Reproducing and Reviewing "Quantum Adaptive Excitation Network with Variational Quantum Circuits for Channel Attention"

CPEN 416
University of British Columbia

Chirag Raisingh
Evan Nawfal
Quinn Senych
Robert Walsh

Professor Olivia Di Matteo

December 8, 2025

Introduction

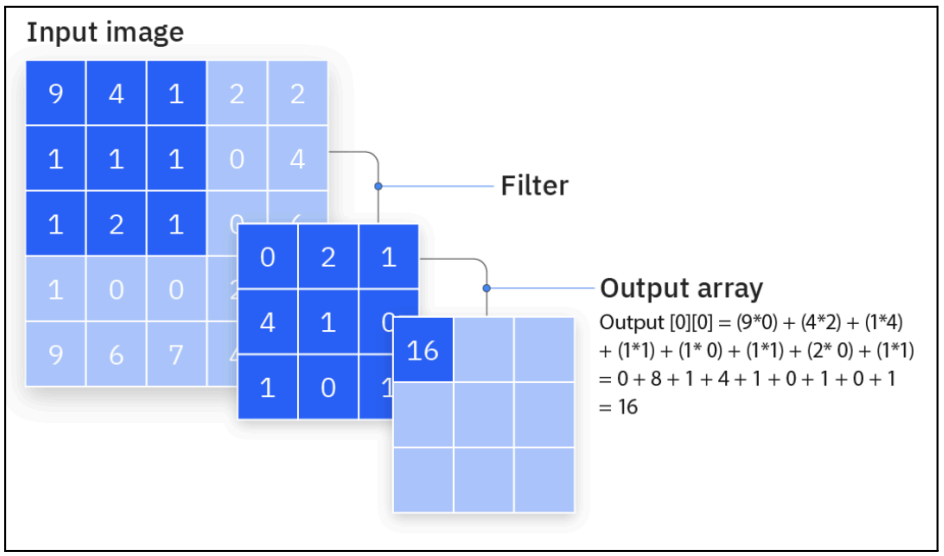
This final report reviews and tries to reproduce the results of "Quantum Adaptive Excitation Network with Variational Quantum Circuits for Channel Attention" [1], which proposes the idea of replacing the classical squeeze and excitation block inside of a convolutional neural network with a variational quantum circuit. This report begins by providing background theory that is necessary for understanding convolutional neural networks, squeeze and excitation networks, and variational quantum circuits, as well as the MNIST, Fashion-MNIST, and CIFAR-10 datasets that were used in the paper [1] and the reproduction effort. Using PennyLane and Pytorch, the QAE-Net architecture and its classical SENet were reimplemented and compared against the original paper, where the accuracy and training curves of the models were obtained. Practical issues in reproducibility and differences in performance were discussed, as well as what may have caused them, especially for CIFAR-10 where we saw a large difference in results. Finally, an analysis of the paper and of the reproduction results were provided where the relevance of the paper and its implementation decisions were discussed.

Background Theory

In order to properly replicate the paper, a proper understanding of Convolutional Neural Networks, Squeeze and Excitation Networks, and Variational Quantum Circuits was required. Each of these components are described below.

Convolutional Neural Networks (CNNs)

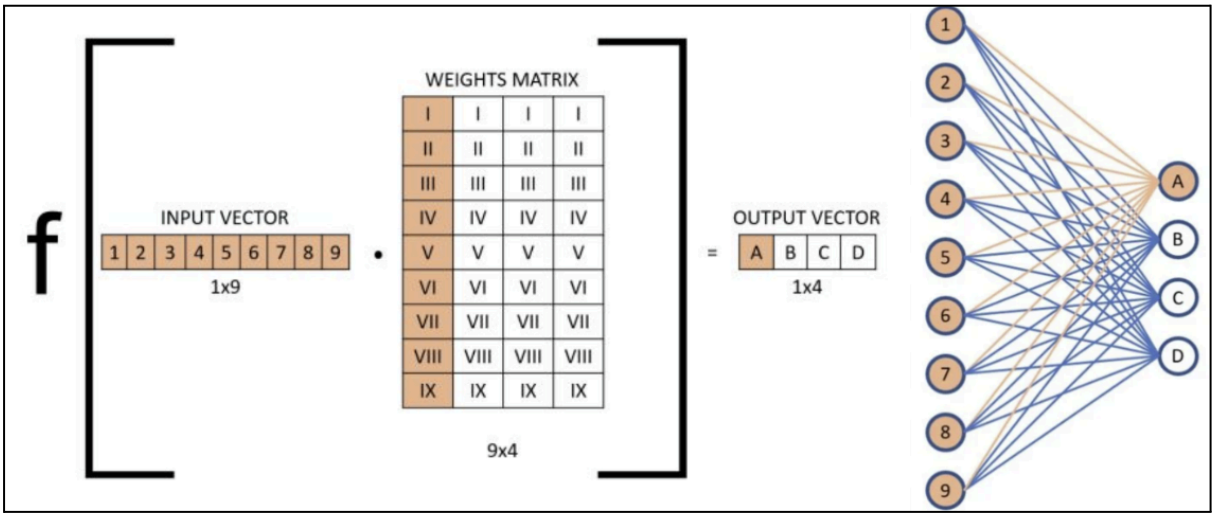
The heart of this research paper involves implementing and optimizing a convolutional neural network (CNN) with the use of quantum technologies. Hence, it makes sense to explore in depth what a CNN is and how it functions. What differentiates CNNs from other types of neural networks is its unique ability and superior performance with image, speech or audio signal inputs [7]. Typically CNNs consist of 3 different layers: the convolutional layer, the pooling layer and the fully connected layer, all of which work in conjunction with each other to produce a desired outcome. In the convolutional layer, a kernel strides over the input matrix, taking a dot product of all the input data pixels and the values in the kernel. The operation of a convolutional layer can be seen in [Figure 1]. The output of these iterations result in a populated feature map. Additional convolutional layers can be used as well to interpret and extract relevant patterns in greater detail.



[Fig 1: Operation of a Convolutional Layer, Retrieved from [7]]

The next layer is the pooling layer whose main function is to reduce complexity and increase efficiency by reducing the number of samples of the input. It does this in a similar fashion as the convolutional layer but instead of a feature detector with multiple weights, it instead aggregates all of the input pixels it covers and runs them through an aggregating function, more specifically max pooling or average pooling [7].

Finally, the last layer is referred to as the fully connected layer. The name aptly describes its function, as it serves to connect all the input neurons to the output. [8]. Refer to [Figure 2] for a visualization of this layer.



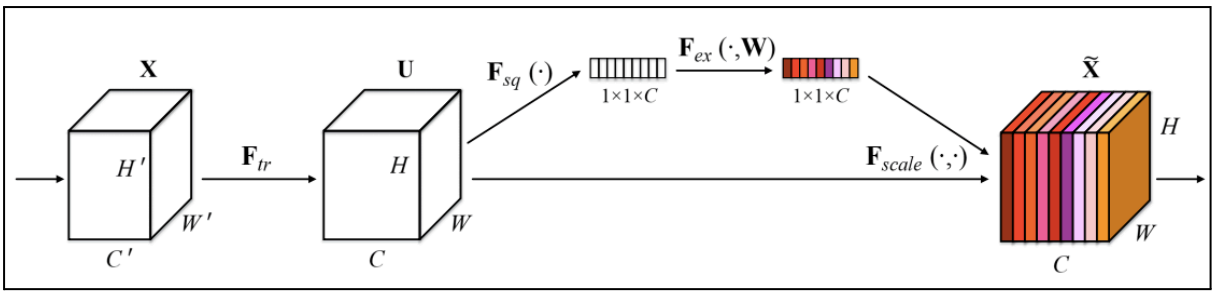
[Fig 2: Operation and Visualization of a Fully Connected Layer, Retrieved from [8]]

Squeeze and Excitation Networks (SENets)

A Squeeze-and-Excitation Network [2] is an improved version of a CNN. It uses a special block known as a squeeze and excitation block to allow it to pick up on important relationships between different channels, and use these relationships to improve its classification ability [2]. As the name suggests, a SENet consists of two main parts; a squeeze block, and an excitement block.

In the squeeze stage, global average pooling is applied across the spatial dimensions of each feature channel [2]. This compresses the spatial information and produces a single value per channel, capturing how important each channel is globally.

In the excitation stage, this vector of channel descriptors is passed through a small neural network called a multilayer perceptron (MLP) [9]. The MLP learns the relationships between channels and outputs a set of normalized weights between 0 and 1, These weights are then used to rescale the original feature map through channel wise multiplication, emphasizing important channels and suppressing less useful ones [9]. The refined feature map is then passed to the rest of the CNN for further processing.



[Fig 3: Squeeze and Excite Network Workflow, Retrieved from [2]]

Variational Quantum Circuits (VQCs)

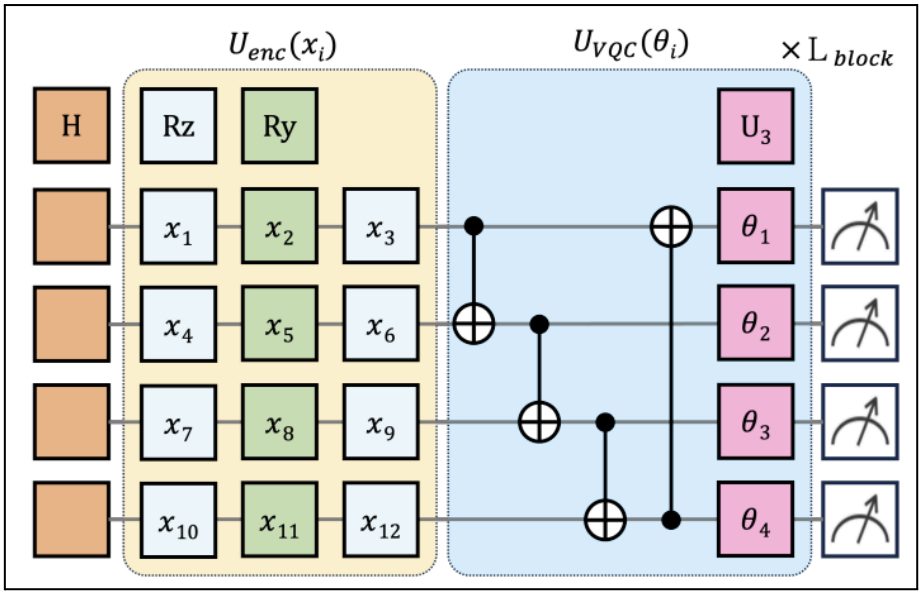
A Variational Quantum Circuit (VQC) is a parameterized quantum model that combines quantum state evolution with classical optimization. The circuit contains a set of tunable rotation angles, and these parameters are iteratively updated based on gradients that come from the measurement outcomes of the circuit. A VQC generally follows these three stages:

1. Preparation of an initial quantum state, where classical inputs are encoded to rotations for a qubit.
2. A quantum circuit U(θ) with some ansatz value θ.
3. Measurement at the end of the circuit, where expectation values of observables (Pauli operators) are passed back to the classical network

A Hybrid loop like this allows the quantum part to represent nonlinear transformations while the classical portion handles gradient descent, enabling VQCs to behave as drop in replacements for neural network layers [1].

In classical Squeeze and Excitation networks, an MLP learns its dependencies between feature channels and outputs a set of reweighting coefficients for the channel. Hsu et al. replace this MLP with a VQC under the hypothesis that quantum entanglement might capture inter-channel relationships better than classical methods, especially for RGB images. By encoding channel descriptors into qubits and allowing entangling gates to represent high order interactions, the VQC can function as a quantum version of the excitation block. Its final expectation value represents attention weights that scale the CNN's feature maps in the same way that the output of a classical SENet excitation block would.

The model used for replicating the paper uses angle encoding to encode classical channel values using angle encoding, where each value x_i becomes a rotation for gates like $RZ(x_i)$ and $RY(x_i)$. The VQC is built from layers of single qubit rotations followed by CNOT gates, and this block is repeated L times to change the circuit depth. After running the circuit, we measure Pauli Z expectation values, which become the channel weightings that replace the excitation step in SENet. This design was implemented using PennyLane[5] and was trained in PyTorch[6]. For CIFAR-10, a four qubit circuit was used, with smaller versions for MNIST and F-MNIST, and trained models with one to three variational layers [1].



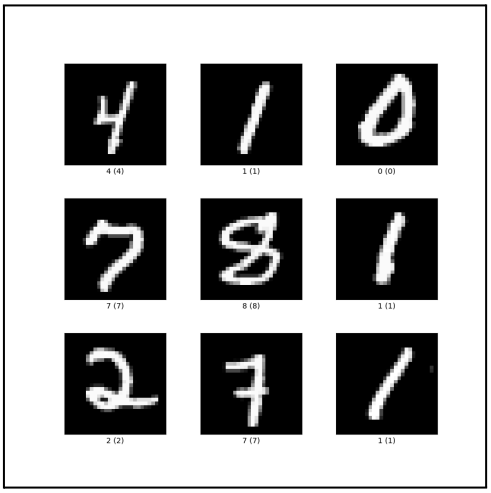
[Fig 4: Variational Quantum Circuit, Retrieved from [1]]

Datasets

We trained our model as the same three datasets as the paper.

MNIST

The MNIST dataset contains 28x28 grayscale images of handwritten digits across ten classes. Because it is simple and low dimensional, it serves as a baseline that helps confirm that the architecture in the CNN and VQC are functioning correctly on easy image classification tasks [10]. The dataset has 60,000 training, and 10,000 testing images [10]. An important thing to note is that MNIST contains only a single channel, meaning channel attention mechanisms (both classical and quantum) are less impactful, making MNIST useful for checking correctness but not for evaluating the claimed benefits of quantum attention.

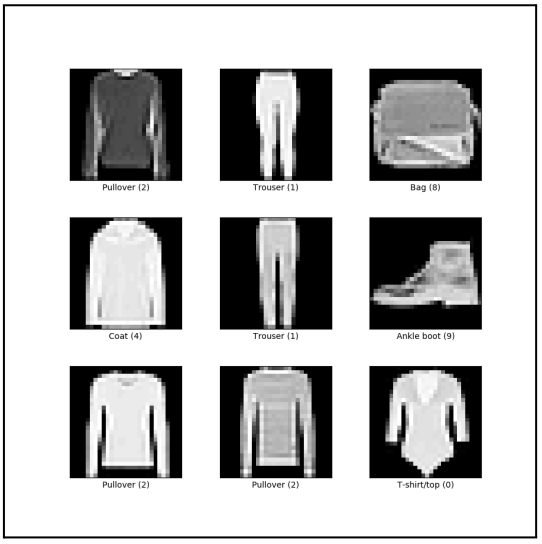


[Figure 5: Examples of MNIST Images, Retrieved from [10]]

F-MNIST

F-MNIST is another 28x28, black and white dataset with 10 classes. In this case it is images of clothes. It has the same 60,000 training, 10,000 testing split of images [11]. Since the images are

slightly more complex than MNIST it is another good test of model accuracy, but the images being black and white lessens the importance of channel relationships, meaning that the Excitation blocks (VQC or SEN) will have less of a chance to improve performance. An example of images from the F-MNIST dataset can be found in Figure 6, below.



[Figure 6: Examples of F-MNIST Images, Retrieved from [11]]

CIFAR-10

CIFAR-10 is a dataset with 60,000 images. 10,000 test and 50,000 training images. There are 10 classes spanning a variety of real world objects (specifically animals and vehicles). All images are in colour [12]. The CIFAR-10 dataset saw the biggest performance difference between quantum and classical networks in the paper [12]. An example of CIFAR-10 images can be found in Figure 7, below.

[Figure 7: Examples of CIFAR-10 Images, Retrieved from [12]]

Results

The results of our implementation were similar to those of the paper when training our models on the MNIST and F-MNIST datasets, however, our models performed very differently on CIFAR-10 than they did in the paper. Not only was the accuracy of our SENet 10% worse than the paper's when

trained on CIFAR-10, but we did not see any increase in accuracy when using QAE-Net [1], compared to the 13% improvement achieved by the paper.

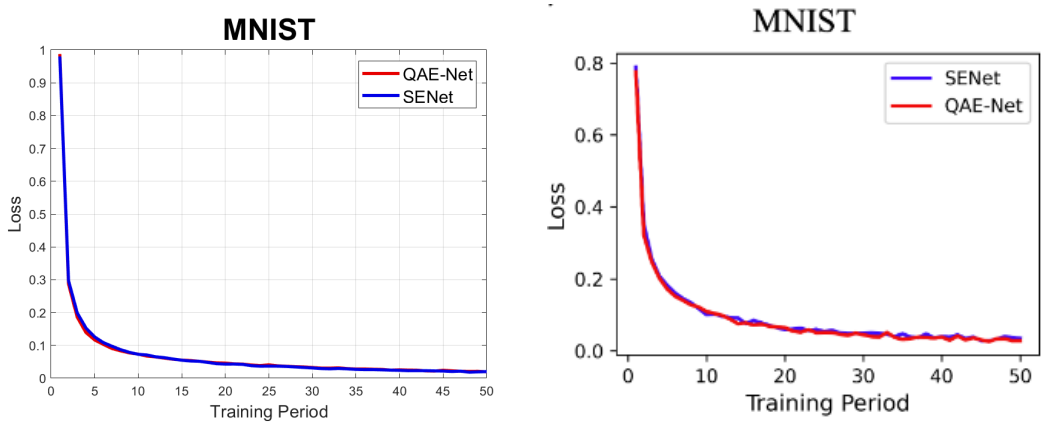
The summarized accuracy results can be found in Table 1, below.

Dataset	Model	Paper Accuracy	Our Accuray
MNIST	SENet[2]	97.9	99.09
	QAE-Net[1]	98.0	99.21
F-MNIST	SENet[2]	91.0	89.85
	QAE-Net[1]	91.3	90.33
CIFAR-10	SENet[2]	76.72	66.32
	QAE-Net[1]	89.08	66.22

[Table 1: Accuracy Results of Original And Replicated Models]

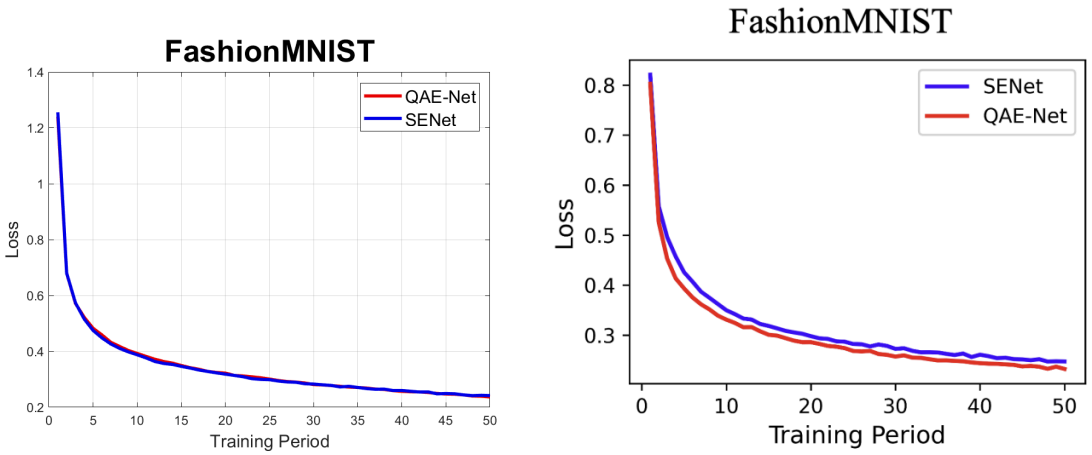
The results of model training were plotted with training period in epochs on the horizontal axis, and loss on the vertical axis. This format allows a visualization of how the model improves with successive rounds of training.

The training curves of all models on MNIST can be found in Figure 8, below. These curves are very similar.



[Figure 8: Visual Comparison of QAE-Net and SENet Training on MNIST from Our Implementation (Left), and the Paper Implementation (Right)]

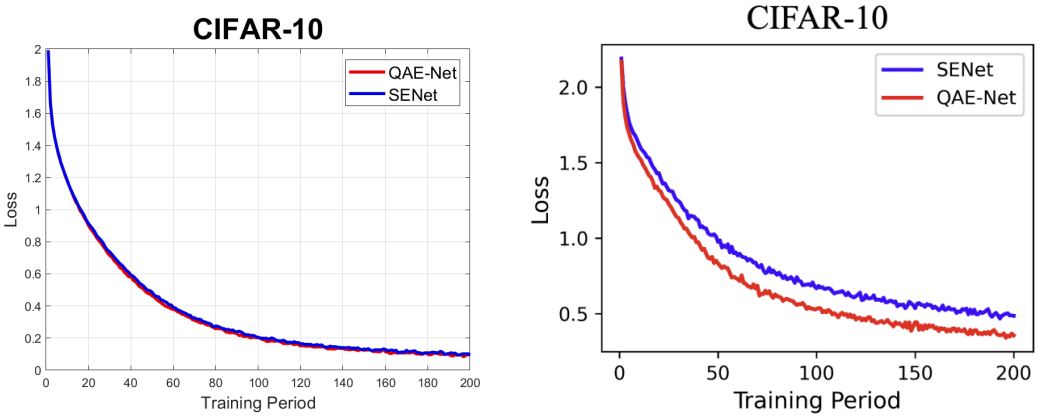
The training curves of all models on F-MNIST can be found in Figure 9, below. Like with MNIST, these curves are very similar.



[Figure 9: Visual Comparison of QAE-Net and SENet Training on F-MNIST from Our Implementation (Left), and the Paper Implementation (Right)]

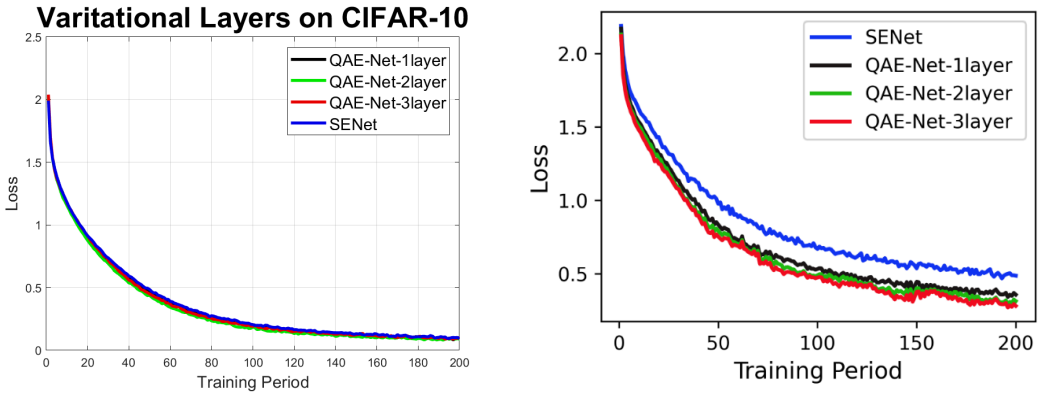
Finally, the training curves of all models on CIFAR-10 can be found in Figure 10, below. Unlike with the MNIST datasets, these curves are clearly very different. In our implementation, the loss values in the final epochs of training appear to be lower, and our QAE-Net [1] and SENet [2] have very similar curves. On the other hand, in the paper implementation, both networks appear to have higher loss values towards the end of training, and between the two, QAE-Net [1] achieves and sustains lower loss values quicker than SENet. The lower loss values displayed by our models compared to those of the paper imply that our models are more confidently wrong, which is reflected in our lower final accuracy in Table 1, above.

While we are unsure of the exact reasons for such a divergence between our results and the paper's when we trained our model on CIFAR-10, one potential reason could be dimensional differences in our CNNs. The paper's CNN was underspecified, and therefore some of the architecture was inferred. For exact specifics, see the section on reproducibility below.



[Figure 11: Visual Comparison of QAE-Net and SENet Training on CIFAR-10from Our Implementation (Left), and the Paper Implementation (Right)]

As well as testing SENet [2] and QAE-Net [1] models on all 3 datasets, the paper trained 2 and 3-Layer QAE-Nets [1] on CIFAR-10 and compared the results to the 1-layer QAE-Net [1] model, as well as SENet [2]. Once again, our models saw a large discrepancy in results compared to the results of the paper. The visualization of our results compared to the papers can be found in Figure 12, below. The paper saw a very clear decrease in loss during training for QAE-Nets [1] with more layers, and all QAE-Nets [1] outperformed SENet [2], whereas our models all had nearly identical training curves.



[Figure 12: Visual Comparison of 1,2, and 3 layer QAE-Net trained on CIFAR-10 from Our Implementation (left), and the paper (Right)]

Overall, the results of our SENet and QAE-Net models were consistently very similar. It is interesting to see that a squeeze and excitation block can in fact be effectively replicated with a VQC, however, it does raise the question of what exactly the paper did differently to make their QAE-Net model so much more effective on colour images specifically, when ours was unable to achieve the same results.

Reproducibility

Replicating the results of QAE-Net [1] presented a challenge. Many configuration details were left out of the paper and our group also made design decisions to diverge from what the paper explicitly called out to increase training speed to be able to collect data in a reasonable way.

Intentional Divergences from the Paper

The paper specifically mentions using Pennylane's [5] "lightning.gpu" device. We found that using lightning.gpu significantly slowed down the simulation, so we changed to "default.qubit". Reading documentation we discovered the "lightning.gpu" device is meant for circuits with a large number of qubits or when running a circuit with a large number of GPUs. It is logical that "default.qubit" works significantly better on our consumer grade hardware as opposed to NVIDIA Tesla V100s.

We also implemented batched training. This was not mentioned in the paper at all but had the greatest effect on speed. Training a model on MNIST before implementing batching would take well over 16 hours, whereas after batching, it would take around 10 minutes. This forced us to make another change: the paper explicitly uses the parameter-shift rule to calculate the gradient for optimization. While this method is how a quantum circuit would actually calculate gradient, as we have access to the internal calculations we can cheat and use a more efficient gradient calculation method such as back-propagation. The change in calculation method should not change the mathematical result, therefore not affecting the training process. We were concerned about the size of batching possibly affecting the accuracy results [3], however upon running tests with batch sizes of 1, 16, and 1024, we found they all had roughly the same loss and accuracy after 10 epochs (taking 8 hours, 1 hour, and 10 minutes respectively).

Ambiguities

The paper touches on optimization, saying that the entire model is optimized using gradient-based back-propagation using a cross-entropy loss function [1]. However, the actual optimizer used in the PyTorch [6] implementation was not specified. Optimizers such as SGD and Adam can have vastly different required learning rates and final accuracy [4]. We decided to use torch.optim.Adam as a

more robust choice, however, SGD also seemed to be a valid optimizer module used for similar models. Had the authors used SGD or a different momentum based optimizer such as RMSprop or AdaGrad, our use of Adam could be a direct cause of the significant divergence in results for the CIFAR-10 dataset.

There were also ambiguities in the dimensioning and complete details of both the CNN backbone as well as the squeeze and excitation block. The CNN backbone was missing details on the inclusion of pooling layers and activation functions which were required to allow the dimensions of the input and output of the attention mechanism to align. These details were likely omitted due to being standard practice in neural network architecture, however, with our limited experience, this was a major source of confusion for us. Additionally, the dimensions of the linear layers within the squeeze and excitation network [2] used in QAE-Net [1] were not specified, we deduced they likely used a reduction ratio of 3 to match the quantum architecture, however, this could be a source of our error.

Analysis

While we were unable to reproduce the exact results of the paper, we still believe that this paper is very relevant to the field of quantum computing, especially for the NISQ era. With machine learning becoming more prevalent every day, a quantum circuit that is purpose-built to help improve model accuracy seems like a very viable investment.

However, the divergence on the CIFAR-10 dataset, where the paper claimed a 13% performance boost that we could not replicate, suggests the quantum advantage is highly dependent on a specific aspect of their implementation.

Furthermore, the paper's claim that increasing the VQC depth (from 1 to 3 layers) results in progressively better performance was not observed in our tests. Our multi-layered QAE-Net tests exhibited virtually identical loss curves to the single-layer version. This raises questions about the VQC's true capacity for this specific application: either the advantage is locked behind an extremely difficult optimization barrier, or the small 4-qubit circuit architecture has limited ability to scale its performance for more complex data like the 3 dimensional CIFAR-10 dataset.

Arguably more importantly, even though the small size of the quantum circuit could be a potential limitation, the size is realistic for our modern era. Unlike some theoretical quantum algorithms which require a number of qubits that is infeasible in the NISQ era, this circuit is only 4 qubits wide, and only has a depth of 5 for the VQC, (or 9 including the encoding block), making it much more likely to be realistic to implement in the near-term. Additionally, presuming the exact optimization and model architecture in the paper, the results that showed improved accuracy with an increased number of VQC layers also indicate that this quantum circuit would hopefully scale well alongside improvements in quantum hardware, potentially consistently improving performance as more hardware becomes available.

Therefore, despite the fact our implementation did not show the same large improvement on the CIFAR-10 dataset as the paper's. Our implementation showed at least comparable performance to its classical equivalent, proving promising for the future.

References

[1] Yu-Chao Hsu, Kuan-Cheng Chen, Tai-Yue Li, and Nan-Yow Chen, “Quantum Adaptive Excitation Network with Variational Quantum Circuits for Channel Attention,” arXiv preprint arXiv:2507.11217, 2025.

[2] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7132–7141, 2018.

[3] Yann A LeCun, L´eon Bottou, Genevieve B Orr, and Klaus-Robert M¨uller. Efficient backprop. In Neural networks: Tricks of the trade, pp. 9–48. Springer, 2012.

[4] S. Ruder, “An overview of gradient descent optimization algorithms,” arXiv preprint arXiv:1609.04747, 2016.

[5] V. Bergholm, J. Izaac, M. Schuld, C. Gogolin, S. Ahmed, V. Ajith, M. S. Alam, G. Alonso-Linaje, B. AkashNarayanan, A. Asadi, et al., “Pennylane: Automatic differentiation of hybrid quantum-classical computations,” arXiv preprint arXiv:1811.04968, 2018.

[6] A. Paszke, “Pytorch: An imperative style, high-performance deep learning library,” arXiv preprint arXiv:1912.01703, 2019.

[7] IBM, “Convolutional Neural Networks,” IBM Think Topics, IBM Corporation, 2024. [Online]. Available: https://www.ibm.com/think/topics/convolutional-neural-networks

[8] Built In, “What Is a Fully Connected Layer?,” Built In, 2024. [Online]. Available: https://builtin.com/machine-learning/fully-connected-layer

[9] DigitalOcean, “Channel Attention: Squeeze-and-Excitation Networks,” DigitalOcean Community Tutorials, 2024. [Online]. Available: https://www.digitalocean.com/community/tutorials/channel-attention-squeeze-and-excitation-networks

[10] TensorFlow Datasets, “mnist | TensorFlow Datasets,” *TensorFlow*, 2010. https://www.tensorflow.org/datasets/catalog/mnist

[11] TensorFlow Datasets, “fashion_mnist | TensorFlow Datasets,” *TensorFlow*. https://www.tensorflow.org/datasets/catalog/fashion_mnist

[12] TensorFlow Datasets, “cifar10 | TensorFlow Datasets,” *TensorFlow*. https://www.tensorflow.org/datasets/catalog/cifar10

[13] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images,” Apr. 2009. Available: https://www.cs.toronto.edu/%7Ekriz/learning-features-2009-TR.pdf