Simulating Quantum Circuit using TensorFlow Quantum

August 10, 2022

1 Project Summary

1.1 Overview:

Quantum Computing brings the concept of Quantum Mechanics and the power of computing together, enabling us to simulate physical systems using computers [5]. Quantum Computers are powerful and can be used to solve many classical algorithmic problems. Our implementation uses the simulation of quantum gates like the **CNOT** gate, **Pauli X** gate, and **Toffoli** gate with multiple inputs using the Tensor Flow Quantum framework (TFQ) [1]. TFQ provides an interface for building "quantum-classical models," allowing the simulation of classical algorithms with quantum machine learning [1].

1.2 Intellectual Merit:

The **XOR** problem is an exciting machine learning problem because a single layer perceptron cannot shatter the data points correctly for an **XOR** operation with two inputs because there are 4 points and the VC-dimension is 3 in this case [13]. However, it is simpler to implement this kind of gates by building a prediction model that can simulate its function. The quantum gates equivalent to **XOR**, **NOT**, and a combination are **Pauli X**, **CNOT**, and **Toffoli** gates, respectively. Our implementation models the behavior of a circuit that uses these gates by using a training set and making predictions on a testing set.

1.3 Broader Impacts of the Proposed Work:

The successful simulation of the quantum circuit with a machine learning model makes it possible to implement more complex algorithms without physically accessing a quantum computer, making TensorFlow Quantum a practical framework for quantum programming. Our experiments have also resulted in a tutorial for practicing the implementation of a prediction model for such circuits. Computer scientists and other researchers can use this tutorial as a base to practice the implementation of quantum gates with machine learning.

2 Project Description

2.1 Introduction

Making a quantum circuit can be time-consuming, especially if too many qubits are involved, and the circuit is lengthy. To avoid the laborious process of building the circuits, we can use machine learning to design a machine learning model for such quantum circuits. Here we use Tensor flow quantum to simulate a custom quantum circuit that consists of several quantum gates. The quantum **CNOT** gate achieves the same function as the classical **XOR** gate [11]. The **Pauli-X** gate is the quantum equivalent of the **NOT** gate [12]. The **Toffoli gate** works with three quantum bits with one target bit and two control bits [4]. It inverts the state of the target bit based on the state of the two control bits [4]. Hence, the **Toffoli** gate achieves the function of combining classical gates in this sense. We have built a prediction model that behaves like a quantum circuit that uses these gates.

2.2 Methodology

2.2.1 Data

We created classical data by initializing binary numbers of 5 digits; since we are creating a quantum circuit using five wires, we can apply the equivalent classical gates to the binary numbers. The circuit below shows an illustration of a quantum circuit using **Pauli X**, **CNOT**, and **Toffoli** gates where the measure on wire 4 is equivalent to the classical gate operation $(((\neg(4 \oplus (0 \wedge 1))) \oplus 3) \oplus 2)$.

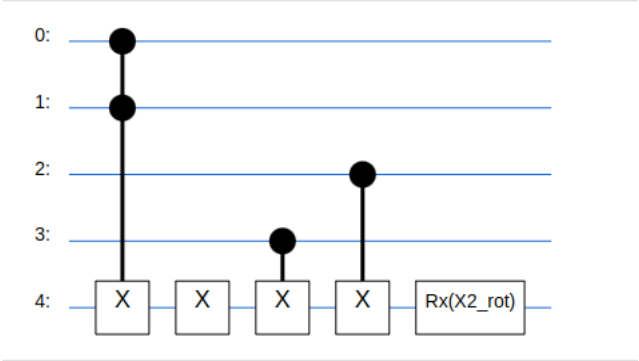

Figure 1: Quantum circuit

There are advanced ways to encode classical data into quantum data like basis embedding [8], amplitude embedding [8], and Qsample Encoding, which are the optimal ways for encoding but are pretty complex [3]. Here we keep the encoding process simple by a more straightforward computational encoding, which takes the binary for each qubit and flips it by applying the X gate. We use the cirq library, a built-in tensor flow quantum natively supported in all TFQ operations.

For instance for binary 00001,

```
cir = cirq.Circuit()
cir.append([cirq.I(qubits[0])])
cir.append([cirq.I(qubits[1])])
cir.append([cirq.I(qubits[2])])
cir.append([cirq.I(qubits[3])])
cir.append([cirq.X(qubits[4])])
```

Among the 32 combinations, we use the first 16 as the training set and the next 16 as the testing set; if the output is 0, we label it as -1, and if the result is 1 then we mark it as 1. It is labeled as so for the hinge loss [6], which we will use for training.

2.2.2 Model

We create the circuit using cirq [7] and sympy [9], which we want to model for simulation. This circuit consists of 5 quantum gates, accepts a list of qubits, and returns the circuit according to the qubits. The configuration and sequence gates are as follows:

Sequence	Gate	Control	Target
1	Toffoli	0, 1	4
2	Pauli X	-	4
2	CNOT	3	4
3	CNOT	2	4
4	RX	-	4

Table 1: Configuration and sequence gates

The final RX rotation is arbitrary and placed there for parametrization purposes only: the rotation gate also reduces superposition in the input quantum data so that we can extract as much valuable information from the measurement as possible [1]. We initialized the parameters using the cirq line qubits and undertook a Z measurement since the Pauli Z matrix establishes the framework for the measurement. For training parameterized quantum models, we use PQC layer: this layer initializes the parameters and manages them in a Keras native manner given a parameterized circuit [2].

2.3 Results and Analysis

We used the Adam optimizer and learning rate of 0.003 and hinge loss as it is convex and suitable for data classification. For accuracy, we use the default TensorFlow Keras accuracy [10]. We fit the model using the train and test data with epochs of 512 and batch size of 4. We chose a small batch size with a smaller number of epochs because too large of batch size or the number of epochs could lead to poor generalization since our data is limited [10]. As we can see,

our model worked reasonably well in this situation. After the first 50 epochs, the loss significantly dropped up to 300 epochs, after which the loss reached its minimum and became constant. Both the training loss and the validation loss follow the same pattern.

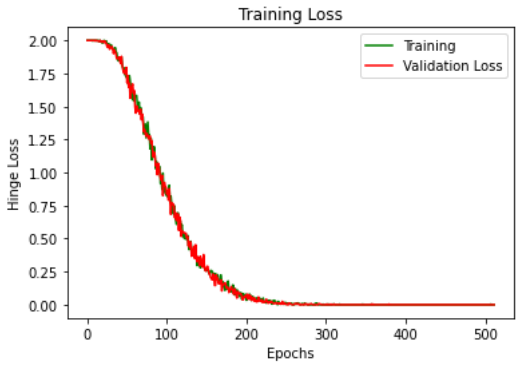

Figure 2: Loss function

The accuracy is similar to the loss in that it is zero for the first 150 epochs, in this case, suggesting that the model did not pick up much information during these early epochs. But after then, accuracy increased significantly, reaching its peak in about 300 epochs. Both the training and testing data show the same trend, but the testing data is slightly more reliable in the last few epochs.

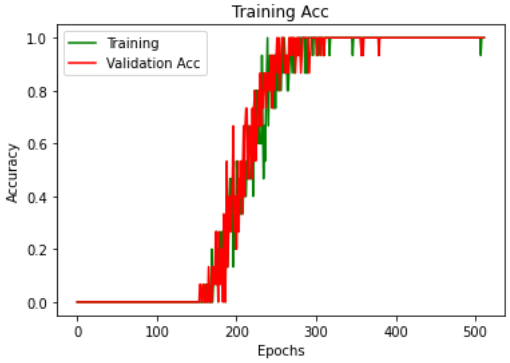

Figure 3: Accuracy

The tutorial and code for this project have been uploaded to github. Github link: GitHub Link

2.4 Conclusion

Based on these results, it seems fair to conclude that quantum circuits can be simulated by a machine learning model using TensorFlow Quantum. Our implementation was focused on the simulation of a circuit with multiple

quantum gates; however, it is possible to achieve similar designs for more complicated circuits as well.

2.5 Acknowledgment

We would like to give special thanks to Dr. Javier Orduz, the instructor, for this course on Quantum Computing. This class was very informative and has provided the two of us with good exposure to different elements of Quantum programming. We believe this experience will be helpful for us in the future, and we look forward to possible future collaboration with Dr. Orduz.

References

[1] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[2] Gavin E Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*, 2019.

[3] DeMarcus Edwards and Danda B Rawat. Quantum adversarial machine learning: Status, challenges and perspectives. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 128–133. IEEE, 2020.

[4] Arkady Fedorov, Lars Steffen, Matthias Baur, Marcus P da Silva, and Andreas Wallraff. Implementation of a toffoli gate with superconducting circuits. *Nature*, 481(7380):170–172, 2012.

[5] Richard P Feynman. Simulating physics with computers. In *Feynman and computation*, pages 133–153. CRC Press, 2018.

[6] Claudio Gentile and Manfred KK Warmuth. Linear hinge loss and average margin. *Advances in neural information processing systems*, 11, 1998.

[7] Andrew Hancock, Austin Garcia, Jacob Shedenhelm, Jordan Cowen, and Calista Carey. Cirq: A python framework for creating, editing, and invoking quantum circuits.

[8] Danyal Maheshwari, Daniel Sierra-Sosa, and Begonya Garcia-Zapirain. Variational quantum classifier for binary classification: Real vs synthetic dataset. *IEEE Access*, 10:3705–3715, 2021.

[9] Aaron Meurer, Christopher P Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov,

Jason K Moore, Sartaj Singh, et al. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, 2017.

[10] Pablo Rivas. *Deep Learning for Beginners: A beginner's guide to getting up and running with deep learning from scratch using Python.* Packt Publishing Ltd, 2020.

[11] Ferdinand Schmidt-Kaler, Hartmut Häffner, Mark Riebe, Stephan Gulde, Gavin Lancaster, Thomas Deuschle, Christoph Becher, Christian F Roos, Jürgen Eschner, and Rainer Blatt. Realization of the cirac–zoller controlled-not quantum gate. *Nature*, 422(6930):408–411, 2003.

[12] Colin P Williams. Quantum gates. In *Explorations in Quantum Computing*, pages 51–122. Springer, 2011.

[13] Peter Wittek. *Quantum machine learning: what quantum computing means to data mining.* Academic Press, 2014.