Task -

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

QGNN Circuit 1

Reference Paper and Its Architecture

I have taken inspiration from the paper **"Financial Fraud Detection using Quantum Graph Neural Networks"** by Nouhaila Innan et al [1]. This research proposes using Quantum Graph Neural Networks (QGNNs) to detect financial fraud, leveraging graph-based data representations and quantum variational circuits (VQCs) to enhance classification accuracy.

The core idea behind the paper's approach is to represent financial transactions as graphs, where nodes correspond to features of transactions and edges represent relationships between them. The researchers use topological data analysis (TDA) to embed graph structures in a lower-dimensional space before encoding them into a quantum state. The encoding process employs angle encoding to transform node features into quantum states, which are then processed using variational quantum circuits (VQCs). These circuits apply RX, RY, and CNOT gates to introduce entanglement and capture higher-order dependencies in the data. The transformed quantum states are measured using Pauli-Z expectation values, which serve as enriched features for a Graph Neural Network (GNN). The final graph-level representation is processed by a classical classifier to distinguish fraudulent and non-fraudulent transactions.

My QGNN Implementation for Jet Classification

Building upon the inspiration from the paper on Quantum Graph Neural Networks (QGNNs) for fraud detection, I implemented a Quantum Graph Neural Network (QGNN) circuit tailored for jet classification rather than financial fraud detection.

Overview of My Implementation

In my implementation, I designed a **hybrid quantum-classical model** that incorporates a quantum circuit for feature transformation before passing the information through a classical GNN. The main architecture consists of:

1. **Graph Construction using k-Nearest Neighbors (k-NN):**

    a. The dataset consists of particle jet features (e.g., transverse momentum pT, rapidity y, azimuthal angle ϕ, and energy).
    b. Each jet is represented as a graph, where particles are nodes, and edges are defined using k-NN.
    c. This representation allows the model to learn relationships between particles in the jet.

2. **Quantum Feature Encoding in a Variational Quantum Circuit (VQC):**

    a. The quantum circuit encodes jet features into qubits using RX gates for angle encoding.
    b. A layered ansatz applies CNOT gates for entanglement, followed by parameterized rotations (RX, RY) to learn quantum feature transformations.
    c. The final Pauli-Z expectation values serve as transformed features, which are then used in the GNN layers.

3. **Graph Neural Network for Classification:**

    a. The Quantum Graph Convolutional Network (QGCN) applies message-passing operations to learn edge and node interactions.
    b. Each quantum-enhanced node representation is passed through graph convolution layers, followed by a global mean pooling layer to generate a graph-level representation.
    c. The final classification layer predicts whether a jet originated from a quark or a gluon based on the processed features.

Results and Performance

The training and validation results of my QGNN implementation showed promising classification accuracy for distinguishing between quark and gluon jets. The training process converged well, achieving a **validation accuracy** of approximately **72%**, with a training accuracy of around 75%.

The circuit diagram for this implementation is drawn within the code file(QGNN_1st_circuit.ipynb).

QGNN Circuit 2

Reference Paper and Its Architecture

For my second Quantum Graph Neural Network (QGNN) circuit, I drew inspiration from the ***"Quantum Graph Neural Network Models for Materials Search"*** [2] paper by Ju-Young Ryu et al. This research explores Quantum Graph Neural Networks (QGNNs) in the context of molecular property prediction, particularly using Equivariantly Diagonalizable Unitary Quantum Graph Circuits (EDU-QGCs). EDU-QGCs introduce equivariant transformations that preserve graph symmetries while minimizing quantum circuit depth. The key idea behind EDU-QGCs is to parameterize unitary operations such that they respect the permutation invariance of nodes, ensuring that learned representations remain consistent across different graph orders. These circuits apply node-local and entangling operations (CNOT and controlled RZ gates) on edges, allowing for efficient information propagation in graph-based quantum learning models.

My QGNN Implementation

Building upon this idea, I first implemented my QGNN circuit on the QM9 dataset, a benchmark dataset for molecular property prediction. Here, each molecule is represented as a graph, where nodes correspond to atoms and edges correspond to chemical bonds. I employed quantum feature encoding to transform atomic features into quantum states and then applied EDU-based transformations to enhance node embeddings. The final quantum states were measured using Pauli-Z expectation values, and the outputs were used for direct regression of molecular properties. The results were strong, with the model demonstrating smooth training convergence and low loss values, indicating that quantum-enhanced representations successfully captured molecular interactions.

Encouraged by the QM9 results, I then applied the same QGNN circuit to jet classification, treating each jet as a graph where nodes represent particles and edges are defined based on k-nearest Neighbors (k-NN). The quantum feature encoding and EDU transformations remained the same, preserving node interactions and capturing relational structures among particles. However, in contrast to QM9, the model performed poorly on jet classification, with loss values failing to converge and validation accuracy remaining significantly lower. This suggests that while EDU-QGC-based transformations effectively represent molecular systems, they may not generalize well to the high-dimensional and stochastic nature of jet classification tasks.

Overview of My Implementation

My implementation follows a **hybrid quantum-classical learning approach.** The key components of the circuit are:

1. **Graph Construction:**
    a. In QM9, molecules are modeled as graphs with atoms as nodes and bonds as edges.
    b. In jet classification, particles are treated as nodes, with k-NN defining edges to model spatial relationships.
2. **Quantum Feature Encoding & EDU-QGC Transformations:**
    a. Each node feature is encoded into quantum states using RY and RZ gates, ensuring information is mapped into a quantum Hilbert space.
    b. EDU-QGC layers process node embeddings through node-local operations and entangling two-qubit interactions on edges, ensuring graph topology is preserved.
    c. The EDU unitary layers introduce correlations between connected nodes while maintaining equivariance, ensuring that relational dependencies are well represented in the quantum model.
3. **Prediction Stage:**
    a. For QM9, the final quantum states are measured using Pauli-Z expectation values, and a classical linear readout layer is applied for molecular property regression.
    b. For jet classification, the quantum embeddings are passed into a Graph Neural Network (GNN), which applies message-passing operations to refine node interactions before classification.

Results and Performance

- QM9 Dataset: The model performed well, achieving low training and validation loss, with smooth convergence. This indicates that EDU-QGC-based quantum representations effectively capture molecular properties.
- Jet Classification: The model struggled, with poor convergence and low validation accuracy, highlighting that the same quantum embedding strategy that worked for QM9 was not directly effective for particle physics data.

This suggests that while EDU-QGC circuits enhance structured molecular data representations, they may require modifications to handle more complex, noisy, and high-dimensional datasets like jet classification.

The circuit for this implementation is drawn within the code file, following the described architecture (QGNN_2nd_circuit.ipynb).

References

[1] Innan, N., Sequeira, A., Canedo, A., & Campbell, T. (2023). ***Financial Fraud Detection using Quantum Graph Neural Networks*****. arXiv preprint arXiv:2309.01127.**
https://arxiv.org/abs/2309.01127

[2] Ryu, J.Y., Elala, E., & Rhee, J.K.K. (2023). ***Quantum Graph Neural Network Models for Materials Search*****.** ***Materials (Basel)*****, 16(12), 4300.**
https://pmc.ncbi.nlm.nih.gov/articles/PMC10304445/