QML-HEP GSoC 2025 Tasks

Instructions

Below are 8 evaluation tasks that you will perform to demonstrate the skills needed for the projects. **All students, regardless of project**, will have to complete **Tasks I, II, and III.**

If you're interested in *"Implementation of Quantum Generative Adversarial Networks to Perform High Energy Physics Analysis at the LHC"*, you will have to complete **Task IV.**

If you're interested in *"Quantum Graph Neural Networks for High Energy Physics Analysis at the LHC"*, please complete **Task V**.

If you're interested in *"Learning quantum representations of classical high energy physics data with contrastive learning"*, you will have to complete **Task VI**.

If you're interested in *"Equivariant quantum neural networks for High Energy Physics Analysis at the LHC"*, you will have to complete **Task VII**.

If you're interested in *"Quantum transformer for High Energy Physics Analysis at the LHC"*, you will have to complete **Task VIII**.

If you're interested in *"Quantum Kolmogorov-Arnold Networks for High Energy Physics Analysis at the LHC"*, you will have to complete **Task IX**.

If you're interested in *"Quantum Foundation Model for High Energy Physics"*, you will have to complete **Task VI**.

If you're interested in *"Quantum Particle transformer for High Energy Physics Analysis at the LHC"*, you will have to complete **Task VIII**.

If you're interested in *"Q-MAML - Quantum Model-Agnostic Meta-Learning for Variational Quantum Algorithms for High Energy Physics Analysis at the LHC"*, you will have to complete **Task VI or Task XI**.

If you're interested in *"Quantum Diffusion Model for High Energy Physics"*, you will have to complete **Task X**.

Students interested in multiple projects should complete all the tasks relevant to the projects.

We *highly* recommend students to complete **all tasks**, regardless of your project of interest, to maximize the chances of being selected.

Submission Instructions

For the exercise, it's recommended to use a Python notebook. There are 2 ways to submit your work:

1. (Recommended) Upload the .ipynb file to Github and send us the link. *Make sure that it is a public repository.*

2. Alternatively, send us the .ipynb file and a PDF version of the .ipynb file with the corresponding cell outputs.

Test Submission Instructions: Please send us your CV and a link to all your completed work (github repo, Jupyter notebook + pdf of Jupyter notebook with output) to ml4-sci@cern.ch with Evaluation Test: QMLHEP in the title.

Task I: Quantum Computing Part

1) implement a simple quantum operation with Cirq or Pennylane
   a) With 5 qubits
   b) Apply Hadamard operation on every qubit
   c) Apply CNOT operation on (0, 1), (1,2), (2,3), (3,4)
   d) SWAP (0, 4)
   e) Rotate X with pi/2 on any qubit
   f) Plot the circuit

2) Implement a second circuit with a framework of your choice:
   a) Apply a Hadmard gate to the first qubit
   b) rotate the second qubit by pi/3 around X
   c) Apply Hadamard gate to the third and fourth qubit
   d) Perform a swap test between the states of the first and second qubit |q1 q2> and the third and fourth qubit |q3 q4>

Task II: Classical Graph Neural Network (GNN)

For Task II, you will use ParticleNet's data for Quark/Gluon jet classification available here with its corresponding description.

- Choose 2 Graph-based architectures of your choice to classify jets as being quarks or gluons. Provide a description on what considerations you have taken to project this point-cloud dataset to a set of interconnected nodes and edges.
- Discuss the resulting performance of the 2 chosen architectures.

Task III: Open Task

Please comment on quantum computing or quantum machine learning. You can also comment on one quantum algorithm or one quantum software you are familiar with. You can also suggest methods you think are good and you would like to work on. Please use your own understanding. Comments copied from the internet will not be considered.

Task IV: Quantum Generative Adversarial Network (QGAN)

You will explore how best to apply a quantum generative adversarial network (QGAN) to solve a High Energy Data analysis issue, more specifically, separating the signal events from the background events. You should use the Google Cirq and Tensorflow Quantum (TFQ) libraries for this task.

A set of input samples (simulated with Delphes) is provided in NumPy NPZ format [Download Input]. In the input file, there are only 100 samples for training and 100 samples for testing so it won't take much computing resources to accomplish this task. The signal events are labeled with 1 while the background events are labeled with 0.

Be sure to show that you understand how to fine tune your machine learning model to improve the performance. The performance can be evaluated with classification accuracy or Area Under ROC Curve (AUC).

Task V: Quantum Graph Neural Network (QGNN)

In task II you already worked with a classical GNN.

- Describe a possibility for a QGNN circuit, which takes advantage of the graph representation of the data

- Implement and draw the circuit.

Task VI: Quantum representation learning

In this task you should implement a simple representation learning scheme based on a contrastive loss:

- Load the MNIST dataset
- Write a function which takes an image and prepares a quantum state. This function should have trainable parameters which we want to learn in order to have good quantum representations
- Create a circuit with which takes two images and embeds both as quantum states with the function you wrote before. Afterwards the circuit should perform a SWAP test between the two states. In the end the measurement should give the fidelity of the quantum states.
- Train the circuit parameters with a contrastive loss: For two MNIST images in the same class the fidelity should be maximized, while for images of different classes the fidelity should be minimized.

Task VII: Equivariant quantum neural networks

In this task you are supposed to get started with equivariant quantum neural networks by implementing a Z_2 × Z_2 equivariant quantum neural network. Z_2 is a symmetry group an as an example we will generate a simple classical dataset which is respects the Z_2 x Z_2 symmetry.

This example is explained in the paper https://arxiv.org/abs/2205.06217 and additional background can be found in https://arxiv.org/abs/2210.08566.

- Generate a classification dataset with two classes and two features x_1 and x_2 which respects the Z_2 x Z_2 symmetry (this corresponds to mirroring along y=x). An example can be found in the first reference paper.
- Train a QNN to solve the classification problem
- Train an Z_2 x Z_2 equivariant QNN to solve the classification problem and compare the results.

Task VIII: Vision transformer/Quantum Vision Transformer

Implement a *classical* Vision transformer and apply it to MNIST. Show its performance on the test data. Comment on potential ideas to extend this classical vision transformer architecture to a quantum vision transformer and sketch out the architecture in detail.

Task IX: Kolmogorov-Arnold Network

Implement a *classical* Kolmogorov-Arnold Network using basis-splines or some other KAN architecture and apply it to MNIST. Show its performance on the test data. Comment on potential ideas to extend this classical KAN architecture to a quantum KAN and sketch out the architecture in detail.

Task X: Diffusion

Complete the specific task 2 from the DeepFalcon test. Comment on potential ideas to extend this classical diffusion architecture to a quantum diffusion and sketch out the architecture in detail.

Task XI:

In this task, you should implement a simple embedding task with few Linear Layers and Parameterized Quantum Circuit(PQC). We recommend using 2-3 Layers of Linear, 4-5 qubits.
Generate Normally Distributed Data: Sample input data from a normal distribution.
Designing a Neural Network (MLP): Use a neural network to estimate PQC parameters from normally distributed data.
Quantum State Preparation: Generate a quantum state using the estimated parameters.
Training: use MSE Loss