] Rigetti. Forest by rigetti. https://www.rigetti.com/forest. Knowing the platform.
[14] Kevin Bonsor Jonathan Strickland. How quantum computers work. https://computer.howstuffworks.com/quantum-computer.htm. Working of Quantum Computers.
[15] Tensorflow. Machine learning on quantum computing using tensorflow. https://tensorflow.org/.
[16] Tensorflow. Quantum computing on tensorflow. https://www.tensorflow.org/quantum. Usage of Tensorflow Quantum Framework.
[17] Wikipedia. Quantum computing. https://en.wikipedia.org/wiki/Quantum_computing. Basics of Quantum Computing.
[18] Wikipedia. Quantum logic gates. https://en.wikipedia.org/wiki/Quantum_logic_gate. Basics of Quantum Logic Gates.
[19] Wikipedia. Quantum machine learning. https://en.wikipedia.org/wiki/Quantum_machine_learning. Basics of Quantum Machine Learning.

## LIST OF FIGURES