QUANTUM MACHINE LEARNING REPORT

Abhinav Dwivedi B20CI003
Tejal Kumari B20CI044

TITLE: QUANTUM MACHINE LEARNING

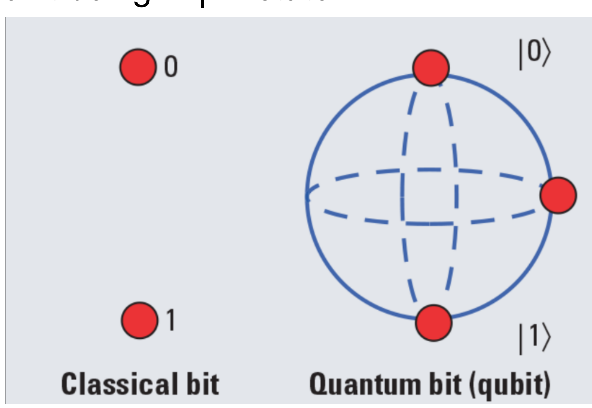
INSTRUCTORS:
V. Narayan and Subhashish Banarjee

ABSTRACT:
Quantum Machine Learning is an interplay of Quantum Computing and Machine Learning. The field of Quantum enhanced machine learning refers to making use of quantum algorithms in machine learning tasks with an aim to speed up the computations. The aim of this project was to study Quantum Computation and explore its useful applications in Machine learning tasks.

INTRODUCTION:
Initially, we had started learning the fundamental concepts of quantum computing and understanding and implementing many machine learning algorithms.
A qubit is made from a two-level system, where the levels are |0> and |1> (Dirac Notation), they are analogous to classical bits 0 and 1. A difference between classical bit and a qubit is that a qubit can exist as a superposition of states |0> and |1>. This is called a superposition state. For a qubit |ψ>, its superposition state can be written as:

$$|\psi\rangle = a|0\rangle + b|1\rangle$$

Where, a and b are complex numbers, and $|a|^2 + |b|^2 = 1$.
If we measure this qubit, there is a $|a|^2$ probability of the qubit being in |0> state and a $|b|^2$ probability of it being in |1> state.



Gates in Quantum computing:
In quantum circuits, the operations(gates) are always reversible. These reversible gates can be represented as matrices, and as rotations around the Bloch sphere.
The Pauli X gate is analogous to the NOT gate. It is represented by the Pauli-X matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0|$$

The Hadamard gate (H-gate) allows us to move away from the poles of the Bloch sphere and create a superposition of |0> and |1>. It has the matrix:

$$H = \tfrac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

The RXX gate is a parametric 2- qubit X⊗X interaction, represented by:

$$R_{XX}(\theta) = \exp\left(-i\frac{\theta}{2} X\otimes X\right) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & 0 & 0 & -i\sin\left(\frac{\theta}{2}\right) \\ 0 & \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 \\ -i\sin\left(\frac{\theta}{2}\right) & 0 & 0 & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

The RZZ gate is a parametric 2- qubit Z⊗Z interaction, represented by:

$$R_{ZZ}(\theta) = \exp\left(-i\frac{\theta}{2} Z\otimes Z\right) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 & 0 & 0 \\ 0 & e^{i\frac{\theta}{2}} & 0 & 0 \\ 0 & 0 & e^{i\frac{\theta}{2}} & 0 \\ 0 & 0 & 0 & e^{-i\frac{\theta}{2}} \end{pmatrix}$$

$$XX(\chi) = \begin{pmatrix} \cos(\chi) & 0 & 0 & -i\sin(\chi) \\ 0 & \cos(\chi) & -i\sin(\chi) & 0 \\ 0 & -i\sin(\chi) & \cos(\chi) & 0 \\ -i\sin(\chi) & 0 & 0 & \cos(\chi) \end{pmatrix}.$$

Hinge Loss $\rightarrow \ell(y) = \max(0, 1 - t \cdot y)$

PROBLEM STATEMENT:
Binary Image Classification on Fashion MNIST dataset using Quantum Neural Network, with the libraries TensorFlow-Quantum and Cirq. TensorFlow-Quantum is a python framework used for quantum machine learning. Cirq is a python software library for writing, manipulating and optimizing quantum circuits.

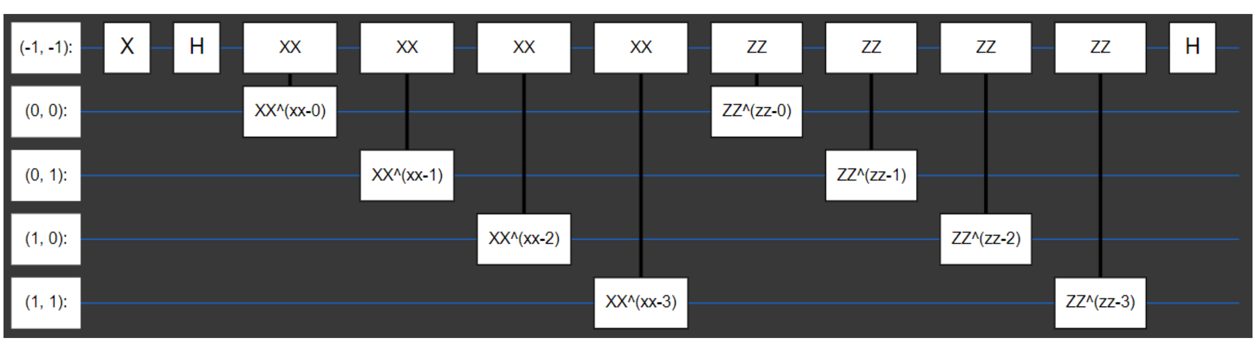
METHOD:
A quantum neural network is used.

1. Libraries used and their versions:
   i. tensorflow → 2.3.1
   ii. tensorflow_quantum → 0.4.0
   iii. cirq → 0.9.1

   Dataset: Fashion-MNIST is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

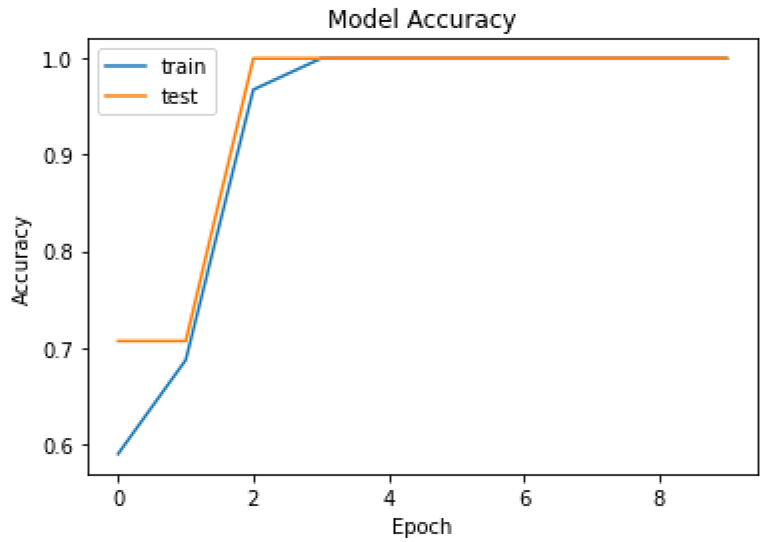
2. The x_train, y_train, x_test and y_test sets have been imported from the dataset.
3. In the preprocessing step, we have considered only classes labeled 5 and 9 out of the total 10 classes for our binary classification.
4. The images have been downsized from 28x28 to 2x2. Pixels of the images have been flattened. So now we have 4 pixels in total, each corresponding to qubits 00, 01, 10 and 11.
5. The test set, x_test and y_test have been further split into test and validations set with a size ratio of 0.15.
6. Binary encoding has been performed on x_train with a threshold of 0.5.
7. In the next step, we have converted all the images of x_train, x_test and x_valid into circuits, and an X gate has been applied to the qubit if its value is 1, using the create_circuit_from_image function that we have created.
8. The final steps involve building the Quantum Neural Network. A QNN class has been made which has the following functions:
    a. add_singleQubit_gate: Adds a single qubit gate to the circuit, has parameters - Cirq circuit,cirq gate(gate to append to the circuit), qubit_index(index to qubits to apply the gate).
    b. add_twoQubit_gate: Adds two qubit gate to the circuit, has parameters - Cirq circuit,cirq gate, qubit_index
    c. add_layer: Adds new gates/layers to the circuit, has parameters - Cirq circuit, cirq gate, symbol_gate(symbol of the gate in string).

9. A function create_qnn has been made to design the following circuit:

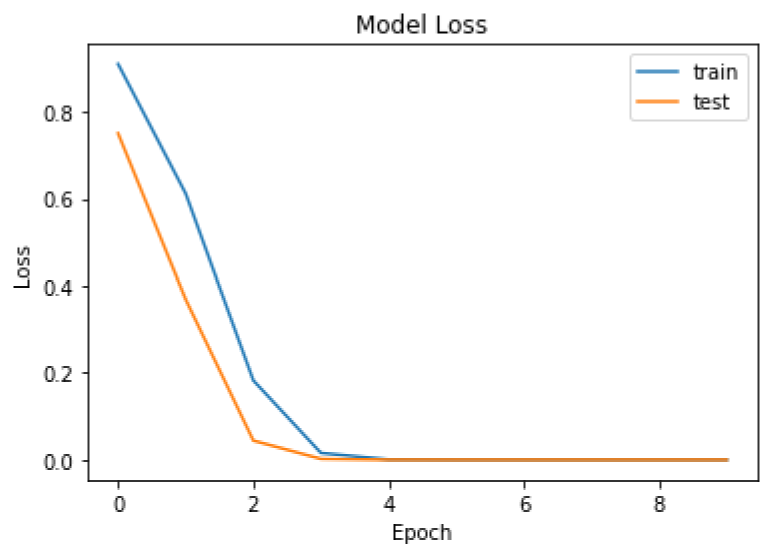


10. The usual steps of model training and testing are done along with the accuracy calculations.

RESULTS AND ANALYSIS:



The model accuracy increases with increase in the number of epochs.



The model loss decreases with increase in the number of epochs.

Code: Colab link

References:

1. https://qiskit.org/textbook/preface.html
2. https://onlinecourses.nptel.ac.in/noc21_cs103/preview
3. https://www.ibm.com/quantum-computing
4. https://www.tensorflow.org/quantum/tutorials/qcnn