Project Report: Hybrid Quantum-Classical Stock Predictor (VQC Model 1)

Based on Hybrid/model1

/main.ipynb

1. Project Objective

The objective of this project was to design, train, and evaluate a hybrid quantum-classical model for time-series forecasting, specifically to predict the 'Close' price of a stock. The model leverages a classical recurrent neural network (LSTM) for primary pattern recognition and a quantum-powered circuit to predict and correct the residual error.

2. Data and Preprocessing

- Data Source: The model was trained on X_train.csv and used to predict values for X_test.csv.
- Feature Engineering: 10 distinct features were used for training. In addition to the standard 'Open', 'High', 'Low', 'Close', and 'Volume', advanced technical indicators were engineered, including:
    - MA10 (10-day Moving Average)
    - MA30 (30-day Moving Average)
    - RSI (Relative Strength Index)
    - Lag_1 (Lagged 'Close' price by 1 day)
    - Lag_5 (Lagged 'Close' price by 5 days)
    - (Bollinger Bands were also created but not used in the final feature list to avoid data leakage).
- Scaling: RobustScaler was used to normalize the 10 features. This scaler is less sensitive to outliers than standard scalers.
- Windowing: The time-series data was structured into sequences (windows) of 20 timesteps to be fed into the LSTM, creating 145 training samples.

3. Model Architecture (Quantum Residual)

The model is a hybrid PyTorch-Qiskit architecture named QuantumResidualModel.

- Classical Backbone: A 2-layer, bidirectional LSTM with 64 hidden units and a Dropout layer (0.2) was used as the primary classical predictor. This backbone processes the 20-day window and outputs a preliminary price prediction via two linear layers.

- Quantum Solution: A 4-qubit Variational Quantum Circuit (VQC) was designed using Qiskit.
    - Design (Model 1): This circuit was defined manually using ParameterVector. It encodes 4 features (derived from the LSTM's 128-unit output) using RY gates. The trainable ansatz consists of 2 repetitions of RY rotations followed by CX (CNOT) entangling gates.
    - Observable: The measurement was defined as the average magnetization of all qubits (Pauli "Z" operator on all 4 qubits).
- Integration: The Qiskit circuit was made compatible with PyTorch's autograd system using the TorchConnector.
- Forward Pass: The model's final output is the sum of the classical prediction and the quantum circuit's prediction (the "quantum solution").

4. Training Protocol

- Loss Function: MSELoss (Mean Squared Error).
- Optimizer: AdamW, a modern variant of Adam with improved regularization.
- Learning Rate Scheduler: CosineAnnealingWarmRestarts was used to cyclically adjust the learning rate, helping the model escape local minima.
- Training: The model was trained for 100 epochs with a batch size of 32. The final average training loss was 0.013426.
- Output: The trained model weights were saved to qiskit_residual_model.pth.

5. Evaluation & Results

Performance was measured on a validation set (the last 15% of the training data). The model demonstrated solid predictive capability.

- R-squared (R^2) Score: 0.7052
- Root Mean Squared Error (RMSE): 36.58
- Mean Absolute Error (MAE): 28.24

The final trained model was then used to perform an iterative prediction on the X_test.csv data, and the results were saved to predictions.csv.
