Quantum Convolutional Neural Networks via Variational Quantum Circuits for Efficient Image Classification

Authors

Diptanshu Sikdar *

Max Cui *

Adrian Kao *

Abhik Das *

Ahmet T. Bayrak *

Jagannath Prabhakaran *

(*Aspiring Scholars Directed Research Program:

46307 Warm Springs Blvd, Fremont, CA 94539)

Advisor & Corresponding Author:

Dr. Larry McMahan (larry.mcmahan@asdrp.org)

Abstract

Quantum Machine Learning (QML) is a multidisciplinary field involving machine learning algorithms and quantum computing concepts, and it focuses on the construction of the quantum model representation and optimization of machine learning. It has the potential to outperform classical machine learning algorithms due to the significant computational speedup on parallel data. Because machine learning relies heavily on probability, the quantum computing environment is beneficial since it can significantly reduce resources that a machine learning (ML) model needs when learning from high dimensional data. To test this quantum advantage in the Noisy Intermediate Scale Quantum (NISQ) era, we implemented a Quantum Convolutional Neural Network (QCNN) for image recognition using Tensorflow Quantum. We hypothesized that our QCNN would have a higher accuracy and greater efficiency in training and runtime than a classical Convolution Neural Network (CNN). Our QCNN architecture parallels a classical CNN structure except that the bulk of the work is done in the quantum domain. It consists of a classical-to-quantum image data encoder, a cluster state quantum circuit, series of Quantum Convolutional and Quantum Pooling Layers via Variational Quantum Circuits (VQCs) and trainable parameters, and a measurement layer leading to the output. The data encoding/decoding and the cost function optimization are performed on a classical computer, while a quantum computer simply calculates the probabilities of the states in the VQCs. Based on probabilities generated by the quantum computer and optimization algorithm executed on a classical computer, the rotation parameters in the VQCs are updated after each backpropagation cycle.

Keywords

Quantum Machine Learning, Noisy Intermediate Scale Quantum (NISQ), Convolutional Neural Networks, Cluster State Quantum Circuit, Variational Quantum Circuits (VQC), Adaptive Moment Estimation (ADAM), Convolutional Layer, Pooling Layer

Author Summary

Our work focuses on the quantum-computing-based implementation of Convolutional Neural Networks (CNNs). CNN's are a machine learning model architecture primarily used to classify images due to efficient feature extraction. For instance, a CNN can recognize specific characteristics like a bird's beak or a wing. However, the limitations of a CNN become evident when training with large datasets with many features. Here, quantum computing may prove to be more efficient. The main difference between classical (conventional) computing and quantum computing environments is the number of possible states represented within a single element at an instant. While classical computers represent either a '0' or a '1' via bits, a quantum computer can represent a superposition of zeros and ones using qubits. Furthermore, qubits can be strongly coupled together through a phenomenon called quantum entanglement, which enables extreme parallelization. We designed and implemented a Quantum Convolutional Neural Network to classify images.

Introduction

The predecessor to Quantum Machine Learning is Machine Learning, which is performed entirely on classical, bit-based computers and mimics how the human brain learns from data. Input data is fed into an untrained model, which makes an initial inference about the data. The predicted values are compared to the actual values, yielding the loss calculation, and the Mean Squared Error (MSE) can then be calculated (Figure 1).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

Figure 1: Mean Square Error Formula. There are *n* data points.
Y_i is the observed value, and $\hat{Y}_i$ is the predicted value.

The loss (MSE) is then back-propagated through the model, and the parameters are adjusted accordingly, thus allowing the model to learn from its mistakes. As the errors are back-propagated through the model, the parameters (or neuron weights) are readjusted by an optimization algorithm such as the Adaptive Moment Estimation (ADAM) algorithm. In the context of QML, the model has layers of quantum circuits which can be trained by passing in input states and an optimization algorithm to adjust circuit parameters.

Convolutional Neural Networks (CNN) are a variation of Artificial Neural Networks (ANN) and are most helpful in applications like image recognition. Increasing the number of hidden layers or neurons in an ANN is inefficient because it leads to overfitting, which is bad for the model because it compromises the accuracy during training and testing. A CNN avoids this by not making connections with all the neurons in a model. This reduces the chance of overfitting, thereby making the CNN a better model than ANN.

The internal architecture of a CNN usually consists of three layers: a convolutional layer, a pooling layer, and a fully connected layer. In the convolutional layer, the dot product of the filter and the input pixels is calculated, which is then repeated after the filter (filter is a set of weights that is used to scan for specific features of an image) shifts by strides until the entire image is scanned. This forms a feature map of the whole image with only regions having a specific feature. The filters' height and width are already defined, but the depth does not — it depends on the number of filters used. In deep CNNs, multiple convolutional layers can be

stacked together to recognize more complex patterns. To reduce the dimensionality, complexity, and size of the model, pooling layers are used to compress information by scanning a filter across the image and using an aggregation function. Although pooling reduces the size of the feature map, it does not reduce the features themselves. There are various kinds of pooling methods, but the most commonly used methodology is Maximum Pooling returns the pixel with the maximum value to the output array. Finally, In the fully connected layer, an activation function is applied for classification of inputs based on the feature maps. Several fully connected layers can be used for classification of more advanced datasets.

Quantum Convolutional Neural Networks can help with classification problems that are too complex for classical CNNs, due to recent advancements in quantum computing. Currently, QCNNs have been applied to speech recognition. In such applications, they are coupled with a variational quantum circuit (VQC), which is a specialized quantum algorithm that consists of three elements: a vacuum state, a quantum circuit which contains a set of free parameters, and measurement of local observable outputs. We have enhanced our QCNN framework and applied this algorithm for efficient image recognition.

Our objective was to improve performance with a QCNN in image recognition. The input data was passed through an encoder into qubits, which allowed the QCNN to use superposition and parallelism to do further computations. This should improve the performance for high dimensional data such as medical-image data. If a large amount of data is given to the QCNN model, then the QCNN may perform better than a CNN because the QCNN is utilizing the quantum computing environment to make probability calculations.

Others have implemented QCNNs for character recognition, like ones that have been used to learn the MNIST character dataset, but we used the MNIST handwritten digit dataset.

According to Yann Lecun, this dataset consists of several thousand training samples of 28 by 28 pixel-size handwritten digits from 0 to 9, which works well as a test for studying neural networks [16]. Another paper referenced MNIST for quantum sampling in quantum tomography and used it to gain an intuitive understanding about the runtime for multiple simulations [4]. Much work has yet to be done in the research on QCNNs as their capabilities and their applications are sparse due to their novelty.

Methods

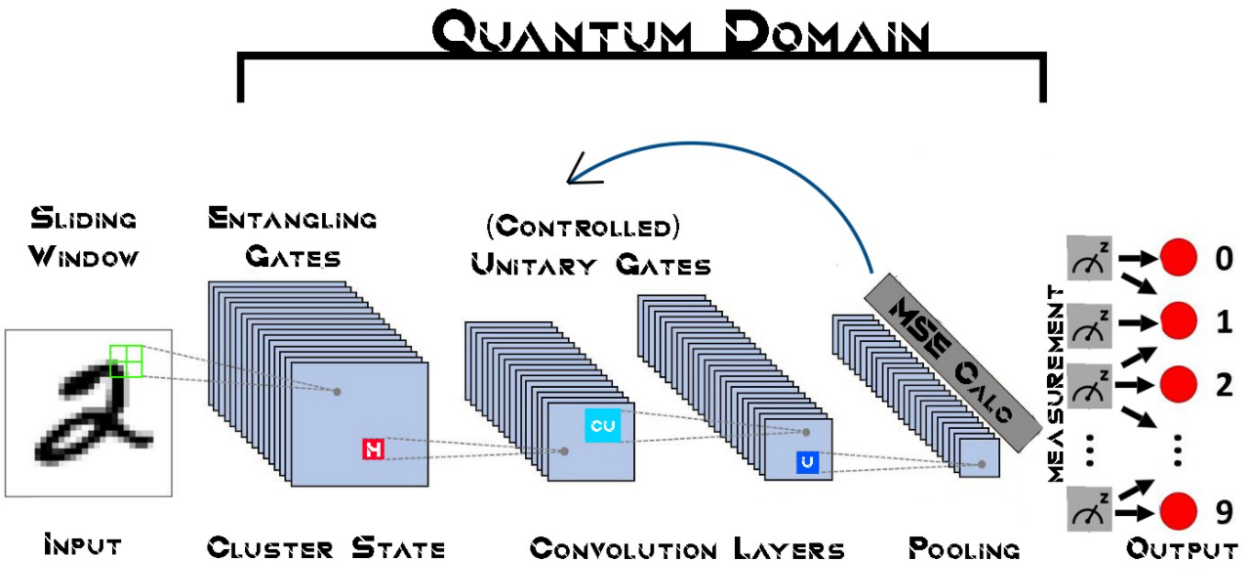

Figure 2: A Brief Overview of our QCNN Architecture

Our methods for improving the performance of Convolutional Neural Networks are described as follows: We have augmented the proven framework of QCNNs and Variational Quantum Circuits and applied this algorithm for efficient image recognition (Figure 2). In addition, an encoder is added for ease of inputting data and a Cluster State is applied to ensure maximum entanglement of the initial quantum states.

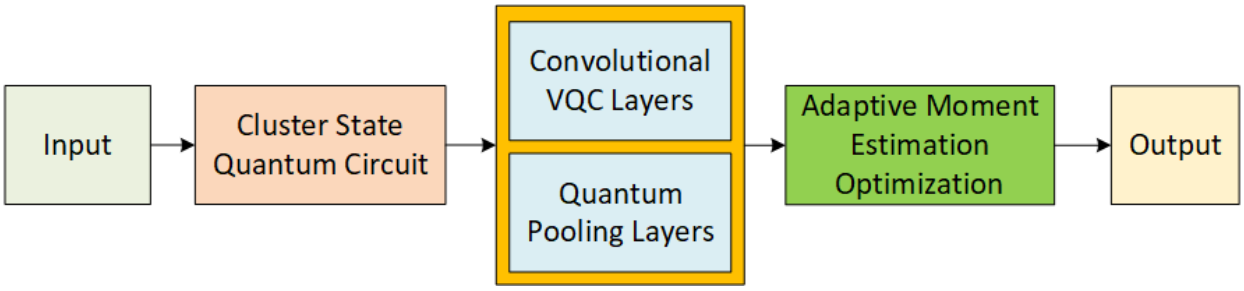

Figure 3: Simplified Flow Chart Describing the Flow of Information from Input to Output

To encode the images from classical data to quantum, , we chose to make the grayscale images into a two dimensional array of simple 1s and 0s instead of taking the value of each pixel and divide it by a coefficient to have numbers under 1 to apply gates to. This converts each image into a binary type system, which is done by selecting a threshold through testing of different values in order to remove blurry parts of the image while keeping key features of each digit. Higher thresholds made images with lower pixel values have key features disappear, while lower thresholds made images bulky and lost key features. The threshold was decided by testing the algorithm on 9s, 6s, and 0s, as these numbers had curves that could easily be lost in the encoding process. We then use the threshold to convert each grayscale pixel (containing a byte value of 0 to 255) to the binary system which is easier to work with in the quantum domain. Any number below the specified threshold is redefined as zero, and any value above it is redefined as one. This novel method of encoding data allows sections of the images that are divided for pooling so that a 2x2 pixel slice can be mapped to 4 qubits.

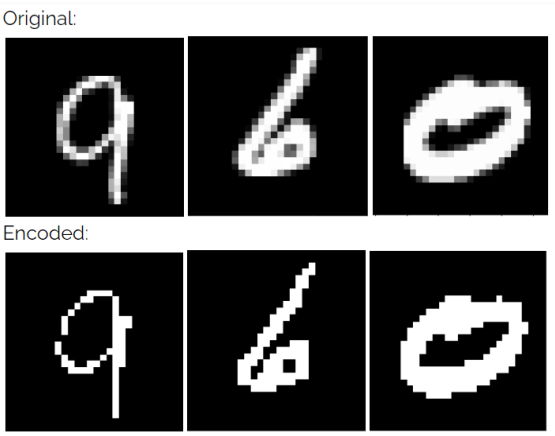

Figure 4: Data Before and After Quantum Embedding Process

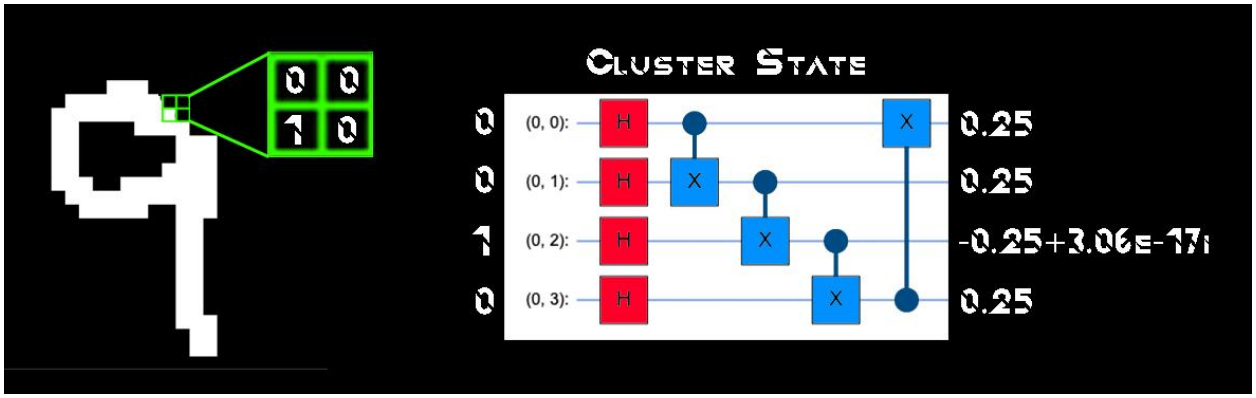

Figure 5: Cluster State Quantum Circuit

These binary values are then inputted into the Cluster State circuit, which achieves a complete entanglement of the input states. This is done by applying a Hadamard gate to each qubit, followed by Controlled-Not (CNOT) gates on each pair of consecutive qubits (1 and 2, 2 and 3, 3 and 4, 4 and 1), followed by Pauli-Y gates with a rotation angle of pi radians (Figure 5).

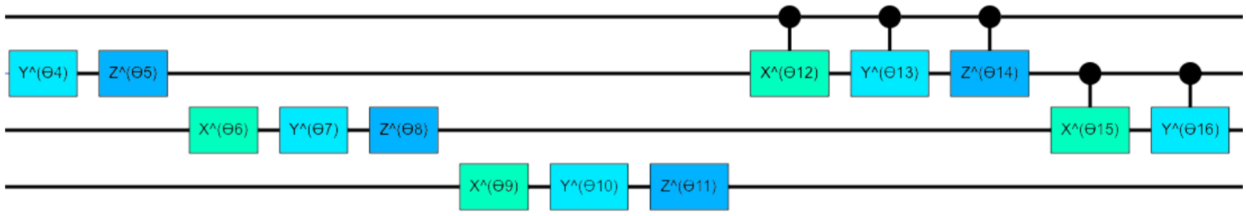

Figure 6: Unitary and Controlled Unitary Gates

Next, we utilized unitary and controlled unitary gates to perform quantum convolution. Instead of having unitary circuits with rotation angles chosen from a uniformly random distribution inside these layers, we implemented single and dual wire unitary gates with theta, phi, and lambda as parameters (Figure 6), which are trainable and correspond to rotations about the X, Y, and Z axes. Some layers will also have CNOT gates in them to reinforce the entanglement previously established in the Cluster State.

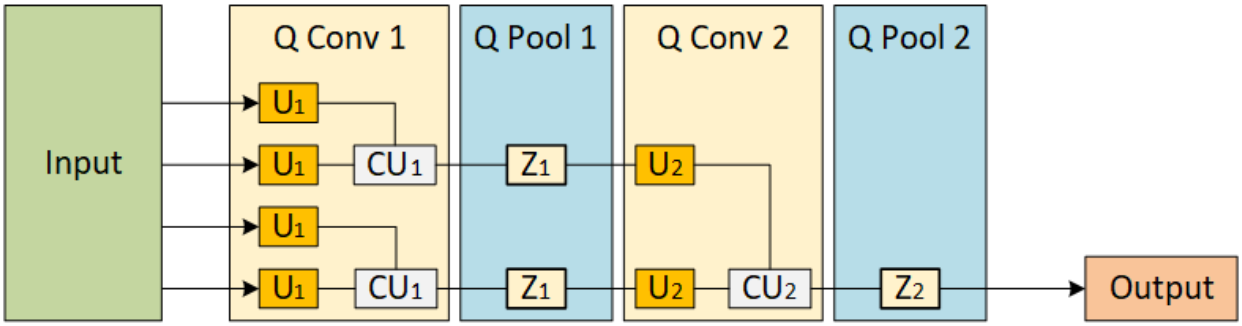

Figure 7: Alternating Convolution and Pooling Layers

We complemented this with a measurement, or pooling layer. A pooling layer consists of various gates which helps condense information from N qubits to N/2 qubits. An alternating pattern of quantum convolution and quantum pooling layers can be used, but we limited our circuit to two pairs of layers for simplicity of training and testing. The final measurement/pooling layer then yields the probability result of the model via measurement gates.

We optimize the rotation angle parameters in the VQCs with the Adam optimizer, which is proven to be more accurate and faster-converging than other optimizers out there [6][10][12]. This will help us have a higher probability result and thus greater prediction accuracy during training and testing.

Finally, once we have acquired sufficient parameters for the VQCs, we will decode the output and be able to recognize and predict which of the handwritten digits the input originally was. This, and the previous optimization step, will be done on a classical computer, as there are no superior quantum alternatives to Adam yet. The results and performance metrics of the QCNN will be compared to that of a CNN, thus proving/disproving our hypothesis.

In summary, we have theorized that more single and dual unitary and controlled unitary gates with more parameters will result in greater accuracy when testing the model, although

optimization speed will be slightly compromised due to the large number of parameters which must be adjusted. So, as a result of these methods, the theorized performance for our QCNN model was testing and training accuracy in the eighties and above (calculated by looking at existing scientific literature), and training time in under 10 minutes for our small dataset of 1000 images.

Results

We have completed the internal architecture of our QCNN, including all the Quantum Convolution and Pooling Layers, as is demonstrated by our code (which is open-source) (Figures 8, 9, 10).

```
[ ]
def QConvLayer1(qubits):
    circuit = cirq.Circuit()
    for first in qubits:
        circuit.append(cirq.Ry(rads=np.pi)(first))
    for first, second in zip(qubits[0::1], qubits[1::1] + [qubits[0]]):
        circuit.append(cirq.CNOT(first, second))
    return circuit
```

Figure 8: Code for a Quantum Convolution Layer

```
[ ]
def QPoolLayer(source, sink, params):
    poolCircuit = cirq.Circuit()
    sinkBasisSelector = U3_Gate(sink, params[0:3])
    sourceBasisSelector = U3_Gate(source, params[3:6])
    poolCircuit.append(sinkBasisSelector)
    poolCircuit.append(sourceBasisSelector)
    poolCircuit.append(cirq.CNOT(control=source, target=sink))
    poolCircuit.append(sinkBasisSelector**-1)
    return poolCircuit
```

Figure 9: Code for a Quantum Pooling Layer

```
    modelCircuit = cirq.Circuit()
    modelCircuit += QConvLayer1(qubits)
    modelCircuit += QConvLayer2(qubits, symbols[0:24])
    modelCircuit += QConvLayer3(qubits, symbols[24:36])
    modelCircuit += QPoolLayer(qubits[0], qubits[2], symbols[36:42])
    modelCircuit += QConvLayer4(qubits, symbols[42:57])
    modelCircuit += QConvLayer5(qubits, symbols[57:67])
    modelCircuit += QPoolLayer(qubits[1], qubits[3], symbols[67:74])
    modelCircuit += QConvLayer6(qubits)
    return modelCircuit
```

Figure 10: Code for the QCNN Model

Currently we are still working on training our model, so our results are not available yet. Our next action item is to transform the quantum data in a way which can be optimized in the classical Adam optimizer. After this, we will train and test the model. Our theorized success criteria for our QCNN model was testing and training accuracy somewhere above 80%, and training time in under 10 minutes for our small dataset of 1000 images. However, we have tested our QCNN on the Tensorflow State Excitations Dataset (predicting whether a particle is excited or not), and we found we had an accuracy of 98% on average.

Discussion

Lessons we have learned are that encoding classical data into the quantum domain requires some creative thinking, which is demonstrated in our specialized encoding algorithm. However, this has a net positive effect as it allows for easy computation in the quantum domain. We have also learned how to construct customized QCNNs and VQCs, and train their parameters and optimize using the Adam optimizer.

Along the way, we faced several challenges, including encoding the classical data into quantum information using the sliding window algorithm, as our QCNN was generalized to have any number of qubits, resulting in a sliding window of varying size and numerous bounds errors. This could possibly lead to loss in information, as the information in each bit is rounded based on

a threshold. In addition, the layers of the QCNN were created manually and gates were selected based on personal experimentation, so some bias could be involved. The biggest challenge we faced was training the model; specifically, transforming the data in a way that could be input into the Adam optimizer. Currently, we have not overcome this challenge, but are working to implement this.

We feel that these results will greatly affect the research field of Quantum Machine Learning, as it demonstrates the capability of QCNNs and their superior performance and efficiency over that of a classical CNN. The wide variety of applications of Quantum Convolutional Neural networks and its high efficiency further asserts quantum computing's leading role in the future of machine learning and artificial intelligence.

The QCNN created over the course of research has a projected accuracy of X%, trained in Y minutes with 1024 shots per input image. On average, CNNs recognize images with 70% accuracy trained with the same number of shots [8], which is less efficient compared to the QCNN. More layers can be easily added to the QCNN, increasing efficiency and complexity, to approach near-optimal accuracy and efficiency.

In the near future, we could train our current QCNN with more images (i.e a larger dataset), as well as use a larger test dataset for more accurate results or to work with images with color. We could also change the internal architecture and adjust the gates in the layers to improve performance and optimization. Later on, QCNNs can be employed in a wide variety of other tasks, such as acoustic data (also known as speech recognition) [17]. We could also experiment with different QML algorithms and compare their performances, such as Hybrid Quantum-Classical Convolutional Neural Networks, or Quantum Generative Adversarial Neural Networks in image recognition. In addition, we could also experiment with different

optimization algorithms for improving performance, such as Quantum Natural Gradient Descent [3][13][15], or implement our own custom optimizer for various applications in Quantum Machine Learning.

Conflicts of Interest/Disclosure

The authors of this paper guarantee that they have no affiliations with or involvement in any organization with any financial or non-financial interests in the content presented in this research manuscript.

Acknowledgements

First and foremost, we would like to thank our research advisor, Dr. Larry McMahan. We would also like to thank the Aspiring Scholars Directed Research Program (ASDRP) for providing us with the opportunity to conduct this research. Lastly, I would like to personally thank my team members for their amazing hard work, patience, and perseverance on this project.

References

[1] Abeer, V., Aishwarya, S., Subramanya, K., Sathish, B. (2021) Quantum Computational Techniques For Prediction Of Cognitive State Of Human Mind From EEG Signals [online], Techscience.com, available: https://www.techscience.com/jqc/v2n4/41123.

[2] Bisarya, A., El Maouaki, W., Mukhopadhyay, S., Mishra, N., Mishra, S., Behera, B. (2020) Breast Cancer Detection Using Quantum Convolutional Neural Networks. [online], medRxiv.org - the preprint server for Health Sciences, available: https://www.medrxiv.org/content/10.1101/2020.06.21.20136655v2.full.pdf.

[3] Bozanic, L. (2020) Quantum Natural Gradient From The Ground Up [online], Medium, available:

https://medium.com/@lana.bozanic/quantum-natural-gradient-from-the-ground-up-983db57cbf6.

[4] Cong, I., Choi, S., Lukin, M. (2019) Quantum Convolutional Neural Networks. [online], Nature, available: https://www.nature.com/articles/s41567-019-0648-8.pdf.

[5] Diego, V., Hans, T., Sneyder, G. (2020) Supervised Learning Using Hybrid Quantum-Classical Neural Networks [online], Universidad Nacional de Colombia, available: https://fagonzalezo.github.io/qcp-2020-2/project/qcp-project-hmtoquicac-davegav-esgantivar.pdf.

[6] Doshi, S. (2020) Various Optimization Algorithms For Training Neural Network [online], Medium, available: https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6.

[7] Du, Y., Hsieh, M., Liu, T., You, S., Tao, D. (2020) On The Learnability Of Quantum Neural Networks [online], arXiv.org e-Print archive, available: https://arxiv.org/pdf/2007.12369.pdf.

[8] Henderson, M., Shakya, S., Pradhan, S., Cook, T. (2019) Quanvolutional Neural Networks: Powering Image Recognition With Quantum Circuits [online], arXiv.org, available: https://arxiv.org/abs/1904.04767.

[9] Houssein, E., Abhohashima, Z., Elhoseny, M., Mohamed, W. (2021) Hybrid Quantum Classical Convolutional Neural Networks For COVID-19 Prediction Using Chest X-Ray Images[online], arXiv.org e-Print archive, available: https://arxiv.org/pdf/2102.06535.pdf.

[10] Kan, E. (2020) How To Implement An Adam Optimizer From Scratch [online], Medium, available: https://towardsdatascience.com/how-to-implement-an-adam-optimizer-from-scratch-76e7b217f1cc.

[11] Liu, J., Lim, K., Huang, W., Guo, C., Huang, H. (2021) Hybrid Quantum-Classical Convolutional Neural Networks. Arxiv.Org E-Print Archive. [online], arXiv.org e-Print archive, available: https://arxiv.org/pdf/1911.02998.pdf.

[12] Musstafa (2021) Optimizers In Deep Learning [online], Medium, available: https://medium.com/mlearning-ai/optimizers-in-deep-learning-7bf81fed78a0\.

[13] Stokes, J., Izaacs, J., Killoran, N., Carleo, G. (2019) Quantum Natural Gradient. [online], ML4PhysicalSciences, available: https://ml4physicalsciences.github.io/2019/files/NeurIPS_ML4PS_2019_7.pdf.

[14] Telahun, M. (2020) Exploring Information For Quantum Machine Learning Models. [online], Ir.library.louisville.edu, available: https://ir.library.louisville.edu/cgi/viewcontent.cgi?article=4808&context=etd.

[15] YouTube (2021) An Intro To Quantum Natural Gradient Descent [online], available: https://www.youtube.com/watch?v=NhAv76nFIs8.

[16] LeCun, Y., Cortesk, C. (n.d.) THE MNIST DATABASE Of Handwritten Digits [online], available: http://yann.lecun.com/exdb/mnist/

[17] Lee, Y., Lim, S., Kwak, I. (2021) *CNN-Based Acoustic Scene Classification System* [online], electronics: Seoul, available:

https://res.mdpi.com/d_attachment/electronics/electronics-10-00371/article_deploy/electronics-10-00371.pdf.