**Methodology**

**1. Introduction and Rationale**

This chapter outlines the overall methodological framework used to investigate two approaches for portfolio optimization under uncertain market conditions:

1. **Predicting Key Inputs for a Traditional Portfolio Optimization** (volatility forecasts and covariance matrix estimation).
2. **Direct Portfolio Weight Prediction** (supervised learning and deep reinforcement learning).

Much of the existing portfolio management literature focuses on a two-step procedure, commonly referred to as the “estimate-then-optimize” paradigm. Traditional econometric models (e.g., various GARCH specifications) and the seminal Heterogeneous AutoRegressive (HAR) model (Corsi, 2009) have historically guided the estimation of crucial inputs such as expected returns, volatilities, and covariances (Andersen, Bollerslev, & Diebold, 2007; Corsi, 2009). More recent work has embraced tree-based machine learning and deep neural networks, as exemplified by a growing array of volatility forecasting competitions and machine-learning-based papers (Christensen, Siggaard, & Veliyev, 2021; Zhang et al., 2024). Nevertheless, the modern proliferation of market data and computationally enhanced forecasting techniques suggests we should explore both input-driven approaches and direct weight-prediction paradigms.

The overall research design includes:

* **Data Preprocessing and Feature Engineering** from minute-level crypto prices and macroeconomic series.
* **Two main methodological approaches**:
  1. A *two-step* volatility/covariance forecasting pipeline followed by a classical mean–variance style optimization (including robust variants).
  2. A *one-step* direct weight prediction approach, split into supervised learning (predicting historical realized weights) and deep reinforcement learning (PPO-based).

Each section describes the theoretical underpinnings (including references to the user-uploaded studies) and justifies how each component (XGBoost, LSTM, Autoformer, ADCC, PPO, etc.) was chosen in line with recent advances from the literature (e.g., Brauneis & Sahiner, 2024; Kwan, 2017; Kuhn, Shafiee, & Wiesemann, 2024).

**2. Data Collection and Preprocessing**

**2.1 Data Sources**

1. **Cryptocurrency Minute-Level Prices (Binance API)**  
   We gathered 1-minute interval price data for major crypto assets (e.g., BTC, ETH) to capture intraday fluctuations. Minute-level data provide sufficient detail to construct daily realized volatility while still being computationally manageable. This minute-bar data includes open, high, low, close, and volume (OHLCV).
2. **Macroeconomic Indicators (Yahoo Finance)**  
   We pulled daily macro features such as equity indices (e.g., S&P 500), volatility indices (VIX), interest rates, or Fed macro releases. Although crypto may show weaker ties to conventional macro signals, recent evidence (Guidolin, Panzeri, & Pedio, 2024) suggests that broad economic conditions can still inform risk sentiment for crypto markets.
3. **Data Range and Coverage**  
   The primary sample stretches from mid-2018 to late 2023 to ensure coverage across multiple volatility regimes. The dataset was split into:
   * **Training Window:** 2018-06-01 to 2022-12-31
   * **Validation Window:** 2023
   * **Testing Window:** Post-2023 data for final out-of-sample performance checks.

**2.2 Data Cleaning and Alignment**

* **Resampling to Daily**  
  Although prices are recorded at minute intervals, we aggregate them to daily frequency for both price returns and realized volatility calculations. Missing data (e.g., from API downtime) is imputed through forward-filling or, if necessary, interpolation.
* **Constructing Realized Volatility (RV)**  
  For each crypto asset iii on a given day ttt, we approximate realized volatility by summing squared intraday log returns: RVi,t  =  ∑τ=1Mri,t,τ2, RV\_{i,t} \;=\; \sum\_{\tau=1}^{M} r\_{i,t,\tau}^2,RVi,t​=τ=1∑M​ri,t,τ2​, where ri,t,τr\_{i,t,\tau}ri,t,τ​ is the log return for asset iii in sub-interval τ\tauτ of day ttt and MMM is the number of intervals in a trading day. This daily RV is crucial for training each volatility forecasting model (Brauneis & Sahiner, 2024).
* **Aligning Macroeconomic Data**  
  Macroeconomic factors, usually available only on business days, are merged with the crypto daily data. Non-trading days for macros are forward-filled, acknowledging potential biases.

**2.3 Feature Engineering**

* **Time Series Features**  
  We produce:
  1. **Lagged Realized Volatilities** at daily, weekly, and monthly horizons (the standard HAR decomposition; see Corsi, 2009).
  2. **Lagged Returns** (e.g., 5 lags of daily crypto returns).
  3. **Technical Indicators:** RSI, moving averages (e.g., EMA12, EMA26), MACD (Zhang et al., 2024).
* **Macro and Cross-Asset Indicators**  
  Incorporate VIX, S&P 500 returns, 10-year Treasury yields, and crypto-specific sentiment indexes (when available).
* **Normalization**  
  Given the wide range of feature scales (volume vs. RSI vs. macro indexes), all features are standardized (z-score) within the training set and consistently applied to validation/test sets.

**3. Approach 1: Predicting Inputs for Portfolio Optimization**

This approach follows the classic “estimate-then-optimize” route but with modern enhancements:

1. **Volatility Forecasting** for each asset iii.
2. **Correlation/ Covariance Estimation** using a dynamic correlation model (ADCC) or direct Cholesky factor approach.
3. **Shrinking/Filtering** the resulting covariance.
4. **Mean–Variance Portfolio Optimization** or robust variants (distributionally robust optimization, shrinkage of correlation matrix, etc.).

**3.1 Volatility Forecasting Models**

**3.1.1 XGBoost with HAR Features**  
XGBoost (Chen & Guestrin, 2016) is a gradient-boosted decision tree algorithm that has proven successful in volatility forecasting (Christensen et al., 2021; TODIM-XGBoost references in Jain, Sahay, & Nupur, 2024). Here, we feed in:

* HAR-based lags (RVt−1RV\_{t-1}RVt−1​, RVt−1:t−5RV\_{t-1:t-5}RVt−1:t−5​, RVt−1:t−22RV\_{t-1:t-22}RVt−1:t−22​)
* Additional lags (returns, macro variables, etc.)
* Relevant technical indicators.

We train a separate XGBoost regressor for each asset’s daily volatility. Using regularized objective functions, XGBoost typically avoids overfitting even with many correlated features (Zhang, Zhang, Cucuringu, & Qian, 2024).

**3.1.2 Direct Prediction of Cholesky Decomposition (LSTM, Transformer, Autoformer)**  
While XGBoost plus HAR features works at the univariate level, alternative methods attempt a more holistic approach by directly predicting the entire covariance matrix. Instead of forecasting each pairwise correlation separately, we parameterize the covariance matrix through its Cholesky factor (Bucci, 2020). The neural networks used include:

* **LSTM** (Long Short-Term Memory): Suited for time series with memory effects. It processes sequences of daily data, aiming to model both short- and long-horizon dependencies.
* **Basic Transformer** (Vaswani et al., 2017): Uses self-attention to capture patterns across time steps. Shown to excel in sequence tasks, though often data-intensive.
* **Autoformer** (Wu, Xu, Wang, & Long, 2021): A specialized Transformer variant for long-term time series forecasting, featuring a decomposition mechanism that isolates trends and seasonal components.

Each network outputs a vector of parameters (the Cholesky components), which we reshape into the forecast covariance matrix Σ^t+1\hat{\Sigma}\_{t+1}Σ^t+1​. This direct approach is conceptually elegant but can be more complex to implement and tune (Huang, Newton, Platanakis, & Sutcliffe, 2024).

**3.2 Correlation Estimation via ADCC**

Dynamic conditional correlation (DCC) methods (Engle, 2002) and their asymmetric extension (ADCC; Cappiello, Engle, & Sheppard, 2006) have become standard for capturing time-varying correlations among asset returns. In an ADCC model:

1. Univariate GARCH (or in our case, machine-learning-based volatility forecasts) standardize each asset’s returns.
2. A correlation update equation then evolves over time, allowing negative returns to increase correlation more strongly (an asymmetric effect).

**3.3 Covariance Matrix Robustification**

Following the logic of Ledoit and Wolf (2004) and Kwan (2017), the estimated covariance matrix Σ^t\hat{\Sigma}\_tΣ^t​ may be excessively noisy in high-dimensional settings. Two steps are standard:

1. **Shrinkage**  
   Form a convex combination of Σ^t\hat{\Sigma}\_tΣ^t​ with a more stable target (e.g., scaled identity or historical average). This reduces variance in the estimate:

Σ~t  =  δ Σtarget  +  (1−δ) Σ^t, \tilde{\Sigma}\_t \;=\; \delta \,\Sigma\_{\text{target}} \;+\; (1 - \delta)\,\hat{\Sigma}\_t,Σ~t​=δΣtarget​+(1−δ)Σ^t​,

where δ\deltaδ is chosen via a closed-form formula or cross-validation.

1. **Eigenvalue Filtering**  
   After shrinkage, any negative or extremely small eigenvalues are floored to a small positive threshold to ensure positive semi-definiteness (Zhang et al., 2024; Kwan, 2017).

**3.4 Portfolio Optimization**

With Σ~t\tilde{\Sigma}\_tΣ~t​ in hand, we solve a standard mean–variance problem or a robust extension. The user can specify whether to include an estimate of expected return. Typical solutions include:

* **Global Minimum Variance (GMV):**  
  min⁡w  w⊤Σ~tw,s.t.  1⊤w=1,  w≥0\min\_{w} \; w^\top \tilde{\Sigma}\_t w,\quad \text{s.t.} \;\mathbf{1}^\top w=1,\; w\ge 0minw​w⊤Σ~t​w,s.t.1⊤w=1,w≥0 (if no shorting).
* **Mean–Variance:**  
  max⁡w  w⊤μ^t  −  λ w⊤Σ~tw,\max\_{w} \; w^\top \hat{\mu}\_t \;-\; \lambda \, w^\top \tilde{\Sigma}\_t w,maxw​w⊤μ^​t​−λw⊤Σ~t​w, subject to feasible constraints.
* **Distributionally Robust Optimization (DRO):**  
  Instead of plugging in one covariance, we define an ambiguity set P\mathcal{P}P of possible distributions around Σ^t\hat{\Sigma}\_tΣ^t​ and solve min⁡wmax⁡P∈PVarP(w)\min\_{w}\max\_{P \in \mathcal{P}} \mathrm{Var}\_P (w)minw​maxP∈P​VarP​(w), in line with Kuhn, Shafiee, & Wiesemann (2024) and references therein.

Rebalancing occurs every 14 days with the updated Σ~t\tilde{\Sigma}\_tΣ~t​ and, if needed, updated μ^t\hat{\mu}\_tμ^​t​. Transactions costs are either ignored in this method or accounted for ex post (see, e.g., Kwan, 2017, for a shrinkage illustration).

**4. Approach 2: Direct Portfolio Weight Prediction**

Rather than forecasting inputs, a second approach learns the mapping from historical data to optimal weights directly. This can bypass the possible error compounding in the “estimate-then-optimize” pipeline (Huang et al., 2024). We implement two sub-methodologies:

1. **Supervised Learning** to mimic historically optimal weights from realized daily volatility.
2. **Deep Reinforcement Learning** (PPO) to learn trading actions that maximize cumulative return or Sharpe-like reward.

**4.1 Supervised Learning on Historical Realized Weights**

**4.1.1 Historical Realized Weights Computation**  
We first compute daily ex post “optimal” weights using realized data. For instance:

* **Ex Post GMV Weights**  
  If Σrealized\Sigma^\text{realized}Σrealized is the realized covariance matrix of day ttt, the ex post GMV weights might be w^\* \propto \Sigma^\text{realized^{-1}} \mathbf{1}.
* Alternatively, one can use ex post utility-based allocations or risk-parity logic.

These weights (vectors wt∗w^\*\_twt∗​) become the training target in a supervised model that sees feature vectors xtx\_txt​ from day ttt (lagged volatilities, returns, indicators, etc.) and aims to predict wt∗w^\*\_twt∗​. The underlying premise is that certain “rules” from ex post allocations can generalize to new data (XGBoost or neural networks can discover complex patterns).

**4.1.2 Model Architectures**

1. **LSTM**  
   Similar to volatility forecasting, an LSTM can process multiple time steps of input features to produce a predicted weight vector w^t\hat{w}\_tw^t​. This captures dynamic relationships, e.g., how recent volatility spikes or macro shifts prompt a portfolio tilt.
2. **Transformer**  
   The self-attention mechanism can glean patterns from both near and distant past. Transformers have shown strong performance in financial time series tasks that involve longer sequence dependencies (Wu et al., 2021).
3. **Autoformer**  
   Specifically designed for time series forecasting, Autoformer’s seasonal-trend decomposition may help highlight cyclical patterns in crypto markets. It may generate more stable or interpretable direct weight predictions (Wu et al., 2021).

**4.1.3 Loss Functions and Constraints**  
We use a standard mean-squared error (MSE) loss between w^t\hat{w}\_tw^t​ and wt∗w^\*\_twt∗​, potentially with a simplex or positivity constraint: ∑iw^t,i=1,w^t,i≥0\sum\_i \hat{w}\_{t,i} = 1, \hat{w}\_{t,i} \ge 0∑i​w^t,i​=1,w^t,i​≥0. Alternatively, a softmax layer can ensure nonnegative weights that sum to 1. Some practitioners incorporate advanced custom losses that reflect portfolio variance or drawdown, but MSE is the simplest.

**4.2 Deep Reinforcement Learning (DRL) with PPO**

**4.2.1 Motivation**

The key difference between DRL and supervised approaches is that DRL learns an *optimal policy* of actions (weight adjustments) by interacting with a simulated environment that reflects returns, transaction costs, and risk constraints (Guidolin et al., 2024; Jain et al., 2024). This agent-based approach can, in principle, adapt to shifting market regimes better than static supervised methods, because it is not locked into the ex post weighting solutions from a possibly outdated period.

**4.2.2 PPO Algorithm**

Proximal Policy Optimization (PPO) is an on-policy, actor-critic method proposed by Schulman et al. (2017). We build a custom environment that mimics crypto trading on Binance with:

* **State sts\_tst​:** includes current prices, past returns, realized volatility, open positions, account balance, etc.
* **Action ata\_tat​:** rebalancing the portfolio weights across N crypto assets (and possibly a cash position).
* **Reward rtr\_trt​:** typically daily or weekly PnL adjusted by a risk measure (e.g., negative of realized variance or a Sharpe ratio).
* **Episode Termination:** occurs after a fixed horizon (e.g., 120 days) or triggered by margin calls, large drawdowns, or user-defined conditions.

The **actor** network outputs a probability distribution over possible weight choices (continuous action space). An **LSTM layer** processes the state, followed by feed-forward layers that produce the final policy parameters. The **critic** network estimates the expected cumulative return from the current state. PPO’s clipped loss function ensures stable updates. Transaction fees, slippage, and leverage constraints are built into the environment (see the step function code snippet in the user-provided environment; the environment enforces real-world constraints such as total leverage and net exposure).

**4.2.3 Training Protocol**

* **Sampling Episodes**: We randomly start the agent in different market segments from 2018–2022 to ensure broad coverage of volatility regimes. Each episode simulates 120 days.
* **Policy Updates**: The agent’s transitions (st,at,rt,st+1)(s\_t, a\_t, r\_t, s\_{t+1})(st​,at​,rt​,st+1​) are aggregated into mini-batches, used to update the actor–critic networks via stochastic gradient descent.
* **Reward Shaping**: We add mild risk penalties for large drawdowns or excessive turnover to encourage stable performance.
* **Validation**: Periodically test on an unseen portion of data (the 2023 set) to check for overfitting.

**4.2.4 Final Evaluation**

Out-of-sample performance from a trained PPO agent is assessed on the post-2023 test set. Key metrics include annualized return, volatility, Sharpe ratio, and max drawdown. Since DRL typically includes transaction costs in the reward function, we compare net returns to the standard benchmarks (e.g., Markowitz, 1/N naive portfolio, or the best supervised model).

**5. Metrics, Benchmarking, and Implementation Details**

**5.1 Model Training and Validation**

1. **Volatility Forecasting**:
   * Models (XGBoost, LSTM, Transformer, etc.) are trained on 2018–2022 data.
   * Hyperparameters tuned by validation (2023).
   * Performance measured by RMSE or MAPE of daily volatility predictions.
2. **Portfolio Outcomes**:
   * For Approach 1: We input predicted volatilities/correlations into GMV/MDP/robust optimization. Rebalance every 14 days or monthly.
   * For Approach 2: We predict or compute daily weights directly (supervised) or run the DRL agent.
3. **Optimization or DRL**:
   * Evaluate final weight allocations on out-of-sample data post-2023.

**5.2 Performance Criteria**

1. **Volatility/Correlation Forecast Metrics**
   * **RMSE** of one-day-ahead realized volatility.
   * **Diebold–Mariano Tests** to check statistical significance of forecasting improvements.
2. **Portfolio Performance Metrics**
   * **Annualized Return (Rˉ\bar{R}Rˉ)** and Volatility (σ\sigmaσ)
   * **Sharpe Ratio** = Rˉ/σ\bar{R} / \sigmaRˉ/σ
   * **Max Drawdown**: largest peak-to-trough decline during the test period.
   * **Turnover**: trading frequency is relevant for cost analysis.
3. **Benchmarks**
   * **Historical Sample Covariance** with a standard Markowitz or GMV solution.
   * **Naive 1/N** equal weighting.
   * Compare results to both supervised approach (Approach 2a) and PPO RL approach (Approach 2b).

**5.3 Software Implementation**

* **Python** with libraries: NumPy, pandas, PyTorch, TensorFlow, or R for XGBoost, random forest, LSTM, Transformer, and Autoformer.
* **Reinforcement Learning**: Stable Baselines or custom PPO code.
* **Hyperparameter Tuning**: We rely on random search or Bayesian optimization for each model.
* **Computational Resources**: At least one high-end GPU is recommended for deep networks (Autoformer, PPO) given the large data dimension.

**6. Integration with Literature**

The method draws extensively from:

* **HAR & ML Volatility Forecasting**: Corsi (2009), Zhang et al. (2024), Christensen et al. (2021), Brauneis & Sahiner (2024).
* **Covariance Shrinkage**: Kwan (2017) describing constant-correlation shrinkage, along with the classic Ledoit–Wolf approach.
* **Distributionally Robust Optimization**: Kuhn, Shafiee, & Wiesemann (2024) for worst-case approaches around an empirical reference distribution.
* **Direct Weight Prediction**: Huang, Newton, Platanakis, & Sutcliffe (2024) for single-stage automated learning of portfolio allocations, plus references to earlier supervised approaches (Guidolin et al., 2024).
* **DRL**: The PPO-based approach references the surge in reinforcement learning for portfolio decisions (Jain et al., 2024; “Machine Learning in Portfolio Decisions” by Guidolin, Panzeri, & Pedio, 2024).

By blending volatility modeling, robust covariance estimation, and advanced direct weight-prediction strategies, this methodology addresses the multifaceted nature of portfolio optimization in highly volatile and rapidly evolving crypto markets. It ultimately provides a comparative study of:

1. Traditional approaches (HAR + ADCC + Markowitz),
2. Machine learning–augmented input prediction (XGBoost, LSTM, Transformer, Autoformer),
3. Direct weight inference from supervised learning,
4. Dynamic DRL-based trading policies.

**7. Expected Outcomes and Significance**

We anticipate the following from this methodological design:

1. **Improved Volatility Forecast Accuracy**:
   * XGBoost and neural networks often outperform baseline HAR in daily volatility forecasting (Christensen et al., 2021; Brauneis & Sahiner, 2024).
   * Direct forecasting of covariance via LSTM/Transformer could yield more nuanced correlation structure, though possibly data-hungry.
2. **Better Portfolio Performance**:
   * Input-driven (Approach 1) may still produce robust allocations if the correlation matrix shrinkage is effective.
   * Direct weight methods (Approach 2) could surpass the “estimate-then-optimize” pipeline by minimizing propagation of forecast errors.
3. **Scalability and Practicality**:
   * The DRL approach is promising yet can be more complex and sensitive to hyperparameters.
   * For practitioners with limited resources, an XGBoost-based volatility forecast plus simpler correlation estimation might suffice.

From an academic standpoint, comparing these strategies in identical data settings contributes to the ongoing debate about direct vs. two-stage portfolio construction (Huang et al., 2024). Practically, the combination of robust volatility estimates, advanced machine learning, and state-of-the-art DRL can help professional portfolio managers navigate cryptomarkets’ high volatility, benefiting both hedge funds and sophisticated retail investors.

**References**

* Andersen, T. G., Bollerslev, T., & Diebold, F. X. (2007). Roughing it up: Including jump components in the measurement, modeling, and forecasting of return volatility. *The Review of Economics and Statistics*, 89(4), 701–720.
* Brauneis, A., & Sahiner, M. (2024). Crypto Volatility Forecasting: Mounting a HAR, Sentiment, and Machine Learning Horserace. *Asia-Pacific Financial Markets*.
* Bucci, A. (2020). Deep learning the Cholesky decomposition for large-scale covariance forecasting. *arXiv preprint*.
* Cappiello, L., Engle, R. F., & Sheppard, K. (2006). Asymmetric dynamics in the correlations of global equity and bond returns. *Journal of Financial Econometrics*, 4(4), 537–572.
* Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.
* Christensen, K., Siggaard, M., & Veliyev, B. (2021). A Machine Learning Approach to Volatility Forecasting. *CREATES working paper*.
* Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2), 174–196.
* Guidolin, M., Panzeri, G., & Pedio, M. (2024). Machine Learning in Portfolio Decisions. SSRN working paper no. 4988124.
* Huang, X., Newton, D. P., Platanakis, E., & Sutcliffe, C. (2024). Single-stage portfolio optimization with automated machine learning for M6. *International Journal of Forecasting*.
* Jain, V., Sahay, R. R., & Nupur (2024). TODIM with XGBOOST and MVO metaheuristic approach for portfolio optimization. *International Journal of System Assurance Engineering and Management*.
* Kwan, C. C. Y. (2017). Shrinkage of the sample correlation matrix of returns towards a constant correlation target: A pedagogic illustration based on Dow Jones stock returns. *Spreadsheets in Education*, 10(1), 1–25.
* Kuhn, D., Shafiee, S., & Wiesemann, W. (2024). Distributionally Robust Optimization. *arXiv preprint* arXiv:2411.02549.
* Ledoit, O., & Wolf, M. (2004). A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2), 365–411.
* Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint* arXiv:1707.06347.
* Vaswani, A., et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.
* Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34, 22419–22431.
* Zhang, C., Zhang, Y., Cucuringu, M., & Qian, Z. (2024). Volatility forecasting with machine learning and intraday commonality. *Journal of Financial Econometrics*, 22(2), 492–530.