Pair Trading with Dynamic Portfolio Construction

Edward SIN Chi Man edwardsin0214@gmail.com

MSc Quantitative Finance
Department of Mathematics
National University of Singapore

Abstract

This paper presents a quantitative analysis of a statistical arbitrage pair trading strategy. Each pair of assets are constructed through a straight forward procedure, where the cointegrated relationship of each pair of assets are identified through the Augmented Dickey-Fuller (ADF) test. Dynamic hedge ratios between pairs are estimated using a Kalman filter to adapt to changing market conditions. We implement different portfolio allocation strategies including Equal Weight (EW), Equal Risk Contribution (ERC), and Maximum Sharpe Ratio (MSR) approaches. Additionally, the application of machine learning technique is implemented, such as Recurrent Reinforcement Learning Algorithm (RRL) and Long Short-Term Memory (LSTM) networks in Recurrent Neural Network (RNN) Algorithm, where they both aim to dynamically allocate portfolio weights over time. The proposed methodology is demonstrated through historical data of securities from the New York Stock Exchange (NYSE), and the performance is reviewed through Backtrader.

1 Introduction

Pairs trading is a market-neutral strategy that dates back to the mid-1980s when a group of technical analysts at Morgan Stanley first introduced the approach. It is designed to take advantage of price discrepancies between two securities by simultaneously taking a long position in the undervalued security and a short position in the overvalued one. The fundamental assumption behind pairs trading is that the relative mispricing between the two securities will eventually correct itself, allowing investors to profit from the convergence. This pair trade strategy, known as statistical arbitrage, is very famous among high-frequency trading firms, hedge funds and asset management companies in order to seek out potential market-neutral profits and generate alpha to compete against the market portfolio.

There are plenty of elements when implementing pair trading strategies, including formation of security pairs, number of units to buy for two securities, optimal trading signals and portfolio constructions. In section 1, we introduce how the set of candidate pairs are formed by cointegration approach. Then we determine dynamic hedge ratios between two securities (number of units to buy for the overvalued asset) by Kalman Filter method and optimize trading signal with the best standardized z-score thresholds. In section 2, we allocate our portfolio with two styles: Static Weighting and Dynamic Weighting. For Static Weighting we implement Equal Weight (EW), Portfolio Optimization strategies including Equal Risk Contribution (ERC) and Max Sharpe Ratio (MSR) methods. For Dynamic Weighting we propose adaptaion of Recurrent Reinforcement Learning Algorithm (RRL) and Long Short-Term Memory Algorithm (LSTM) in Recurrent Neural Networks (RNN) to provide dynamic weights for each pair. Finally, the performance of these strategies is reviewed through Backtrader.

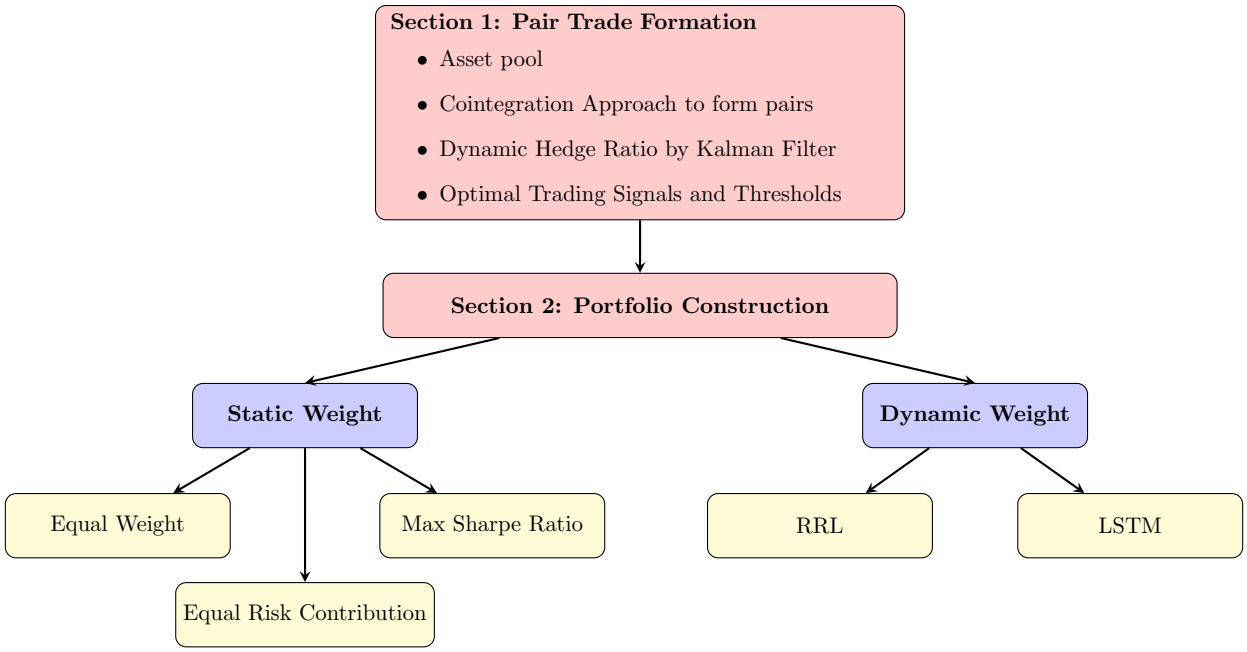

Figure 1: Flowchart of Pair Trade Mechanism

2 Pair Trade

2.1 Asset pool

Securities are selected from New York Stock Exchange (NYSE) to form a asset pool including equities, ETF, commodities, market index, spanning various sectors in energy and commodities, technology, semiconductor, healthcare and biotechnology, banking, fixed income, real estate, retails, etc. These assets have large market capitalization with high liquidity, which are optimal for pair trade since they have small bid-ask spread with lower transaction costs.

The historical data of these securities are collected from Yahoo Finance, starting from Jan 2013 to Dec 2023. The dataset was segmented into training set (80%: Jan 2013 to Oct 2021) and testing set (20%: Oct 2021 to Dec 2023).[2]

GDX	GLD	AAPL	GOOGL	META
AMD	NVDA	CSCO	ORCL	TTWO
EA	HYG	LQD	JNK	SLV
SIVR	USO	UWT	QQQ	SPY
VOO	VDE	VTI	EMLP	VDC
FSTA	KXI	IBB	VHT	VNQ
IYR	MSFT	PG	TMF	UPRO
WFC	JPM	GS	CVX	XOM
INTC	COST	WMT	T	VZ
CMCSA	AMZN			

Table 1: Asset pool

2.2 Cointegration Approach

Two securities x_t and y_t are formed as a pair trade if they are cointegrated. If the linear combination of two time series is tested to be stationary, which implies that the $spread_t$ between two securities prices has mean-reverting property back to a constant mean, then they are said to be cointegrated. Here are the detailed procedures:

- Determine the absolute price spread $spread_t = y_t - (\beta x_t)$.
- Implement Engle and Granger method [1] by using Augmented-Dickey Fullers (ADF) test to determine whether $spread_t$ is stationary.
- If P-value for ADF test is less than critical value = 0.0015, we can conclude that two time series x_t and y_t are cointegrated to form a pair trade.

We formed 4 cointegrated according the rules above.

Pair #	**Asset 1**	**Asset 2**
1	KXI	CMCSA
2	ORCL	GOOGL
3	UPRO	ORCL
4	VDC	CMCSA

Table 2: List of Cointegrated Pairs

2.3 Dynamic Hedge Ratio by Kalman Filter

After forming pair trades, we use β units of x_t to hedge against y_t, where β is also known as the hedge ratio. We can simply find the hedge ratio by conducting simple linear regression.

$$y_t = \beta x_t$$

However, since market conditions are constantly changing, a static hedge ratio may not be optimal in the face of shifting market dynamics. A more adaptive approach is to estimate a dynamic hedge ratio β_t, which can evolve in response to new information.

Kalman Filter is a recursive algorithm that update and predict the unknown estimates in each time steps by incorporating new data as they become available. It dynamically adjusts β_t to better reflect current market conditions for two stocks hedging against each other.

At $t = 0$, given known historical security prices x_0 and y_0, we initialize hedge ratio $\hat{\beta}_0$ and variance of initial hedge ratio Σ_0 from the estimated parameter and the variance of estimated parameter of simple linear regression model $y_t = \hat{\beta}_0 x_t$ respectively. Then, we predict the unknown $\hat{\beta}_{1|0}$ and $\Sigma_{1|0}$. After inital setup, Kalman Filter recursively updates and predicts the unknown parameters at each time step when new securities prices are available.

To dive into the mathematics, we identify state equation and measurement equation as below[5]:

$$\textbf{State Equation:} \quad \beta_t = \beta_{t-1} + \epsilon_t \qquad \text{where} \quad \epsilon_t \sim N(0, V_\epsilon)$$
$$\textbf{Measurement Equation:} \quad y_t = \beta_t x_t + \varepsilon_t \qquad \text{where} \quad \varepsilon_t \sim N(0, V_\varepsilon)$$

where ϵ_t and ε_t are both Guassian noises with variance V_ϵ and V_ε respectively.

- **Predicted variables from previous time step:**
  We estimate state estimates at time t, based on the previous time step:

$$\textbf{State Variable:} \quad \hat{\beta}_{t|t-1} = \hat{\beta}_{t-1|t-1},$$
$$\textbf{Covariance of State Variable:} \quad \Sigma_{t|t-1} = \Sigma_{t-1|t-1} + V_\epsilon$$

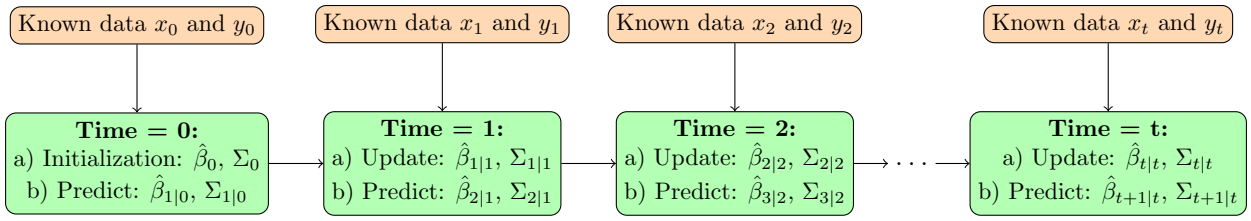

Figure 2: Flow of Kalman Filter

- **Update Step:**
  We refine the predicted state estimates after acquiring the new observations x_t and y_t.
  - **Compute Kalman Gain K_t:**
$$K_t = \frac{\Sigma_{t|t-1}x_t}{x_t^\top \Sigma_{t|t-1}x_t + V_\varepsilon}$$
  - **Update $\hat{\beta}_{t|t}$ and Update $\Sigma_{t|t}$:**
$$\textbf{State Variable:} \quad \hat{\beta}_{t|t} = \hat{\beta}_{t|t-1} + K_t\left(y_t - x_t\hat{\beta}_{t|t-1}\right),$$
$$\textbf{Covariance of State Variable:} \quad \Sigma_{t|t} = \Sigma_{t|t-1} - K_t x_t \Sigma_{t|t-1}$$
- **Prediction step for the next time step:**
  After updating the estimates, we predict state estimates for the next time step:
$$\textbf{State Variable:} \quad \hat{\beta}_{t+1|t} = \hat{\beta}_{t|t},$$
$$\textbf{Covariance of State Variable:} \quad \Sigma_{t+1|t} = \Sigma_{t|t} + V_\epsilon$$

2.4 Trading Signals and Optimal Trading Thresholds

To determine when to buy or sell a pair of stocks, we first need to calculate $spread_t$ after obtaining the dynamic hedge ratio β_t as described in section 2.3. The pair trading strategy opens a position when the standardized spread z_t exceeds the optimized entry Z-score and closes the position when the spread reverts to the optimized exit Z-score as described below:

$$spread_t = y_t - \beta_t x_t$$
$$z_t = \frac{spread_t - \mathbf{E}[spread_t]}{\sqrt{Var(spread_t)}}$$

Signals	Condition		
Long Entry	$z_{t-1} >$ long_entry_Zscore	and	$z_t <$ long_entry_Zscore
Long Exit	$z_{t-1} <$ long_exit_Zscore	and	$z_t >$ long_exit_Zscore
Short Entry	$z_{t-1} <$ short_entry_Zscore	and	$z_t >$ short_entry_Zscore
Short Exit	$z_{t-1} >$ short_exit_Zscore	and	$z_t <$ short_exit_Zscore

Table 3: Trading Signal Conditions

The optimal signals are identified by evaluating all combinations of Z-scores within a defined parameter grid which consists of 100 evenly spaced values for both the long entry, long exit, short entry and short exit Z-scores.

$$param\_grid = \begin{Bmatrix} long\_entry\_Zscore : \mathcal{L}(-3, 0, 16), & long\_exit\_Zscore : \mathcal{L}(-3, 0, 16), \\ short\_entry\_Zscore : \mathcal{L}(0, 3, 16), & short\_exit\_Zscore : \mathcal{L}(0, 3, 16) \end{Bmatrix}$$

where $\mathcal{L}(a, b, n)$ denotes a linearly spaced vector of n values between a and b.

We select combinations of Z-scores which generates the most cumulative cash gain for each cointegrated stock pair during training set as described below:

Pair	Asset 1	Asset 2	Optimal Z-scores (Long Entry, Long Exit, Short Entry, Short Exit)
1	KXI	CMCSA	(-2.4, -0.4, 0.4, 0.2)
2	ORCL	GOOGL	(-2.4, -0.0, 1.4, 1.2)
3	UPRO	ORCL	(-2.4, -1.0, 0.6, 0.4)
4	VDC	CMCSA	(-2.2, -0.2, 0.2, 0.0)

Table 4: Optimal Z-scores for Cointegrated Pairs

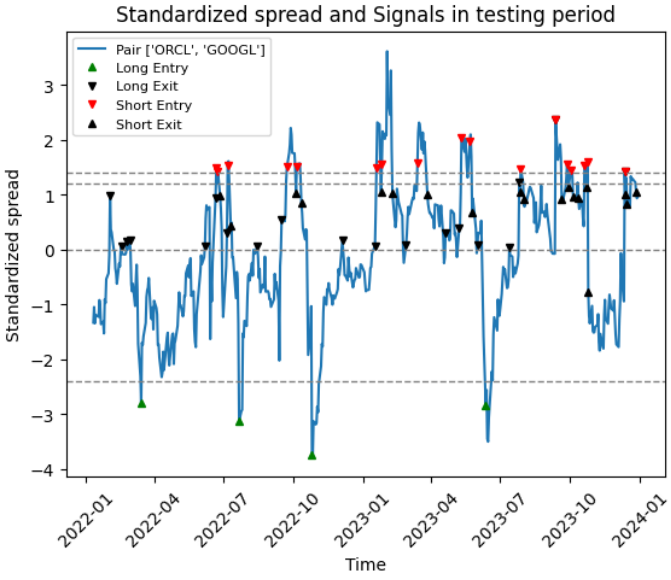

Figure 3: Illustration of indicating trading signals with pair ORCL vs GOOGL in testing set.

3 Portfolio Construction

3.1 Equal Weight

The Equal Weight portfolio construction method allocates an equal proportion of capital to each pair of assets. Let $N = 4$ denote the total number of cointegrated pairs in the portfolio. The weight assigned to each pair, $\mathbf{w}$, is given by:

$$\mathbf{w} = \left[\frac{1}{N}, \frac{1}{N}, \frac{1}{N}, \frac{1}{N}\right] = \left[\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right]$$

This approach does not take into account the risk or expected return of each pair but simply ensures that the capital is distributed evenly across all pairs, which is also used as a benchmark for more sophisticated weighting schemes in the later part.

3.2 Equal Risk Contribution

The Equal Risk Contribution (ERC) portfolio aims to allocate capital in such a way that each asset contributes equally to the overall portfolio risk. Let Σ denote the covariance matrix of historical returns for the cointegrated pairs, and $\mathbf{w}$ be the vector of portfolio weights.

The portfolio risk σ_p is given by:

$$\sigma_p = \mathbf{w}^\top \Sigma \mathbf{w}$$

The risk contribution of pair i is given by:

$$RC_i = w_i \cdot (\Sigma \mathbf{w})_i$$

where $(\Sigma \mathbf{w})_i$ represents the marginal risk of the i-th pair.

In the ERC approach, we solve for weights $\mathbf{w}_{ERC}$ such that the risk contributions are equal across all pairs by minimizing Objective Function:

$$\textbf{Objective Function:} \quad \mathbf{w}_{ERC} = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \left(RC_i - \frac{\sigma_p}{n} \right)^2$$

$$\text{subject to} \quad \begin{cases} w_i \cdot (\Sigma \mathbf{w})_i = w_j \cdot (\Sigma \mathbf{w})_j \,; \\ \sum_{i=1}^{N} w_i = 1 \quad \text{and} \quad w_i \geq 0 \end{cases} \qquad \forall i, j = 1, 2, \ldots, N$$

3.3 Maximum Sharpe Ratio

The Maximum Sharpe Ratio (MSR) portfolio construction method seeks to maximize the Sharpe ratio, which is defined as:

$$\text{Sharpe Ratio} = \frac{\mathbf{E}[r_p] - r_f}{\sigma_p}$$

where r_p is the portfolio return and r_f is the risk-free rate.

To find the weights that maximize the Sharpe ratio, we solve the following optimization of Obejective Function:

$$\textbf{Objective Function:} \quad \mathbf{w}_{MSR} = \underset{\mathbf{w}}{\operatorname{argmax}} \frac{\mathbf{w}^\top \mathbf{E}[r] - r_f}{\sqrt{\mathbf{w}^\top \Sigma \mathbf{w}}}$$

$$\text{subject to} \quad \sum_{i=1}^{N} w_i = 1 \quad \text{and} \quad w_i \geq 0 \qquad \forall i = 1, 2, \ldots, N$$

3.4 Dynamic Weights with Recurrent Reinforcement Learning[3]

Apart from using static weighting, we also construct a portfolio with weights changing over time. Here we use Recurrent Reinforcement Learning (RRL) to dynamically adjust the portfolio weightings for portfolio allocation. It allows the model to capture temporal patterns and make more informed decisions for trade executions of all pairs of securities.

The model is divided into two main components: Forward Pass and Backpropagation. In the Forward Pass, the algorithm begins by feeding the input return vector into the model. The portfolio weighting parameters are initialized and passed through a Softmax function to ensure that the sum of all portfolio weights is constrained to 1. Once the portfolio weightings are computed, sharpe ratio is calculated based on the portfolio returns.

In the Backpropagation phase, the model parameters are updated by maximizing sharpe ratio through gradient ascent. By adjusting the weighting parameters, the model learns to dynamically optimize the portfolio allocations, improving its ability to respond to changing market conditions.

3.4.1 Forward Feed

For each asset pair $p \in [1, 4]$, we have input vectors $x_{t-1}^{(p)}$ which includes spread returns $r_t^{(p)}$ with rolling-window periods of $m = 60$ as well as the previous portfolio weighting $F_{t-1}^{(p)}$. Additionally, we have parameter vectors $\theta_t^{(p)}$, which weight the inputs to construct a linear output $z_t^{(p)}$ as follows:

$$\textbf{Inputs:} \qquad x_t^{(p)} = [1, r_t^{(p)}, r_{t-1}^{(p)}, \ldots, r_{t-m}^{(p)}, F_{t-1}^{(p)}],$$

$$\theta_t^{(p)} = [\theta_{(t,0)}^{(p)}, \theta_{(t,1)}^{(p)}, \theta_{(t,2)}^{(p)}, \ldots, \theta_{(t,m+1)}^{(p)}, \theta_{(t,m+2)}^{(p)}],$$

$$\textbf{Linear output:} \qquad \begin{aligned} z_t^{(p)} &= x_t^{(p)} \cdot \theta_t^{(p)} \\ &= \theta_{(t,0)}^{(p)} + \theta_{(t,1)}^{(p)} r_t^{(p)} + \theta_{(t,2)}^{(p)} r_{t-1}^{(p)} + \cdots + \theta_{(t,m+1)}^{(p)} r_{t-m}^{(p)} + \theta_{(t,m+2)}^{(p)} F_{t-1}^{(p)} \end{aligned}$$

The Softmax activation function is then computed to determine the portfolio weighting $F_t^{(p)}$ to ensure that weightings across all asset pairs sum to 1.

$$\textbf{Weightings:} \qquad F_t^{(p)} = Softmax(z_t^{(p)}) = \frac{e^{z_t^{(p)}}}{\sum_{k=1}^{pair=4} e^{z_t^{(k)}}} \quad \text{where} \quad \sum_{k=1}^{pair=4} F_t^{(k)} = \sum_{k=1}^{pair=4} e^{z_t^{(k)}} = 1$$

Once the portfolio weightings $F_t^{(p)}$ are determined for each asset pair, the overall portfolio return for time t is computed as:

$$\textbf{Portfolio Return:} \qquad R_t = \sum_{p=1}^{4} F_t^{(p)} r_t^{(p)},$$

To evaluate the performance of the portfolio, the Sharpe Ratio S_n is computed as:

$$\textbf{Sharpe Ratio:} \qquad S_t = \frac{mean(R_t)}{std(R_t)} = \frac{A_m}{K_m \sqrt{B_m - A_m^2}},$$

where:

$$A_m = \frac{1}{m} \sum_{t=1}^{m} R_t, \quad B_m = \frac{1}{m} \sum_{t=1}^{m} R_t^2, \quad K_m = \sqrt{\frac{m}{m-1}}.$$

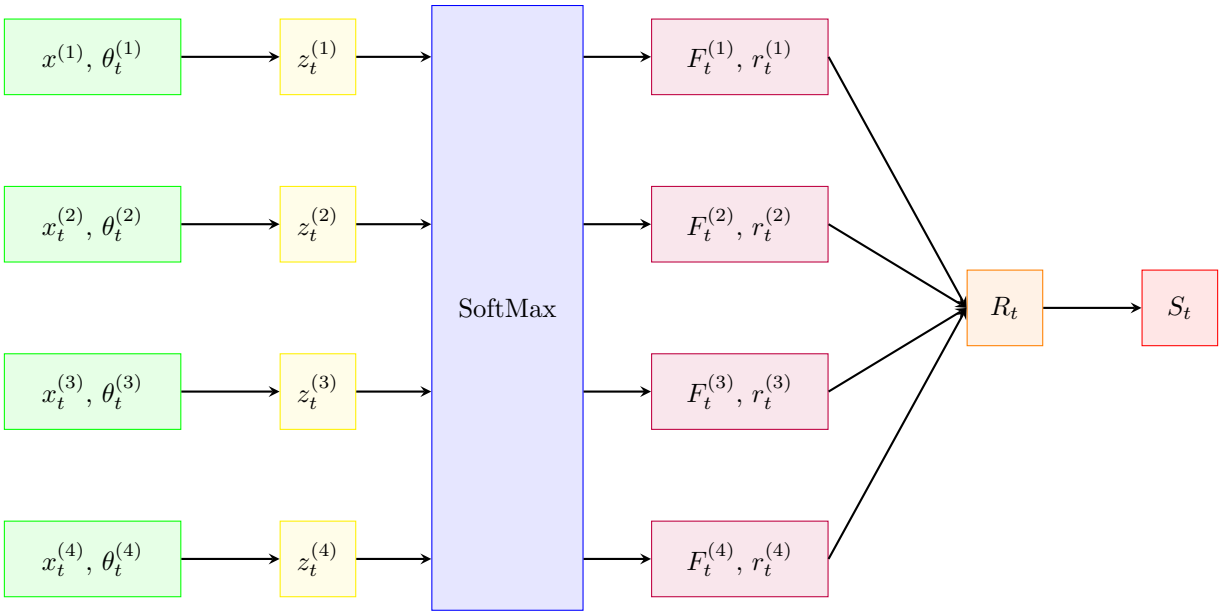

Figure 4: Forward Feed Mechanism

3.4.2 Backpropagation

In order to maximize the Sharpe Ratio, we implement Gradient Ascent method to update weight parameters $\theta_t^{(p)}$ in each iteration of Backpropagation alorithm in order to adjust portfolio weightings F_t to each pair of securities and optimize Sharpe Ratio at latest time T: $S_T(\theta_t^{(p)})$. In each iteration, we update $\theta_t^{(p)}$ by the general algorithm as follows:

$$\begin{aligned}\theta_t^{(p)} &\leftarrow \theta_t^{(p)} + \Delta\theta_t^{(p)} \\ &= \theta_t^{(p)} + \rho \frac{\boldsymbol{dS_T(\theta_t^{(p)})}}{\boldsymbol{d\theta_t^{(p)}}}\end{aligned}$$

where ρ is the constant learning rate of Gradient Ascent.

We continue to loop through each iteration of Backpropagation to update $\theta_t^{(p)}$ until the gradient of $S_T(\theta_t^{(p)})$ with respect to $\theta_t^{(p)}$ is minimized, which means portfolio Sharpe ratio at time T is maximized.

Let start the Backpropagation mechanism with the equation of Sharpe ratio function:

$$\textbf{Sharpe Ratio:} \qquad S_T(\theta_t^{(p)}) = \frac{mean(R_t)}{std(R_t)} = \frac{A_m}{K_m\sqrt{B_m - A_m^2}},$$

where:

$$A_m = \frac{1}{m}\sum_{t=1}^{m} R_t, \quad B_m = \frac{1}{m}\sum_{t=1}^{m} R_t^2, \quad K_m = \sqrt{\frac{m}{m-1}}.$$

To compute the gradient of $S_T(\theta_t^{(p)})$ with respect to $\theta_t^{(p)}$, we apply Chain Rule to calculate gradients of

$S_T(\theta_t^{(p)})$ with respect to A_T and B_T.

$$\begin{aligned}\frac{\boldsymbol{dS_T(\theta_t^{(p)})}}{\boldsymbol{d\theta_t^{(p)}}} &= \sum_{t=1}^{T}\left(\frac{dS_T}{dA_T}\frac{dA_T}{dR_t}+\frac{dS_T}{dB_T}\frac{dB_T}{dR_t}\right)\frac{dR_t}{d\theta_t^{(p)}} \\ &= \frac{1}{T}\sum_{t=1}^{T}\left(\frac{B_T - A_T R_t}{K_T(B_T - A_T^2)^{3/2}}\right)\frac{dR_t}{d\theta_t^{(p)}}\end{aligned}$$

Then we continue to compute the derivative of R_t with respect to $\theta_t^{(p)}$ by Chain Rule again:

$$\frac{dR_t}{d\theta_t^{(p)}} = \frac{\partial}{\partial\theta_t^{(p)}}\sum_{k=1}^{pair=4} F_t^{(k)} r_t^{(k)} = \sum_{k=1}^{pair=4} r_t^{(k)}\frac{\partial F_t^{(k)}}{\partial z_t^{(p)}}\frac{\partial z_t^{(p)}}{\partial\theta_t^{(p)}}$$

Since F_t^p is a SoftMax function, the partial derivative of F_t^p with respect to $z_t^{(p)}$ is:

$$\frac{\partial F_t^{(i)}}{\partial z_t^{(p)}} = \begin{cases} F_t^{(i)}(1-F_t^{(j)}), & \text{for } i=j \\ -F_t^{(i)}F_t^{(j)}, & \text{for } i\neq j\end{cases}$$

Given the partial derivative of F_t^p with respect to $z_t^{(p)}$, we expand the derivative of R_t with respect to $\theta_t^{(p)}$:

$$\frac{dR_t}{d\theta_t^{(p)}} = r_t^{(1)}\left(-F_t^{(1)}F_t^{(p)}\right)\frac{\partial z_t^{(p)}}{\partial\theta_t^{(p)}} + ... + r_t^{(p)}\left(F_t^{(p)}\left(1-F_t^{(p)}\right)\right)\frac{\partial z_t^{(p)}}{\partial\theta_t^{(p)}} + ... + r_t^{(4)}\left(-F_t^{(4)}F_t^{(p)}\right)\frac{\partial z_t^{(p)}}{\partial\theta_t^{(p)}}$$

As $z_t^{(p)}$ depends on the previous weighting $F_{t-1}^{(p)}$, the partial derivative of $z_t^{(p)}$ respect to $\theta_t^{(p)}$ is as follows:

$$\frac{\partial z_t^{(p)}}{\partial\theta_t^{(p)}} = x_t^{(p)} + \theta_{(t,m+2)}^{(p)}\frac{\partial F_{t-1}^{(p)}}{\partial\theta_t^{(p)}}$$

The partial derivative of $F_{t-1}^{(p)}$ with respect to $\theta_t^{(p)}$ is:

$$\frac{\partial F_{t-1}^{(p)}}{\partial\theta_t^{(p)}} = F_{t-1}^{(p)}(1-F_{t-1}^{(p)})\frac{\partial z_{(t-1)}^{(p)}}{\partial\theta_t^{(p)}}$$

To find out whether the gradient is minimized after updating $\boldsymbol{\theta}_t$ for all pairs $p\in[1,4]$, we calculated the Euclidean Norm of the gradient of $S_T(\theta_t^{(p)})$ with respect to $\theta_t^{(p)}$:

$$\|\nabla S_T(\boldsymbol{\theta}_t)\|_2 = \sqrt{\begin{array}{c}\left[\left(\frac{\partial S_T}{\partial\theta_{(t,0)}^{(1)}}\right)^2+\left(\frac{\partial S_T}{\partial\theta_{(t,1)}^{(1)}}\right)^2+\cdots+\left(\frac{\partial S_T}{\partial\theta_{(t,m+2)}^{(1)}}\right)^2\right] \\ +\cdots+ \\ \left[\left(\frac{\partial S_T}{\partial\theta_{(t,0)}^{(4)}}\right)^2+\left(\frac{\partial S_T}{\partial\theta_{(t,1)}^{(4)}}\right)^2+\cdots+\left(\frac{\partial S_T}{\partial\theta_{(t,m+2)}^{(4)}}\right)^2\right]\end{array}}$$

If the norm is less than tolerance level (0.00001), we conclude that the sharpe ratio is maximized with updated $\theta_t^{(p)}$, then we can use it to update portfolio weightings F_t. If not, iterate the whole process again until it reaches the level.

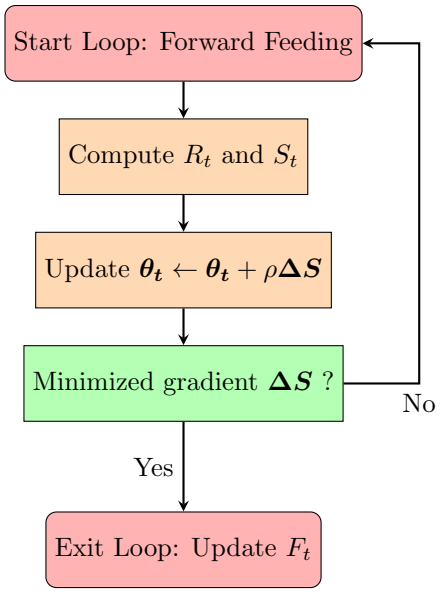

Figure 5: Forward Feeding and Backpropagation at each time t for all pairs $p \in [1, 4]$

3.5 Dynamic Weights with LSTM

Additionally, we enhance the pair trading strategy by incorporating a Long Short-Term Memory (LSTM) network. LSTM is an advanced form of Recurrent Neural Networks (RNN) that can capture long-short term dependencies in sequential data. This makes it well-suited for predicting dynamic portfolio weights based on the historical rolling return series.

LSTM operates through three main gates:

- **Forget Gate:** Allows the model to forget and discard previous irrelevant information (including cell state (Long term memory) C_{t-1} and hidden state (Short term memory) H_{t-1} as described in the diagram).
- **Input Gate:** Updates current cell state C_t and hidden state H_t.
- **Output Gate:** Determines what part of information should be output to the next timestep, preserving temporal dependencies.

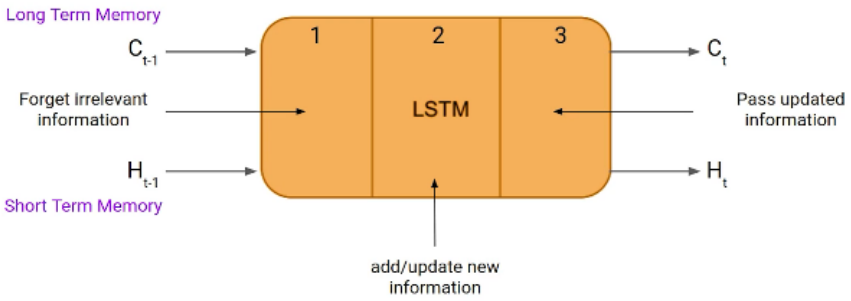

Figure 6: LSTM structure showing Forget Gate(1), Input Gate(2), and Output Gate(3).[4]

The LSTM network for dynamic portfolio weight prediction is implemented using `tensorflow` and consists of the following layers:

- **Input Layer:** A sequence of returns with rolling window of 60: $\mathbf{r_t}$ is fed into the model.
- **LSTM Hidden Layers:** Two LSTM layers (with 128 units of neurons and 64 batches each) are used to process the sequence of rolling returns $\mathbf{r}_t$ and capture the temporal dependencies in the data.
- **Dropout Layer:** A dropout layer with 30% dropout rate is applied to prevent overfitting by randomly setting units to zero during training.
- **Output Layer:** The final dense layer outputs the portfolio weights $\mathbf{w}_t$. A `softmax` after a `relu` activation function is used to ensure that the portfolio weights are non-negative and sum up to 1.

The LSTM modelis trained to predict portfolio weights $\mathbf{w}_t$ that maximize the Sharpe ratio by using the Adam optimizer with a learning rate of 0.0001.

4 Performance review with BackTrader

Table 5: Backtrader Results for Different Strategies

	Final Value($)	**CAGR**	**Sharpe Ratio**	**VaR 95%**	**VaR 99%**	**Annualized Volatility**
EW	1,487,274	+21.95%	1.45	-1.01%	-1.98%	13.01%
RP	1,395,247	+18.12%	1.27	-0.97%	-2.14%	12.39%
MSR	1,341,964	+15.84%	0.83	-1.42%	-2.47%	17.63%
RRL	1,819,245	+34.88%	0.97	-1.92%	-4.33%	29.62%
LSTM	1,602,371	+26.58%	1.50	-1.00%	-2.28%	13.46%
DJI	1,066,098	+3.25%	0.20	-1.60%	-2.75%	15.72%
SPY	1,051,924	+2.56%	0.16	-1.93%	-3.34%	19.06%
NASDAQ	995,823	-0.21%	0.08	-2.51%	-4.10%	25.05%

All strategies start with initial value of $ 1,000,000 and transaction cost is set at 0.01% for each trade. We compare five pair trade weighting strategies against buying and holding market indices such as DJI, S&P 500 and NASDAQ during the backtesting period from Oct 2021 to Dec 2023.

All weighting strategies beat market indices with much higher sharpe ratio (+0.63 to +1.42) and CAGR (+13% to +35%). For static weighting approaches, Equal Weight (EW) performed best among the static strategies, achieving a solid sharpe ratio of 1.45 and CAGR of 21.95%. Risk Parity (RP) followed closely, slightly reduced sharpe ratio of 1.27 and return at 18.12% CAGR but balancing risk well with a lower volatility of 12.39%. For Maximum Sharpe Ratio (MSR), while designed to optimize risk-adjusted returns, underperformed with a sharpe ratio of 0.83 and with the highest volatility of 17.63% among the static models. The unexpected underperformance from MSR model is likely due to market volatile movement in post-COVID period, causing over-concentration in riskier assets during the testing period.

The dynamic weighting strategies, Recurrent Reinforcement Learning (RRL) and Long Short-Term Memory (LSTM), both outperformed static weighting strategies by adapting to dynamic market environment. RRL model delivered the highest CAGR of 34.88%, although it came with significant risk with volatility of 29.62%. On the other hand, LSTM model achieves the highest sharpe ratio of 1.50 with a CAGR of 26.58% and relatively low volatility at 13.46%. This demonstrates the LSTM's ability to capitalize on market trends while effectively managing risk compared to the more aggressive RRL approach.

5 Conclusion

This project implemented pair trading strategies using both static and dynamic portfolio construction methods. Through the use of cointegration tests and Kalman filters, we identified pairs of securities and dynamically adjusted hedge ratios to form candidate pairs. The performance of static weighting strategies like Equal Weight (EW), Risk Parity (RP), and Maximum Sharpe Ratio (MSR) was compared to more advanced dynamic methods, including Recurrent Reinforcement Learning (RRL) and Long Short-Term Memory (LSTM) networks. The results showed that dynamic models, especially LSTM, outperformed static weighting strategies by capturing market trends and adjusting portfolio weights in response to market shifts. While RRL delivered the highest returns, it also came with higher risk, whereas LSTM demonstrated a better balance between risk and reward, achieving the highest sharpe ratio.

While the project yielded promising results, there is room for improvement. Extending the training and testing period beyond 2013 to 2023 could potentially enhance the robustness of the models, especially for dynamic strategies like RRL and LSTM that require extensive data to train effectively. Additionally, There are some ways to improve the optimization of hyperparameters in machine learning models for both RRL and LSTM models, such as `CrossValidation`, `GridSearch` and `RandomSearch` functions. However, due to the computational intensity of these models, extending the data period and performing hyperparameter optimization require significantly more time and GPU power. Future work could focus on scaling up computational resources to fine-tune these models further and explore additional deep learning architectures for even better performance.

References

[1] Robert F Engle and Clive WJ Granger. Co-integration and error correction: representation, estimation, and testing. *Econometrica: journal of the Econometric Society*, pages 251–276, 1987.

[2] Goldinlocks. Pairs trading with a kalman filter. `https://goldinlocks.github.io/PAIRS-TRADING-WITH-A-KALMAN-FILTER/#PAIRS-TRADING-WITH-A-KALMAN-FILTER`.

[3] Franco Ho Ting Lin. Dynamic asset allocation for pairs trading, 2018.

[4] Analytics Vidhya. What is lstm? introduction to long short-term memory. `https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/`.

[5] Jia Yu. Cointegration approach for the pair trading based on the kalman filter. In *2022 2nd International Conference on Business Administration and Data Science (BADS 2022)*, pages 633–642. Atlantis Press, 2022.

6 Appendix

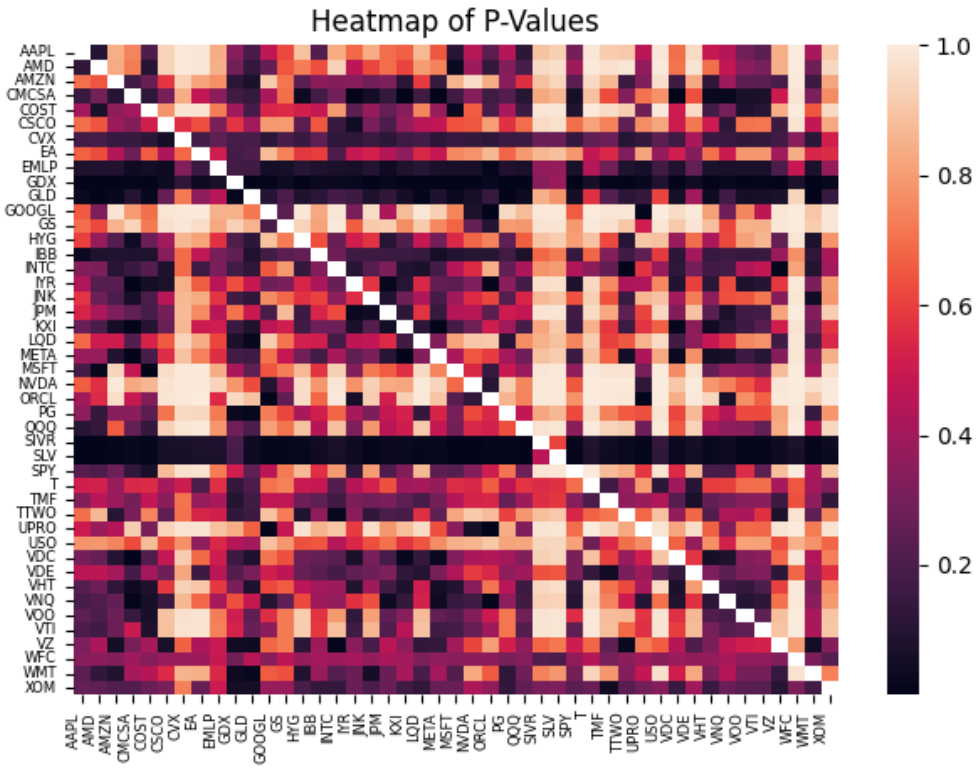

Figure 7: P-value of cointegration test

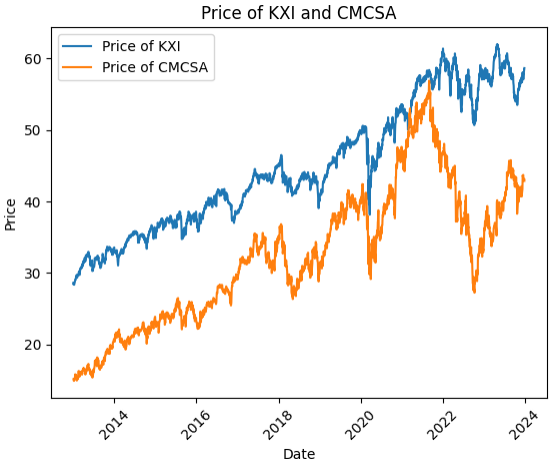

Figure 8: Price of KXI and CMCSA

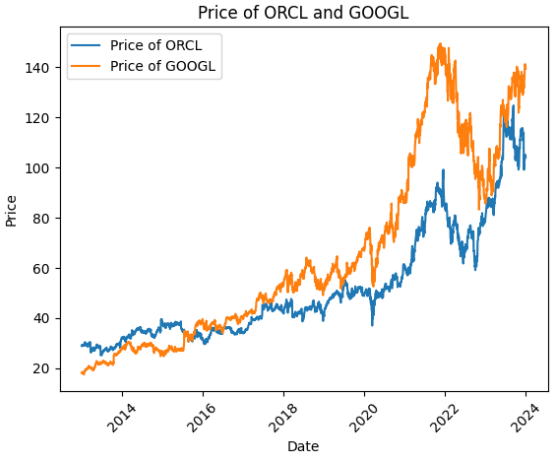

Figure 9: Price of ORCL and GOOGL

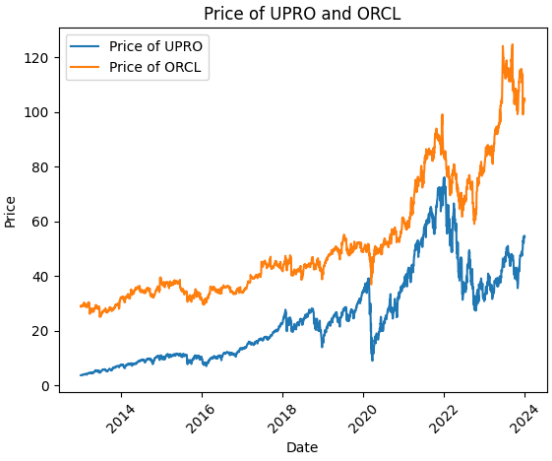

Figure 10: Price of for UPRO and ORCL

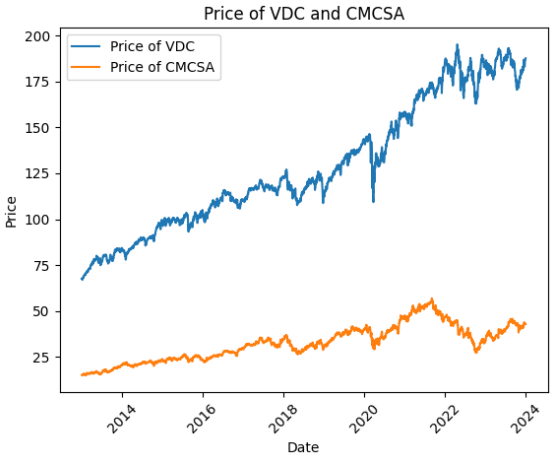

Figure 11: Price of VDC and CMCSA

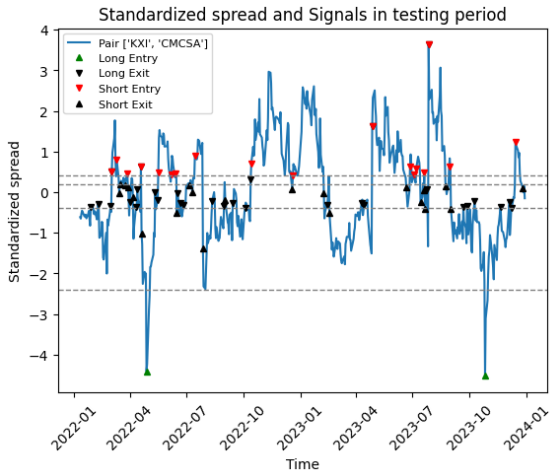

Figure 12: Spread of KXI vs CMCSA

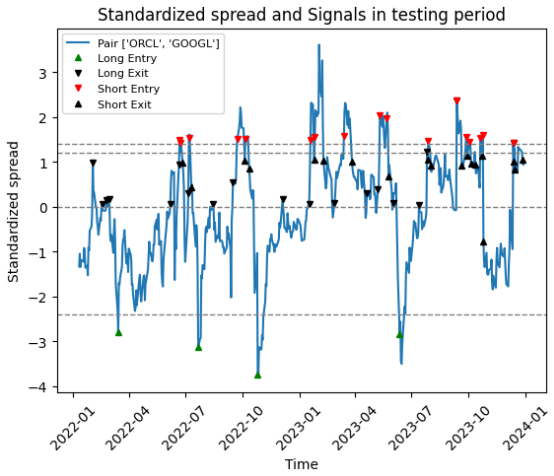

Figure 13: Spread of ORCL vs GOOGL

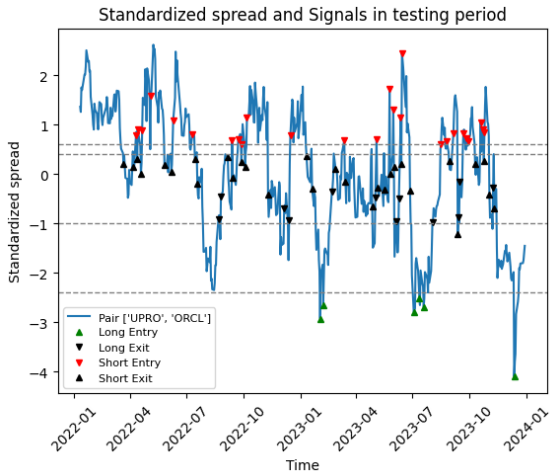

Figure 14: Spread of for UPRO vs ORCL

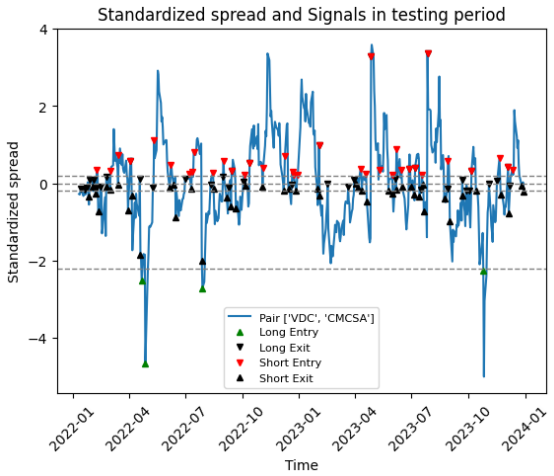

Figure 15: Spread of VDC vs CMCSA

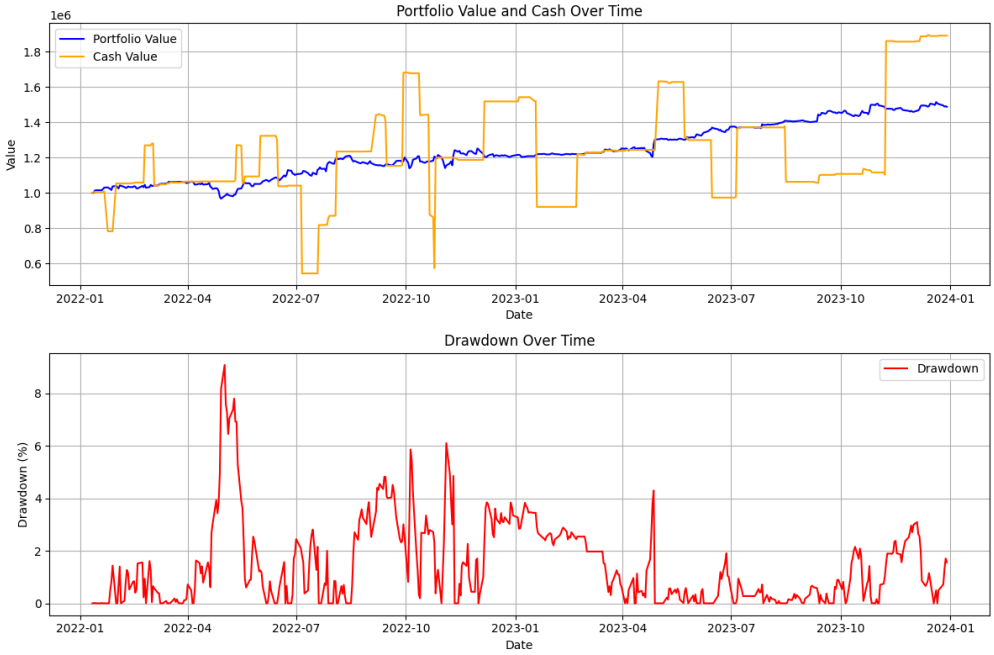

Figure 16: Equal Weight Portfolio Backtesting

Figure 17: Risk Parity Portfolio Backtesting

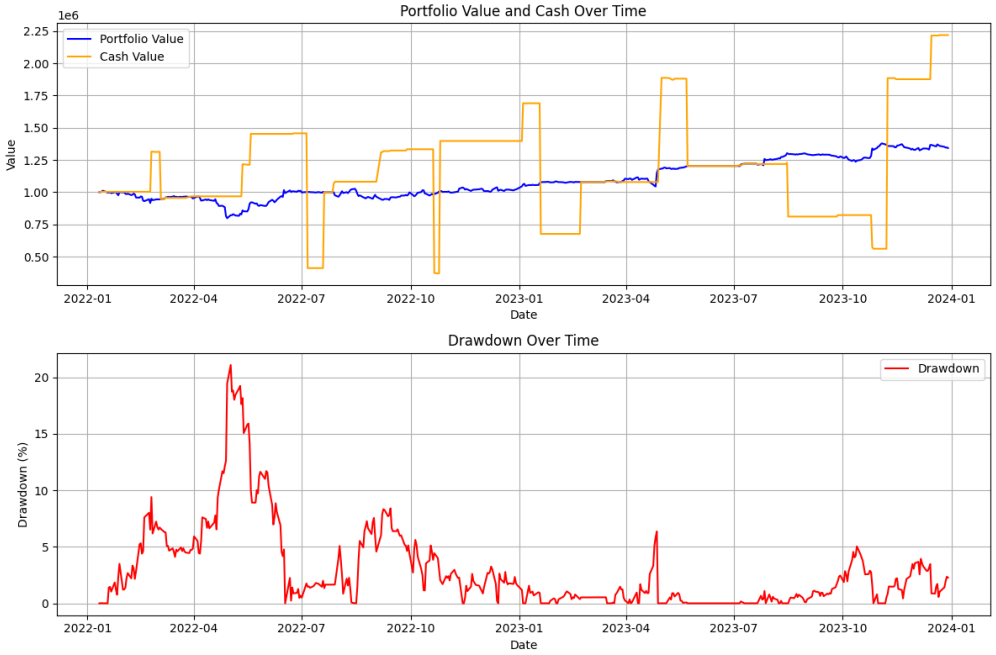

Figure 18: Max Sharpe Ratio Portfolio Backtesting

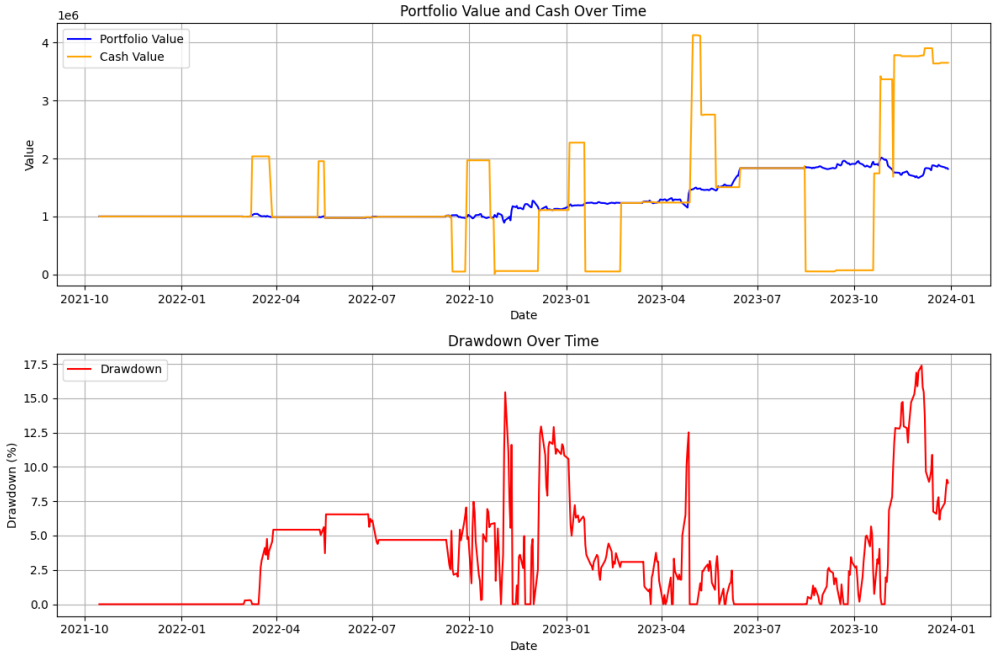

Figure 19: Recurrent Reinforcement Learning Portfolio Backtesting

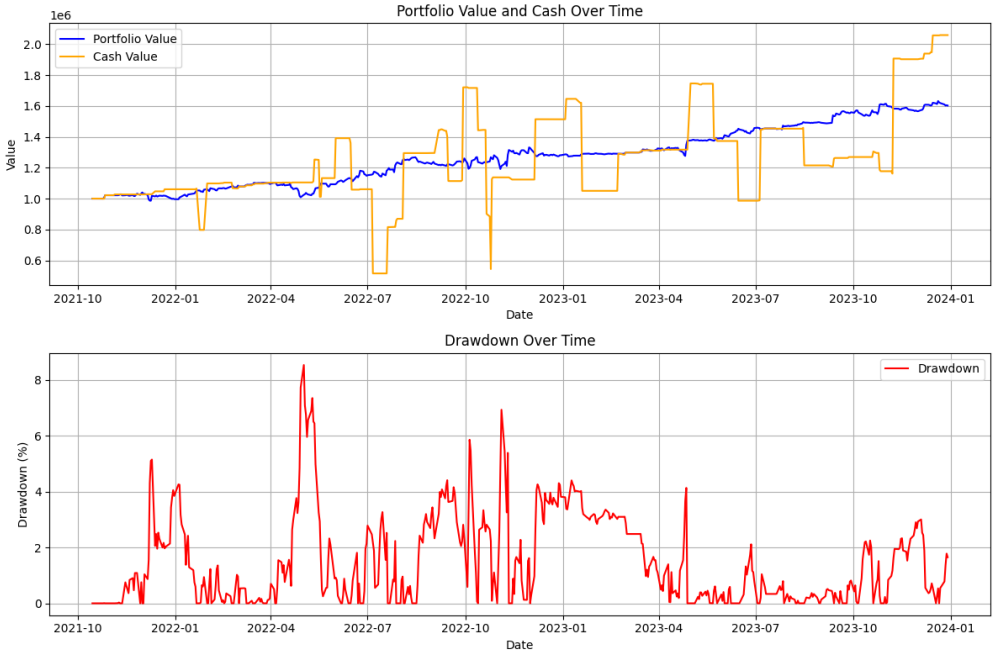

Figure 20: Long Short-Term Memory Portfolio Backtesting

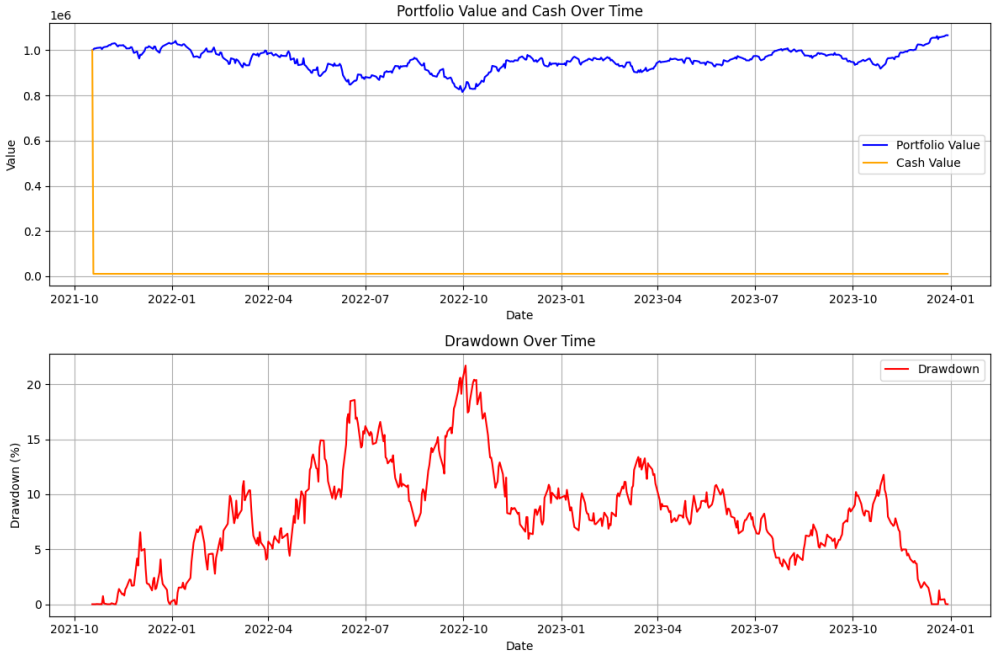

Figure 21: Buy and Hold DJI Backtesting

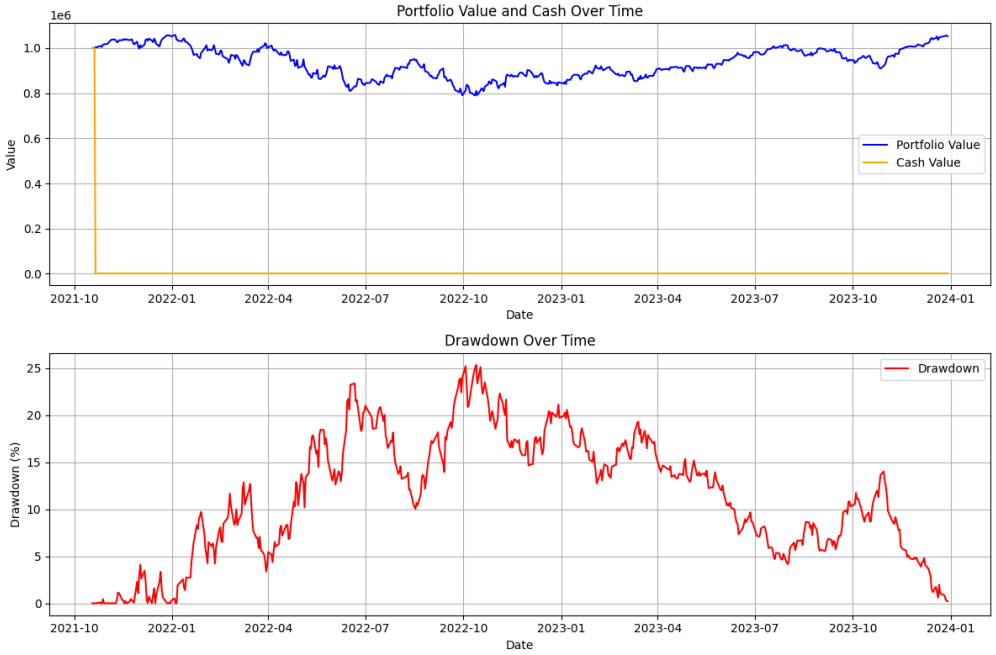

Figure 22: Buy and Hold SPY Backtesting

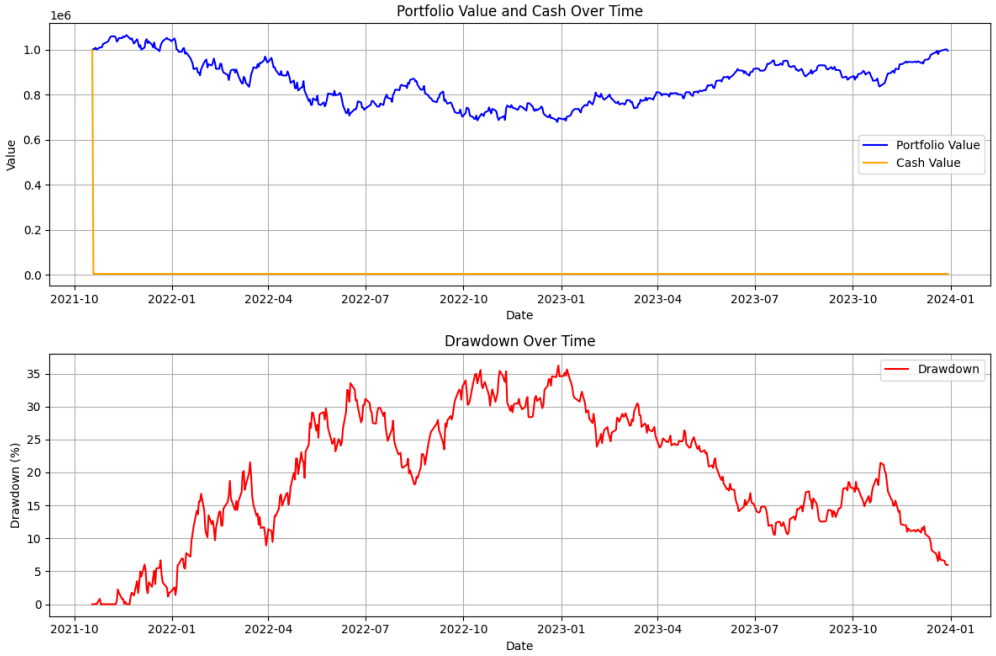

Figure 23: Buy and Hold NASDAQ Backtesting