CPEN400Q W2023 Final Project: Replicating “Identifying overparameterization in Quantum Circuit Born Machines”

sonozaki@ubc.ca

sgonugun@ubc.ca

haria01@ubc.ca

oreade16@ubc.ca

A high-level summary of the key methods and results of paper

The paper explores overparameterization in Quantum Circuit Born Machines (QCBMs) by studying their trainability using gradient descent and the effect of circuit depth. Employing the Jensen-Shannon loss, they identify a critical depth (pc) where training efficiency improves, indicating overparameterization. This critical depth is compared to theoretical quantum bounds like Quantum Fisher Information and Dynamical Lie Algebra, serving as approximate bounds for overparameterization. The findings suggest overparameterized QCBMs can be efficiently trained, with (pc) depending on the target distribution (Uniform, Sparse, Bell, W, HEP), but further investigation into the causes and effects of overparameterization is needed.

An outline of the software you need to write (proposed architecture, experiment pipelines, etc.) (Ore)

The software will be written in Python and will use Pennylane to do so. In order to meet the demands mentioned in the paper on overparameterization and Quantum Circuit Born Machines, the software architectures and experimental pipelines must be able to accommodate both empirical experiments. Here's a proposed outline:

The suggested architecture
1. Quantum Simulation Environment: Put in place a simulator for quantum circuits that can execute QCBMs.
2. Framework for Parameter Optimization: Create a gradient-based optimization framework specifically for quantum circuits.
3. Experiment Management System: Establish a system for the administration, monitoring, and logging of experiments, together with their setups, outcomes, and analysis.
4. Tools for Data Analysis and Visualization

Experiment Pipelines
1. Preparation and basic Configuration
2. Training and Optimization
3. Overparameterization Analysis
4. Phase Transition Studies
5. Performance Assessment

A rough plan for how your group will be distributing the workload

Task	Assigned To

Setting up the GitHub repository	Shashi
Implementing the Quantum Simulation Environment	Akshat and Ore
Implementing the QCBM	So and Akshat
Testing models for robustness and generalization	Akshat
Preparing the training and testing datasets	So and Shashi
Implementing function to plot the data for Analysis	Shashi
Conducting training and optimization experiments	Ore
Writing the final report	The entire team

A risk assessment (discuss any missing information, or challenges with the framework)

Risks related to the replication of the research

- In terms of preparing the data samples, The HEP distribution was constructed from approximately 4 million samples drawn from the kinematic distributions from the Large Hadron Collider. This will not be possible to replicate.
- Potential run time of the code that we have to run might be very long and might cause troubles if we have to run multiple iterations of it. We are expecting to have a maximum number of code for each of the 5 target distributions for 5 qubit configurations which results in at most 25 configured QCBM training to be done.
- Additionally, the paper uses Hardware Efficient Ansatz where they were able to run 8 qubit with 510 layers which would be hard to run on classical computing devices.
- Code scalability across all configurations. As the number of qubits increases, there might be complications as to having to change the code quite a bit and it might take some time.

Use JAX for speeding up the computation
For the actual coding implementation, use black box abstraction to make parts of it configurable to run different code depending on the target and the number of qubits