

# Problem Context

**Objective:** Predict and execute trades on 28 currency pairs by leveraging advanced feature extraction, dimensionality reduction, and graph neural networks to process and interpret market data.

## Key Components of the Model

1. Currency Pairs and Variables:
- P: Set of currency pairs,  $P = \{EURUSD, GBPUSD, \dots, GBPAUD\}$  (total: 28 pairs).

• Variables represent price changes, volatilities, clustering coefficients, candlestick patterns, and adjacency matrices for relationships between pairs.
2. Feature Extraction:
- Log Returns:  $r_t = \log(\frac{p_t}{p_{t-1}})$ , where  $p_t$  is the close price at time  $t$ .

• Volatility: Rolling standard deviation:  $\sigma_t = \sqrt{\text{SMA}(r_t^2, 20)}$ .

• Volatility Clustering: Defined as  $C_t = \frac{\text{SMA}(\sigma_t, 10)}{\text{stddev}(\sigma_t, 10)}$ .

• Candlestick Features:

• Body =  $|p_{\text{open}} - p_{\text{close}}|$ ,

• Upper Shadow =  $p_{\text{high}} - \max(p_{\text{open}}, p_{\text{close}})$ ,

• Lower Shadow =  $\min(p_{\text{open}}, p_{\text{close}}) - p_{\text{low}}$ .
3. Dimensionality Reduction (PCA):
- Extract principal components from Candlestick Features + Volatility Metrics.

• Eigen-decomposition:  $\mathbf{K} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T$ ,

•  $\mathbf{K}$ : Covariance matrix.

•  $\mathbf{V}$ : Eigenvectors (principal components).

•  $\mathbf{\Sigma}$ : Diagonal matrix of eigenvalues.
4. Relational Graph Construction:
- Adjacency Matrices ( $\mathbf{A}_r$ ): Define relationships between currency pairs for different criteria ( $r = 1, 2, \dots, 5$ ).

• Weighted relationships:

•  $\mathbf{A}_r[i, j] = 1$  if  $i = j$ ,

•  $\mathbf{A}_r[i, j] = \text{Random Weight (0.1)}$  otherwise.
5. Graph Neural Network (R-GCN):
- Input Features:  $\mathbf{X}_t \in \mathbb{R}^{P \times F}$ , where  $F = 3$  (candlestick features).

- Propagation Rule:

$$\mathbf{H}^{(l+1)} = \sigma \left( \sum_{r=1}^R \mathbf{A}_r \mathbf{H}^{(l)} \mathbf{W}_r^{(l)} \right),$$

- $\mathbf{H}^{(l)}$ : Node features at layer  $l$ .
- $\mathbf{W}_r^{(l)}$ : Learnable weight matrices.
- $\sigma(x)$ : Activation function (ReLU).

## 6. Output Layer and Decisions:

- Softmax probabilities for actions (Buy, Sell, Hold):

$$P(a|\mathbf{H}) = \frac{\exp(\mathbf{z}_a)}{\sum_{b=1}^3 \exp(\mathbf{z}_b)},$$

- $\mathbf{z}_a$ : Logits for action  $a$ .

## 7. Trading Signals:

- Compute signals:

$$\text{Signal}[i] = P(\text{Buy}) - P(\text{Sell}),$$

- $\text{Signal}[i] > 0.5$ : Strong Buy (Action = 1).
- $\text{Signal}[i] < -0.5$ : Strong Sell (Action = -1).
- Otherwise, Action = 0 (Hold).

# Optimization and Execution

## 1. Dynamic Adjustment:

- Optimize  $\mathbf{W}_r^{(l)}$  using stochastic gradient descent (SGD) or similar methods to minimize prediction error.

## 2. Execution of Trades:

- Implement trades based on the signal using `enterLong()` or `enterShort()` functions.

# Challenges Identified

- Resource Bottlenecks:** GPU contention due to simultaneous operations.
- Trade-off Optimization:** Synchronization introduced delays, impacting overall speed.
- Adaptive Improvement:** Fine-tuning model hyperparameters and system configurations for real-time adaptability.