Quantum Computers:

“The ultimate Quality Assessment machine”

Similar to the BMW M3 around the Nurburgring, qBraid is fast. We’re a rapidly growing start up with a mission to accelerate the quantum computing revolution. Our platform allows seamless access to various quantum computers and simple interfaces to write quantum code in seconds. **We at qBraid believe that quantum machine learning will be a vital feature of our platform and are honored to demonstrate our quantum machine learning based solution to improve automated quality assessment.**

Team Members: Kanav Setia, Ryan Hill, Ricky Young

Table Of Contents

Description

Although current machine learning algorithms can detect scratches and bumps in images, our quantum machine learning algorithm in this submission presents an opportunity to use less computational resources and achieve the same results or better in the near future. While much work is needed for quantum computers to become scalable, our algorithm uses much less trainable parameters compared to a classical equivalent, thus providing an indication of the unrealized potential in quantum machine learning. Together with BMW and AWS, qBraid is eager to **demonstrate the advantage of quantum machine learning algorithms for quality assurance datasets across AWS devices using the qBraid-sdk (to be released early October).** When quantum computers are fully realized, qBraid intends to have an arsenal of machine learning algorithms that will be available to its users including QCNN (Quantum Convolutional Neural Networks) and, while not strictly machine learning, quantum kernel methods such as QSVM (Quantum Support Vector Machines). These algorithms will provide the quantum advantage that will reduce the computational cost compared to their classical counterparts. The BMW Challenge is a crucial step towards realizing a quantum machine learning suite that is coding and device agnostic.

Not only does our submission exhibit potential, but it also is developed in qBraid's Lab coding environment! Lab removes the hassle of installing and fixing quantum computing packages and allows any user to simply code as soon as they log in. We're excited to mention that Lab will eventually use the qBraid software development kit allowing for easy access to quantum computing machines, much like how AWS allows software engineers to access different classical computation resources. Our engineers have already demonstrated their capabilities in the quantum machine learning realm, recently placing first at the CDL (Creative Destruction Lab) Quantum Hackathon using a weather anomaly detection algorithm with grover's search + QNN and successfully verifying variational quantum classifiers.

Timeline

If we are selected this would be our proposed timeline for our solution to:
Develop quantum machine learning algorithms for quality assurance datasets and benchmark them across AWS devices using the qBraid-sdk.

Oct 15 ~ Nov 5, 2021

- Finish developing three promising quantum machine learning algorithms on qBraid Lab.
- Train the algorithms on the provided dataset using a single device or simulator, polish.
    - This portion will require the help from AWS and BMW to better define the kinds of improvements necessary

Oct 5 ~ Nov 19, 2021

- Benchmark across AWS devices collect results and interpret.
- Compare and analyze results with each of the other device results as well as a classical counterpart for the three quantum machine learning algorithms.

Nov 19 ~ 29, 2021

- Depending on the results present findings and demonstrate that an increased number of qubits could provide a quantum advantage.
- Prepare to turn in submission.

Solution

We demonstrate a small scale solution consisting of a hybrid approach where a quantum neural network layer is used conjointly with limited use of classical machine learning layers. Given the current limitations of quantum computing we downsample the images to be a size of 8 x 8. Then, they are fed into a quantum machine learning algorithm which **achieves a performance of 95% test accuracy.** This result demonstrates the effectiveness of quantum algorithms *even without a quantum or classical convolutional neural network.* We acknowledge that our model is simplistic, but we encountered challenges in run time using larger circuits and an appropriately large amount of samples. We found the current circuit to be the best size for demonstration purposes. Later down the road, we intend to use more elaborate quantum machine learning algorithms of which we have already developed in our Lab environment such as the Circuit Centric Classifier [1].

Implementation:

The implementation of our demo primarily uses PennyLane and Tensorflow. In future uses we would benchmark our algorithm across all of the AWS devices (as well as other devices such as Google's Sycamore) using various languages Cirq, PennyLane, Bracket with our very own qBraid-SDK.

Preprocessing

Given the current limitations of quantum computers, the images have been reduced in size from the original 277 x 277 to 8 x 8 pixels. Then a threshold is applied to highlight the specific portions allowing the algorithm to train upon the "brightness" of an image.

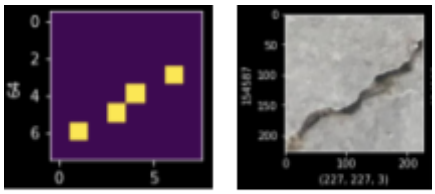

Fig.0 Preprocessing Images

Amplitude Embedding

Encoding the classical data into quantum circuits is an important step as we move toward quantum machine learning applications. Out of the many ways to encode the scratch detection data [2,4], we started with the simple amplitude encoding, $|\psi>=\sum_{i=1}^{N} x_i|i>$ where x_i is the classical data, and N is the total number of the data we are interested in encoding. There is a proposal to embed the RGB data [1], but for simplicity, we encode the data as a grayscale image into quantum states. The purple section in Fig.1 illustrates the amplitude encoding process.

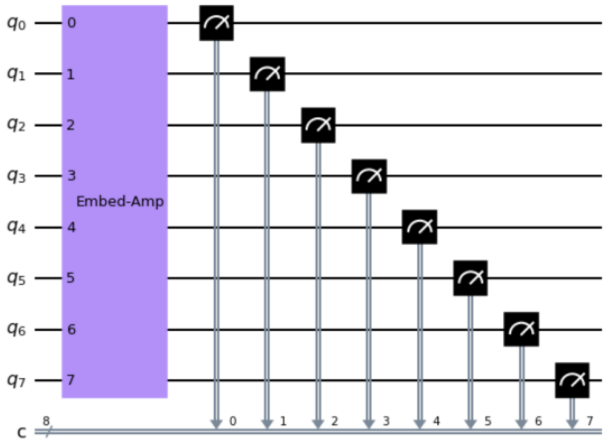

Fig.1 Amplitude Encoding

Strongly Entangled Layers

Strongly entangled layers similar to the Circuit Centric Classifier [1] were used to create our trainable parameters for the PQC (Parametrized Quantum Circuit). The circuit defaults to using CNOT gates where the second qubit is determined by $(i+r) \mod M$, where r is a hyperparameter called the range, and $0 < r < M$ [3]. We used the strongly entangled layers given their performance capabilities, however we limit to only 1 layer of gates given that the run time quickly increases when the depth of the circuit increases to 1+.

Classical Dense Layers

The classical layers are used to provide an input for the images as well as an output from the quantum circuit. We note that the first classical dense layer is not trainable, indicating that the total number of trainable parameters is close to much less than 4,174.

Advantages of a quantum algorithm:

This demo illustrates the quantum advantage which can be achieved with some heavy preprocessing. Below we provide two machine learning model summaries which were both run on the surface crack detection set. One can note the stark difference in the amount of

parameters used in the classical algorithm. Although it can be qualified that it was trained on a much larger image 277 x 277, we expect that the algorithms we develop will still require a much smaller amount of parameters as demonstrated by this submission.

We also see promise in quadratic speed ups in unstructured searching, such as for dot like imperfections in a quality assessment trial using Grover's Search Algorithm.

Model: "quantum"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	4160
keras_layer (KerasLayer)	(None, 6)	0 (unused)
dense_1 (Dense)	(None, 2)	14

Total params: 4,174
Trainable params: 14
Non-trainable params: 4,160

Model: "classical" Accuracy: 99.8%
Courtesy of:
KUNDAN KUMAR MANDAL

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 227, 227, 64)	1792
conv2d_1 (Conv2D)	(None, 227, 227, 64)	36928
max_pooling2d (MaxPooling2D)	(None, 113, 113, 64)	0
conv2d_2 (Conv2D)	(None, 113, 113, 128)	73856
conv2d_3 (Conv2D)	(None, 113, 113, 128)	147584
max_pooling2d_1 (MaxPooling2	(None, 56, 56, 128)	0

conv2d_4 (Conv2D) (None, 56, 56, 256) 295168

conv2d_5 (Conv2D) (None, 56, 56, 256) 590080

conv2d_6 (Conv2D) (None, 56, 56, 256) 590080

max_pooling2d_2 (MaxPooling2 (None, 28, 28, 256) 0

conv2d_7 (Conv2D) (None, 28, 28, 512) 1180160

conv2d_8 (Conv2D) (None, 28, 28, 512) 2359808

conv2d_9 (Conv2D) (None, 28, 28, 512) 2359808

max_pooling2d_3 (MaxPooling2 (None, 14, 14, 512) 0

conv2d_10 (Conv2D) (None, 14, 14, 512) 2359808

conv2d_11 (Conv2D) (None, 14, 14, 512) 2359808

conv2d_12 (Conv2D) (None, 14, 14, 512) 2359808

max_pooling2d_4 (MaxPooling2 (None, 7, 7, 512) 0

flatten (Flatten) (None, 25088) 0

dense (Dense) (None, 4096) 102764544

dense_1 (Dense) (None, 4096) 16781312

dense_2 (Dense) (None, 2) 8194

Total params: 134,268,738
Trainable params: 134,268,738
Non-trainable params: 0

References

[1] Maria Schuld, Alex Bocharov, Krysta Svore, and Nathan Wiebe. Circuit-centric quantumclassifiers.Physical Review A, 101, 04 2018.

[2] Mohd. Hussain Mir, Harkirat Singh,Pixel identification in an image using Grover Search Algorithm, https://arxiv.org/ftp/arxiv/papers/2107/2107.03039.pdf

[3] "Source code for pennylane.templates.layers.strongly_entangling (PennyLane)," https://pennylane.readthedocs.io/en/stable/_modules/pennylane/templates/layers/strongly_entangling.html#StronglyEntanglingLayers, Accessed: 2021-09-23.

[4] Yu Jing, Yang Yang, Chonghang Wu, Wenbing Fu, Wei Hu, Xiaogang Li, Hua Xu, RGB Image Classification with Quantum Convolutional Ansaetze, https://arxiv.org/abs/2107.11099