Global Quantum Hackathon: Hybrid QML Solution for Stock Index Prediction

1. Introduction

This document details a hybrid quantum machine learning (QML) solution developed for the Global Quantum Hackathon. The challenge requires predicting the future Close prices of a stock index based on historical Open, High, Low, Close, and Volume data for the next 10 trading days.

Our approach utilizes a combination of classical and quantum techniques:

1. **Feature Engineering:** We transform the time-series data into 2D image-like representations and matrices using Quantum Gramian Angular Fields (QGAF).
2. **Hybrid Model:** We employ a hybrid architecture consisting of a classical Convolutional Neural Network (CNN) for feature extraction, which feeds into a trainable Quantum Variational Circuit (QVC) implemented using PennyLane and TensorFlow/Keras.
3. **Delta Prediction:** Instead of directly predicting the Close price, the model predicts the *difference* (delta) between the Close price and the mid-price ((High + Low) / 2.0) for that day.

2. Methodology

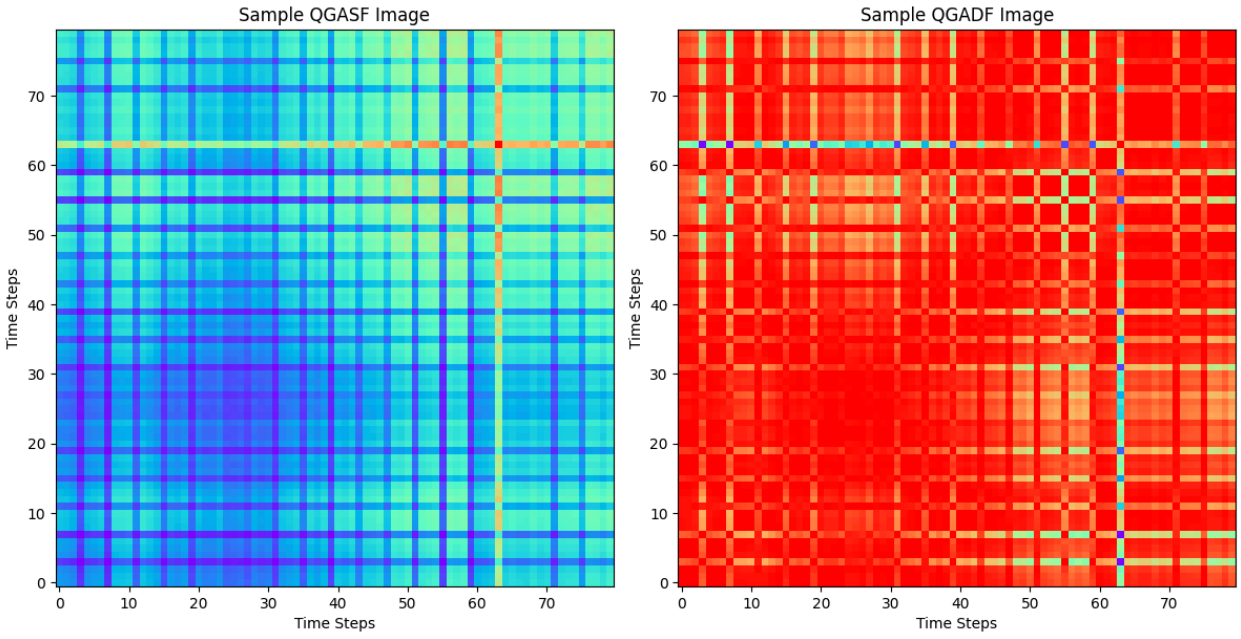
2.1 Data Loading & Preprocessing

- The Date column is converted to datetime objects, and the data is sorted chronologically.
- A StandardScaler from scikit-learn is fitted only on the training data's features (Open, High, Low, Volume) to prevent data leakage. This scaler is used later to transform both training and testing data windows.

2.2 Feature Engineering: Quantum Gramian Angular Fields (QGAF)

To leverage the pattern-recognition capabilities of CNNs, the 1D time-series data is converted into 2D, two-channel images using QGAF.

1. **Sliding Window:** A sliding window of length WIN_N = 20 days is used. For each prediction step, the preceding 20 days of the four features (FEATS = ['Open','High','Low','Volume']) are considered.
2. **Flattening & Scaling:** The (20, 4) window data is flattened into a vector of 80 features. This vector is then transformed using the pre-fitted StandardScaler.
3. **Angle Encoding:** The scaled 80-feature vector is mapped to angles in the range [0, \pi], a mapping which involves a tanh activation followed by np.arccos. This encodes the time-series information into quantum-suitable angles.
4. **QGAF Matrix Calculation:** Two Gramian matrices (80x80) are computed using the qgaf function based on these angles:
    - **QGASF (Sum Field):** $G_{sum} = \cos(\phi_i + \phi_j)$
    - **QGADF (Difference Field):** $G_{diff} = \cos(\phi_i - \phi_j)$



5. **Image Stacking:** The QGASF and QGADF matrices are stacked along the channel

dimension to create an (80, 80, 2) image for each time step. These images form the input X_train and X_test for the model.

6. **Test Image Generation:** The first test image uses the last 20 days of training data. Subsequent test images incorporate the newly available test data points while maintaining the 20-day window size, ensuring no future information is leaked.

2.3 Target Variable Strategy: Delta Prediction

To improve prediction stability and leverage the provided High and Low values in the test set, the model is trained to predict the *delta* from the mid-price:

- y_delta = Close - (High + Low) / 2.0

This transforms the task into predicting the deviation from a known baseline (mid_price), which is often easier than predicting the absolute Close value.

2.4 Hybrid Model Architecture (CNN-QVC)

1. **Classical CNN Head:**
    - Takes the (80, 80, 2) QGAF image as input.
    - Consists of stacked Conv2D and MaxPooling2D layers, followed by Flatten and Dense layers.
    - Acts as a feature extractor, identifying relevant patterns in the QGAF images and producing a classical feature vector (128 dimensions).
2. **Quantum Layer (QuantumLayer):**
    - **Interface:** A Dense layer maps the 128 classical features down to N_QUBITS = 4 features. These are scaled by π using a tanh activation and a Lambda layer to serve as input rotation angles (between $-\pi$ and π) for the quantum circuit.
    - **Quantum Node (qnode_tf):** A PennyLane qnode is defined to run on a simulator (default.qubit) using the TensorFlow interface and backpropagation for gradients.
    - **Quantum Circuit:**
        - **Encoding:** The 4 input features (angles) are encoded onto 4 qubits using qml.RY gates.
        - **Variational Layers:** qml.StronglyEntanglingLayers are applied. These layers contain **trainable parameters** (self.w).
        - **Measurement:** The expectation value of the Pauli-Z operator (qml.expval(qml.PauliZ(i))) is measured for each qubit, yielding 4 output features from the quantum layer.
    - **Trainability:** The QuantumLayer defines **self.w (shape (N_LAYERS, N_QUBITS, 3))** as a Keras trainable weight. This ensures the quantum circuit's parameters are optimized during model training.
3. **Classical Tail:**
    - The 4 outputs from the QuantumLayer are fed into a final Dense layer.
    - An output Dense(1, activation='linear') layer produces the final prediction for y_delta.

2.5 Training

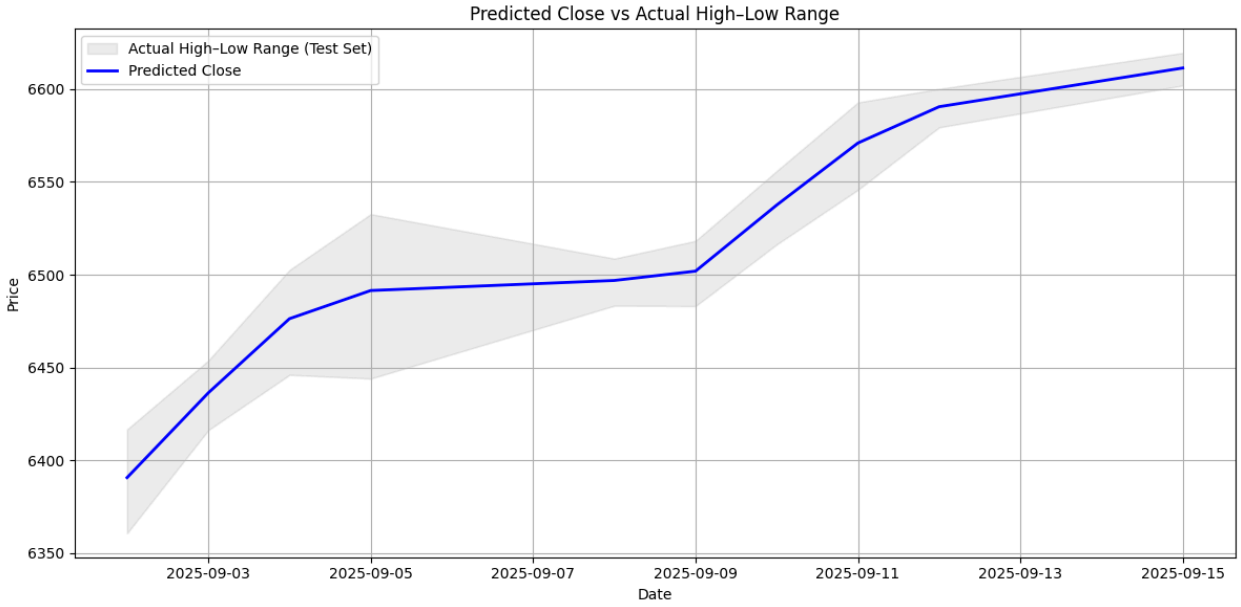
- The model is compiled with the Adam optimizer and mean squared error (mse) loss function.
- Training uses model.fit with standard Keras callbacks:
  - EarlyStopping
  - ReduceLROnPlateau
- An 80/20 time-based split is used for training and validation sets (X_tr, y_tr, X_va, y_va).

2.6 Prediction

1. The trained hybrid model predicts the y_hat_delta values for the X_test QGAF images.
2. The known mid-prices (mid_te = (High + Low) / 2.0) for the test days are extracted from X_test.csv.
3. The final Predicted_Close is calculated by adding the predicted delta back to the known mid-price: y_hat_close = mid_te + y_hat_delta.
4. These predictions are saved in the csv format.

3. Results

The model produces predictions for the 10 required test days. A plot generated in the notebook visually confirms that the Predicted_Close values consistently fall within the actual High–Low range provided in the X_test.csv dataset, demonstrating the effectiveness of the delta prediction strategy.



4. Conclusion

The combination of QGAF feature engineering, a CNN feature extractor, a trainable

PennyLane QVC, and a delta prediction strategy provides a robust and innovative approach to the time-series forecasting problem.

5. Future Enhancements

Another feature mapping strategies could be tested to enrich the feature set, including specific aspects of market behavior, such as momentum, trend, and volatility.