ISSN:0975-9646
Agastya Kompella/ (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 13 (1) , 2022, 24-27

Think Different: Applications of Artificial Intelligence in Quantum Computing

Agastya Kompella
Sanskriti School, Chanakyapuri, New Delhi

Abstract-The paper will aim to be a primer on Quantum computing and its basics in addition to analyzing the current state of Quantum Computing and its limitations in the field of Artificial Intelligence by comparing the results between a Quantum Neural Network and Classical Neural Networks when trained, validated and tested on the Modified National Institute of Standards and Technology (MNIST) datasets.

1. INTRODUCTION

A traditional computer that we know of represents information in the form of tiny switches also known as bits (0 or 1). Every digital object we know of is made up of millions of combinations of these bits. This form of representation however poses limitations and challenges as it is not an accurate representation of the world around us. For one, this form of representation limits even the state-of-the-art supercomputers to execute sequentially. This means that to test every possibility or outcome for a problem, each of them needs to be run individually and validated to get the results leading to an immense amount of time to be spent on operations.

Another limitation is that representing information in the form of bits does not reflect the way the universe works or interacts. In reality, there may be a correlation between two pairs of information or events. This can be represented accurately in quantum computers due to the concept of quantum entanglement which in binary form cannot be inherently factored in.

In addition, in the real world every outcome is uncertain and probabilistic and a binary form of representing information does not factor that in. The paper will now aim to explain some of the main concepts that govern quantum computing and will provide analogies to the concepts for a clear understanding.

Qubits

Quantum computing, unlike a regular computer, leverages quantum mechanics and the concept of qubits. Qubits are typically subatomic particles such as electrons and photons that do not represent information as either on or off, but as both on and off at the same time or they are always somewhere in between. However, when they are measured each qubit can be only in a 1 or 0 state. To give an analogy, the concept of qubits is similar to flipping a coin.

When a coin is flipped the result can be either heads or tails. When the coin is flipping the coin is both 1 or 0. However, when an individual measures the result of the coin flip the outcome is always either heads or tails and never a state in between. To achieve this property of existing in multiple states at the same time until measured, qubits rely on two principles: Superposition and entanglement.

Superposition

Superposition in the context of quantum computing refers to the idea that qubits can exist in both states 0 and 1 at the same time. That is that they can be present in either state with some relative probability. In superposition, the state is a linear combination of an infinite number of states between 0 and 1. The state can be determined only when measured and this is known as quantum measurement. This means that for an n number of qubits, they can exist in a superposition of 2^n states allowing us to store as many configurations at any given point in time. This in contrast to n classical bits can only represent one single configuration or state at any given point in time.

An example illustrating the concept of superposition is that of a single photon passing through a beam splitter. Until the photon is detected, the photon has a state of presence in both arms of the splitter with some probability. Again, when measured through the photon is either on one side of the splitter or the other.

Quantum Entanglement

Quantum entanglement is the phenomenon where two qubits have a close connection where they react to each other state changes regardless of the distances between them. Changing the state of one qubit instantaneously allows for the state of another qubit to change predictably. This means that when one qubit is measured, the state and properties of the other entangled qubit can also be deduced. This property allows for an exponential increase in a quantum machine's crunching ability as to describe entangled states using ordinary bits is extremely time computation intensive.

To provide an analogy on the phenomena, the idea of entanglement is similar to that of reading a book. Reading a singular page might not give us enough information about the book but when all the pages are read together and the correlations between the pages are accounted for then all the information in the book can be deduced.

2. HYPOTHESIS

I believe that quantum computing is the future of modern computing and harnessing its properties will allow us to solve many interesting problems. In this paper, I aim to show that the current state of quantum computing allows us to build meaningful applications that are comparable with the applications and systems built on the classical computers we have today.

The paper aims to compare results for the accuracy and training times between a quantum neural network and classical neural networks when trained and validated on the MNIST dataset. I hypothesize that the results obtained

from each of the methods are comparable and will result in accurate prediction models.

3. DATASET

The MNIST dataset is a large collection of handwritten numbers that are used to train image recognition models. Each image is black and white and is of size 28 x 28 pixels. The background is black and the number in the image is represented by white pixels. In addition, the dataset contains 60,000 training images and 10,000 testing images. This gives us the training set to have a dimension of 60,000 x 28 x 28 and the testing set to have a dimension of 10,000 x 28 x 28. Each image has an associated label with labels ranging from 0 - 9.

4. CLASSICAL NEURAL NETWORK

To implement the classical forward neural network (FNN) where the model is a linear stack of layers, I used the Keras module in python while running each of the cells on a Jupyter notebook to produce the results.

Preprocessing

To pass the training data into the sequential model, I flattened each of the data inputs from 28 x 28 into a 1-dimensional vector of size 784 x 1.

In addition, I scaled each of the images from the 0 - 255 scale which represents each pixel to a normalized scale where each value lies between 0 and 1.

The last preprocessing step before forming the model was representing the result in a more desirable fashion where the result is not represented by a singular class value between 0 - 9 but by a vector of length 9 where the expected label has a value of 1 while other values in the vector are 0. For example, the class label 3 would be translated to be: [0, 0, 1, 0, 0, 0, 0, 0, 0, 0]. This way the result from our neural network while testing can be a vector of probabilities where each position represents the likelihood of the class label being represented in the image. This way, the argmax (position associated with the maximum value) of the output vector from the neural network would give us the required class label.

Model Layers

Layer 1: Dense neural network, Nodes: 1024
  Activation Function: ReLU
  Dropout: 0.2
Layer 2: Dense neural network, Nodes: 1024
  Activation Function: ReLU
  Dropout: 0.2
Layer 3: Dense neural network, Nodes: 10
  Activation Function: Softmax
Loss Function: Cross-Entropy
Optimizer: Adam
Batch Size: 512
Epochs: 5

Results

Accuracy of the model on test data: **97.96 %**

5. CONVOLUTIONAL NEURAL NETWORK (CNN)

The second neural network that will be built to be trained on the data is a convolutional neural network. The network aims to identify key features in the image by passing filters also known as kernels over the image which is a dot product application over the entire image. This along with a process of max pooling which essentially picks the max value in a subset leaves us with a feature map with only the important details.

Preprocessing

In the case of a CNN, we do not flatten the image unlike in the case of the fully connected neural network. Every other preprocessing step remains the same as before.

Model Layers

Layer 1:
  Convolution Layer, Kernels: 32, Kernel Shape: 3 x 3
  Activation Function: ReLU
  Dropout: 0.2
Layer 2:
  Convolution Layer, Kernels: 32, Kernel Shape: 3 x 3
  Activation Function: ReLU
  Max Pooling: 2 x 2
Layer 3:
  Convolution Layer, Kernels: 64, Kernel Shape: 3 x 3
  Activation Function: ReLU
  Max Pooling: 2 x 2
Layer 4: Dense neural network, Nodes: 512
  Activation Function: ReLU
  Dropout: 0.2
Layer 5: Dense neural network, Nodes: 10
  Activation Function: Softmax
Loss Function: Cross-Entropy
Optimizer: Adam
Batch Size: 512
Epochs: 6

Results

Accuracy of the model on test data: **98.94 %**

6. QUANTUM NEURAL NETWORK (QNN)

A quantum neural network allows for classical or quantum data to be trained by supervised learning as was the case with the above two models. They consist of a sequence of quantum circuits that transform input quantum states to a qubit result which is then measured to get the results. In order to implement a QNN we use the TensorFlow Quantum library and its in-built functions.

Preprocessing

Due to the limitations of the quantum computing resources available to perform the experiment the data had to be reduced to just two classification values. The two classes I chose to train the QNN on were classes 0 and 1. Due to the size of the current quantum computers, the images also had to be downsized from a 28 x 28 pixel size to 4 x 4.

To pass an input into a quantum neural network, data needs to be represented in the form of quantum circuits. Since we are working with images, a direct comparison can be made

between pixels and images where each pixel value can be represented by a qubit. The state of each qubit, since limited to either 0 or 1, leaves us to transform each of the pixels to conform with the values by normalizing the 4 x 4 image and applying a round function to each of the pixels. Each of the normalized data as qubits needs to be transformed and represented in the form of quantum circuits now. A quantum circuit is a combination of quantum gates that are applied to transform the qubits individually and in a correlated fashion. An example of a quantum gate is an X gate which is like the traditional NOT gate we are familiar with but one that is applied on qubits and helps switch a qubit with a probabilistic value of 0 to be 1 and vice-versa. There is also a designated output qubit assigned that will allow us to determine the prediction of the model.

Model Layers

In the model, we apply multiple layers of gates such that each of the qubits representing data transformed from the pixel representation acts and transforms the output qubit. This is possible due to the principle of quantum interference we discussed above.

We use the quantum circuit that we created that is a combination of X, Y, XX and ZZ gates to pass into a Parameterized Quantum Circuit (PQC). A PQC is a hybrid-classical machine learning quantum model that trains on quantum circuits to produce results.

Layer 1: Parametrized Quantum Circuit,
  Input: Quantum Circuit applied on 4 x 4 Qubit grid
  Result: Output Qubit
Loss Function: Cross-Entropy
Optimizer: Adam
Batch Size: 32
Epochs: 4

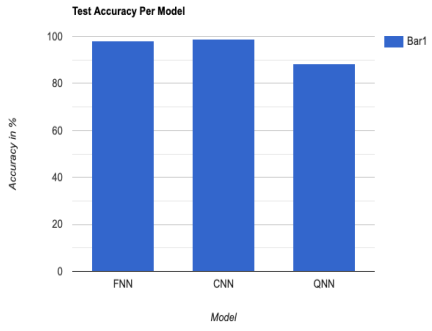
Results

Accuracy of the model on test data: **87.50 %**

7. Comparison Of The Results

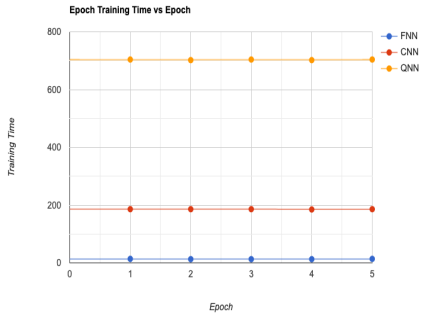
Below we shall compare the accuracy, number of epochs taken to train the model before overfitting, and the training time per epoch across the three models.

Accuracy comparison of each of the models



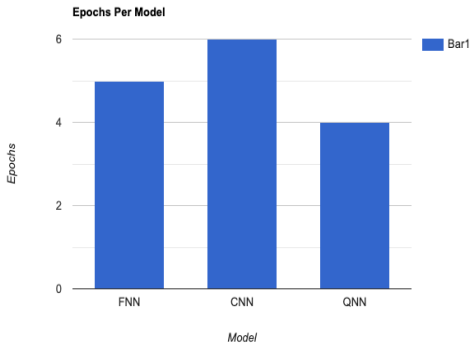
As one can see the quantum neural network was the least accurate of the three models with the convolutional neural network being the most accurate.

Training time per epoch



The training times of the three models are vastly different for each epoch given similar batch sizes. The reason can be attributed to the number of layers in each model and the complexity in the computations in finding the right weight vectors *for each of the nodes in the neural network.*

Number of epochs to minimize loss function and not overfit the model



In this comparison, each of the models has a comparable number of epochs to train the model before overfitting the data. I came to this conclusion by validating the model on a subset of the training data and found the optimal number of epochs at the point where validation accuracy started to reduce.

8. Conclusion

In today's day and given the current problem space, classical models are superior to the quantum systems existent when applied to classical data. This can be attributed to the fact that classical data can be accurately represented and trained on by the current systems as opposed to the quantum neural network which requires for the data to be transformed and represented in a quantum fashion. In addition, the computing power and number of coherent qubits available also limit the amount of information that can be encoded and used as features in order to make predictions. In our case, even after reducing the output classes and the training size of the data, the accuracy of the quantum model was lower than the classical neural networks. This just shows that as quantum computers become a more prevalent computing system, classical computers will still remain in existence with both solving problems that they are each best suited for.

9. FUTURE WORK

This paper aims to apply the quantum neural network system to a classical problem where the classical models outperformed the quantum neural network. The next step would be to harness some of the inert properties of the quantum neural network and apply it on a quantum system (e.g. particle reactions) and then compare the results and accuracy of a classical neural network with a quantum neural network.

10. REFERENCES

1. Beatrice, Adilin. "Every Thing You Need to Know about Quantum Computers." *Artificial Intelligence, Big Data Analytics and Insight*, 22 Feb. 2021, https://www.analyticsinsight.net/every-thing-you-need-to-know-about-quantum-computers/.
2. Bradben. "Understanding Quantum Computing - Azure Quantum." *Azure Quantum | Microsoft Docs*, https://docs.microsoft.com/en-us/azure/quantum/overview-understanding-quantum-computing.
3. Fisher, Chris. "IBM: What Is Quantum Computing?" *IBM Quantum*, 2 Apr. 2009, https://www.ibm.com/quantum-computing/what-is-quantum-computing/.
4. Giles, Martin. "Explainer: What Is a Quantum Computer?" *MIT Technology Review*, MIT Technology Review, 20 Oct. 2021, https://www.technologyreview.com/2019/01/29/66141/what-is-quantum-computing/.
5. Giles, Martin. "Explainer: What Is a Quantum Computer?" *MIT Technology Review*, MIT Technology Review, 20 Oct. 2021, https://www.technologyreview.com/2019/01/29/66141/what-is-quantum-computing/.
6. "Google Colaboratory." *Google Colab*, Google, https://colab.research.google.com/github/AviatorMoser/keras-mnist-tutorial/blob/master/MNIST%20in%20Keras.ipynb#scrollTo=J2_7X96U3-cU.
7. Katwala, Amit. "Quantum Computing and Quantum Supremacy, Explained." *WIRED UK*, 5 Mar. 2020, https://www.wired.co.uk/article/quantum-computing-explained.
8. Kerenidis, Iordanis, and Alessandro Luongo. "Quantum Classification of the MNIST Dataset with Slow Feature Analysis." *ArXiv.org*, 19 Nov. 2020, https://arxiv.org/abs/1805.08837.
9. "MNIST Classification : Tensorflow Quantum." *TensorFlow*, https://www.tensorflow.org/quantum/tutorials/mnist.
10. Qiskit. "Building a Quantum Variational Classifier Using Real-World Data." *Medium*, Qiskit, 9 Apr. 2021, https://medium.com/qiskit/building-a-quantum-variational-classifier-using-real-world-data-809c59eb17c2.
11. Siegelwax, Brian N. "Quantum Mnist." *Medium*, Towards Data Science, 16 Sept. 2020, https://towardsdatascience.com/quantum-mnist-f2c765bdd478.