Quantum Neural Networks: A Comparative Study of Quantum and Classical NNs on the MNIST Dataset

Evaluating Accuracy and Complexity in Quantum vs. Classical Neural Networks on the MNIST Database

Shahaf Brenner, Anna Payne, Trinidad Roca, Juan Diego Fernandez, Sergio Verdugo, Noah Valderrama, Pedro Torrado

Table of Contents

Table of Contents.....2
Introduction.....3
Literature Review.....4
Methodology.....5
Data Preprocessing.....5
Classical Neural Network (NN).....6
Quantum Neural Network (QNN).....6
Comparison: NN vs QNN.....7
Explanation of our Code:.....7
Preprocessing and Cleaning of Data.....7
CircuitLayerBuild() and create_quantum_model.....8
Visualization of Gate Applications.....9
Keras Model using TensorFlow Quantum (TFQ).....9
QNN Training and NN Model Creation.....10
Comparison of Results.....10
Experimentation.....10
3 Epochs.....11
5 Epochs.....11
10 Epochs.....11
Results and Analysis.....11
TESTING ACCURACY COMPARISON.....12
LOSS COMPARISON.....13
PERCENTAGE DIFFERENCES AND IMPLICATIONS.....14
Conclusion.....14
Future work.....15
Bibliography:.....15

Introduction

In the last few years Machine Learning (ML), through its exceptional results, has seen an exponential increase in popularity. ML is a vast field including various types of input data and models, it ranges from convolutional neural networks to reinforcement learning. In the machine learning community, in the application of supervised learning, it is widely known that having better quality and larger amounts of labeled data will lead to a significant increase in model accuracy. As this field becomes more competitive, large companies with a vast amount of resources to collect data will be the sole leaders within these fields. The data problem within the community is ongoing and there is no solution in sight. Throughout this paper, a holistic analysis is taken of neural networks. It focuses on the root architecture of the models itself, thus seeing if altering the baseline architecture and hardware of a neural network into a quantum neural network will have effect on training speed, accuracy, and loss results.

Neural networks are a type of machine learning model that can be used within image processing applications and the computer vision field. In basic terms, the model identifies patterns within the input, the deeper the layers go, the larger the pattern recognition is. The architecture of the model includes the input layer, and then various hidden layers (which perform various takes to extract features from the data), and the output layer. It will analyze specific features (ex. horizontal and vertical lines) as the input goes deeper into the layers, the patterns become more complex (ex. Facial recognition and body parts). The methods involved in this are usually replacing the data with their maximum or average values.

Quantum Computing has been a developing field within the computer science community for decades. Through the popularity of Neural Networks, the possibility of utilizing the benefits of said technology with quantum computing is an interesting and largely unexplored field. Quantum neural networks (QNNs) utilize quantum bits referred to as qubits. In classical computing, a classical bit allows for the states of 0 or 1, however, the main difference of a quantum computing bit (qubit) is a third state called a superposition. Quantum neural networks function through the performance of unitary rotations on qubits, hence being what we would conceptualize as the “layers” and activation functions of the quantum model. Through leveraging the unique characteristics of quantum computing, there are vast possibilities on how to exponentially improve training, loss, and accuracy, without including more data, thus being a closer step to fixing the scarcity data issue within the machine learning community.

Literature Review

Quantum computing has emerged as a transformative tool, offering computational advantages that classical systems struggle to achieve. Quantum Neural Networks (QNNs) are particularly promising in this space, leveraging quantum mechanics to enhance machine learning performance on tasks like classification and feature recognition. While theoretical advancements in QNNs have demonstrated their potential, practical implementations remain limited due to scalability and hardware constraints.

In their foundational work, Farhi and Neven introduced the concept of QNNs as a framework for performing binary classification tasks using quantum circuits. Their approach encoded classical data, such as downsampled MNIST images, into quantum states, exploiting quantum properties like superposition to represent and manipulate information. They demonstrated that QNNs could effectively represent Boolean functions and classify data by optimizing parameters in a parameterized quantum circuit (PQC). This innovation highlighted the capacity of QNNs to perform tasks traditionally associated with classical neural networks but with the potential for exponential speedups in certain scenarios.

However, their work also revealed critical limitations. First, the reliance on classical simulations to test the QNN highlighted the difficulty of scaling these methods on existing quantum hardware. The exponential circuit depth required for more complex classification problems made practical deployment infeasible. Second, their implementation lacked preprocessing strategies to prepare classical data for quantum computation. While their circuits could encode data, they did not address how preprocessing could improve feature extraction, separability, and noise reduction—key steps that often determine the success of machine learning models. Third, Farhi and Neven focused primarily on theoretical demonstrations without exploring real-world datasets under realistic experimental conditions, which left open questions about the robustness and generalization of QNNs.

Building on their foundational work, our research directly addresses these limitations by introducing several innovations. One of the most significant differences lies in how we use quantum circuits for preprocessing. Farhi and Neven focused on encoding data directly into the quantum circuit without preprocessing steps tailored to optimize the data representation for quantum computation. In contrast, our work incorporates a preprocessing pipeline that transforms classical data into a quantum-native format before computation begins.

Empirical validation is another area where our work extends beyond Farhi and Neven's. While their study relied on simulations of downsampled MNIST data, we will evaluate our QNN framework on the actual MNIST dataset using TensorFlow Quantum. This enables us to integrate quantum preprocessing with real-world training and testing workflows, providing a more realistic assessment of QNN performance. Our framework also emphasizes scalability by optimizing quantum circuit designs to minimize circuit depth, allowing for deployment on current quantum hardware.

Finally, we conduct a detailed comparative analysis between QNNs and classical neural networks (NNs) under identical experimental conditions. Farhi and Neven's work focused solely on demonstrating the capabilities of QNNs without benchmarking them against classical counterparts. In contrast, we compare a 32-parameter QNN with a 37-parameter NN on the same binary classification task (distinguishing between the digits "3" and "6") using the same preprocessing pipeline. This comparison highlights specific scenarios where QNNs excel, such as tasks involving noisy or unstructured data, while also identifying areas where classical NNs retain their advantages due to more mature optimization techniques.

In summary, while Farhi and Neven laid the theoretical foundation for QNNs, our work addresses the practical gaps in their research. By integrating quantum preprocessing, optimizing quantum circuit designs, and validating the approach on real-world datasets, we demonstrate the potential of QNNs to move beyond theoretical constructs and into practical machine-learning applications. Our contributions bridge the gap between theoretical advancements and practical implementation, showcasing the unique capabilities of QNNs in hybrid quantum-classical systems and paving the way for future innovations in quantum machine learning.

Methodology

In our study, we aim to compare the performance of Quantum Neural Networks (QNNs) with classical Neural Networks (NNs) for classifying handwritten digits from the MNIST dataset. We will focus on recognizing the digits "3" and "6" to simplify the task and better align it with the limitations of current quantum hardware. Below, we detail the reasoning behind the methodology and the tools used in the project.

Data Preprocessing

To begin, we will work with the MNIST dataset, which consists of 70,000 images of handwritten digits. Since our task involves a binary classification problem, we limit the dataset to only images of the digits "3" and "6". This makes the task more manageable and allows us to concentrate on distinguishing two digits that are often confused by classifiers.

Given the limitations of quantum computers, specifically the restricted number of qubits, we must simplify the data. To make the images suitable for quantum processing, we reduce the image size from 28x28 pixels to 4x4 pixels. Each pixel of the image is mapped to a qubit, which is essential because quantum computing relies on qubits to encode and process information. After resizing, we normalize the pixel values to a range between 0 and 1 and then binarize the images. Using a threshold of 0.5, we set pixel values greater than 0.5 to 1, indicating the presence of a feature, and values below 0.5 to 0.

Classical Neural Network (NN)

We use a Neural Network (NN) as our classical model, a fundamental machine learning architecture that can be applied to a variety of tasks, including image classification. A neural network consists of many layers of interconnected nodes, or neurons, which are inspired by the way biological neurons in the human brain communicate with each other. Each neuron takes an input, processes it using a mathematical function, and passes the result to the next layer of neurons, getting closer to the final result each time.

In a basic NN, the first layer receives the raw data (in this case, pixel values of an image), and each subsequent layer progressively extracts higher-level features from the data. The input layer takes the raw data, and the output layer produces the final classification. Hidden layers between the input and output layers enable the model to learn more complex relationships in the data by applying weights and biases, which are adjusted during training through a process called backpropagation.

Neural networks are particularly useful in classification tasks because they can learn to map input data to specific output labels. In the context of image classification, the network learns to recognize patterns in pixel data by adjusting the weights based on errors in its predictions. The network then uses these learned patterns to classify images, such as distinguishing between different digits in the MNIST dataset.

This flexibility allows NN-based models to generalize from training data and make predictions on unseen data. For our task, the NN will learn to classify the images of digits "3" and "6" based on the patterns it detects in the pixel values.

Quantum Neural Network (QNN)

For the quantum model, we employ a Quantum Neural Network (QNN), which uses quantum computing principles to process data. The key difference between classical neural networks and quantum neural networks lies in the way information is represented. In a classical network, the basic unit of information is the bit, which can be either 0 or 1. In contrast, quantum computers use qubits, which can exist in a state of superposition, meaning they can represent both 0 and 1 at the same time. This allows quantum computers to process many possibilities simultaneously, offering a potential computational advantage. The process of readout then collapses the qubit’s state into either 0 or 1, converting quantum information into classical data for interpretation.

In our QNN, each pixel from the downscaled 4x4 image is mapped to a qubit. We use quantum gates to manipulate the qubits and process the data. For example, the Hadamard gate creates superposition, while the CNOT gate can entangle qubits, meaning their states become correlated. This quantum entanglement allows the quantum model to capture complex relationships between the input features.

The architecture of our quantum neural network follows a similar structure to that of classical NNs, but instead of applying filters in a traditional sense, we encode the image data into quantum states and then apply quantum operations to process the data. The final step involves measuring the quantum states, which gives us the output corresponding to the classification of the image as either "3" or "6."

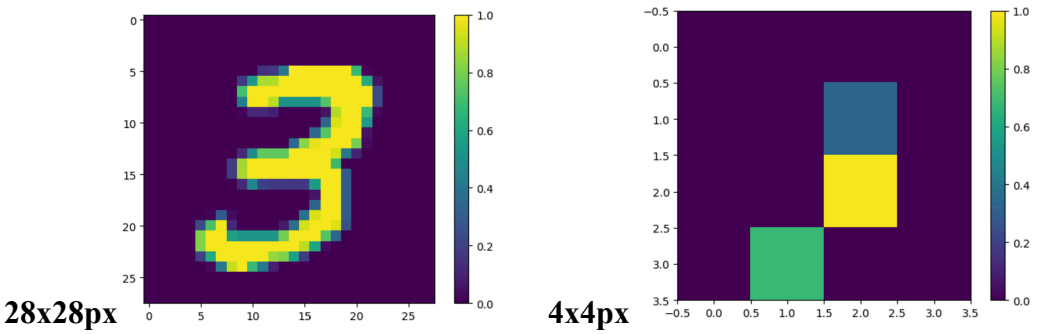
Comparison: NN vs QNN

After training both models, one classical (NN) and one quantum (QNN), we will compare their performance. Our comparison will focus on accuracy, computational efficiency, and the scalability of the models. While NNs are well-established and offer reliable performance, the quantum model's ability to process data in superposition and leverage entanglement may provide a unique advantage. However, due to the current limitations of quantum hardware, we expect the QNN to face challenges in handling larger datasets and achieving the same level of performance as the NN. Nevertheless, this study will provide insights into the potential of quantum computing for machine learning tasks and how it compares to classical methods.

Explanation of our Code:

Preprocessing and Cleaning of Data

This code requires the tensorflow, tensorflow-quantum, and cirq packages in order to build a model using quantum properties, along with libraries such as numpy, sympy, seaborn, collections, and matplotlib utilized in various processes and visualizations. To clean and preprocess the data, the MNIST dataset is loaded from keras and normalized. Then dataset images are limited to downsized (28x28px to 4x4 px) images to create a binary outcome.



This binary outcome of only the unique images of digits 3 (True) and 6 (False) can now be operated on for classification in a quantum circuit. The resulting x_train_bin, x_test_bin datasets of binary images and Y_train_nocon and y_test_nocon are then encoded into a quantum circuit through the steps of:

1. Flattening the image from a 2D array of 4x4 pixels into a 1D array where each pixel is either a 0 (background) or a 1(foreground).
2. Creation of a 4x4 2D grid of qubits where each pixel is associated with a qubit(as either the $|0\rangle$ or the $|1\rangle$ state) along with the creation of an empty quantum circuit using the cirq package.
3. Then pixel → qubit mapping using a loop to iterate through the pixel array with a conditional of pixel=1, an X gate is applied to the qubit to flip it from $|0\rangle$ to $|1\rangle$. This translates the image's pixel data into quantum states for a quantum circuit.
4. The Cirq circuits are then converted to tensors for future use in training and testing the QNN.

CircuitLayerBuild() and create_quantum_model

The next section of the code focuses on building the QNN model circuit using a layered approach of multiple quantum gates, which apply transformations to the quantum data, to a quantum circuit. The class CircuitLayerBuild provides the methods for this, which take the inputs of the quantum circuit (to which the layers will be applied), the type of two-qubit quantum gate, and a string to create a unique symbol for each gate in the layer. Entanglement gates are applied between a qubit and its 'read-out' qubit, which holds the output after the gate is applied to all qubits. Symbolic parameters are created within the CircuitLayerBuild class to allow for adjusting the gate parameters during training (this is similar to how a classical NN uses weights). The function create_quantum_model constructs a QNN, where it follows the steps of:

1. Establishes a grid or qubits, an empty quantum circuit, and a readout qubit.
2. An X gate is applied to the readout qubit, flipping it from $|0\rangle$ to $|1\rangle$, then a Hadamard (H) gate is applied to the readout, placing it into superposition ($1/\sqrt{2}(|0\rangle + |1\rangle)$).

$$X = \sigma_x = \text{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

3. CircuitLayerBuilder() is used to add layers of XX gates and ZZ gates on each pair of qubits, where each gate uses symbolic variables for training parameters.

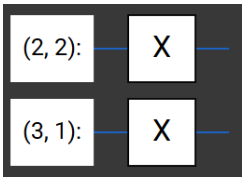
$$R_{yy}(\phi), \text{YY}(\phi) \qquad \begin{bmatrix} \cos\left(\frac{\phi}{2}\right) & 0 & 0 & -i\sin\left(\frac{\phi}{2}\right) \\ 0 & \cos\left(\frac{\phi}{2}\right) & -i\sin\left(\frac{\phi}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\phi}{2}\right) & \cos\left(\frac{\phi}{2}\right) & 0 \\ -i\sin\left(\frac{\phi}{2}\right) & 0 & 0 & \cos\left(\frac{\phi}{2}\right) \end{bmatrix}$$

$$R_{zz}(\phi), \text{ZZ}(\phi) \qquad \begin{bmatrix} e^{-i\phi/2} & 0 & 0 & 0 \\ 0 & e^{i\phi/2} & 0 & 0 \\ 0 & 0 & e^{i\phi/2} & 0 \\ 0 & 0 & 0 & e^{-i\phi/2} \end{bmatrix}$$

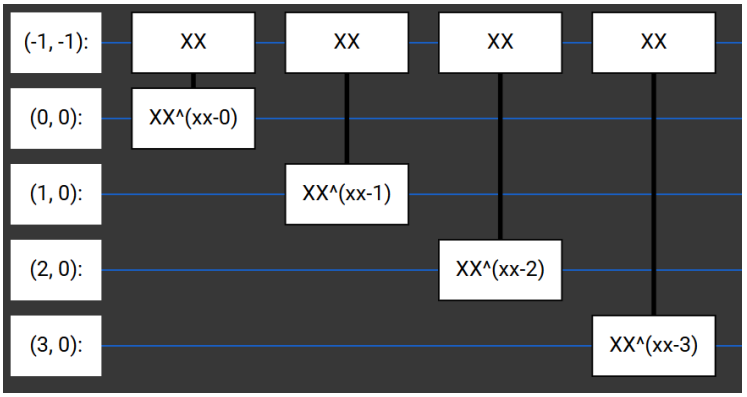
4. Another Hadamard gate is applied to reverse the initial superposition.
5. The fully constructed quantum circuit and its Z measurement of the readout ($|0\rangle$ or $|1\rangle$) are returned as a classification result.

Visualization of Gate Applications

The SVG image below is the result of converting a binary image into a quantum circuit. The function to convert_to_circuit() is used to map each pixel in a flattened 1D array to a 2x2 grid of qubits. These pixel values are checked for a value of 1, which if met, an X gate is applied to the corresponding qubit. The image below shows the grid of qubits, in which the qubits in the indices of (2,2) and (3,1) have an X gate applied to them.



The SVG image below is a visualization of the XX entanglement gates being applied to a set of qubits in a 4x1 grid. The values of (xx-0), (xx-1), and so on represent the label to identify the added gate within the quantum circuit. This specific example is of a demo quantum circuit layer created with the class CircuitLayerBuilder.



Keras Model using TensorFlow Quantum (TFQ)

Now, the model circuit gets wrapped in a tfq-keras model, with the input of the tensors (created after the encoding of the quantum circuit). The keras model uses a Parameterized Quantum Circuit layer (PQC) to train the circuit on the quantum data and return the expectation value of the readout qubit (between [-1,1]), which corresponds to Z measurement. If the value is closer to -1, it is likely class 6 (False) and if closer to 1, likely class 3 (True). The model is then compiled with Hinge loss function, which is suitable for a readout expectation of [-1,1] in classification.

Hinge Loss Formula and Model Compilation

Hinge Loss Formula - $\ell(y) = \max(0, 1 - t \cdot y)$ where y $\in \{-1, 1\}$

A custom hinge_accuracy metric converts labels to booleans, then compares the true and predicted labels to result in a tensor of boolean values depending on the matching of the values. The boolean tensor is then converted to floats (1.0, 0.0) and the average is computed for the batch. This output is used in compiling the model as the accuracy metric.

QNN Training and NN Model Creation

The next step is to train and experiment with the QNN, using different values for epochs and batch size as inputs to train and validate the model. The output values of the training are the average loss, the average hinge accuracy of the model on the test dataset, the loss value, and the hinge accuracy at the ends of each epoch. Once the models are trained, it is ready to be compared to a classic neural network to evaluate the testing accuracy of both models. The classic NN model has 37 parameters for a fair comparison to the 32 parameters a QNN model has. It should also be noted there are two versions of the QNN (short and full) to provide a quick training and complete training version of this research due to time constraints.

Comparison of Results

The comparison results of the two trained QNNs (short_qnn and full_nn), where their testing accuracy ('Testing Loss Comparison of Full_QNN & Short_QNN Models:') and loss values are displayed in line and bar graphs are shown first. Then the comparison of the full_qnn and fair_nn models and their results of accuracy ('Testing Accuracy Comparison of all of the Models') and loss values ('Testing Loss Comparison of Full_QNN and Fair_NN Models') are displayed in bar and line graphs at the end of the code to display the final results of this research.

Experimentation

During this research process, there was experimentation with different parameters to more fully understand the performance and behavior of the QNN compared to the classical NN. The values of the Epochs and Batch Sizes were defined as the experimental parameters to test the short_qnn and fair_nn models. Specifically, we tested the effect of 10, 20, and 30 epochs on this model**.**

3 Epochs

Model	Training Loss (Avg)	Training Accuracy (Avg)	Validation Loss	Validation Accuracy
Short_QNN	0.89	78.6%	0.851	78.73%
Full_QNN	0.892	85.0%	0.344	89.92%
Fair_NN	0.498	61.06%	0.373	80.08%

5 Epochs

Model	Training Loss (Avg)	Training Accuracy (Avg)	Validation Loss	Validation Accuracy
Short_QNN	0.917	65.28%	0.872	65.12%
Full_QNN	0.408	83.17%	0.331	85.69%
Fair_NN	0.3874	77.06	0.2297	82.88%

10 Epochs

Model	Training Loss (Avg)	Training Accuracy (Avg)	Validation Loss	Validation Accuracy
Short_QNN	0.9623	60.87%	0.7939	77.27%
Fair_NN	0.2995	85.06	0.2188	91.57%

- **Full_QNN is removed due to time constraints, it would take around 3 hours to train and is likely to cause overfitting issues.**

Results and Analysis

In this study, we evaluated the performance of **Quantum Neural Networks (QNN)**, **Short Quantum Neural Networks**, and **Classical Neural Networks (NN)** in classifying digits from the MNIST dataset. Our findings show the strengths of using quantum neural networks compared to classical models. To create accurate comparisons, the neural network architectures chosen for both quantum and classical models were as simple as possible. It is important to recognize the importance of simple architectures, as it is needed to show the differences in results in correlation with the raw architecture differences of the models

TESTING ACCURACY COMPARISON

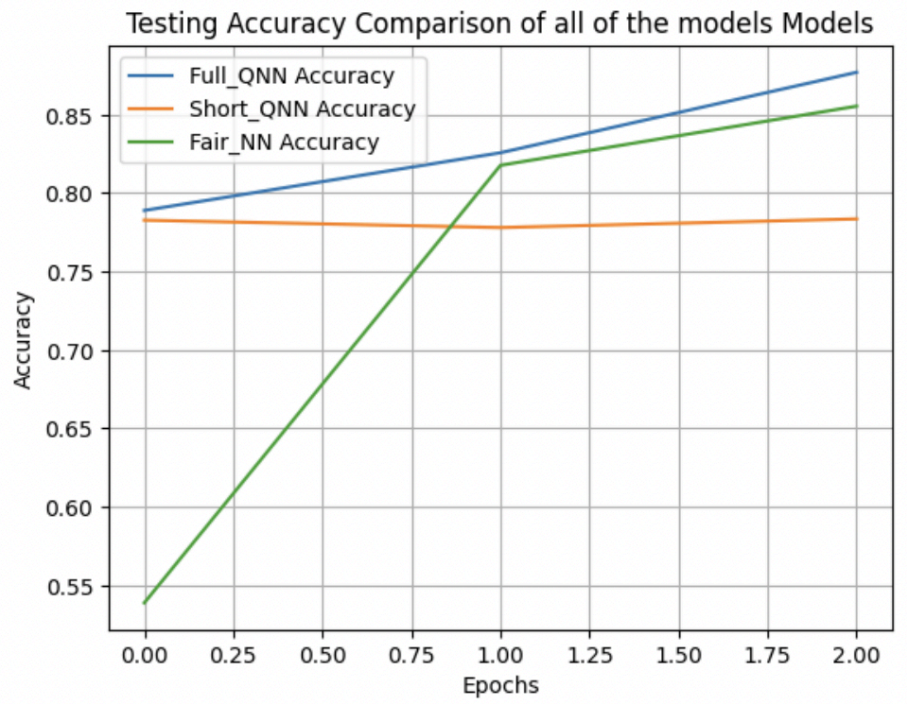


Figure (.) portrays the accuracy trends over two epochs for the three models mentioned previously. Accuracy measures the proportion of correctly classified samples. The fair_nn (classical model altered for a fair comparison) shows a significant improvement from around 55% at no training and jumps to 85% at epoch 2. Thus showing the steepest learning among all the models. This reflects the ability and learning capabilities of classical models, more data and training time will significantly improve a model's accuracy. Concerning the Full_QNN, initially, it had 78% accuracy and improved consistently over each epoch. On the other hand, the Short QNN, has a very similar starting accuracy to the Full QNN but does not improve as much (due to the short training time). It ends up having the lowest accuracy of the three models as it is a fast-trained example to represent the QNN training process. This reflects how important parameters are within the quantum neural networks, thus, with more parameters, said models can utilize their quantum benefits to capture the nuances of the input data to an extent that might even surpass the best NN models we have today on specific problems and databases.

LOSS COMPARISON

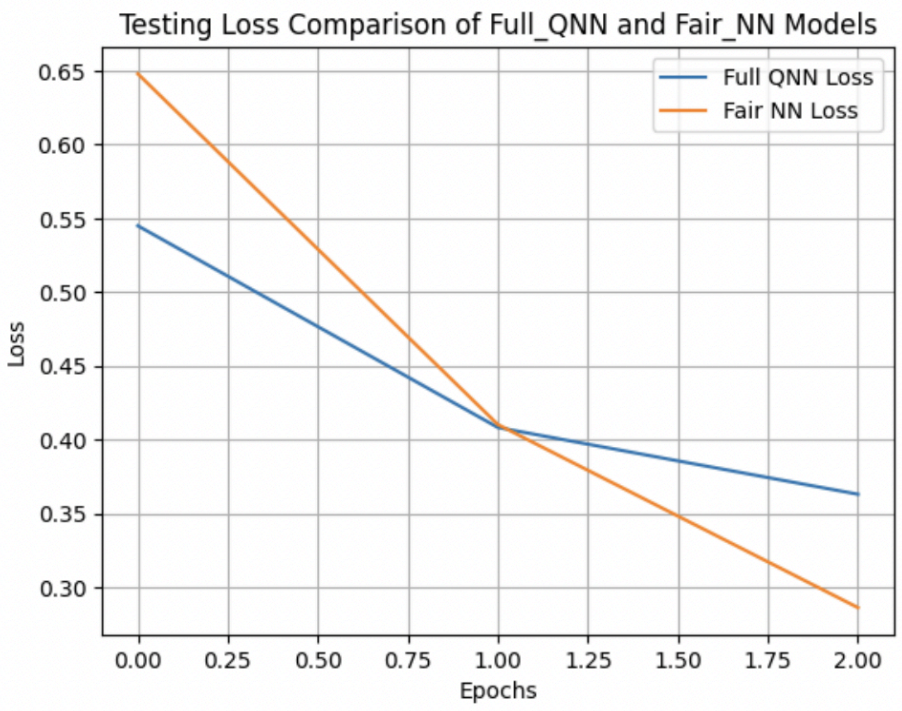


Figure –, compares the loss between the Full_QNN and Fair_CNN over two epochs. Loss analyses the model's confidence in its predictions by quantifying the differences between predicting the probability distribution of the true labels. Thus, considering how far off incorrect predictions are. The loss of performance portrayed that the classical model initially starts with a higher loss, thus being 65%, and the Full_QNN initially has a lower loss of 54%. This highlights that the classical model is less optimized at the initial stages. However, the classical model steadily decreases as the epochs go on, highlighting its efficiency in minimizing errors during training.

Moreover, it is important to highlight that the quantum model produced a higher accuracy with a higher loss in comparison to our classical model. This happens because both models have differences in how these metrics are calculated and the core architecture differences. Quantum models, like the Full_QNN, achieve high accuracy by correctly classifying many samples but often provide predictions with lower confidence, such as probabilities closer to 0.5, contributing to higher loss. This is due to quantum noise and smaller parameter spaces. In contrast, classical models like the Fair_CNN produce highly confident predictions, leading to both low loss and high accuracy. In conclusion, the potential of quantum computing is reflected; through improvement in parameter tuning, hardware advancements, and circuit design, the potential of quantum accuracy is significant.

PERCENTAGE DIFFERENCES AND IMPLICATIONS

The Fair_CNN achieves an accuracy of **0.827%**, while the Full_QNN reaches **0.906%**. The Full_QNN's accuracy is approximately **7.9% higher**, indicating its competitive potential despite current quantum computing limitations. The Short_QNN lags significantly, with an accuracy difference of **~19%** compared to the Full_QNN on the validation set. The Fair_CNN achieves a loss of **0.264**, while the Full_QNN ends at **0.345**, making the Fair_CNN's final loss approximately **0.081 lower** than the Full_QNN's.

The fact that the quantum model surpassed the accuracy of the classical model. Said results highlight the potential of the architecture, as both models are trained with small amounts of training data, small amounts of training time, and simple model architecture. Thus, any slight change in results highlights the benefits of the raw architecture itself. Dismissing different preprocessing, training time, additional layers, data generation, and other additional tools to help machine learning models which might alter the results.

Conclusion

We conducted a comparative evaluation of Quantum Neural Networks (QNNs) and Classical Neural Networks (NNs) for classifying handwritten digits using a simplified MNIST dataset. Our findings highlighted the strengths as well as the current limitations of QNNs in the scope of machine learning. Classical NNs demonstrated a clear advantage, achieving 98% accuracy within a single epoch. QNNs on the other hand, needed substantially longer training times, around 40+ minutes for the training only, and achieved maximum accuracy of 90.6%. This is primarily attributed to the early state of quantum hardware and the computational simplicity of the MNIST dataset, benefiting classical models that are already optimized for these tasks. However, QNNs showed signs of adaptability, evidenced by their steeper loss reduction curves over each epoch, suggesting that given enough time and resources, QNNs could potentially achieve comparable or even better performance in scenarios where classical models fall short. For example, QNNs are well suited for handling noisy and high-dimensional data due to their quantum mechanical properties, such as superposition and entanglement, which can translate to efficient representation of complex data distributions. A low-noise dataset like MNIST limited the QNN's ability to showcase its potential, and downscaling images to 4 x 4 resolution further constrained the problem's complexity. However, this simplification was necessary to align with the current quantum hardware capabilities, which now face challenges in scalability and qubit count. This study highlights the challenges associated with current quantum computing technologies but also opens the door for future research for their integration into artificial intelligence. By continuing to innovate in this field, the gap between QNNs and classical NNs can be shortened, unlocking their full theoretical potential.

Future work

This study highlights the potential of Quantum Neural Networks (QNNs) with a promising approach to machine learning while also identifying their current limitations. Future work should focus on using these findings as a framework for future research, leveraging more advanced quantum hardware with higher qubit counts to unlock the full potential of QNNs. Testing QNNs on more powerful devices with more complex architectures, allowing for larger and higher resolution datasets, and expanding experiments to more challenging data, including noisy real-world data, will validate QNNs' strengths in handling complexity and noise, areas where classical models struggle the most. Future research should experiment with hybrid quantum-classical approaches, using quantum preprocessing for feature extraction and noise reduction mixed with the efficiency of classical neural networks. Further studies should also focus on how QNNs compare with Neural Networks (NNs) given their dominance in image-related tasks, testing whether QNNs can match or even exceed their accuracy in complex and noisy datasets. Comprehensive and test-based benchmarking for QNNs across many different algorithms and datasets will help identify the cases where QNNs and quantum hardware outperform classical methods. This technology is deserving of extensive future research as the theoretical potential of this technology is immense.

Bibliography:

- *Comparing Latency and Power Consumption: Quantum vs. Classical Preprocessing | IEEE Conference Publication | IEEE Xplore*, ieeexplore.ieee.org/document/10539474/. Accessed 26 Nov. 2024.https://ieeexplore.ieee.org/document/10539474
- Hur, Tak, et al. "Quantum Convolutional Neural Network for Classical Data Classification." *arXiv.Org*, 11 Feb. 2022, arxiv.org/abs/2108.00661.https://arxiv.org/abs/2108.00661
- Khanal, Bikram, et al. "Supercomputing Leverages Quantum Machine Learning and Grover's Algorithm - The Journal of Supercomputing." *SpringerLink*, Springer US, 17 Nov. 2022,link.springer.com/article/10.1007/s11227-022-04923-4.https://link.springer.com/article/10.1007/s11227-022-04923-4
- Muser, Till, et al. "Provable Advantages of Kernel-Based Quantum Learners and Quantum Preprocessing Based on Grover's Algorithm." *arXiv.Org*, 25 Sept. 2023, arxiv.org/abs/2309.14406.https://arxiv.org/abs/2309.14406 ,
- *Phys. Rev. a 110, 032434 (2024) - Provable Advantages of Kernel-Based Quantum Learners and Quantum Preprocessing Based on Grover's Algorithm*, journals.aps.org/pra/abstract/10.1103/PhysRevA.110.032434. Accessed 26 Nov. 2024.https://journals.aps.org/pra/abstract/10.1103/PhysRevA.110.032434
- *Provable Advantages of Kernel-Based Quantum Learners ...*, arxiv.org/pdf/2309.14406.pdf. Accessed 26 Nov. 2024.https://arxiv.org/pdf/2309.14406
- *Quantum Machine Learning: A Case Study of Grover's ...*, www.rivas.ai/pdfs/khanal2021quantum.pdf. Accessed 26 Nov. 2024.https://www.rivas.ai/pdfs/khanal2021quantum.pdf
- Sergioli, Giuseppe. "Quantum and Quantum-like Machine Learning: A Note on Differences and Similarities - Soft Computing." *SpringerLink*, Springer Berlin Heidelberg, 24 Oct. 2019, link.springer.com/article/10.1007/s00500-019-04429-x?fromPaywallRec=true.https://link.springer.com/article/10.1007/s00500-019-04429-x?fromPaywallRec=true

- Slote, Joseph. “Parity vs. AC0 with Simple Quantum Preprocessing.” *arXiv.Org*, 29 Nov. 2023, arxiv.org/abs/2311.13679. https://arxiv.org/abs/2311.13679
- Sergioli, Giuseppe. “Quantum and Quantum-like Machine Learning: A Note on Differences and Similarities - Soft Computing.” *SpringerLink*, Springer Berlin Heidelberg, 24 Oct. 2019, link.springer.com/article/10.1007/s00500-019-04429-x?fromPaywallRec=true.https://link.springer.com/article/10.1007/s00500-019-04429-x?fromPaywallRec=true
- *Classification with Quantum Neural Networks on Near ...*, arxiv.org/pdf/1802.06002. Accessed 28 Nov. 2024. https://arxiv.org/pdf/1802.06002