QPU Tasks, Summer school

Testing TQ quantum layers with IBM QPU

Author:

Stepanov Timofei

Evaluating the performance of different TQ quantum layers for an IBM Quantum Processing Unit (QPU) and establishing best practice.

1 Introduction

This project is aimed at benchmarking hybrid quantum neural networks with different quantum layer architectures using the Hidden Manifold. The quantum layers tested are: parallel quantum network (PQN), quntum depth-infused network (QDI) and parallel exponential network (EFQ)

2 Dataset description

The Hidden Manifold problem is a classification task mimicing the idea, that the data is labeled on a manifold that is embedded in a space of different dimentionality.

Input vectors of m dimensions are sampled from a normal distribution in a low-dimensional space and labeled by a single-layer neural network initialised at random. The inputs are then projected to the final d-dimensional space. There are two different dataset collections in this task, one only varying the dimension of the input vectors while keeping the manifold dimension constant, while the other, vice versa, keeps the d constant and varies the dimensionality of the manifold.

In this project we decided to use the first collection with $d = 5$ and $m = 6$ while larger vector dimensions lead to use of larger models, which significantly increases training time.

3 Hybrid parallel quantum network

3.1 General information

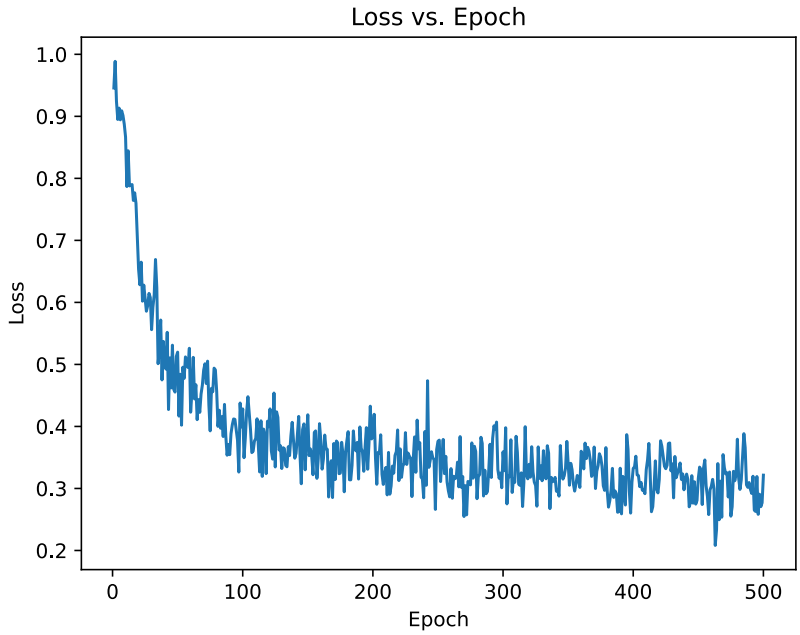
We propose the following architecture for Hybrid PQN networks: the input data go into a fully connected network, which then connects to the PQN Quantum circuit (or a few of them). The PQN quantum circuit consisit of X rotation gates for input feature embeddings and repetitions of Z, Y, Z weighed rotation gates with CNOT gates for entanglement. After that each qubit is measured in Z axis and the result is forwaded to another FCNN.

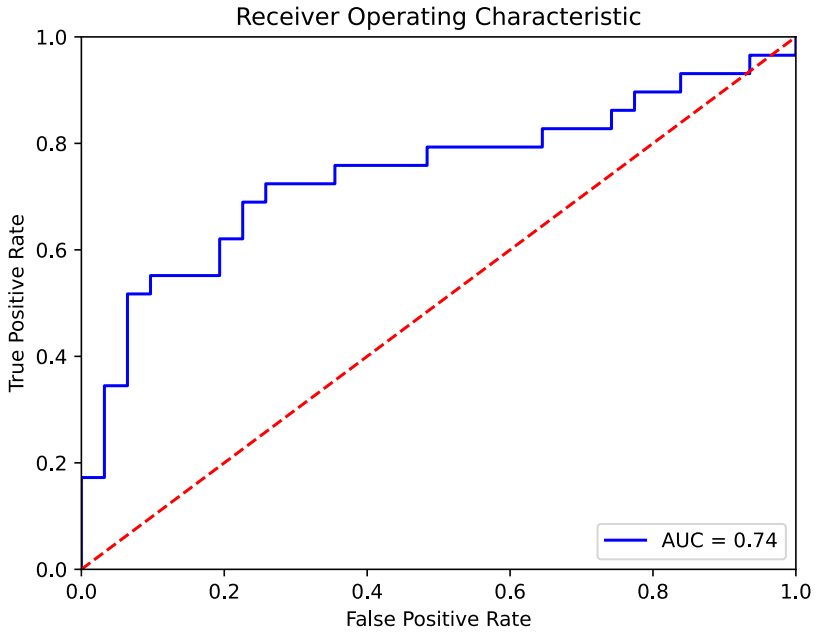
3.2 Architecture 1

The input FCNN consists of fully connected layers with sizes: 48-24-12 (in that particular order). The second dense layer's architecture is 64-32-16-8-4. In both networks we use batch normalization and dropout.

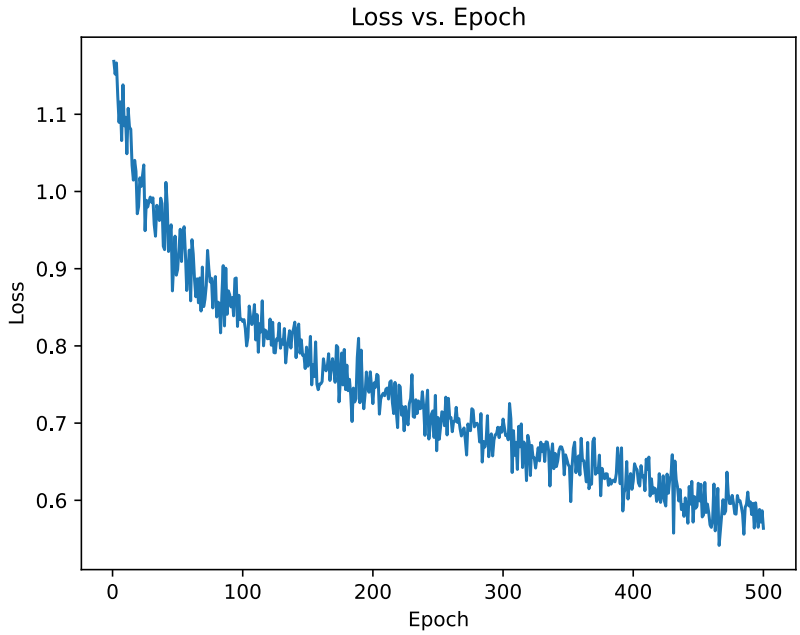
The quantum layer has 5 qubits and two repetitions of weighed gates, totalling 5544 trainable parameters.

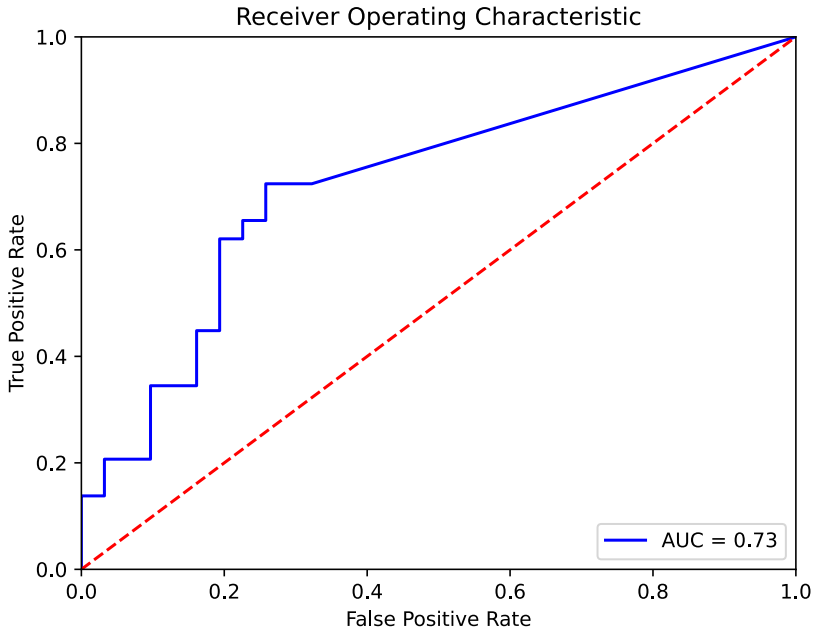
3.2.1. Noiseless case





3.2.2. Noisy case





4 Quantum depth-infused network

4.1 General information

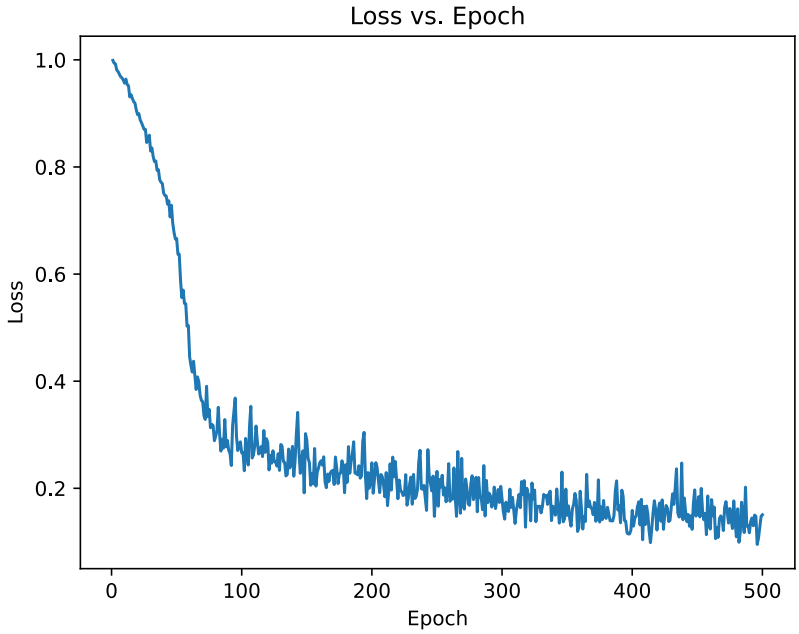
Hybrid QDI network consists of single FCNN and QDI quantum layer. The inputs go into the dense layers and then the results are forwaded to the quantum layer using rotation embedding. The QDI quantum circuit has the following architecture: first R_x weighed gates with CNOT gates are repeated i times. Then embedding R_z, weighed R_x (repeated n times)

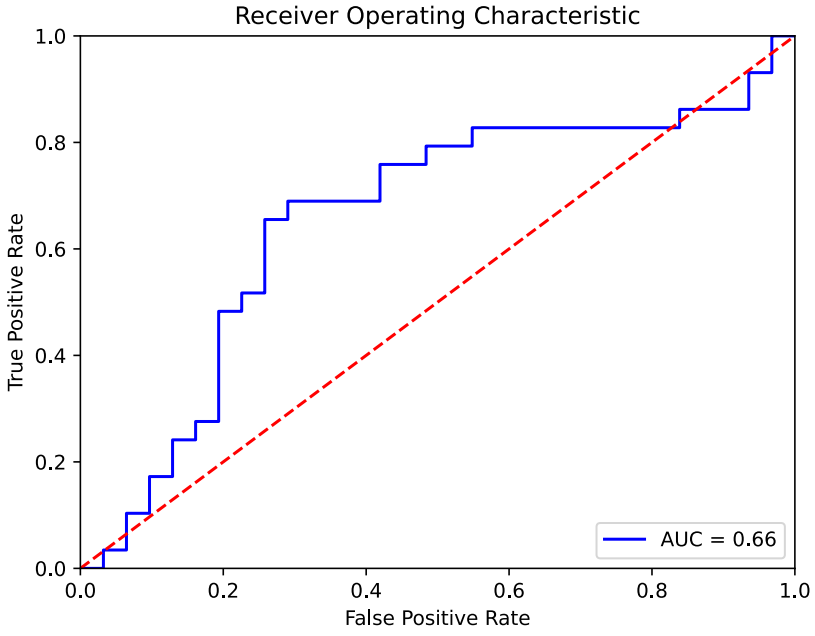
and CNOT gates are repeated j times. Then the first qubit is measured in the Z axis and passed as an output.

4.2 Architecture 1

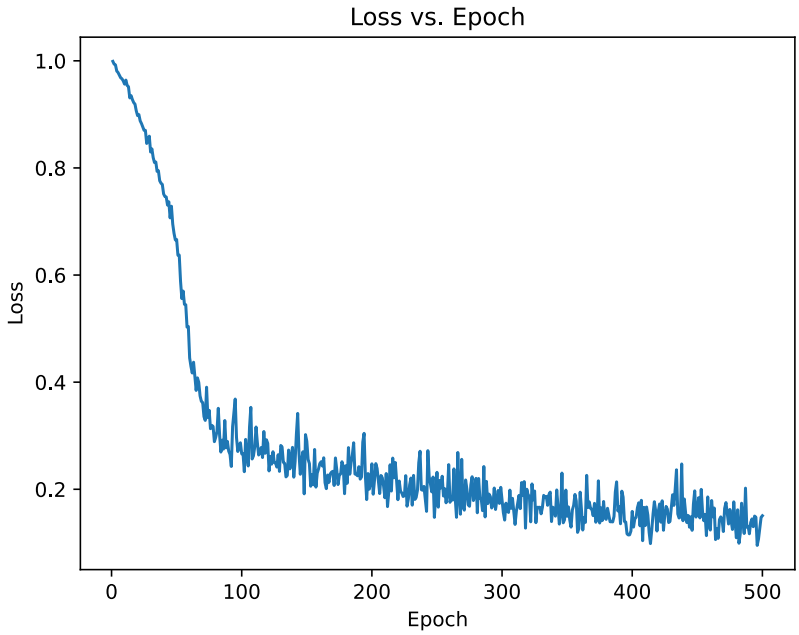
This architecture proposes FCNN with layer sizes 64-32-32-16-8. The QDI layer parameters are as following: $i = 1, n = 2, j = 2$. Total amount of trainable parameters -

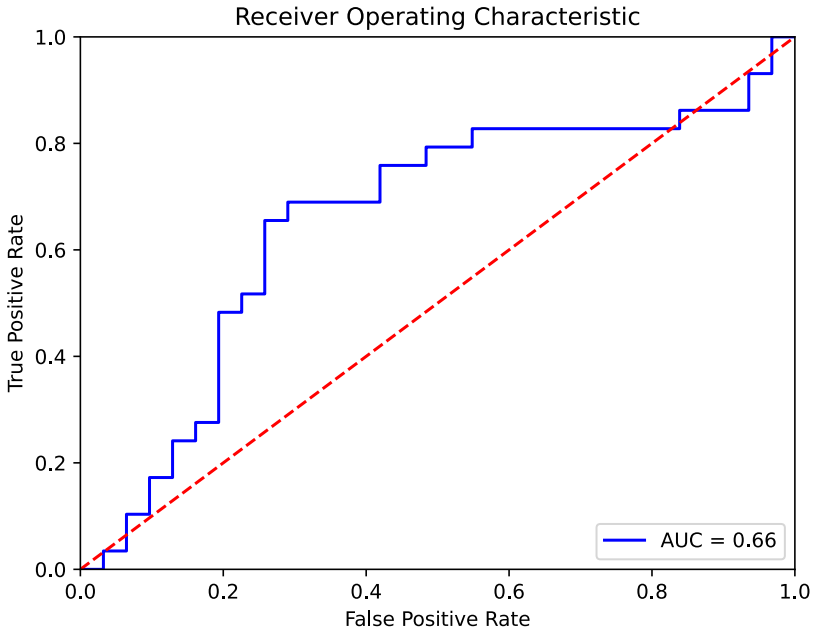
4.2.1. Noiseless case





4.2.2. Noisy case



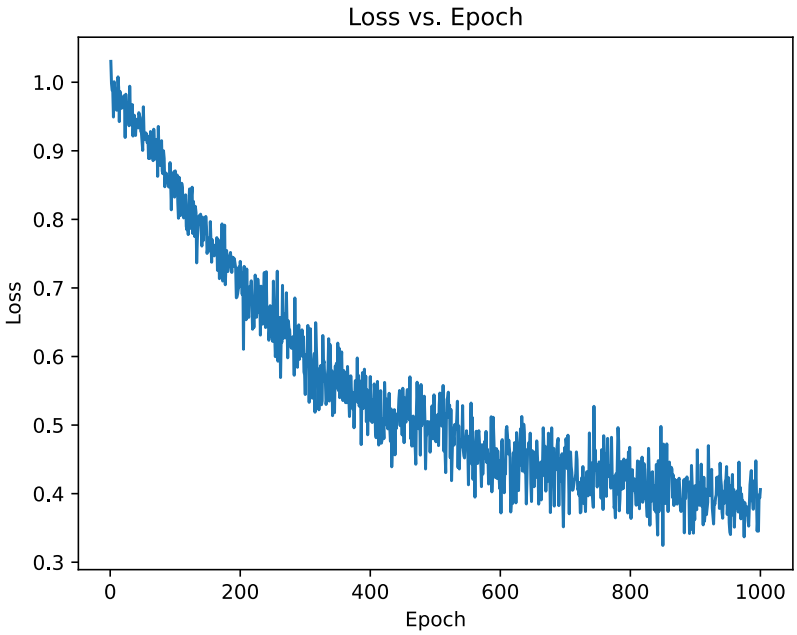


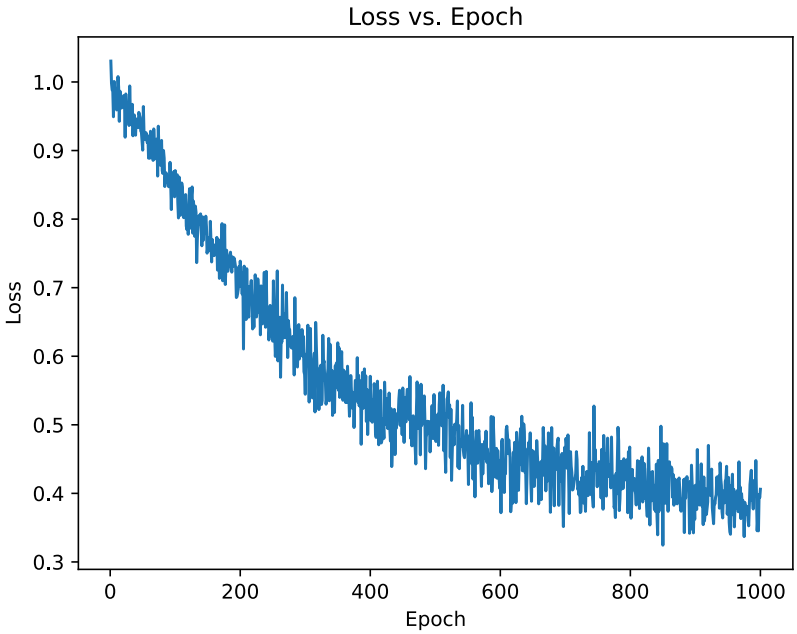
5 Parallel exponential network

5.1 General information

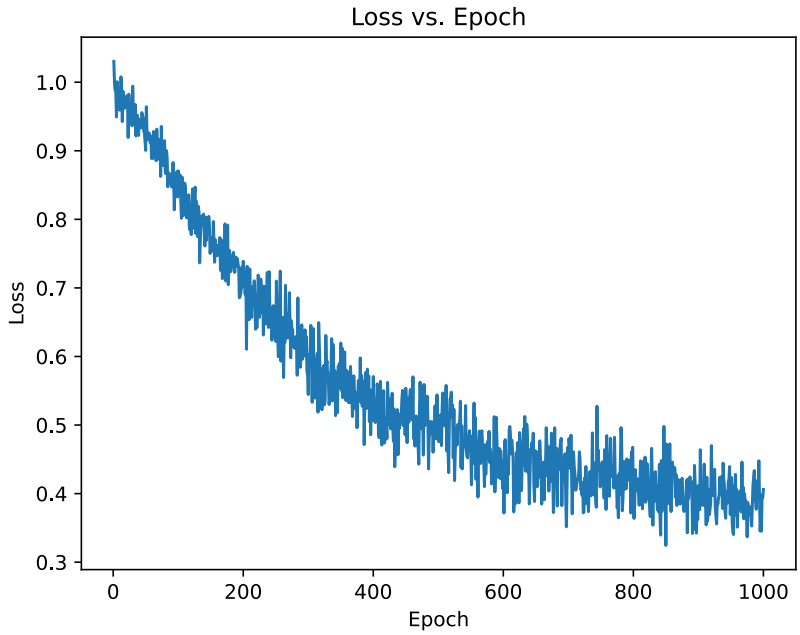
5.2 Architecture 1

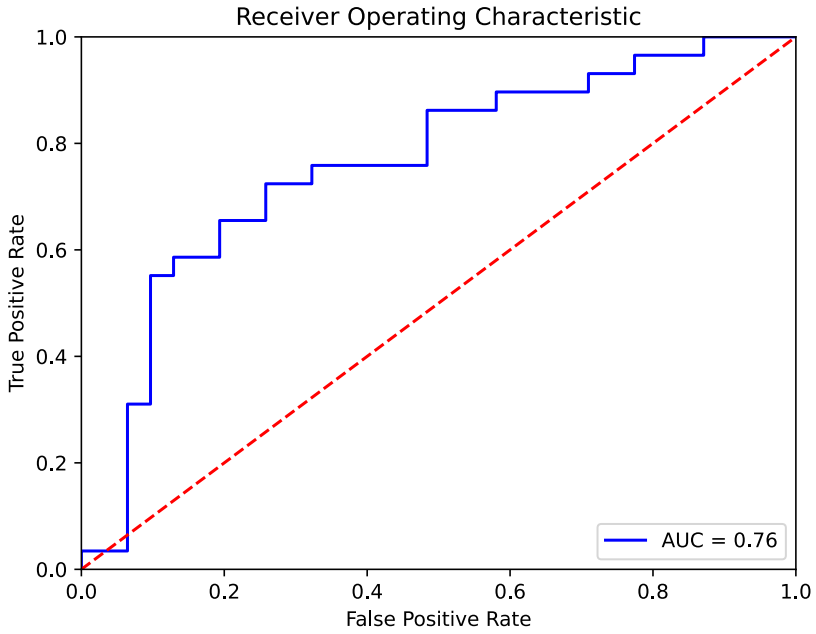
5.2.1. Noiseless case





5.2.2. Noisy case





6 Conclusion

6.1 HPQN vs. FCNN

We have successfully trained and verified 3 different HPQN architectures. The results show that hybrid quantum neural networks performance is on the same level or even better than of the fully connected architectures with the same amount of parameters.

6.2 Noise influence

As we can see in some cases noise can slow down the training process and lead to bigger loss values in the same amount of epochs, however in case of small models and long training (500 epochs took 16 hours) it's influence is not crucial.

It should be noted that when emulating noise we did not enable readout errors for faster computing times.

6.3 Improvement points

To improve this project following steps can be made:

- Benchmark more different architectures and hidden manifold dataset collections for a wider range of data
- Verify the results with a real device
- Enable full-scale gpu support for Qiskit Algorithms gradient framework