Technical Report: A Dual-Head Hybrid LSTM Model for Clamped Stock Price Prediction

October 26, 2025

This report details the architecture and methodology of the stock price prediction model implemented in the notebook `Stock_Price_Prediction_with_Quantum_Enhanced_CNN_and_QGAF (1).ipynb`. The executed code implements a **Dual-Head Hybrid LSTM model**. This model predicts the normalized daily return, which is then used to forecast the 'Close' price with a high-low clamping constraint.

1 Objective and Core Strategy

The primary objective is not to predict the exact 'Close' price directly, but rather to predict the **normalized return (r_{norm})** relative to the 'Open' price.

$$r_{\text{norm}} = \frac{\text{Close} - \text{Open}}{\text{Open}} \tag{1}$$

This r_{norm} is then decomposed into its **sign (direction)** and **magnitude (volatility)**, which are predicted by two separate network heads. This dual-head strategy allows the model to decouple the classification task (is the price going up or down?) from the regression task (by how much?).

The final 'Close' price is then reconstructed using the predicted r_{norm} and clamped to the day's actual 'Low' and 'High' values to ensure a realistic prediction.

2 Feature Engineering and Preprocessing

The model uses a sequence of 60 days as a lookback window. Each day in this sequence is described by a 24-dimensional feature vector.

- **Input Data:** The model ingests time-series data containing `Date`, `Open`, `High`, `Low`, `Close`, and `Volume`.
- **Engineered Features:**
  - **Basic:** `Open`, `High`, `Low`, `LogVolume` (using `log1p`).
  - **Price-Based:** `High_Low` (daily range).
  - **Cyclical Time Features:** `DOW_sin`/`DOW_cos` (Day of Week) and `month_sin`/`month_cos` (Month of Year) are encoded using sine/cosine transformations.

  - **Lagged Features:** `Close_lag1` and `Close_lag2`.
  - **Technical Indicators:**
    * **Returns/Volatility:** 5-day and 10-day log returns (`ret_5`, `ret_10`) and standard deviation (`vol_5`, `vol_10`).
    * **Moving Averages:** 5, 10, and 20-day Simple Moving Averages (`SMA_5_lag1`, etc.).
    * **Momentum/Oscillators:** 14-day RSI (`RSI_14_lag1`), EMA Gap, and a 60-day Drawdown.
    * **Other:** 14-day Average True Range (`atr14`) and a Volume Z-score.
- **Preprocessing:** All 24 features are scaled using `sklearn.preprocessing.StandardScaler`, which is `fit` on the training data only.

3 Model Architecture

The model is a hybrid dual-head architecture implemented in TensorFlow/Keras, combining a classical LSTM stack with a variational quantum circuit (VQC) from PennyLane.

1. **Input Layer:** `layers.Input(shape=(60, 24))` — Accepts a sequence of 60 days, each with 24 features.
2. **Classical RNN Trunk:**
   - A stacked LSTM layer: `layers.LSTM(64, return_sequences=True)`.
   - A second LSTM layer: `layers.LSTM(32, return_sequences=False)` outputs a single vector representing the 60-day lookback.
   - A dense embedding layer: `Dense(64, activation="relu")`.
3. **Quantum Branch:**
   - The 64-feature embedding is compressed via `Dense(4, activation="tanh")`.
   - These 4 features are fed into a `QLayer` (a custom Keras layer).
   - **Quantum Circuit:** Inside `QLayer`, a 4-qubit PennyLane QNode (`qml.device("default.qubit", ...)`) performs:
     - **Encoding:** `qml.AngleEmbedding` encodes the 4 classical features.
     - **Variational Layer:** `qml.StronglyEntanglingLayers` (with 2 layers) acts as the trainable quantum network.
     - **Measurement:** `qml.expval(qml.PauliZ(0))` measures the expectation value of the first qubit, returning one classical value.
4. **Hybrid Concatenation:** The output from the classical LSTM trunk (64 features) and the output from the quantum circuit (1 feature) are concatenated (`layers.Concatenate()`) into a 65-feature hybrid vector.
5. **Dual Prediction Heads:** This hybrid vector is fed into two parallel heads:
   - **Classification Head (Direction):**

  - `Dense(32, activation="relu") -> Dropout(0.3) -> Dense(1, activation="sigmoid")`
  - **Name:** `cls_out`
  - **Target:** `sign` (0 or 1)
  - **Loss:** `BinaryCrossentropy`
- **Regression Head (Magnitude):**
  - `Dense(32, activation="relu") -> Dense(1, activation="relu")`
  - **Name:** `reg_out`
  - **Target:** `mag` (absolute normalized return)
  - **Loss:** `Huber`

4 Training and Loss Function

The model is trained to minimize a weighted sum of the two head-specific loss functions.

- **Optimizer:** `optimizers.Adam(learning_rate=1e-3)`
- **Losses:** {`"cls_out": BinaryCrossentropy, "reg_out": Huber`}
- **Loss Weights:** {`"cls_out": 1.0, "reg_out": 1.0`. The total loss is the simple sum of the two.
- **Callbacks:**
  - `EarlyStopping`: Monitors `val_loss` with `patience=15`.
  - `ReduceLROnPlateau`: Reduces learning rate if `val_loss` stagnates.

5 Prediction and Reconstruction

The model's predictive power is demonstrated in its walk-forward forecasting method on the test set.

1. **Seeding:** The loop is initialized with the last 60 days of scaled training data (`X_train_tail`).
2. **Recursive Prediction:** The model iterates one day at a time:
   - **Predict r_{norm}:** The model predicts `p_up` (direction) and `mag` (magnitude). These are combined:
     $r_{\text{norm\_pred}} = (2.0 \cdot p_{\text{up}} - 1.0) \cdot \text{mag}$
   - **Reconstruct 'Close' Price:** The 'Close' price is calculated:
     $C_{\text{pred}} = \text{Open}_{\text{actual}} \cdot (1.0 + r_{\text{norm\_pred}})$
   - **Clamping:** The predicted price is clamped to the actual day's trading range:
     $C_{\text{final}} = \texttt{np.clip}(C_{\text{pred}}, \text{Low}_{\text{actual}}, \text{High}_{\text{actual}})$
3. **Feature Update:** This C_{final} value is used to update the `Close_lag1` and `Close_lag2` features for the *next* day's prediction. The 60-day window slides forward, and the process repeats.

This recursive, clamped forecasting method makes the model's output robust and realistic.