Quantum Neural Network Benchmarking with MNIST Dataset

Matthew Dobson, Glen S. Uehara, Dr. Andreas Spanias

Abstract - **The rise of machine learning as a commonplace technique for categorizing and classifying large sets of data has led to demand in greater computational power. Quantum computing is poised to provide drastically more number-crunching capability in smaller amounts of time using considerably less energy. The synthesis of these fields as quantum machine learning has the capacity to change modern data-driven technology sectors. To measure and evaluate potential capabilities of quantum machine learning models, this project will see the development of a quantum neural network that can operate on trivial databases such as the Modified National Institute of Standards and Technology. These quantum neural networks will be benchmarked on simulated quantum systems initially, with the future goal of operation on actual quantum hardware. This will help create a baseline of understanding how to leverage the inherent advantages of quantum computers for machine learning applications.**

Index terms – Machine learning, quantum computing, neural networks, MNIST

I. Introduction

Quantum computing (QC) is a pillar of future-leaning research in computation. Phenomena from quantum mechanics principles such as superposition and entanglement are applied to quantum bits or qubits, then leveraged for greater parallel computation power than classical Boolean computing. Unlike with classical binary bits, that can take on the value of 0 or 1, qubits are able to take on the value of 0, 1, or a superposition of both states. This will allow for the computation of enormous data sets at a fraction of the power cost, in a fraction of the time. The inherent parallelism of quantum computers lends itself to application areas such as Big Data, cryptography, machine learning, and more.

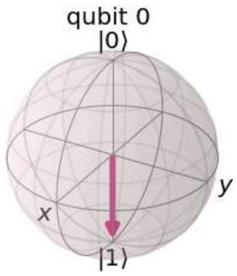


Figure 1: Bloch sphere, graphical representation of the possible qubit states.

Prior to discussion on the field of quantum machine learning (QML), it is important to provide some background on traditional machine learning, and application areas within. Machine learning (ML) is a field within computer science but has applications in several industries including but not limited to telecommuncations and power distribution [2]. Algorithms in ML are designed to disseminate through data sets, and differentiate data based on predefined categorizations. The process involves training an ML algorithm on a given data set, for example data on properties of a species of flower, to separate data based on trends. This generally requires larger sets of data to be performed adequately. Algorithms can then be tested against data of similar categorization to evaluate overall efficacy of the learning model. A popular area within ML is development of artificial neural networks (NN), which mimic the way that neurons in the brain process information and are helpful in fields of pattern recognition, outcome prediction and data classification [3] for innumerable various industries. More discussion regarding the synthesis between QC and NNs will occur later in this document. By performing these operations and techniques, the result is a programmatic method of "teaching" a computer how to classify types of data.

The synthesis of QC and ML comes in the form of QML where the powerful parallelized advantages of QC can be leveraged in the training of ML models. This could present itself in the form of improved model training times, as well as actual model operation on much larger data sets [4]. To benchmark potential improvements gained from QML methodologies, a common and well-known dataset will be able to guide technology development. For this initial research proposal, the Modified National Institute of Standards and Technology (MNIST) database comprised of handwritten numerical digits will be the target focus. MNIST is a well-known and comprehensively solved dataset within the field of ML [4].

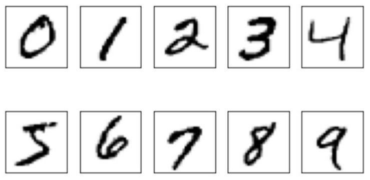


Figure 2: Selection of handwritten characters from MNIST dataset.

Using the python programming language and the quantum software development kit (SDK) Qiskit, the goal is to develop a type of quantum neural network (QNN). As QC hardware is still early in development, much of the modern work and research in the field is performed on quantum simulators that run on classical computer hardware [5]. This MNIST QNN will be benchmarked using quantum simulations initially, with the eventual possibility of operation on IBM's cloud-based quantum computer hardware. Several research groups are currently investigating the application of experimental machine learning algorithms on the MNIST dataset for its well-solved status [6][7][8], which further reinforces the selection justification.

II. Design

Primary project focus has been on investigating hybrid neural networks that leverage the quantum advantages during the training process, while keeping the simpler stages on the classical side for easier development. Design of the system will leverage the IBM quantum software dev kit called Qiskit for the quantum components, and the python machine learning package Pytorch for the classical components.

Qiskit and IBM quantum computers rely on programmatic abstraction of quantum circuits as the logical representation of a quantum algorithm. Through the Qiskit development kit, circuits are constructed from python language functions, and form the basis of a quantum computer program. A basic circuit for creating a data flow pipeline called a feature map is demonstrated in Figure 3.

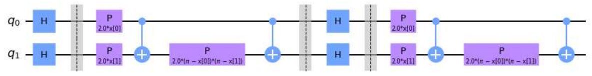


Figure 3: 2 qubit feature map quantum circuit.

Classical state preparation includes conditioning the data in a useful format using convolution image processing operations [9] and passing it into the quantum neural network (QNN) which performs forward and backward propagation operations to determine the network weights. The quantum component hooks into the classical layers with a torch connector module that performs the data pipeline construction between classical and quantum. Once complete, a single sample is output to be placed in the proper classification as decided by the QNN [10]. The dataset element is "guessed" as a true or false Boolean tested on each corresponding digit, for example going through a true or false assignment to determine if it is a 0, a 1, and so on [11].

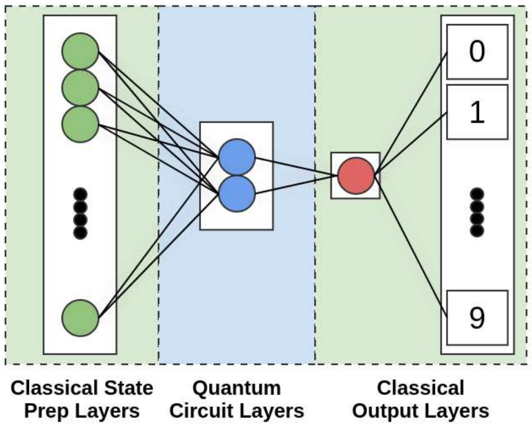


Figure 4: Hybrid Quantum Neural Network Architecture.

The example shown is relevant to the MNIST dataset. This is a well-defined, solved dataset that is excellent for benchmarking neural network efficacy. Two separate architectures were investigated when implementing this hybrid model to operate on. As quantum computing is still early in its capabilities, and being limited to a small number of qubits, the hybrid approach is necessary to perform this low-resolution machine learning [12][13].

A. Two-class Selection from MNIST

Initial designs revolve around training the hybrid model with a selection of two classes, or characters, from the MNIST dataset. This began using handwritten samples of 0 and 1 for training and testing and evolved to compare each pairing of characters. Training the model with only two classes simplified the quantum layers of the neural network, only requiring two qubits for operation. The efficacy of this approach is demonstrated in Table (). For the sake of brevity only classes 0 and 1-9 are presented. Code was modularized to ease development and testing of this system, where data loading and preprocessing, as well as training and testing were separated into discrete packages. This allowed for a looping approach to process each pairing of two handwritten digits for model training and testing as well as minimized the complexity of the program.

Table 1: Results from two-class loop design.

Class Selection (Digits)	Accuracy (%)
0 and 1	52.8
0 and 2	61.1
0 and 3	63.9
0 and 4	68.9
0 and 5	63.9
0 and 6	65.0
0 and 7	62.2
0 and 8	72.2
0 and 9	76.1

B. Full Ten-class MNIST

Final goal of the project was to develop a hybrid QNN that would be capable of operation on multi-class datasets, and in this specific case on the ten class MNIST dataset for benchmarking and development. Implementation of the full dataset led to significant challenges, as the number of classes would generally require more complex quantum layers with higher qubit counts. Several tests were conducted on model training efficacy while varying the number of qubits. Two, three and four qubit systems were tested, with results in Figure 5, with no other alterations performed on the test or model elements.

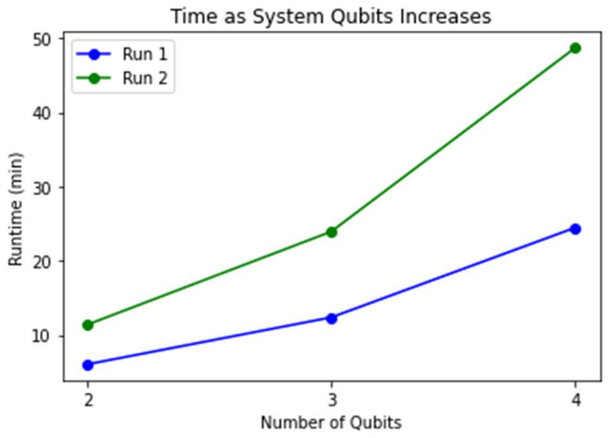


Figure 5: Time cost for increasing number of qubits.

These tests demonstrate that using the same underlying architecture methodology, increasing the number of qubits adds runtime duration. This is most likely an artifact of the simulation process, as the advantage of having more qubits in an actual quantum system would be fewer operations through more parallelization and overall higher resolution and performance. More on the simulation process in the next section. The resolution improvements should cross over into the realm of simulation, but results were inconclusive for the design thus far.

III. Simulations

Quantum computer hardware is not easily accessible or readily available, for this reason algorithm development occurs primarily on quantum computer simulators. These simulators operate on classical computers, where the possible quantum states and errors are built prior to the program runtime, which will generate an environment that works similarly to that of a quantum computer. The drawback of simulations is that the program runtime is considerably longer than would be in a real quantum computer architecture.

There are several varieties available within the qiskit software development kit, but focus is placed on qasm and statevector. Qasm executes quantum circuits that simulate noise and measurement errors within the entangled qubits. This creates the environment that more closely resembles actual quantum hardware. Statevector simulates a more idealized environment for the circuit, eliminating measurement errors allowing for focus to be placed on validating algorithms.

Statevector is a preferred environment to develop and evaluate algorithms, and as such is the starting point. As shown in Figure 6, the cost reduction exhibits the behavior desired from a neural network training process with the elbow-shape curve in the statevector sim, whereas the qasm simulation in Figure 7 is a less refined outcome.

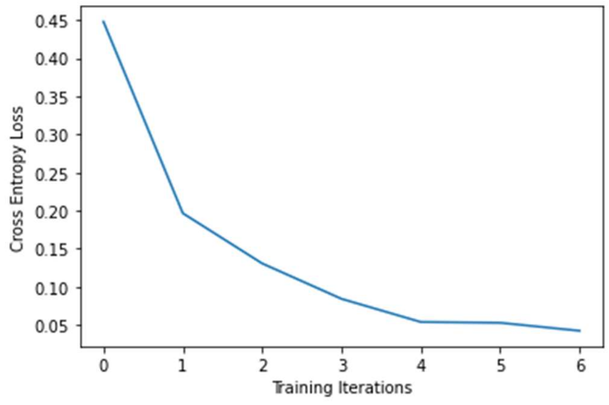


Figure 6: Statevector training cost reduction.

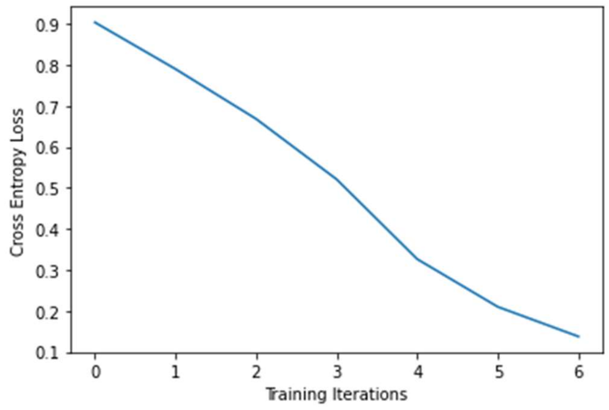


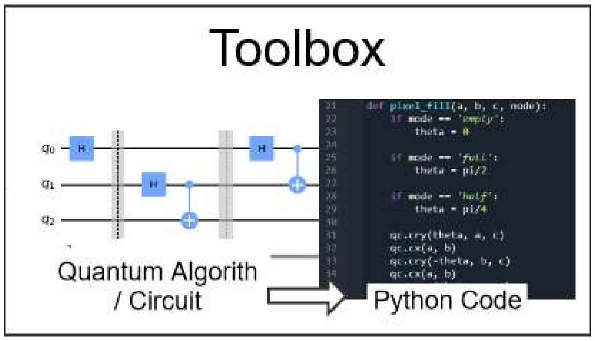
Figure 7: Qasm training cost reduction.

What this will introduce with the qasm sim is more randomness in results. With only two MNIST classes this isn't as readily apparent, accuracy is for all intents and purposes identical between the two simulators, but as the number of classes and number of qubits increases, the complexity of the system results in more noise that crosses over into runtime efficiency and accuracy.

IV. Conclusions and Future Work

Predominant takeaway from this project has been that the technical readiness level of machine learning applied to quantum computing is still in its early stages. There is inherent randomness in the quantum component of the models, in addition to the increased runtime duration seen from simulating quantum systems on classical hardware. This is expected to improve as the technology matures, and new algorithm implementations are developed in the quantum side.

Another deliverable for the project comes from performance improvements for photovoltaic fault detection using quantum machine learning. The QNN architecture developed significantly reduced machine learning model training time, leading to some optimizations in a related project and co-authorship of a research paper [14].



```
def pixel_fill(a, b, c, mode):
    if mode == 'empty':
        theta = 0

    if mode == 'full':
        theta = pi/2

    if mode == 'half':
        theta = pi/4

    qc.cry(theta, a, c)
    qc.cx(a, b)
    qc.cry(-theta, b, c)
    qc.cx(a, b)
```

Figure 8: Concept architecture for quantum computer program toolboxes.

Future work on this design will be expanded upon in undergraduate research programs moving forward. A direct run-on for this project is going towards developing machine learning, computer vision and image processing techniques applied to quantum computing systems. The emphasis will be on creating reusable modules, or toolboxes, that act as a black box solution and can speed up the development and testing of quantum algorithms, through abstracting away much of the complexity of the system. Secondary goal is to investigate capabilities and cross-over areas where quantum advantages can be leveraged in the advancement of machine learning applications and beyond [15].

V. Acknowledgment

This project is sponsored by the NSF Award 1659871.

References

[1] U. S. Shanthamallu, A. Spanias, C. Tepedelenlioglu, M. Stanley, "A Brief Survey of Machine Learning Methods and their Sensor and IoT Applications" SenSIP Center, School of ECEE, Arizona State University, NXP Semiconductors

[2] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, H. Arshad, "State-of-the-art in artificial neural network applications: A survey" Heliyon, Volume 4, Issue 11, Nov. 2018, e00938

[3] G. Uehara, A. Spanias, W. Clark, "Quantum Information Processing Algorithms with Emphasis on Machine Learning" School of ECEE, Arizona State University, General Dynamics Mission Systems

[4] A. Baldominos, Y. Saez, P. Isasi, "A Survey of Handwritten Character Recognition with MNIST and EMNIST" Applied Sciences, 3169;doi:10.3390/app9153169, Aug. 2019

[5] G. Uehara, "Quantum Machine Learning using Quantum Simulation" School of ECEE, Arizona State University.

[6] B. O. Kaziha "A comparison of Quantized Convolutional and LSTM Recurrent Neural Network Models Using MNIST". International Conference on Electrical and Computing Technologies and Applications, ICECTA 2019 (2019)

[7] A. Palvanov, Y. Choy "Comparisons of deep learning algorithms for MNIST in real-time environment". International Journal of Fuzzy Logic and Intelligent Systems (2018), 126-134, 18(2)

[8] D. Pedamonti “Comparisons of non-linear activation functions for deep neural networks on MNIST classification task”. Department of Computer Science, University of Edinburgh, (2018)

[9] S. Y. Simard, D. Steinkraus, J. C. Platt “Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis”. IEEE Computer Society, (2003)

[10] V. P. Ngoc, H. Wiklicky “Tunable Quantum Neural Networks for Boolean Functions” Imperial College London. ArXivID: 2003.14122v2

[11] E. Grant, M, Bendetti, S. Cao et al “Heirarchical Quantum Classifiers” Quantum Information, (2018), 1-8, 4(1)

[12] G. Carleo and M. Troyer, “Solving the quantum many body problem with artificial neural networks,” Science 355, 602 (2017).

[13] Bausch, J. and Leditzky, F., 2020. Quantum codes from neural networks. New Journal of Physics, 22(2), p.023005.

[14] G. Uehara, S. Rao, M. Dobson, C. Tepedelenlioglu, A. Spanias “Quantum Neural Network Parameter Estimation for Photovoltaic Fault Detection” IEEE IISA 2021 Conference, SenSIP Center, School of ECEE, Arizona State University

[15] Jia, Z.A., Yi, B., Zhai, R., Wu, Y.C., Guo, G.C. and Guo, G.P., 2019. Quantum neural network states: A brief review of methods and applications. Advanced Quantum Technologies, 2(7-8), p.1800077. [23] G. Carleo and M. Troyer, “Solving the quantum many body problem with artificial neural networks,” Science 355, 602 (2017)