QWORLD

Project - 19

QML for Image Processing

Mentor:

Viratkumar Kothari | Co-founder and CTO, Xporium, India | virat.Kothari@gmail.com
Devang Gajjar | Chief Solution Architect, HealthTech India, India | devang.gajjar@gmail.com

Interns:

Anurag Kulkarni, Azza Fadhel, Pushkal Shukla, Rajatav Dutta, Rifatul Islam Himel, Riya Malani, Vishal Kumar

Research Problem

Image Classification in computer vision in machine learning is a very famous use case in classical computers. This may be binary or multi-class classification. In binary classification, images are classified in two categories, whereas in multi-class classification, images are classified in more than two categories. In a classical computer, we generally use the Central Processing Unit (CPU) or Graphical Processing Unit (GPU) to solve this problem. The GPU processes data faster compared to the CPU because GPU is better at image processing. Still, processing the data more quickly is always a problem in classical computing.

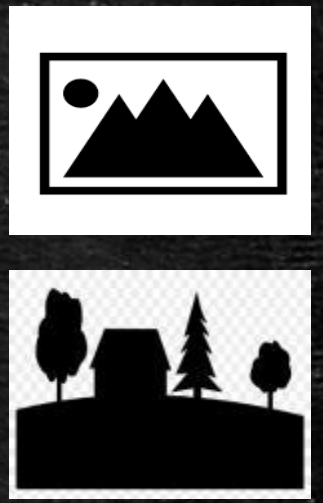
We want to solve the same image classification problem using quantum computing, assuming it may process the data faster than the GPU. Quantum Machine Learning, also known as QML, integrates quantum algorithms in Machine Learning programs. The QML refers to analysing classical data executed on a quantum computer, i.e. quantum-enhanced machine learning. It is assumed that the specialized algorithm may give better and quicker results on Quantum computers. A famous approach to this is to solve the problem using the Hybrid method, which involves both classical and quantum processing, where computationally difficult things are sent to a quantum device.

Workflow: Image classification using Classical Machine Learning

Dataset

Library

Processor



Classification

Workflow: Image classification using Quantum Machine Learning

Dataset

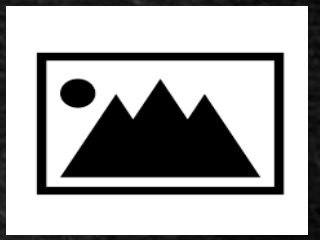
Qiskit

Cirq

Pennylane

Library

Processor



Classification

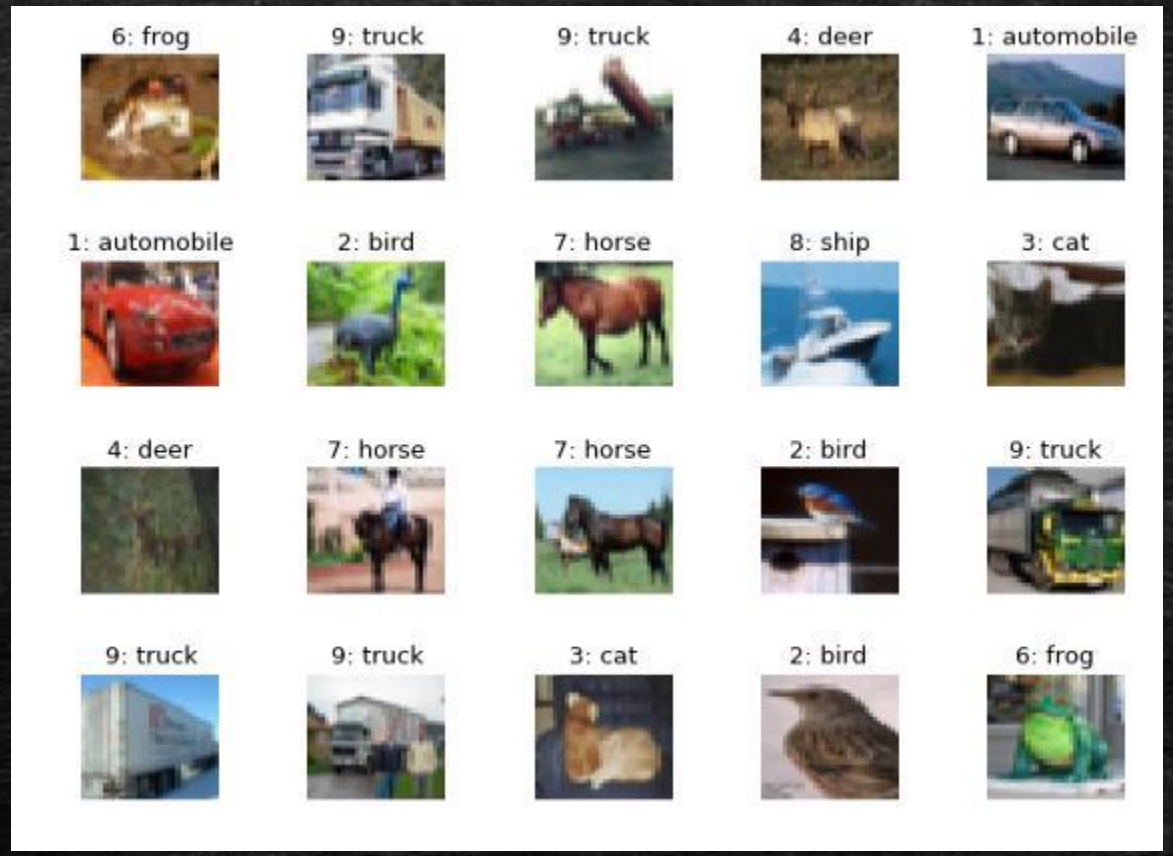
Dataset, Approach and Team

We have digital images, which are part of the renowned digital archive. It is envisaged to classify these images using Machine Learning or Quantum Machine Learning. Unfortunately, the archival images may be subject to copyright, so we cannot directly use them in the public domain. But the project can begin using open data available image classification problems at Kaggle, e.g., CIFAR, MNIST, Animal images or Chest X-ray images etc. Then the model obtained using open data may be implemented for the digital archival images. Of course, this may reduce the accuracy, but the model can be once again tuned for the archival images to get better accuracy. The problem may be solved using various frameworks, including Qiskit, TensorFlow, and Pennylane etc.

Dataset, Approach and Team

Sample images

Actual Images



CIFAR-10 images

Team 1: Qiskit

1) Pushkal Shukla
2) Rifatul Islam Himel
3) Vishal Kumar

Team 2: TensorFlow/Cirq

1) Anurag Kulkarni
2) Rajatav Dutta

Team 3: PennyLane

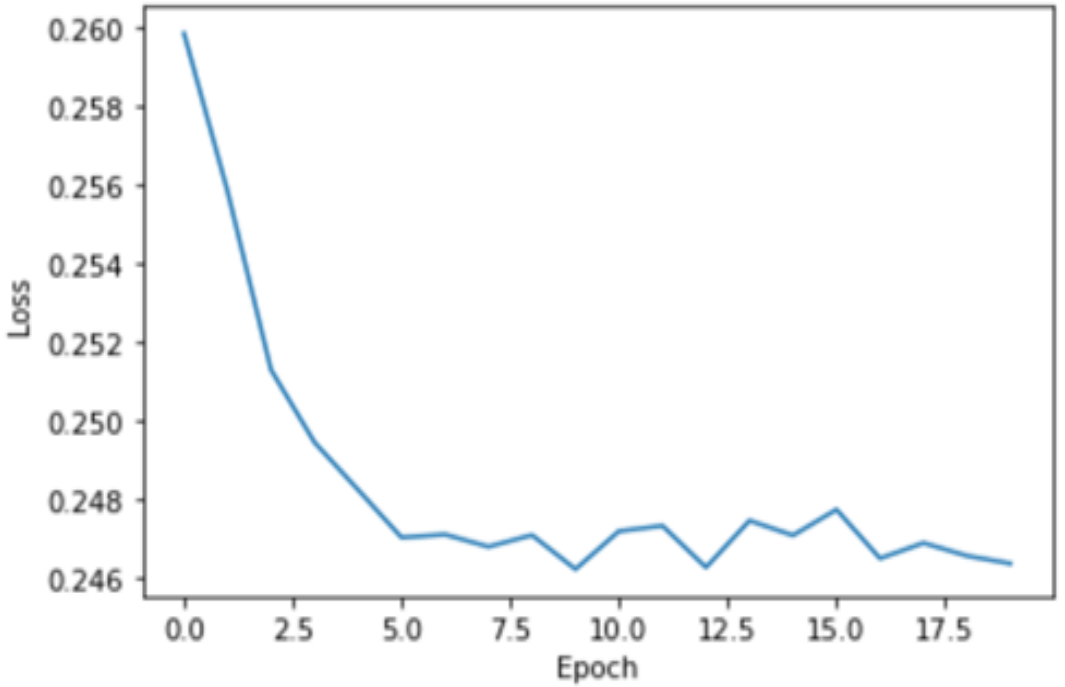
1) Azza Fadhel
2) Riya Malani


Qiskit
TensorFlow / Cirq
PennyLane
Transfer Learning

Approach 1 - Qiskit

Variational Quantum Classifier:

1. Data Encoding Circuit
2. Parameterized Circuit
3. Cost Function
4. Gradient Descent Optimizer

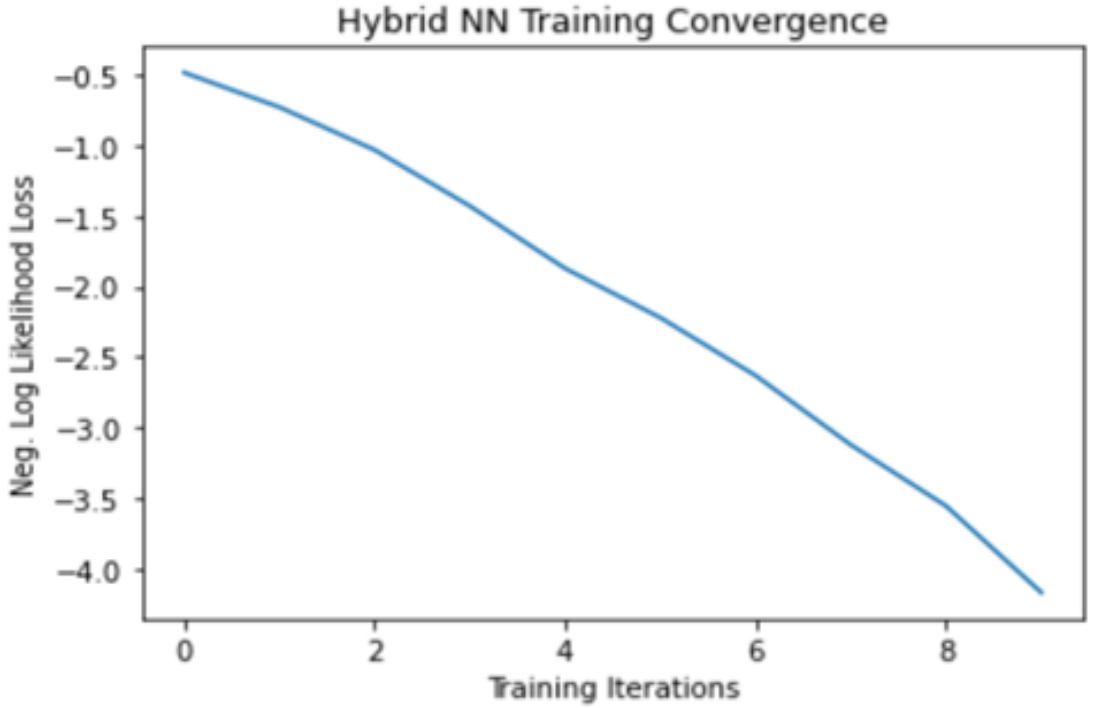


- VQC Accuracy-46%

Approach 1 - Qiskit

Hybrid Quantum-Classical machine learning architectures:

1. Quantum Circuit
2. Circuit QNN to PyTorch
3. Cost Function and Optimization



- Hybrid QNN Accuracy- 83.4%

Approach 1 - Qiskit

Comparative Analysis and Learnings:

	Variational Quantum Classifier	Hybrid QNN
Image Preprocessing	Drastic dimension reduction	Slight dimension reduction
Number of Qubits Required	More (Depends on dimension of the feature)	Less(Just need to link OpflowQNN to PyTorch)
Encoding and Operation	Complicated and Manual approach	Straightforward approach
Processing Time	More	Less
Accuracy	Less	More
Model Performance Stability	Much Stable	Unstable


Qiskit
TensorFlow / Cirq
PennyLane
Transfer Learning

Approach 2 – TensorFlow/Cirq

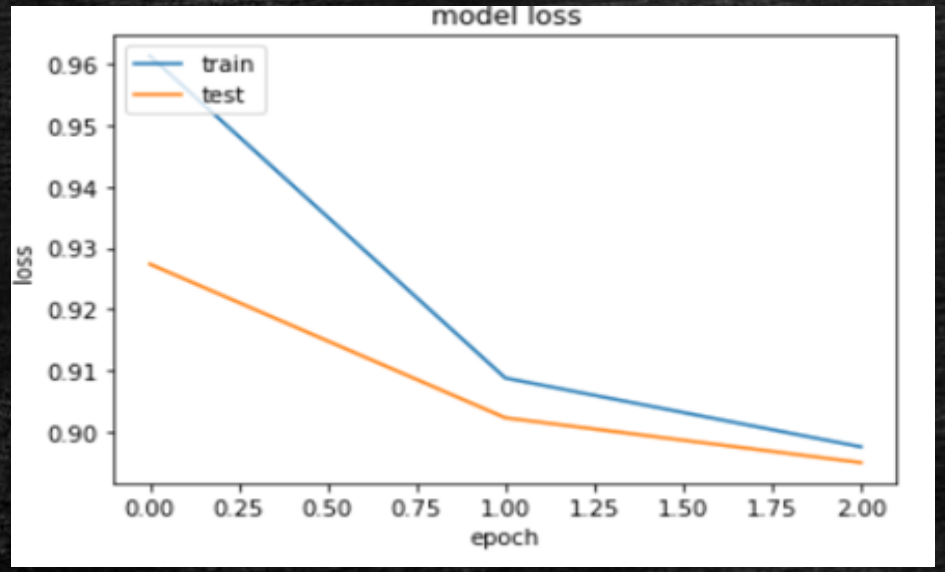
- TensorFlow Quantum is a quantum machine learning library for rapid prototyping of hybrid quantum-classical machine learning model It integrates quantum computing algorithms and logic designed in Cirq
- Cirq is an open source Python Library framework for programming quantum computers
- During pre processing stage 2 Classes of image (Dog and Aeroplane) were filtered from entire CIFAR10 data set
- Then the features were converted into gray scale and downscaled to 4X4 grid
- VQC algorithms was implemented

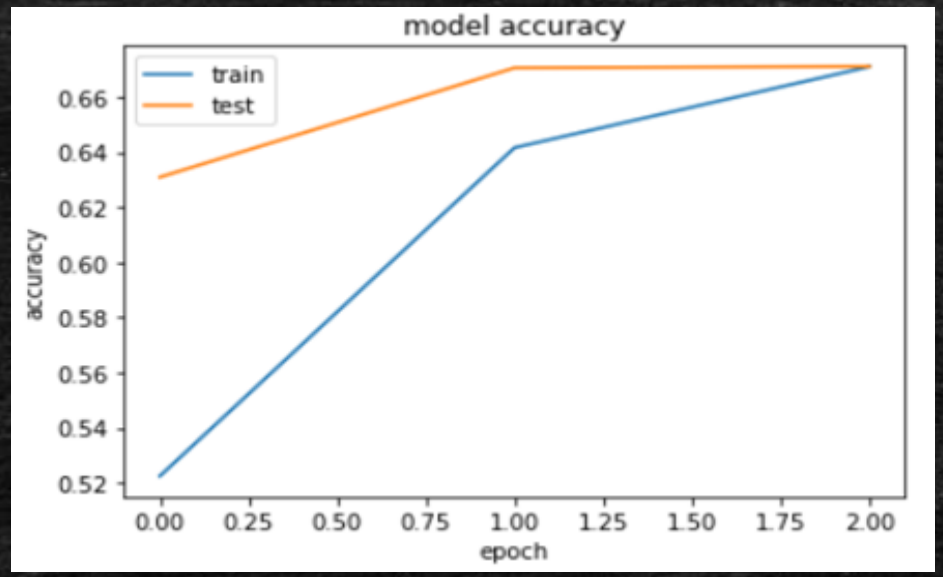
Approach 2 – TensorFlow/Cirq

Variational Quantum Classifier:

- Data Encoding was done by 1 layer
- Quantum model was created by 16 qubit and 1 layers

Result:

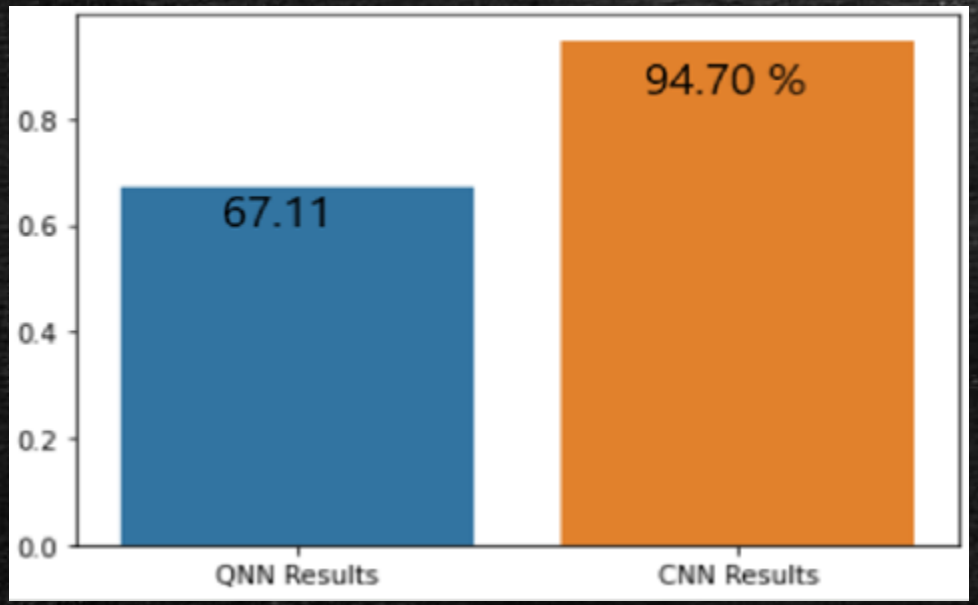





```
Accuracy : 67.11309552192688 %
```

Approach 2 – TensorFlow/Cirq

Comparison of results: QNN Vs. CNN




Qiskit
TensorFlow / Cirq
PennyLane
Transfer Learning

Approach 3 - PennyLane

- It's a cross-platform python library for differentiable programming of quantum computers.
- It trains a quantum computer the same way as a neural network.
- In comparison with Qiskit and TensorFlow approach, PennyLane can perform:
  - Multi-class classification (not only binary classification)
  - Better visualization of the images after encoding and processing.
- Three algorithms are implemented :
  - Quanvolutional neural network algorithm
  - Ensemble classification algorithm
  - Transfer learning algorithm

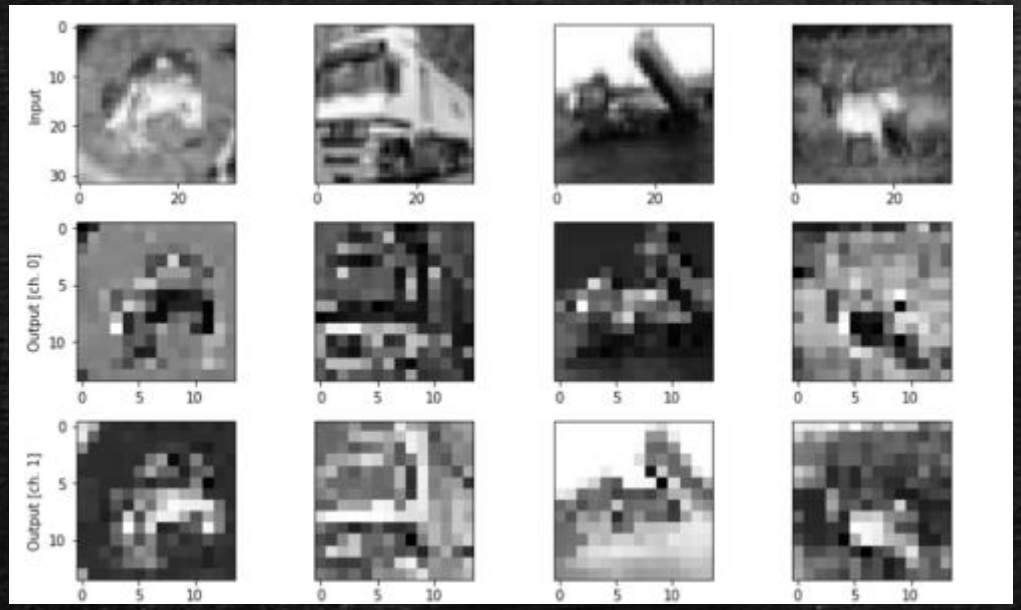
Approach 3 - PennyLane

Quanvolutional Neural Networks Approach

- Reference notebook: https://pennylane.ai/qml/demos/tutorial_quanvolution.html
- It's the quantum equivalent of the CNN model in classical computer.
- The model is based on the idea of a quanvolutional layer where, instead of processing the full input data with a global function, a local quanvolution is applied. What did we try to boost the result :
  - Changed the number of layers. Changed the size of train and test data
  - Changed the number of qubits used from 4 to 9 : It didn't boost result but increased the run time.
  - Changed the device: use of default device, Cirq, Forest and Qiskit device.
  - Tried a lot of Optimizers. The best one was the "adagrad" optimizer
  - Tried a lot of Loss Function. The best one was "sparse_categorical_crossentropy".

Approach 3 - PennyLane

Results: Overfitting




```
Epoch 21/30
13/13 - 0s - loss: 1.7516 - accuracy: 0.4400 - val_loss: 2.4597 - val_accuracy: 0.1333
Epoch 22/30
13/13 - 0s - loss: 1.7357 - accuracy: 0.4200 - val_loss: 2.4655 - val_accuracy: 0.1333
Epoch 23/30
13/13 - 0s - loss: 1.7209 - accuracy: 0.4400 - val_loss: 2.4686 - val_accuracy: 0.1333
Epoch 24/30
13/13 - 0s - loss: 1.7048 - accuracy: 0.4600 - val_loss: 2.4752 - val_accuracy: 0.1333
Epoch 25/30
13/13 - 0s - loss: 1.6896 - accuracy: 0.4600 - val_loss: 2.4816 - val_accuracy: 0.1333
Epoch 26/30
13/13 - 0s - loss: 1.6774 - accuracy: 0.4600 - val_loss: 2.4881 - val_accuracy: 0.1333
Epoch 27/30
13/13 - 0s - loss: 1.6635 - accuracy: 0.4600 - val_loss: 2.4904 - val_accuracy: 0.1000
Epoch 28/30
13/13 - 0s - loss: 1.6519 - accuracy: 0.4600 - val_loss: 2.4937 - val_accuracy: 0.1000
Epoch 29/30
13/13 - 0s - loss: 1.6380 - accuracy: 0.4800 - val_loss: 2.4960 - val_accuracy: 0.1000
Epoch 30/30
13/13 - 0s - loss: 1.6263 - accuracy: 0.5000 - val_loss: 2.4992 - val_accuracy: 0.1000
```

Approach 3 - PennyLane

Ensemble classification approach

- Reference notebook: https://pennylane.ai/qml/demos/tutorial_ensemble_multi_qpu.html
- This method combines two QPU in parallel to help solve ML classification problem.
- What did we try to boost the result :
  - Run the algorithm with only two classes.
  - Changed the size of train and test data
  - Run the algorithm with different devices: Build in device, Qiskit devices, Cirq devices and Forest devices.
  - Changed the number of layers.

Result: accuracy is always <0.2.

Approach 3 - PennyLane

- Device 0 : forest.numpy_wavefunction
- Device 1 : qiskit.basicaer

- Train result :

```
Training accuracy (ensemble): 0.16
Training accuracy (QPU0):  0.16
Training accuracy (QPU1):  0.14
```

- Test result:

```
Test accuracy (ensemble): 0.1
Test accuracy (QPU0):  0.13333333333333333
Test accuracy (QPU1):  0.03333333333333333
```


Qiskit
TensorFlow / Cirq
PennyLane
Transfer Learning

Approach 4 Additional: Quantum Transfer Learning

Reference notebook : https://pennylane.ai/qml/demos/tutorial_quantum_transfer_learning.html

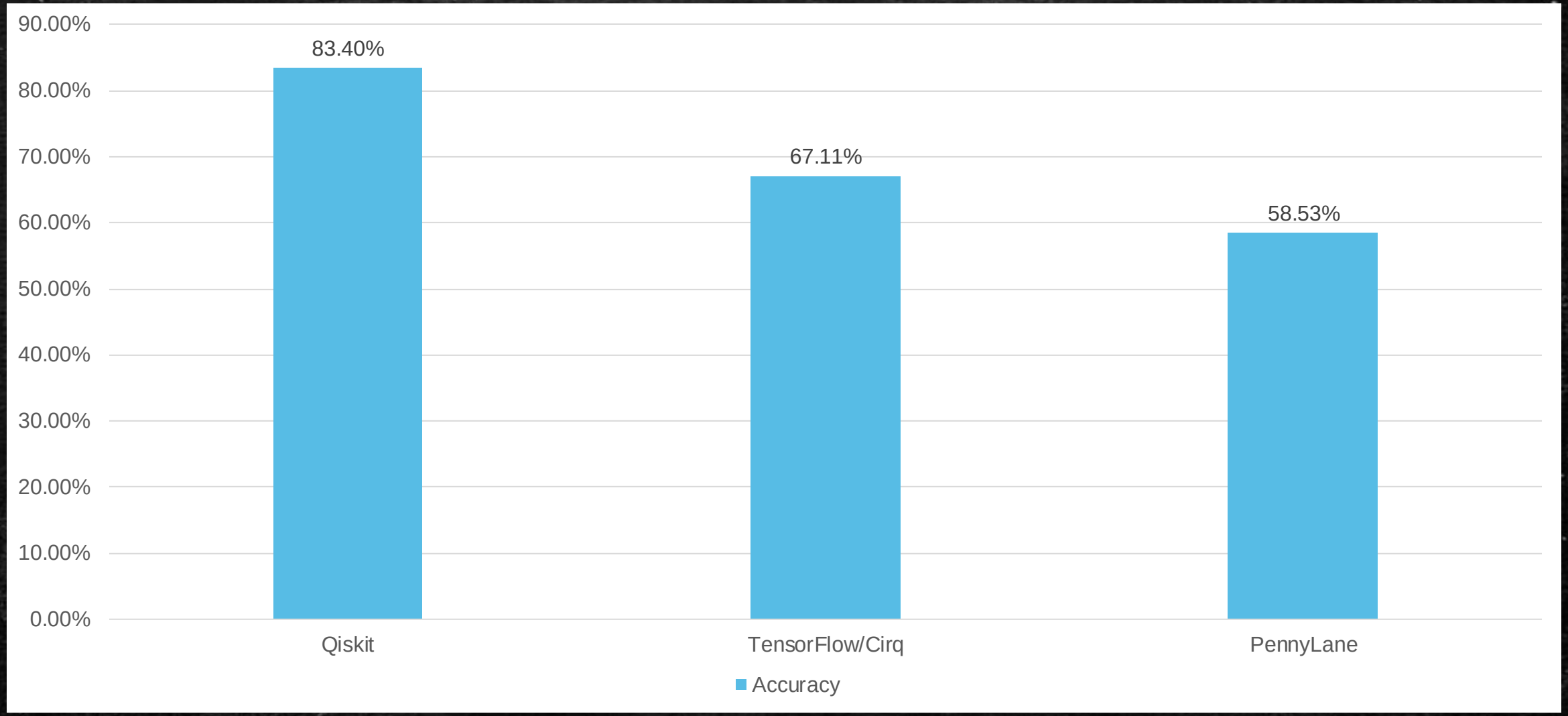
- It's the quantum equivalent of the transfer learning model in classical computer.
- It's a hybrid quantum-classical machine learning model.
- It's a binary classification model based on Transfer learning.
- Number of qubits = 4 qubits.
- Device used : default.qubit device.
- Train data size : 150 for each class
- Test data size : 100 for each class

Approach 4 Additional: Quantum Transfer Learning

Results: Overfitting


```
Phase: train Epoch: 1/1 Loss: 0.6903 Acc: 0.5700
Phase: validation   Epoch: 1/1 Loss: 0.6734 Acc: 0.5853
Training completed in 3m 10s
Best test loss: 0.6734 | Best test accuracy: 0.5853
```

Conclusion


90.00%
80.00%
70.00%
60.00%
50.00%
40.00%
30.00%
20.00%
10.00%
0.00%
83.40%
67.11%
58.53%
Qiskit
TensorFlow/Cirq
PennyLane
Accuracy

Way Forward

- Applying transfer learning for 10 class classification using PennyLane approach
- Redesigning the QNNs (Qiskit+PyTorch) approach
- Implementation of Higher Order Embedding encoding technique using Qiskit
- Better Data Pre-processing before classification
- Try with other complex image dataset like the Chest X-Ray images etc.

- Finally using the best model with actual archival images

Thank you!