
QISKIT RUNTIME APPLICATIONS IN SURFACE CRACK DETECTION

A PREPRINT

Anuj Mehrotra
IOTAONEIQ Solutions Pvt. Ltd.
India
cto@iotasolutions.com

Meghashrita Das
Department of Agricultural and Food
Financial Engineering
Indian Institute of Technology, Kharagpur
India
meghashrita99@gmail.com

Mitesh Adake
Department of Computer Engineering
Pune Institute of Computer Technology
India
miteshadake29@gmail.com

Rajatav Dutta
GreenTech
India
rajatavdutta@gmail.com

Vardaan Sahgal
Department of Physics
University of Delhi
India
sahgalvardaan@gmail.com

Vishal Sharathchandra Bajpe
Department of Mechanical Engineering
Sahyadri College of Engineering and Management
India
vishal.bajpe@gmail.com

August 31, 2021

ABSTRACT

Quantum machine learning has established as an interdisciplinary field to overcome thermodynamic limitations of classical neural networks and machine learning. A classical neural network can be partially quantized to create a hybrid quantum-classical neural network which is used mainly in classification and image recognition. In this report, a basic a multi-label quantum neural network (QNN) model is made with image classification datasets where classical convolutional layers and quantum fully-connected layers are added. Next, with the help of a high performance quantum simulator, Quantum Kernel Alignment (QKA) is used to execute parametrized QNN circuit for evaluating the quantum kernel matrices with Quantum Estimator (QKE). Quantum computers allow evaluating kernels to be used in Support Vector Machines (SVM) on exponentially large feature spaces. This may help to improve the performance of neural network models. Qiskit Runtime is a new architecture offered by IBM Quantum that streamlines computations requiring many iterations. So, a comparison based research has been done in between the classical CNN efficiencies and the improved QNN structured with QKA applied in surface crack detection.

Keywords Quantum Neural Network · Quantum Kernel Alignment · Qiskit

1 Introduction

1.1 Motivation

Although Convolution Neural Network (CNN) has proved to be a very valuable tool in computer vision, many real-world problems are still very hard to solve classically. In the present study, an attempt to formulate a solution to this problem

has been suggested. Convolutional Neural Network (CNN), among many classification models, has shown very high performance at a relatively low processing cost in computer vision. By using real-world images or photographs as the input to ensure practical usability, and by using quantum computing systems to effectively solve large datasets, this report introduces the Quantum Neural Network (QNN) and one of its possible supplementing components to solve real-life problems. In the present study, we have used the 'detection of cracks in surfaces' as an example to highlight the potential QNN model holds. We have added specific quantum circuit filters as additional layers in the QNN model to attain maximum efficiency and practical usability of surface crack detection. The present example of surface crack detection, alone, is of great importance in various manufacturing and production industries, where the measure of efficiency, quality control and an attempt to demonstrate performance improvement which can be easily measured and improved upon by the suitable use of the present model using Qiskit Runtime is being demonstrated in this report.

1.2 Innovation

The innovation behind this paper is to explore a new application of Qiskit runtime algorithms in image processing applications, specifically, an instance of quality control has been demonstrated in the code repository linking to this paper. Through this report, an attempt to demonstrate QKA applied as a part of QNN as a demonstration of Qiskit Runtime capabilities in a real world surface crack finding problem describing a classical and quantum hybrid implementation on an image recognition data sets. This kind of real life application have not been demonstrated as an application of Qiskit Runtime to the author's knowledge.

1.2.1 CNN

Classical and quantum deep learning is used to build an accurate model for crack detection. The data set consists of 20,000 images of concrete structures with cracks and 20,000 images without cracks. Each image in the data set is a 227 x 227 pixels RGB image. A classical Convolution Neural Network (CNN) is made using the Pytorch library for convenient data transformation as well as its compatibility with Qiskit modules. Pytorch has a native connector for Qiskit which reduces the development overhead and augment the model creation process. For this specific implementation, a pretrained Resnet 50 model is used and re-purposed for the surface crack detection datasets by implementing it using transfer learning. Model parameters were untouched and Adam optimizer with a default learning rate was utilized. The model was retrained for 5 epochs on the training data set sample while measuring loss and accuracy on the validation set.

1.2.2 QNN

Quantum neural networks are computational neural network models which are based on the principles of quantum mechanics and acts application-agnostic computational units that can be used for many different use cases, such as Image Processing , Natural Language Processing, computer games, function approximation, handling big data, in modelling social networks, associative memory devices, and automated control systems etc.

For this stage, a hybrid QNN model is prepared simultaneously using Pytorch and Qiskit. The approach was to use a classical Neural network and add in a few quantum layers which are in practise, governed by a pre-set quantum circuit. This complete implementation is trained with same surface crack dataset mentioned previously. Here 1000 positive and negative crack images are taken rather than 20000 for training reasons. There were some complications involved with access time and resource allocation on our IBM backend which warranted for a decrease in resolution of the samples and the size of the training set. The details of which are in sub section 1.2.3 and section 2

1.2.3 QNN+QKA

Collaboration has always been an important concept in the scientific community. In addition to using the built-in programs, a new dimension can come up with Qiskit Runtime programs that can help and accelerate the progress in research.

In Quantum Machine Learning (QML), Quantum kernels is the approach of exploiting group structure in data which helps to achieve quantum speedup. Quantum kernels can be optimized with a technique called kernel alignment. QNN is parameterized quantum circuits which acts like linear methods in quantum feature space and suitable case for using quantum kernels and in turn can be optimized using Quantum Kernel Alignment. An attempt is made to demonstrate the application of using QKA with QNN networks in the model to have augmented training or performance improvement.

2 Methodology

Methods for making QNN and QNN with QKA is discussed in this section.

While formulating the QKA implementation, a few challenges were encountered. The limit of Qobj processing only allowed for a resolution of 7x7 with a sample set of 100 images which was again scaled down to 10 samples. To maintain uniformity among the quantum counterparts, the resolution for the QNN implementation was also crippled to 7x7. The customized quantum circuit for the QNN model was made by preparing a quantum class using a simple quantum circuit with RYrotation by the angle θ to train the output of our circuit in figure 5. In order to measure the output in the z basis, σ_z expectation value must be calculated in 1.

$$\sigma_z = \sum_i z_i p(z_i) \quad (1)$$

The calculated expected value be 0.56. Now after defining the quantum circuit, the functions needed for back-propagated are created using PyTorch. The forward and backward passes contain elements from created Qiskit class. The backward pass directly computes the analytical gradients using the finite difference formula. The network will need to be compatible in terms of its dimensionality when the quantum layer (such that in the quantum circuit) will be inserted. Since the quantum circuit in this paper contains 1 parameter, the network should condense neurons down to size 1. A typical Convolutional Neural Network with two fully-connected layers at the end is made up. The value of the last neuron of the fully-connected layer is fed as the parameter into our quantum circuit. The circuit measurement then serves as the final prediction for 0 or 1 as provided by a σ_z measurement. Now by using Adam PyTorch optimiser, learning rate as 0.001 and negative log-likelihood loss function in order to train over multiple epochs. The best validation accuracy is given by the Figure 3.

Now application of QKA for QNN is taken through the following steps. The IBM Q portal had a restriction limit for parsing the Qobj in its public access tier as of the time of writing and the local server for qiskit-runtime unfortunately crashed on the system which was allocated for the same due to misconfiguration on our side. So, a sample-set of 7x7 resolution images was formed as an input with the last column holding the classification value. To reduce the size of the overall input array, the RGB channel was converted to gray scale single channel. The resulting array was made in such a way that the maximum number of feature columns did not exceed twice of the qubit count of the backend. Since we had the ibm_qasm_simulator as a Qiskit-runtime compatible backend as of the time of writing this report, the limit at our disposal was 32 qubits. The whole of 32 qubit or usage of a 8x8 resolution was not possible due to breach of RAM limit on the said backend. Thus, the resolution was again reverted back to 7x7. For stability due to Qobj parsing issues, it was further brought down to 6x6. Initial parameters were set to random array and the alignment protocol on the runtime was run using the default penalty factor and 5 iterations. The obtained aligned matrix was then converted into a pytorch tensor and fed into the nn.Module class model that was previously formulated for the QNN implementation. An additional kernel layer was added with the initial kernel weight parameter set to the be parameters obtained from the QKA protocol run on the qiskit runtime. This whole setup was then set for training and the loss and accuracy metrics of the model were recorded. When considering finite data, quantum kernel is represented as a matrix and each element of this kernel matrix can be calculated on a quantum computer by calculating the transition amplitude, which provides an estimate of the quantum kernel matrix, which can then be used in a kernel machine learning algorithm like QNN.

2.1 Applicability

Quantum kernels are used in a classification framework inherit the convex optimization program and avoid common limitations of variational quantum classifiers. As QKA optimizes the Quantum Kernel Estimation, so in turn it helps QNN sort programs to achieve higher generalization of accuracy and highly required in Automated Quality Management based on Image Processing.

2.2 Role of Qiskit Runtime

Qiskit Runtime serves to be a tool that can provide considerable speed up for iterative quantum algorithms like QKA by co-locating classical computations with the quantum hardware executions. A lot of the implementational overhead is also taken off by the qiskit-runtime modules.

Table 1: Accuracy value table

Method	Accuracy	Loss in training
CNN	98.42%	0.0510
QCNN	69.2%	-0.5855
QCNN+QKA	70.3%	-0.7009

3 Technology Stack

The figure below covers the technology stack for implementing this project. Please Note : kaggle.json credential file is required to download Surface Crack Detection dataset from Kaggle.com. IBM Q API credentials may be needed if running on a platform other than IBM Q Experience.





Programming Language	Classical (Deep) ML Framework	Quantum Computing Development Platform	Simulator	Additional Module	Coding Collaboration Environment
 PYTHON 3.X >= 3.7	 PYTORCH 1.9	 QISKIT 0.29	IBM QASM (With Qiskit Runtime) & AER Simulator (for local)	QKA.PY from https://github.com/Qiskit-Partners/qiskit-runtime	 Open in Colab

Figure 1: Technology stack

4 Result

In this section, results are explained from the above experiments performed. Figure 2a is representing the correctly detected images getting through the model of CNN [1.2.1]. This is a figure of non-cracked surface detected as negative (The image is with crack is shown as positive and without crack is shown as negative). Similarly, Figure 2b is showing the cracked surface image that is detected as positive. Figure 4 is a plot in between the number of training convergence and the loss function. Table 1 is representing the accuracy and corresponding loss in applied models. For the QKA + QNN part, interestingly, the accuracy was not far away from the QNN counterpart for our controlled setup here. Although the accuracy did not improve much with increasing epochs, the median value was achieved in just 2 epochs as compared to close to 20 epochs in the QNN implementation. A fair disclaimer must be added that this may not be indicative of an extrapolation to larger models since the resolution being evaluated is extremely small and has a single channel. Nevertheless, for the same input scenario, the results have been recorded as in 6 and 7

5 Concluding Remarks and Future Applications

This report, although representing a real world problem, has its experiments performed in a very controlled setup which may not be indicative of a real world deployment scenario. Quantum kernels can only be expected to do better than classical kernels if they are hard to estimate classically. This is a necessary but not sufficient condition. Our future work should be designing routines and structures for quantum kernels that exploit structure in data, within the limited constraint resource available in this NISQ era of Quantum computing. Runtime routines like Qiskit-Runtime can hugely reduce the development overhead for implementing and running such routines and with the development of more advanced and larger quantum backends, real world deployments for image processing based applications are not far away.

As QNN is an application agnostic computational unit with uses cases applicable to Image Processing , Natural Language Processing, computer games, function approximation, handling big data, in modelling social networks, associative memory devices, and automated control systems , fraud detection and much more, these applications can be made much more realizable by a larger audience, not just the research community, but also the software development community by usage of tools like the Qiskit Runtime which is being used in this report.

References

- [1] <https://arxiv.org/abs/2009.09423>
- [2] https://qiskit.org/documentation/partners/qiskit_runtime/tutorials/qka.html
- [3] <https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html>
- [4] <https://arxiv.org/abs/2104.03418>
- [5] Bartkiewicz, Karol, et al. "Experimental kernel-based quantum machine learning in finite feature space." *Scientific Reports* 10.1 (2020): 1-9.
- [6] Daskin, Anmer, Ananth Grama, and Sabre Kais. "Multiple network alignment on quantum computers." *Quantum information processing* 13.12 (2014): 2653-2666.
- [7] Hubregtsen, T., Wierichs, D., Gil-Fuster, E., Derks, P. J. H., Faehrmann, P. K., Meyer, J. J. (2021). Training quantum embedding kernels on near-term quantum computers. *arXiv preprint arXiv:2105.02276*.
- [8] Bartkiewicz, K., Gneiting, C., Černoch, A., Jiráková, K., Lemr, K., Nori, F. (2020). Experimental kernel-based quantum machine learning in finite feature space. *Scientific Reports*, 10(1), 1-9.
- [9] Acampora, Giovanni, and Roberto Schiattarella. "Deep neural networks for quantum circuit mapping." *Neural Computing and Applications* (2021): 1-21.
- [10] Glick, J. R., Gujarati, T. P., Corcoles, A. D., Kim, Y., Kandala, A., Gambetta, J. M., Temme, K. (2021). Covariant quantum kernels for data with group structure. *arXiv preprint arXiv:2105.03406*.
- [11] Glick, J. R., Gujarati, T. P., Corcoles, A. D., Kim, Y., Kandala, A., Gambetta, J. M., Temme, K. (2021). Covariant quantum kernels for data with group structure. *arXiv preprint arXiv:2105.03406*.

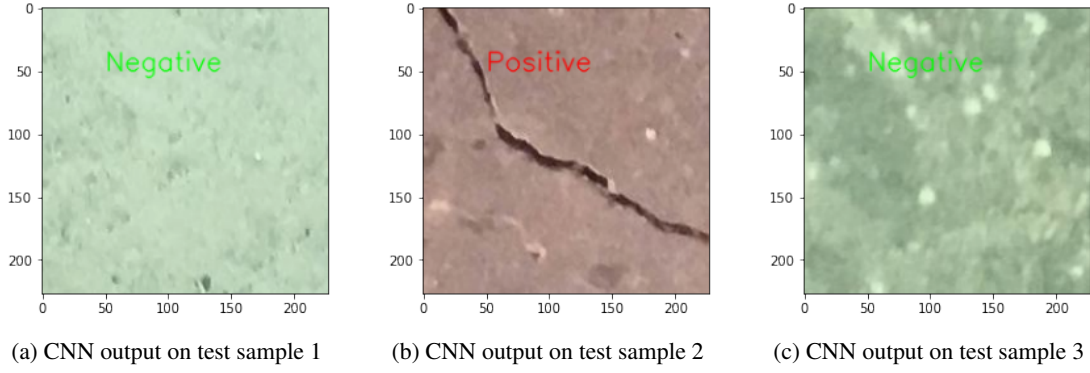


Figure 2: CNN Output

Training complete in 31m 42s
Best val Acc: 0.984209

Figure 3: CNN trained model Best Accuracy on test dataset

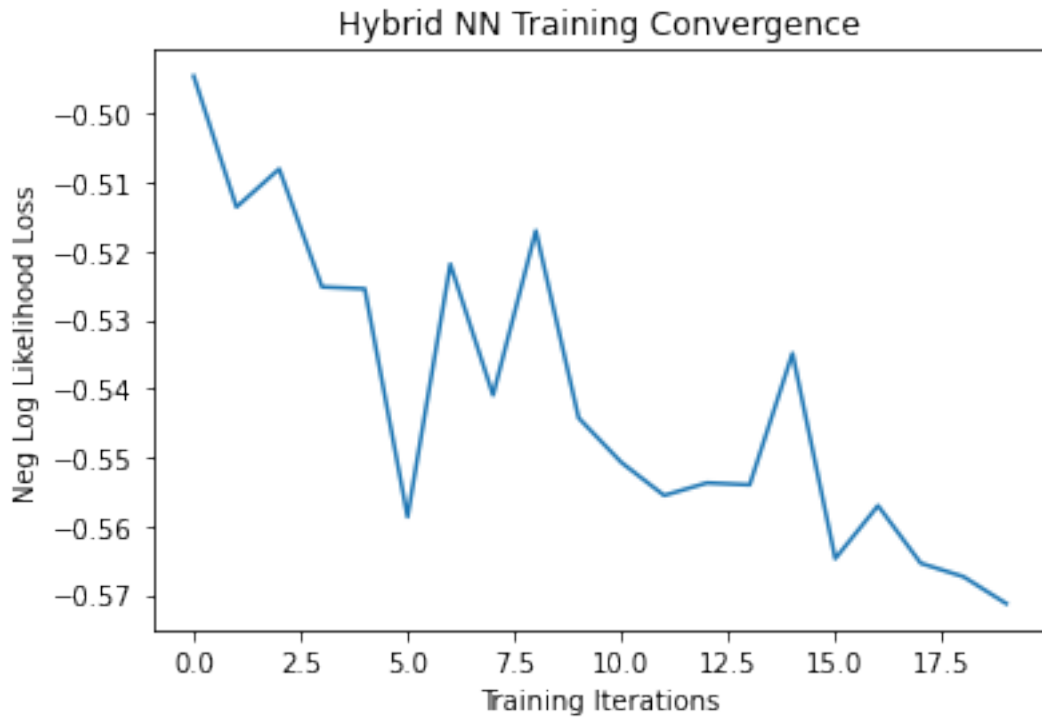


Figure 4: Hybrid NN Training Convergence

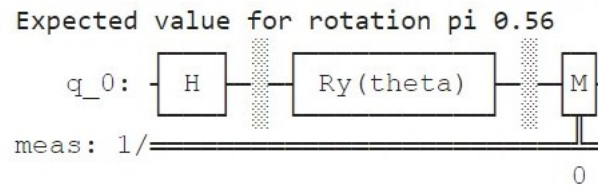


Figure 5: Quantum Circuit in QNN

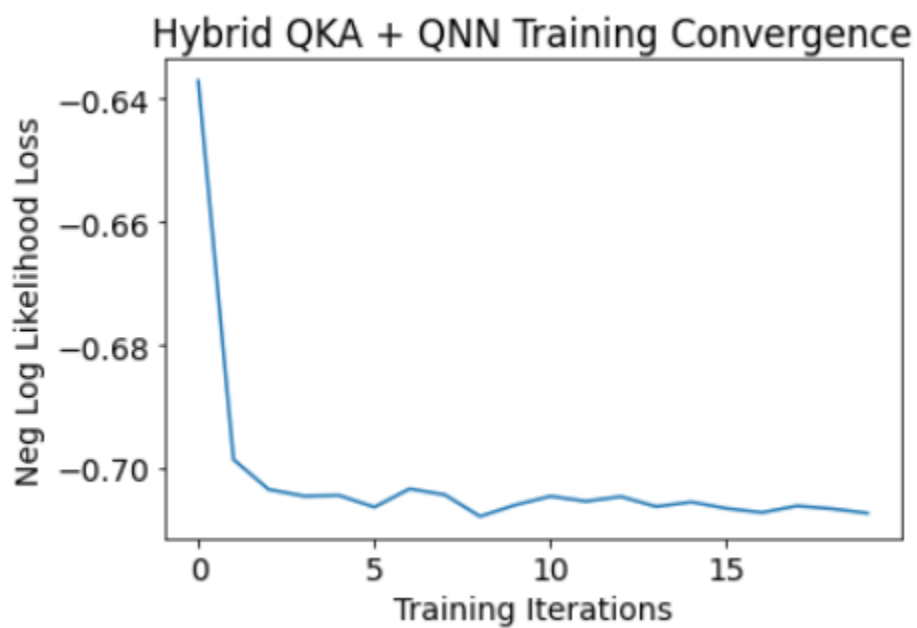


Figure 6: QKA Training epoch losses

Performance on test data:
Loss: -0.7135
Accuracy: 70.8%

Figure 7: QKA loss decrement