PROJECT EXPO'24

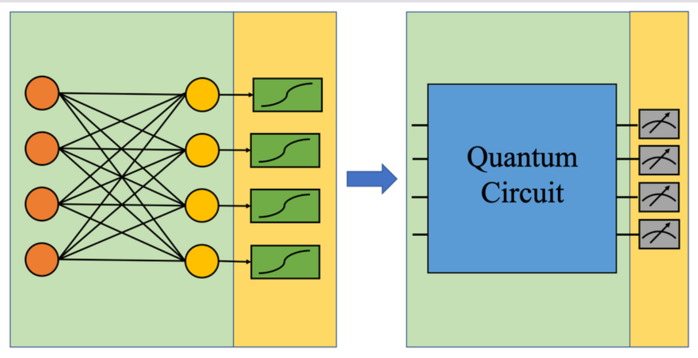
Quantum-Classical Hybrid Machine Learning for Image Classification

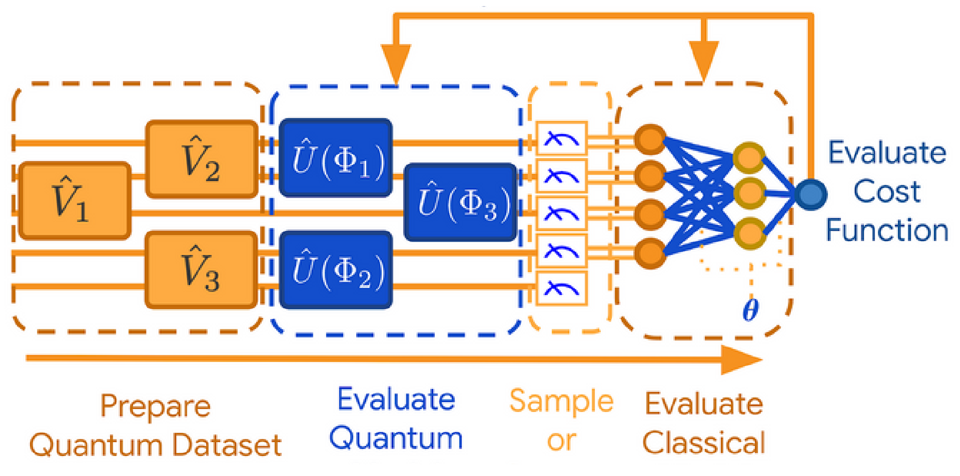
BUILT BY
- UTKARSH KUMAR AWASTHI
- MD SHADAB ALAM
- SRIJAN CHANDGOTHIA
- M.AKHIL TEJA

Introduction

In this Quantum-Classical Hybrid Machine Learning project for Image Classification, we leverage the strengths of both quantum and classical computing paradigms. By combining classical machine learning algorithms with quantum computing techniques, we aim to enhance the efficiency and accuracy of image classification tasks. This hybrid approach harnesses the unique capabilities of quantum computing, such as superposition and entanglement, to address complex feature interactions in images. Stay tuned for exciting advancements in bridging quantum and classical realms for improved image classification performance.

Methodology



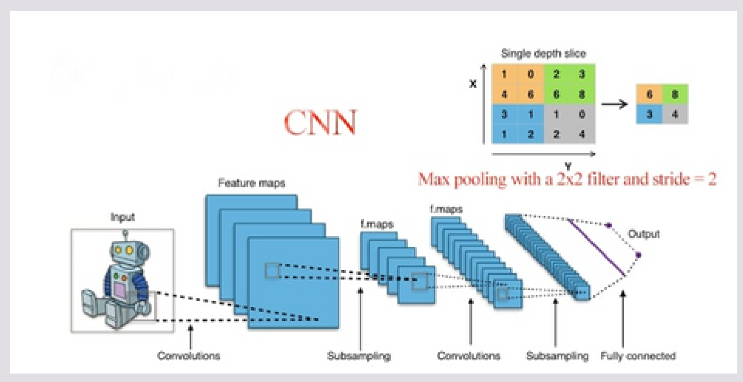


The IMPORT DATASET
• Tensorflow and Keras allow us to import and download the MNIST dataset(Modified National Institute of Standards and Technology) directly from their API.
• The MNIST database contains 60,000 training images and 10,000 testing images

PREPROCESS DATA
•Reshaping the array to 4-dims so that it can work with the Keras API(greyscale image).
• Need to normalize our data as it is always required in neural network models, by dividing the RGB codes to 255.

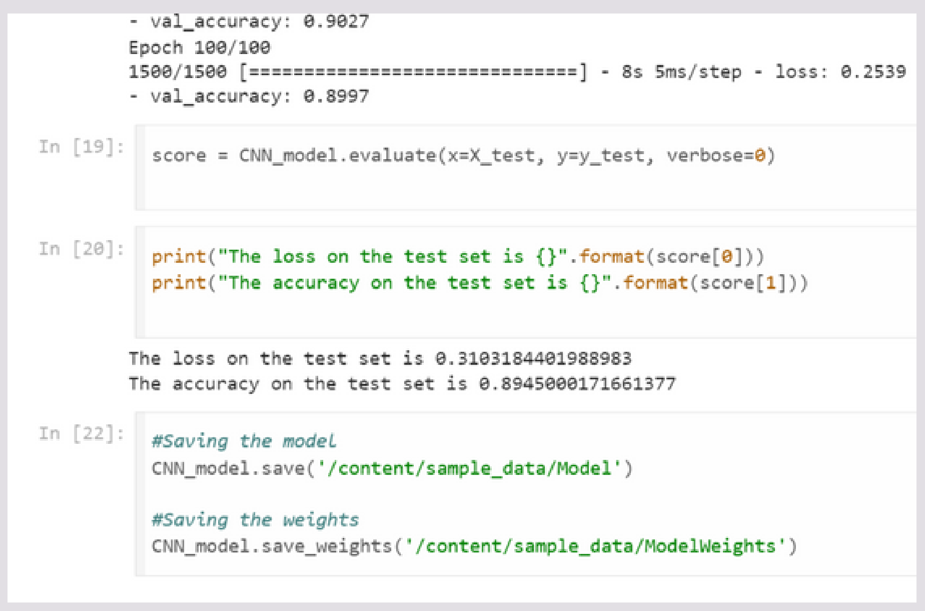
BUILDING THE CONVOLUTIONAL NEURAL NETWORK
•"Sequential model" allows you to build a model layer by layer.model.add(Conv2D(28, kernel_size=(3,3),input_shape=input_shape))#28 number of layers.
• "Max pooling" is a pooling operation that selects the maximum element from the region of the feature map covered by the filter.
• "Flattening" involves transforming the entire pooled feature map matrix into a single column which is then fed to the neural network for processing.
• "DENSE LAYER" , A linear operation in which every input is connected to every output by a weight . It connects neurons in one layer to neurons in another layer. It is used to classify images between different category by training.
• "DROPOUT" : A Simple Way to Prevent Neural Networks from Overfitting.• "SOFTMAX LAYER" is the last layer of CNN.

Results and Analysis



COMPILING AND TRAINING THE MODEL
Compiling the model takes three parameters: Optimizer, Loss and Metrics.
• Optimizer : controls the learning rate. We will be using ‘adam’ as our optmizer. Adam is generally a good optimizer to use for many cases. The adamoptimizer adjusts the learning rate throughout training.
• Loss: that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation.We use fit method then training starts.

EVALUATING THE MODEL
With a simple evaluate function to know the accuracy.
this is what we are doing in the cnn model u have to find a picture which shows the architecture of this model (mainly the preprocessing part)



```
- val_accuracy: 0.9027
Epoch 100/100
1500/1500 [==============================] - 8s 5ms/step - loss: 0.2539
- val_accuracy: 0.8997

In [19]:  score = CNN_model.evaluate(x=X_test, y=y_test, verbose=0)

In [20]:  print("The loss on the test set is {}".format(score[0]))
          print("The accuracy on the test set is {}".format(score[1]))

The loss on the test set is 0.3103184401988983
The accuracy on the test set is 0.8945000171661377

In [22]:  #Saving the model
          CNN_model.save('/content/sample_data/Model')

          #Saving the weights
          CNN_model.save_weights('/content/sample_data/ModelWeights')
```

Conclusion

In conclusion, the Quantum-Classical Hybrid Machine Learning approach showcased promising results for image classification. The synergy of quantum and classical computing demonstrated improved efficiency and accuracy in handling complex tasks. Further research and optimization are essential to unlock the full potential of this hybrid model in real-world applications.