QUANTUM IMAGE CLASSIFICATION WITH QUANTUM CONVOLUTIONAL NEURAL NETWORK

Yingsi Qin*, **Ruoshi Liu***, **Haixu Liu***, **Jack Shi***

1 INTRODUCTION

1.1 CONVOLUTIONAL NEURAL NETWORK

Neural networks are computing systems that are widely used in today's artificial intelligence research areas such as computer vision. Although the neural networks are powerful tools to model complex patterns, training a modern neural network consumes massive computational resources and electric power. Convolutional neural network is a special type of neural network that specializes in extracting spatial information from a matrix of numbers. It has been widely used in computer vision research as the backbone model to perform various advanced computer vision tasks such as object detection, semantic segmentation, visual tracking and etc. Due to the massive amount of parameters in a deep convolutional neural network, training a model to perform a certain task is very computationally expensive, especially when CNN is applied to analyze videos.

1.2 QUANTUM COMPUTING

Quantum Computing uses quantum-mechanical theories such as superposition and entanglement to perform computation. Qubits, fundamental to quantum computing, are analogous to bits in a classical computer, a qubit can be in a 1 or 0 quantum state, or they can also be in a superposition of the 1 and 0 states. However, when we measure the result of a qubit, it is always either 0 or 1, the probabilities of the two outcomes depends on the quantum state that they are in. A quantum computer will have the advantage of using ones, zeros and superposition of ones and zeros, therefore, it can process a vast number of calculations simultaneously.

1.3 QUANTUM MACHINE LEARNING

Quantum machine learning refers to machine learning algorithms for the analysis of classical data executed on a quantum computer, it increases the capability to compute immense quantities of data intelligently, by taking advantage of quantum states and systems.Such technique typically require one to encode the given classical data set into a quantum computer to make it accessible for quantum information processing. Subsequently, quantum information processing routines are applied and the result of the quantum computation is read out by measuring the quantum system.

1.4 RESEARCH PROBLEM

In this project report, we demonstrate a novel approach to image classification by classifying qubit-encoded images with Quantum Convolutional Neural Network (QCNN). The main contributions of our project are twofold: (1) We use quantum image processing to represent pixels as quantum states and use these qubit-encoded images, rather than classically represented images, directly as input of QCNN, thus preserving important spatial information while reducing the computational space complexity by $O(\log(N))$; (2) In prior work, QCNN has only been proved effective on real-world 1-dimensional data, and we confirm QCNN's ability to classify 2-dimensional data and reduce the computational time complexity by $O(\log(N))$. We demonstrate the effectiveness of our approach by training QCNN to classify synthetic 2-dimensional images.

2 RELATED WORK & CONTRIBUTION

2.1 QUANTUM DIGIT CLASSIFICATION

Classical neural networks can classify hand written digits with nearly-perfect accuracy. The archetypal example, developed by LeCun et al. (2010), is the MNIST data set which consists of 55,000 training samples that are 28

*indicates equal contribution.

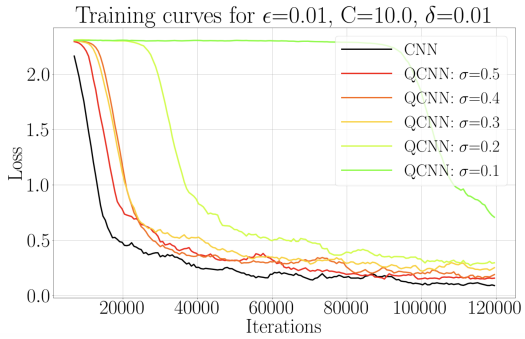

Figure 1: Training curves of classical CNN and QA-CNN when trained to perform image classification on MNIST dataset

by 28 pixelated images of hand written digits ranging from 0 to 9 labeled by humans. MNIST digit classification has since been used as a testbed for studying classical neural networks, so it seems natural that we study the effectiveness of quantum neural networks in this setting. Farhi & Neven (2018) developed a quantum neural network (QNN) framework that learns to correctly distinguish the digits. But the proposed method has several profound limitations. First, the training images are downsampled from 28×28 to 4×4, so important spatial features and correlations are lost. Second, Farhi & Neven (2018) proposed representing each pixel with a qubit, with the state depending on the value of the pixel. Thus, the spatial complexity for the classical and quantum representations of the input images are the same. Finally, QNN only accomplishes to perform classification of two distinct digits, whereas in the MNIST dataset, there are 10 distinct digits.

2.2 Quantum-accelerated CNN (QA-CNN)

Although with the same name as Cong et al. (2019) – Quantum Convolutional Neural Network, Kerenidis et al. (2019) takes a different approach. Cong et al. (2019) processes an image encoded in qubits, while Kerenidis et al. (2019) processes classical images with quantum-accelerated forward and backward propagation. Therefore, we will refer to this model as Quantum-accelerated Convolutional Neural Network (QA-CNN). There are two main contributions by QA-CNN: proposed a QCNN layer and a quantum backpropagation algorithm. The highlight of this paper is an image classification experiment performed on the MNIST dataset. A comparison between classical CNN and QA-CNN is provided in the trainig curve 1.

2.3 Quantum Edge Detection Algorithm

Classically, edge detection methods compute image gradients by different types of filtering masks in $O(2^n)$, a highly efficient quantum algorithm named Quantum Hadamard Edge Detection(QHED) can provide an exponential speed-up to $O(1)$, means the algorithm is independent of the image size. Basically, a Hadamard gate H is applied to detect the boundary. Apply the Hadamard gate to an $2^{n-1} \times 2^{n-1}$ identity matrix, the total operation is then $I_{2^{n-1}} \otimes H$. For an n-qubit input image state $|f\rangle = \sum_{k=0}^{N-1} c_k |f\rangle$ ($N = 2^n$pixels), we have the output image state $|g\rangle = (I_{2^{n-1}} \otimes H) |f\rangle$, here we check the even elements of the resulting state, if the two pixels belong to the same region, their intensity values are identical and the difference vanishes, otherwise their difference is non-vanishing, which indicates a region boundary. Therefore, this procedure yields the horizontal boundaries between pixels at position 0/1, 2/3, etc.

To obtain the boundaries between the remaining pairs 1/2, 3/4, etc., apply the n-qubit amplitude permutation to input, yielding a new image representation $|f'\rangle$ with its odd(even) elements equal to the even(odd) elements of the original input.the quantum amplitude permutation can be performed in $O[poly(n)]$ time. The QHED algorithm generates a quantum state into classical information will require $O(2^n)$ measurements but if the goal is to discover if a specific pattern is present in the picture, a measurement of single local observable may be sufficient.

3 METHOD

There are two stages in our method of classifying 2-dimensional images: first, we process the classically represented images and convert them into images represented by quantum states; next, we use the encoded images directly as input to learn a QCNN classifer. We will describe these two stages in detail below.

3.1 QUANTUM IMAGE PROCESSING

Representing images classically pixel by pixel requires significantly large amount of computational resources, especially when operations are performed on them. As a much more efficient approach, we encode the images in quantum information by representing the classical bits of information using qubits. Specifically, we adopt the method Flexible Representation of Quantum Images (FRQI) proposed by Le et al. (2011). This method allows us to encode an $2^N * 2^N$ image bits of information in $2N + 1$ qubits. The intensity values are encoded in the amplitudes of the quantum states and the images will be retrieved via normalized probability distributions. We encode the states as:

$$|I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} (sin(\theta_i)|0\rangle + cos(\theta_i)|1\rangle)|i\rangle$$

where θ_i corresponds to the intensity of the ith pixel. For example, an $2x2$ image with intensity values

$$\begin{bmatrix} \theta_1 & \theta_3 \\ \theta_2 & \theta_4 \end{bmatrix}$$

will be encoded as:

$$\frac{1}{2}[cos(\theta_1) + sin(\theta_1)|00\rangle + (cos(\theta_2) + sin(\theta_2))|01\rangle + (cos(\theta_3) + sin(\theta_3))|10\rangle + (cos(\theta_4) + sin(\theta_4))|11\rangle]$$

We chose FRQI over other quantum image processing methods such as NEQR and 2D-QSNA, which are discussed by Dendukuri & Luu (2018), due to its low complexity and practicality. The intensity values are encoded in the amplitudes of the quantum state, so that it is straightforward to apply various transformation like QFT directly on the amplitudes later in our QCNN. However, this method limits us to be processing images of dimensions $2^N \times 2^N$ only. It also requires extra qubits to encode the positions along with the pixel intensities.

The FRQI states can be achieved via a circuit of Hadamard and control y rotation gates. Figure 3 shows our implementation of the circuit that encodes FRQI states.

3.2 QUANTUM CONVOLUTIONAL NEURAL NETWORK

Motivated by the classical convolutional neural network architecture, Cong et al. (2019) introduced a quantum circuit model that extends the key properties of CNN to the quantum domain. As shown in Figure 2, the circuit's input is an unknown quantum state ρ_{in}. A convolution layer applies a single quasi-local unitary (U_i)in a translationally invariant manner for finite depth. For pooling, a fraction of qubits are measured, and their outcomes determine unitary rotations (V_j) applied to nearby qubits. Hence, nonlinearities in QCNN arise from reducing the number of degrees of freedom. Convolution and pooling layers are performed until the system size is sufficiently small; then, a fully connected layer is applied as a unitary F on the remaining qubits. Finally, the outcome of the circuit is obtained by measuring a fixed number of output qubits. As in the classical case, circuit structures (i.e. QCNN hyperparameters) such as the number of convolution and pooling layers are fixed, and the unitaries themselves are learned.

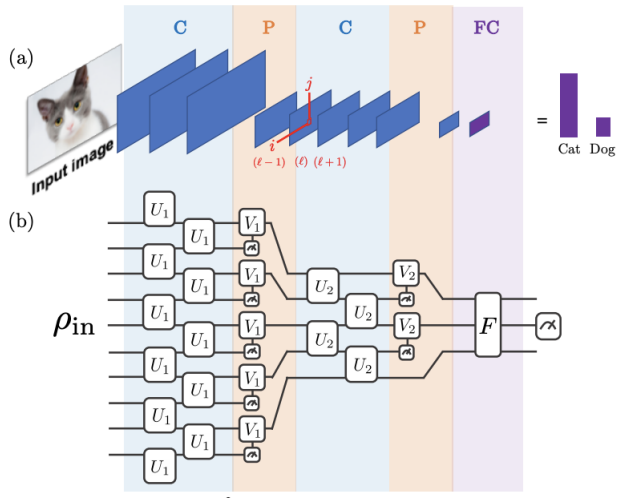

Figure 2: (a) Architecture of classical CNN (b) Architecture of QCNN

4 Experiment

4.1 Quantum Experiments vs. Simulation

Just as most of research in quantum computing, we had the choices of performing our experiments in simulation or on a real quantum computer. There are pros and cons when it comes to which route to follow. The pros of using simulation over quantum computer will be more abundant computation resources. One doesn't need to wait in long queues to put jobs on a quantum computer. Besides, on a quantum simulator is not limited by number of possible qubits that can be accessed and the long queue time for using them. As such, it is more scalable to perform experiments in a quantum simulator. Furthermore, thanks to the nature of simulation, qubits won't decohere over time and thus result in noisy output as much as on real devices. Oppositely, the pros of using quantum computer over simulator will be more convincing conclusions. In addition, in classical computers, we need to explicitly definte the superposition of qubits and store it in memory, whereas if we use a quantum computer, superposition is an intrinsic property, so it's not necessary to store the information. Experiments done on a quantum computer prove the applicability of the concept on an actual quantum computer while a simulator is slightly different.

Considering the pros and cons, we chose to perform our experiments in simulation, since it is relatively cheap and easier to scale up, its results are more stable and won't decohere over time, and we can avoid potentially long queues for using real devices.

4.2 Related Libraries

In our implementation of quantum image classification via QCNN, we used Cirq, an open-source framework for Noisy Intermediate Scale Quantum (NISQ) computers library developed by the Google AI Quantum Team, and the quantum machine learning library, TensorFlow Quantum, developed by Broughton et al. (2020).

4.3 Experiment Workflow

We use FRQI proposed by Le et al. (2011) to first process $2^1 \times 2^1$ images and feed their quantum representation into our QCNN. Any computations in our QCNN will be purely quantum. By demonstrating the feasibility of training a QCNN using qubit-encoded images, we lay the foundation for later studies to investigate QCNN on larger quantum images.

4.3.1 Dataset

we build a dataset of 200 images for our experiment, we tried different type of images, including grayscale images, large-size images, but eventually we reach a final design that all images in the dataset are of size $2^1 \times 2^1$,

which means each of them is made up by 4 pixels, we will use the color angles and position information as a representation of the image, transfer them into quantum superposition using FRQI. Those images whose lower part are black are labeled as -1, those whose upper part are black are labeled as +1, we build this simple dataset manually in order to simplify the task, and make a clear observation of the performance of the model. Finally, we divide this set into two parts, one containing 160 images for training and the other 40 images for testing.

4.3.2 Quantum Image Processing

Once we have an input $2^1 \times 2^1$ image, we use FRQI to transfer it into quantum states, we will use three qubits in this experiment to represent an image, two of the qubits represent the position of the pixel and one qubit is used to represent color angle. Since every pixel in our image has RGB values, therefore, we need to transfer them into a single angle value, which is represented by $\theta \in [0, \frac{\pi}{2}]$. Since R, G and B values are between 0 and 255, we use a normalizer to normalize the sum of the three value between 0 and 1, then multiply by $\frac{\pi}{2}$. The proposed FRQI form is quite flexible because of the way the positions of colors are encoded into computational basis states. In this way, the presentation of the geometric appearance of colors will affect on the quantum representation of the image. In quantum computation, computers are usually initialized in well-prepared states.As a result, the preparation process that transforms quantum computers from the initialized state to the desired quantum image state is necessary. All transforms used in quantum computation are unitary transforms described by unitary matrices. Quantum mechanics ensure the existence of such unitary transforms for the preparation process without pointing out explicitly efficient implementation in the sense of using only simple transforms such as Hadamard transform, rotations, etc. According to the lemma and corollary of the Polynomial Preparation Theorem(PPT), given a vector $\theta = (\theta_0, \theta_1, ..., \theta_{2^{2n}-1})$ of angles, there is a unitary transform P that turns the quantum computers from the initialized state $|0\rangle^{\otimes 2n+1}$, to the FRQI state, by Hadamard, CNOT and $C^{2n}(R_y(\frac{2\theta_i}{2^{2n}-1}))$ gates, where $R_y(\frac{2\theta_i}{2^{2n}-1})$ are the rotations about $\hat{y}$ axis by the angle $\frac{2\theta_i}{2^{2n}-1}$. We know that $R_y(2\theta)$ can be replaced by $cR_y(\theta)$, $cR_y(-\theta)$ and CNOT gates, and $cR_y(\theta)$ can also be replaced by CNOT, Hadamard and $R_y(\theta)$ gates, Fig. 3 shows how to replace $R_y(2\theta)$ using $cR_y(\theta)$, $cR_y(-\theta)$ and CNOT gates. Finally, we build a circuit using three qubits and multiple gates to represent an image.

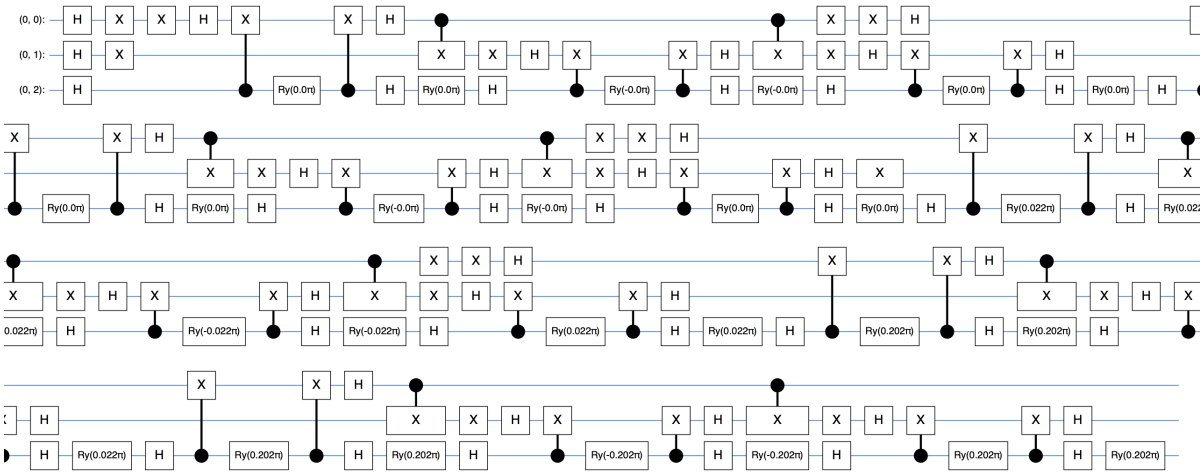

Figure 3: FRQI circuit implemented

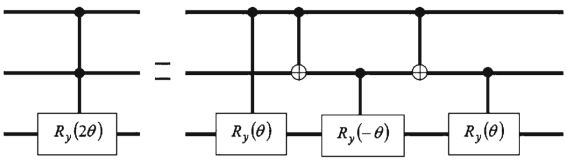

Figure 4: $C^2(R_y(2\theta))$ gates can be built from $C(R_y(\theta)), C(R_y(-\theta))$ and CNOT gates

4.3.3 QUANTUM CNN

We use a quantum CNN architecture built by TensorFlow Quantum(TFQ)to accomplish our task, TFQ provides layer classes designed for in-graph circuit construction, we can use this layer to prepend or append to the inputbatch of circuits, therefore, we can add our FRQI circuit into this architecture, or regard it as the input of the whole network. First we need to transfer the circuit into tensor, then define layers that make up the model. There are a few prerequisites, however, we need to first define a one- and two-qubit parameterized unitary matrices and a general parameterized two-qubit pooling operation. After all these operators have been constructed, we built quantum convolution layer and quantum pooling layer, both of which are quantum circuits as well. The model we defined here will use three quantum convolution layers and three pooling layers. After finish the construction of the model, we can start the procedure of training, testing and plotting the results which is pretty same as CNN in a classical computer. For our experiment, we will use MSE as our loss function, since it is a binary classification problem. Figure 4 shows the architecture of the whole QCNN model.

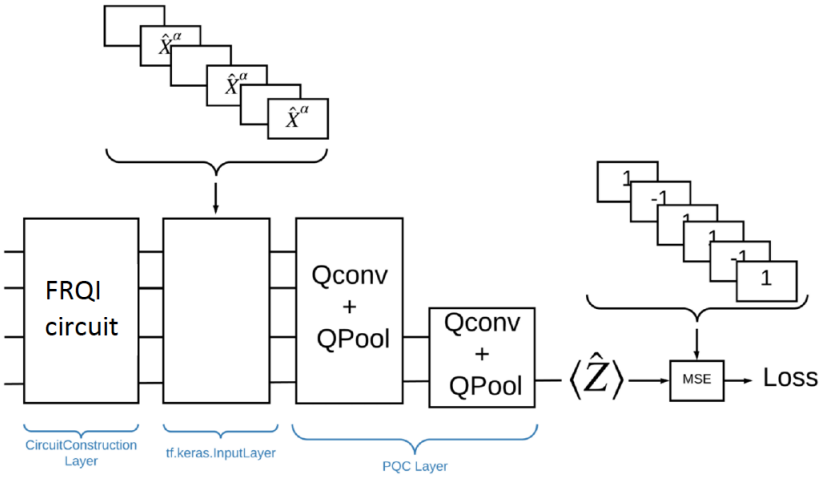

Figure 5: Building blocks of the Quantum CNN of TensorFlow

FIX

4.4 TOY DATASET

As a training data, we constructed a toy dataset with two ground truth labels, 1 and -1. 1 corresponds to 2x2 RGB images where the top row has pixel values [0, 0, 0], and the bottom row contains random pixel values for all three channels 6, vice versa for -1 class. Since our data contains relatively simple pattern, we decided the size of the dataset to be 200 where 160 for training and 40 for validation.

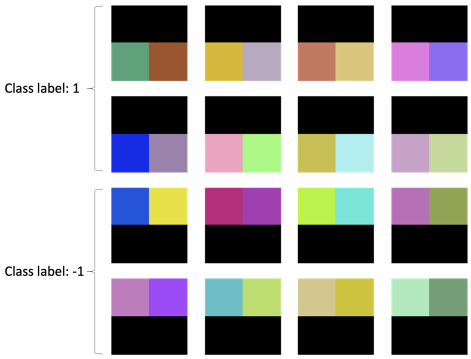

Figure 6: Sample images from toy dataset

5 Results and Discussion

Figure 7 is an example of encoding and decoding a 2x2 image using our FRQI circuit. Comparing the input and output images, we see that the colors do not match exactly. This is due to our algorithm in retrieving the images via a normalized probability distribution, so the θ values for angles of rotation, thus the RGB value for each pixel, might not be exact. However, they remain relatively accurate to other pixels in the same image. It is clear that by having an extra cubit to encode positions along pixel intensities, the spatial information of intensities on each pixel is indeed preserved; this is essential to convolutional neural networks which are good at extracting spatial information and thus features of the images.

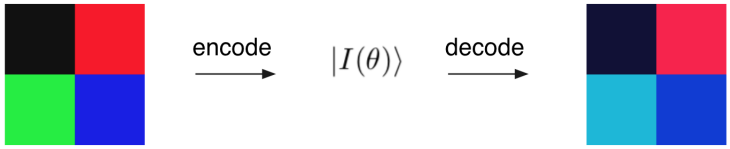

Figure 7: FRQI example

Our QCNN model was trained on 160 samples and validated on 40 samples of 2x2 RGB images. Figure 8 shows the training and validation losses over 20 epochs of training the model with inputs being the FRQI encoded quantum states of the pixels in each image. Figure 9 shows the training and validation accuracies over the same run.

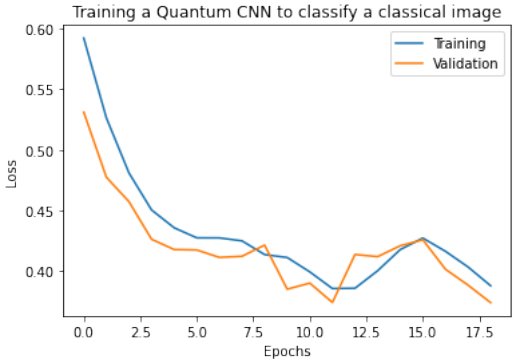

Figure 8: Training and validation losses

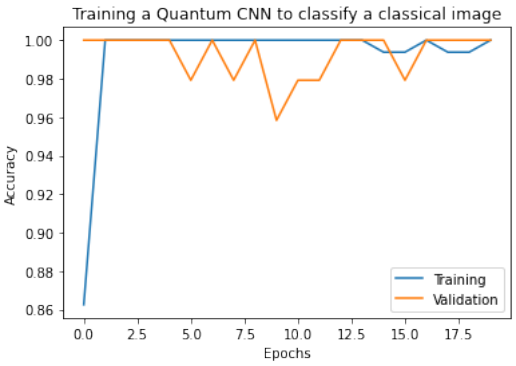

Figure 9: Training and validation accuracies

At the end of epoch No.20, our final training accuracy was 1.0000 while the validation accuracy was also 1.0000; the MSE (mean squared error) training loss after epoch No.20 was 0.3879 while the validation accuracy was 1.0000 and the MSE validation loss was 0.3740. It is shown that the validation loss is lower than the training loss at the beginning of epochs, but the two begin to converge as we go further in epochs. Validation happens in each epoch after the model parameter was updated, and since the training of the model was performed on a toy dataset with only 2 classes, it makes sense that the validation loss appeared lower at the beginning of epochs. Given the simplicity of the dataset, it also makes sense that the training accuracy shown in 9 converges quickly after the second epoch to 1.0000. Increasing the complexity of the classes in the dataset will show a decrease in convergence rate. Nevertheless, it is robust in our results that the images could be classified with both 1.0000 training accuracy and 1.0000 validation accuracy.

Before our approach, the state-of-the-art approach to classifying digits via quantum neural networks had sub-optimal image pre-processing procedures. Farhi & Neven (2018) showed methods of down-sampling 28x28 images to 4x4 images; this approach performs 1-to-1 conversion from pixels to qubits, and loses the important detailed spatial information which the neural network depends on. Our results showed that the approach to encoding pixel intensities as amplitudes of entangled quantum states using FRQI works in that it not only exponentially reduces the spacial and time complexity of computation, but also preserves the original detailed spatial information of intensity value on each pixel.

Before our experiments, Cong et al. (2019) explored only on 1-dimensional input data in the context of quantum experiments, QCNN's ability to classify classical 2-dimensional images remained unknown. Our results can now serve as a proof-of-concept of quantum neural networks ability to classify classical images, lying the founding ground for future research into classifying classical images with quantum neural networks.

6 Future Work

6.1 Scalable FRQI Algorithm

Currently, the quantum gates in FRQI algorithm is hard-coded for the dimension of our toy dataset (2x2x3). As it's been proven to work well in the experiments, the next step will be automating the process of FRQI so that it's adaptable to images of any dimensions.

6.2 Generic Feature Extractor

In the computer vision community, Residual Neural Network (ResNet) He et al. (2016) serves as a backbone for a lot of model because it's proven to be a generic feature extractor for static images. Such type of architecture can learn a semantic representation of an image, and the learned representation can be used to perform custom computer vision tasks. We realized that there's still a long way to go for a quantum convolutional neural network to achieve a comparable ability. The logical first step is to achieve a good classification accuracy on dataset including real-world images. There are a lot of options for this purposes. Famous vision dataset for image classification includes MNIST LeCun et al. (2010), CIFAR-10 Krizhevsky et al. (2009), ImageNet Russakovsky et al. (2015). With an automated FRQI program, we should be able to obtain a preliminary results easily, but the challenge remains improving the architecture of Q-CNN to achieve a better classification results.

References

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

Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15 (12):1273–1278, 2019.

Aditya Dendukuri and Khoa Luu. Image processing in quantum computers, 2018.

Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Iordanis Kerenidis, Jonas Landman, and Anupam Prakash. Quantum algorithms for deep convolutional neural networks. *arXiv preprint arXiv:1911.01117*, 2019.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html*, 6, 2009.

Phuc Q. Le, Fangyan Dong, and Kaoru Hirota. A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Information Processing*, 10(1):63–84, February 2011. ISSN 1570-0755. doi: 10.1007/s11128-010-0177-y. URL https://doi.org/10.1007/s11128-010-0177-y.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. 2010.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.