

Quantum Machine Learning Project

In this project you're supposed to develop a quantum GAN-based method for detecting anomalies in the time series. You can work in teams of up to three students.

Step 1: Data Sets

For this project you need to choose at least **two** datasets from [UCR Time Series Classification Archive](#). Please, use the following data sets:

- 28, 54, 99, 118, 138, 176

You can download the data [here](#). More information about the benchmark could be found [here](#), though the main dataset description is in [UCR_AnomalyDataSets.pptx](#) and from there you'll find out:

- how to split data into train and test
- that train data has no anomalies, test data has only a single anomaly

Here is the description of an example data set from NASA:

UCR_Anomaly_TkeepFirstMARS_3500_5365_5380.txt

This is a real dataset from NASA spacecraft, that appeared in a KDD 2018 paper.

We joined the train and test sets.

The original dataset had two anomalies.

We carefully edited the data, into two datasets, each with *one* anomaly.

This one contains "**unique blip**".

There is also a small amount of level change in this datasets.

However, note that it occurs in both the train and test sets, so it is not the anomaly.

This figure is from the original test split



Please read more about processing of raw time series data [here](#). We recommend using **window_size=128**.

Step 2: QML algorithms

You can apply any neural network architecture (both hybrid classical-quantum and fully-quantum), though it must include a generative adversarial part, i.e. your solution should have a generator and a discriminator.

Recall all the methods discussed in the lecture on Quantum Machine Learning 3. In particular, reconsider some of the code on GANs:

- https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html (classical)
- https://qiskit.org/ecosystem/machine-learning/tutorials/04_torch_qgan.html (quantum)

You can find more recent quantum GANs [here](#).

Step 3: Evaluation

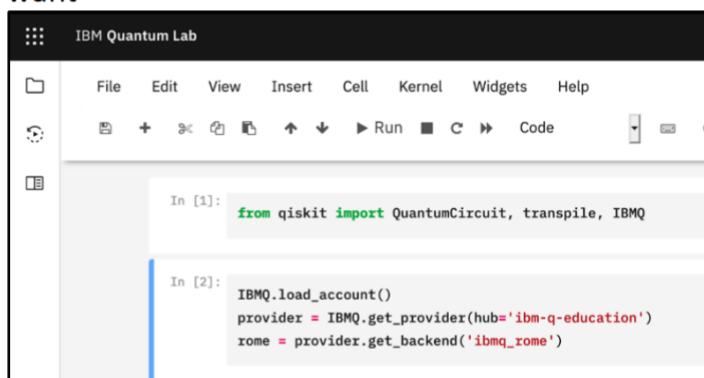
Use the metric of **adjusted recall@k** proposed in [1]. $k=1$ in case of a single anomaly detection. Visualize the results.

Have a look at the description of precision and recall @k for recommender systems. You can adapt these metrics also for anomaly detection:

https://insidelearningmachines.com/precisionk_and_recallk/

Step 4: Execute your experiments also on the quantum computer.

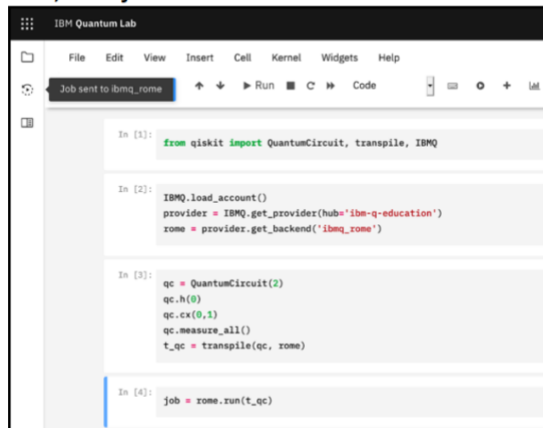
1. Load your account and use the 'ibm-q-education' provider to get the backend you want



```
In [1]: from qiskit import QuantumCircuit, transpile, IBMQ

In [2]: IBMQ.load_account()
        provider = IBMQ.get_provider(hub='ibm-q-education')
        rome = provider.get_backend('ibmq_rome')
```

2. Create and transpile your jobs, then run them on the backend. If using IBM Quantum Lab, the job will show on the left after a few seconds.



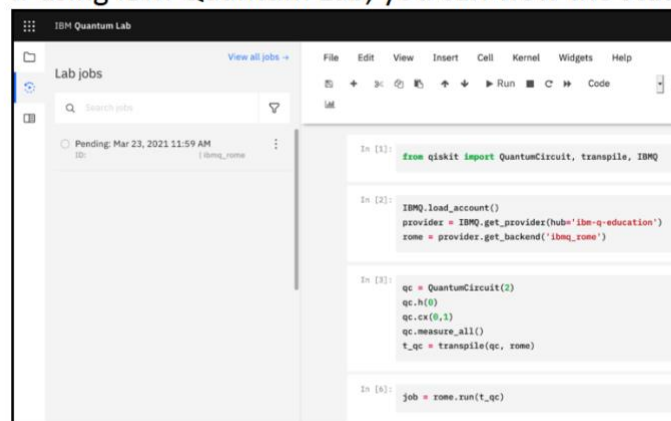
```
In [1]: from qiskit import QuantumCircuit, transpile, IBMQ

In [2]: IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q-education')
rome = provider.get_backend('ibmq_rome')

In [3]: qc = QuantumCircuit(2)
qc.h(0)
qc.cx(0,1)
qc.measure_all()
t_qc = transpile(qc, rome)

In [4]: job = rome.run(t_qc)
```

3. If using IBM Quantum Lab, you can view the status of the job in the side panel



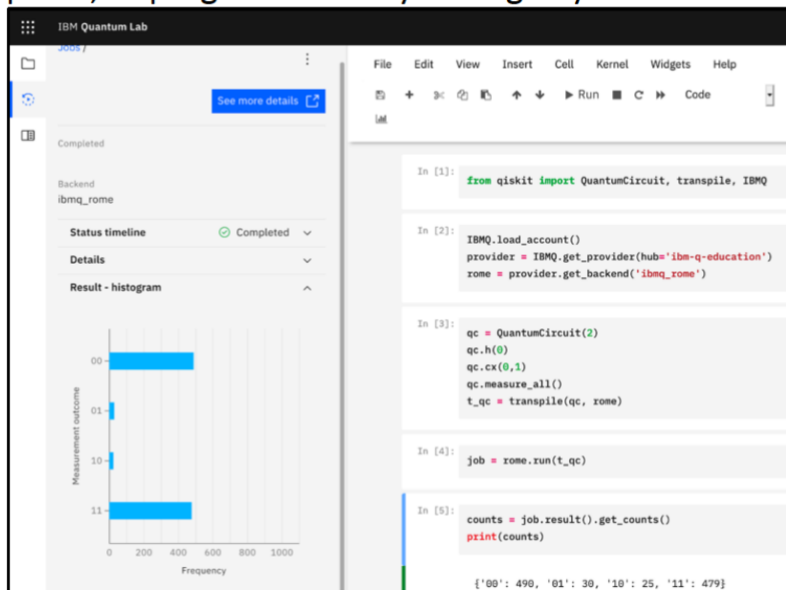
```
In [1]: from qiskit import QuantumCircuit, transpile, IBMQ

In [2]: IBMQ.load_account()
provider = IBMQ.get_provider(hub='ibm-q-education')
rome = provider.get_backend('ibmq_rome')

In [3]: qc = QuantumCircuit(2)
qc.h(0)
qc.cx(0,1)
qc.measure_all()
t_qc = transpile(qc, rome)

In [4]: job = rome.run(t_qc)
```

4. Once the job has completed, you can access the results either through the side panel, or programmatically through Python.





Step 5: Compare to baseline

Compare the results from your method against classical GAN-based algorithm “TadGAN: Time Series Anomaly Detection Using Generative Adversarial Networks”. The PyTorch implementation can be found [here](#).

Step 6: Summarize your results

Write up your results in a max. **10-page power point presentation**. Provide screen shots of your code and comment it. Summarize your main findings and address the following questions:

- Describe the quantum circuit of your chosen architecture
- How does your approach work for the different data sets?
- Were you able to get an improvement compared to the baseline method?
- Why GAN architecture makes sense for anomaly detection, and for time series for instance?
- Which lessons did you learn?
- What would you do differently if you could start all over again?

Send your **presentation and your code** (as one zip-file or as a link to GitHub account) to stog@zhaw.ch and furu@zhaw.ch by **January 14, 2024**.