# *Let the Quantum Creep In:*

## Designing Quantum Neural Network Models by Gradually Swapping Out Classical Components


Peiyong Wang[1], Casey R. Myers[3 1], Lloyd C. L. Hollenberg[2], Udaya Parampalli[1]

[1]School of Computing and Information Systems, The University of Melbourne [2]School of Physics, The University of Melbourne

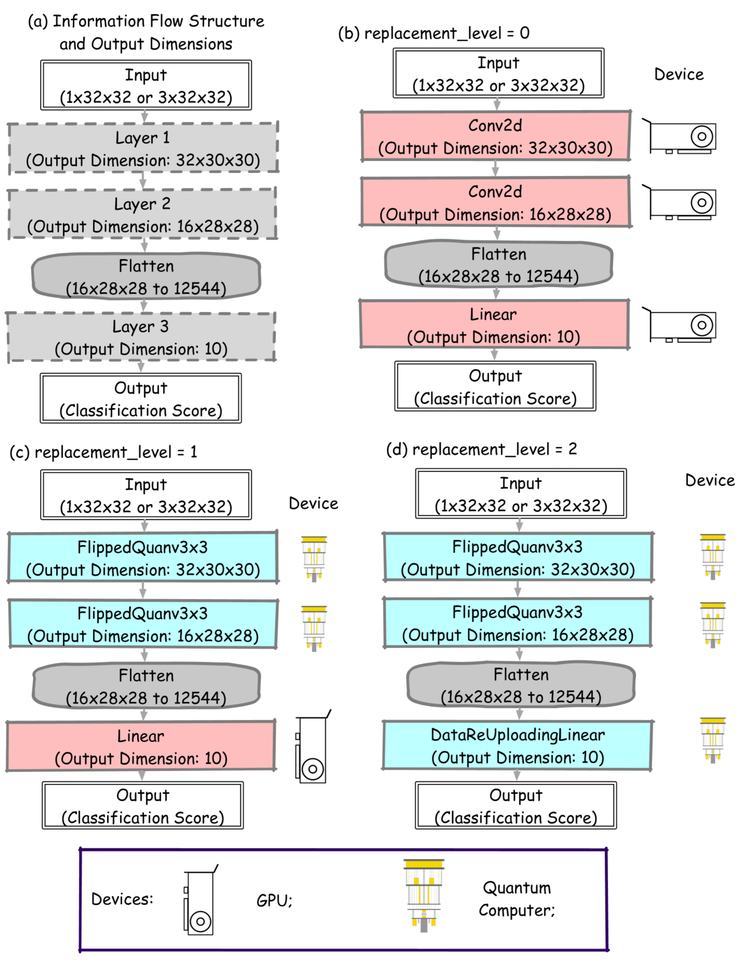
[3]School of Physics, Mathematics and Computing, The University of Western Australia



We propose a framework for integrating quantum computing into AI by gradually replacing classical neural network layers with quantum layers, maintaining consistent input-output flow. Unlike most end-to-end quantum neural network approaches, this method provides a framework for a finer analysis of quantum components' impact on performance. Starting with a simple three-layer classical neural network, we systematically introduce quantum layers and evaluate performance on datasets like MNIST, FashionMNIST, and CIFAR-10. Our results highlight the potential of hybrid classical-quantum designs, offering insights for future quantum neural network development.


## The Framework

For a (more) fair comparison between classical NN and QNN, instead of designing an end-to-end QNN from scratch, we first start with a fully classical model (neural network), and gradually swap the classical NN layers with their quantum counterparts which have the same type of inputs and outputs. Hence the name "Let the quantum creep in".




Overview of the proposed framework. The symbol for the quantum computer is inspired by [1].

- **(a)** the information flow structure and the required dimensions of the input and output of each vacancy for candidate neural network layers. Double-lined boxes are the input and output of the neural network; Dash-lined boxes are layer vacancies for candidate neural network layers. The information passed between layers and the flatten operation are classical, while the candidate neural network layers could be either classical or quantum.
- **(b)** `replacement_level=0`. In this case all vacancies are filled with classical neural network layers (`Conv2d` and `Linear`). All these layers are executed on a GPU with classical neural network libraries.
- **(c)** `replacement_level=1`. In this case, the classical convolution layers `Conv2d` are replaced with its quantum counterpart `FlippedQanv3x3`, while the classical `Linear` layer left unchanged.
- **(d)** `replacement_level=2`. In this case, all the classical layers in (b) are replaced with their quantum counterpart, i.e. `Conv2d` $\rightarrow$ `FlippedQanv3x3` and `Linear` $\rightarrow$ `DataReuploadingLinear`.

All quantum layers are simulated on a GPU when training and testing the neural network model.

## `FlippedQanv3x3`

For a 3 by 3 patch of an image, $x$

$$x = \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix},$$

padded to a 4 by 4 matrix $M_x$

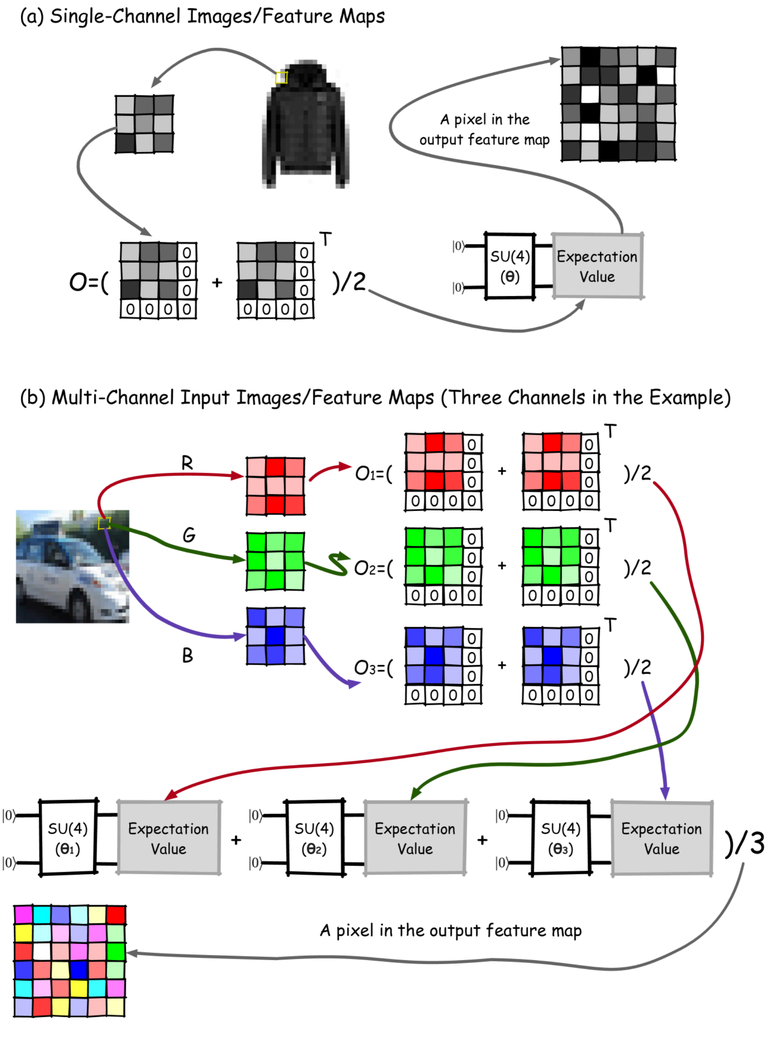
$$M_x = \begin{pmatrix} x_1 & x_2 & x_3 & 0 \\ x_4 & x_5 & x_6 & 0 \\ x_7 & x_8 & x_9 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

the `FlippedQanv3x3`, parameterised by $\boldsymbol{\theta}$ reads as:

$$\texttt{FlippedQanv3x3}_{\boldsymbol{\theta}}(x) = \langle \varphi(\boldsymbol{\theta}) | \mathcal{O}(x) | \varphi(\boldsymbol{\theta}) \rangle,$$

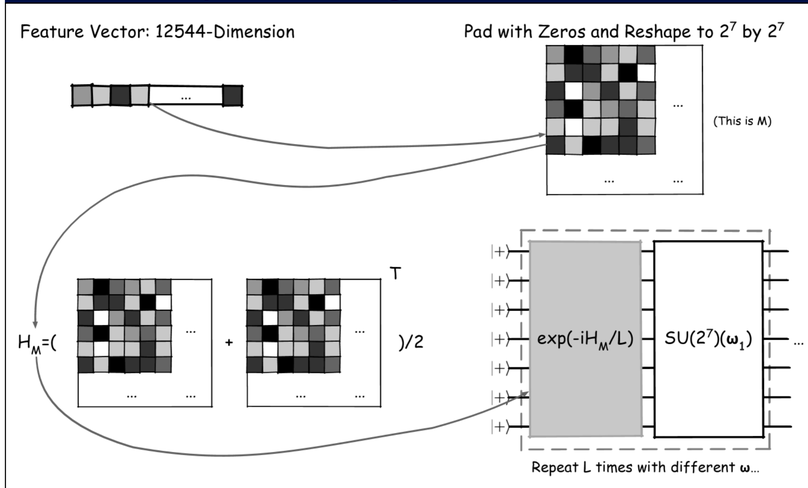
$$\mathcal{O}(x) = \frac{1}{2}(M_x + M_x^{\mathsf{T}}),$$

where $|\varphi(\boldsymbol{\theta})\rangle$ is a four-qubit paraemterised quantum state. In our paper [2], we have shown that mathematically, `FlippedQanv3x3` is equivalent to a classical convolution layer with respect to the input data.




- **(a)** Images and feature maps that only have one channel only need a single circuit for each patch x.
- **(b)** For images and feature maps that have multiple channels, the patch of image within the view of the `FlippedQanv3x3` kernel is a 3-D tensor with the the shape $(C, 3, 3)$. In this example, we take $C = 3$, and in this case, three circuits with different parameters are required from the observables constructed from each channel to calculate the output of the `FlippedQanv3x3` kernel operation.

## `DataReuploadingLinear`




The `DataReuploadingLinear` layer at the end of the hybrid neural network. It takes a $12544$-dimension feature vector from the `Flatten` layer, pad it with zeros and reshape it to a $2^7 \times 2^7$ square matrix $M$. A quantum Hamiltonian $H_M$ is constructed with $M$, and this Hamiltonian will be used to construct the "time"-evolution operator $W = e^{-\frac{iH_M}{L}}$ following the method proposed in our previous research [3], where $L$ is the number of layers of the data re-uploading circuit. In this project, we fix $L = 10$. The trainable layers in the data reuploading circuit are parameterised $\mathrm{SU}(2^7)$ unitaries.

## Results

| Dataset | Metric (Average Value) | Replacement Level 0 | 1 | 2 |
|---|---|---|---|---|
| MNIST | Train Loss | 0.229 | 0.227 | **0.076** |
| | Test Loss | 0.294 | 0.295 | **0.125** |
| | Train Accuracy | 93.5% | 93.6% | **97.8%** |
| | Test Accuracy | 92.3% | 92.3% | **96.9%** |
| FashionMNIST | Train Loss | 0.350 | 0.353 | **0.280** |
| | Test Loss | 0.476 | 0.477 | **0.405** |
| | Train Accuracy | 87.6% | 87.5% | **90.1%** |
| | Test Accuracy | 83.6% | 83.4% | **86.6%** |
| CIFAR-10 | Train Loss | 1.50 | 1.61 | **1.28** |
| | Test Loss | 1.89 | 1.78 | **1.49** |
| | Train Accuracy | 48.9% | 45.0% | **55.0%** |
| | Test Accuracy | 37.0% | 38.9% | **48.9%** |

In our research, we trained all three different replacement levels on three different datasets (MNIST, FashionMNIST and CIFAR-10). Each training combination (replacement_level × dataset) is repeated five times with different parameter initialisations. The loss function is softmax cross-entropy for the 10-label classification task. We used linear algebra functionalities in PyTorch for circuit simulation, as well as `torch.optim.Adam` for training. The numerical experiments are conducted on a single NVIDIA H100 GPU. When replacement_level = 0, all layers in the neural network are classical, i.e. classical Conv2d and classical Linear; when the replacement_level = 1, only the classical convolution layers are replaced with the quantum FlippedQuanv3x3 layer; when the replacement_level = 2, both the classical convlution and linear layers are replced with their quantum counterparts, i.e. Conv2d → FlippedQuanv3x3 and Linear → DataReUploadingLinear. We can see that the DataReUploadingLinear brought a non-trivial performance increase from replacement_level = 1 to replacement_level = 2.

## References


[1] M. Cerezo *et al.*, "Does provable absence of barren plateaus imply classical simulability? Or, why we need to rethink variational quantum computing." [Online]. Available: http://arxiv.org/abs/2312.09121

[2] P. Wang, C. R. Myers, L. C. L. Hollenberg, and U. Parampalli, "Let the Quantum Creep In: Designing Quantum Neural Network Models by Gradually Swapping Out Classical Components." Accessed: Sep. 27, 2024. [Online]. Available: http://arxiv.org/abs/2409.17583

[3] P. Wang, C. R. Myers, L. C. L. Hollenberg, and U. Parampalli, "Quantum Hamiltonian embedding of images for data reuploading classifiers." Accessed: Aug. 09, 2024. [Online]. Available: http://arxiv.org/abs/2407.14055



QTML2024 github.com/peiyong-addwater/Let-The-Quantum-Creep-In arXiv:2409.17583