

## 1 Overview of Pairs Trading Models

I asked GPT 5 to suggest 10 DDQN models to apply to pairs trading exercises via the following prompt:

I have the following dataset of open stock prices (solely equities, no derivatives, futures, or macro instruments). Note that I don't have access to any other data other than this. Please suggest 10 DDQN models to apply to pairs trading on this dataset. The output should be formatted as a table with the following columns:

- \* Model name (description)
- \* States
- \* Actions
- \* Reward

Please render the output and output the LaTeX code to generate the table. Keep all descriptions concise such that the table fits on a page.

After prompting the LLM to export the results to  $\text{\LaTeX}$ , it yielded the following result:

Model name (description)	States	Actions	Reward
Price Spread DDQN (uses raw spread)	Current spread, z-score	Long pair, short pair, flat	$\Delta\text{PNL} - \text{cost}$
Normalized Spread DDQN (scaled inputs)	Normalized spread, rolling mean/var	Long, short, flat	$\Delta\text{Portfolio value}$
Regime-Switch DDQN (spread + regime flag)	Spread, z-score, regime indicator	Long, short, flat	Risk-adjusted $\Delta\text{PNL}$
Volatility-Aware DDQN	Spread, rolling volatility	Long, short, flat	$\Delta\text{PNL}/\text{vol}$
Mean-Reversion Signal DDQN	Spread, residual from OLS	Long, short, flat	Sharpe-scaled $\Delta\text{PNL}$
Multi-Horizon Spread DDQN	Spreads across multiple windows	Long, short, flat	Sum of $\Delta\text{PNL}$
Threshold DDQN	Spread, distance from thresholds	Enter long, enter short, exit, hold	$\Delta\text{PNL} - \text{penalty}$
Spread Momentum DDQN	Spread change, momentum indicator	Long, short, flat	$\Delta\text{PNL} + \text{momentum bonus}$
Rolling Beta DDQN	Spread, rolling beta estimate	Long, short, flat	Hedge-adjusted $\Delta\text{PNL}$
Multi-Asset DDQN (pair-context)	Spread, pair identity embedding	Long, short, flat	Pair-specific $\Delta\text{PNL}$

Table 1: DDQN model designs for pairs trading using only open prices.

## 2 Model Deep Dive: Spread Momentum DDQN

While momentum is a concept heavily discussed in financial literature, its applications in pairs trading are not so well-documented. I prompted GPT 5 to explain this strategy in greater detail:

I'd like to learn more about the spread momentum DDQN model

Below is my own summary of what GPT 5 presented.

As in the example from class, we are following the prices of two securities,  $A$  and  $B$ . Let  $P_t^A$  and  $P_t^B$  denote their respective price<sup>1</sup> time-series. We define the *spread* at time  $t$  to be the difference

$$s_t = \log P_t^A - \beta \log P_t^B$$

---

<sup>1</sup>GPT 5 originally used raw prices, but I had to re-prompt it to use log-prices as we did in lecture.

where  $\alpha, \beta$  are the coefficients of the linear regression  $\log P_t^A = \alpha + \beta \log P_t^B$  modeled over a specified time horizon. The *spread momentum* at time  $t$  is the difference

$$m_t = s_t - s_{t-1}$$

We then define the *momentum volatility* and *normalized spread momentum* as

$$\sigma_t^2 = \frac{1}{n-1} \sum_{i=t-n}^t (m_i - \bar{m})^2, \quad n_t = \frac{m_t}{\sigma_t}$$

Finally, the Z-score of momentum is defined as

$$z_t = \frac{m_t - \bar{m}}{\sigma_t}$$

Given these inputs, the state at time  $t$  is encoded as

$$\mathbf{s}_t = \begin{bmatrix} s_t \\ m_t \\ \sigma_t \\ n_t \\ z_t \end{bmatrix} \in \mathbb{R}^5$$

We model the action space as the set  $\{1, 2, 3, 4\}$ , and the states are encoded as follows:

- (1) Go long spread
- (2) Go short spread
- (3) Hold position
- (4) Liquidate (exit) position<sup>2</sup>

Finally, the reward is simply given by the increase in PnL over one time-step. The motivation behind using momentum as a signal is that a strategy leveraging it admits high gains over short periods of time, while minimizing high reversal. We will be evaluating this statement in section 4.

One key difference between the above proposed model and the example we went over in class is the formation of the action space. In class, the action space was a discrete set of buy, sell, and liquidate thresholds. In this case, the action space is just a set of whether to buy, sell, or liquidate at a given time-step. This is because **the example from class has a trading window, while this model does not**. We keep the concept of a formation window, as all calculations for  $\mathbf{s}_t$  are performed over a rolling-window and we need to eliminate look-ahead bias. Hence, an episode in our setup is the following:

- Start with \$1,000 in cash.
- Sample  $t_0 \in [\text{formation period size}, \text{train size}]$
- For each  $t \in [t_0, \text{train size}]$ 
  - (1) Calculate  $\mathbf{s}_t$ .
  - (2) Sample an action  $a_t$  given  $\mathbf{s}_t$  denoting the new position.
  - (3) Depending on  $a_t$ , change the position to one of  $(-k, k\beta_t), (k, -k\beta_t), (0, 0)$  (or hold).
  - (4) Let  $\Delta_A, \Delta_B$  denote the change in position for securities  $A$  and  $B$ , respectively. The trade notional  $\Delta_A P_A^t + \Delta_B P_B^t$  is then subtracted from the available cash.
  - (5) Let  $x_t, y_t, c_t$  denote the amount invested in securities  $A$  and  $B$  as well as the available cash at time  $t$ . The reward is then

$$r_{t+1} = (c_{t+1} + P_t^A x_{t+1} + P_{t+1}^B y_{t+1}) - (c_t + P_t^A x_t + P_t^B y_t)$$

Note that above,  $k$  is computed as

$$k = \frac{c_t + x_t P_t^A + y_t P_t^B}{P_t^A + |\beta_t| P_t^B}$$

i.e. we are trading proportional to the amount of equity available.

The Q network model architecture was kept simple – just one hidden layer with 128 units.

---

<sup>2</sup>GPT 5 did not originally include this.

### 3 Comments on GPT 5 Code Generation

To keep the attached Jupyter notebook light, I have only included the final code used to create all results and visualizations presented and discussed in this report.

Initially, the AI-generated code did not use a `gymnasium` environment to model state changes. I used the following prompt to force it to do so:

I can use the gym library – could you please rewrite this leveraging gym?

As discussed in the previous section, GPT 5 initially modeled spread using raw prices instead of log-prices. Momentum is defined in almost all literature as the cumulative return on a security, and as such we really want to model the difference in log price (analogous to log of return). Hence, I asked GPT 5 the following:

how would this work with the spread in log prices instead of prices?

can you please regenerate all the code with the adjustments you made?

As discussed in the previous section, GPT 5 initially modeled three actions: long, short, and hold. In class, we discussed the concept of liquidating a position due to stop-loss boundaries being crossed. To include this behavior, I used the following prompt:

Can you regenerate the above, this time adding an additional action for liquidation

The  $\beta$  used to model spread in the initial code was treated as a constant that could be modified by the user, but this does not make sense. Beta varies over time and shifts with market regimes, and as such should not be a constant; rather, it should be calculated on a rolling basis along with momentum. Thus, I asked GPT 5 the following:

can you adjust this to use dynamic beta from a linear regression, and additionally include code in the training loop to plot the reward, loss, and max Q function?<sup>3</sup>

I prompted GPT to keep track of the amount of cash on-hand such that for each episode, we start with a fixed amount of cash and only trade with that, as we had done in class:

That's ok. Can you please modify the code such that for each episode, we start with 1000 dollars in cash available to trade

can you modify the code to keep track of the amount of cash, security A, and security B at each time step? and also generate code to run through an evaluation dataset

We had assumed no transaction cost in the example during class. For simplicity's sake, I prompted GPT to delete all transaction cost adjustments it made in the initial code:

Please delete everything involving transaction cost and regenerate the code. For this exercise we are assuming no tcost

Finally, the code was not trading the number of spread units proportional to the amount of cash available. Hence, I prompted GPT to scale the trade notional proportional to the amount of cash available:

The code is not trading the number of units proportional to the amount of cash available

The adjustments I made to the AI-generated code were only of the following nature:

- I adapted the approach from class to select a co-integrated pair of securities. This was not present in the GPT 5-generated code.
- I wrote my own code to analyze the trading and performance activity on the test set.
- I modified the train performance visualizations to match my desired style.
- I changed the `gym` imports to `gymnasium`.
- I swapped out the randomized price inputs for real data.

---

<sup>3</sup>The latter part of this prompt was included so I could evaluate training results.

## 4 Results: Training and Evaluation

**Disclaimer: Results will likely differ if re-running the supplementary notebook due to random seeds.** I performed a cointegration test on all pairs of securities with correlation greater than 0.9, as we had done in class, then picking the pair with the lowest p-value which turned out to be HBAN and SCHW. The model was trained on 100 episodes and the training set was constructed from all observations prior to January 1st, 2017; after each episode, the reward, average loss, and average max Q function values were recorded. The training performance is illustrated below:

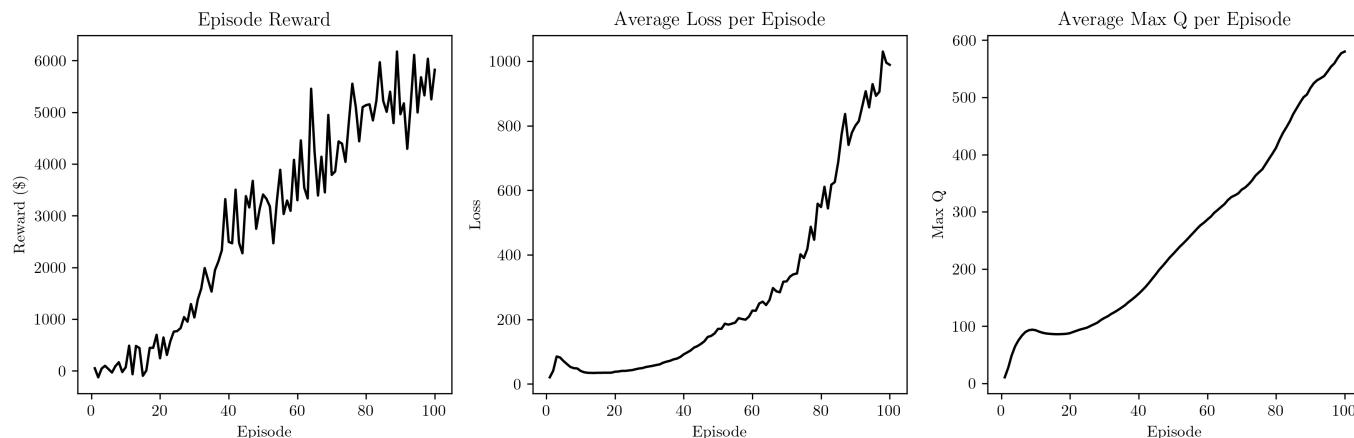


Figure 1: DDQN Training Performance

Although noisy, we see the reward for each episode increasing on average. Although the average loss increased with each episode, the maximum Q function sharply increased, implying that the model was biased towards an increase in the Q function as opposed to the squared difference between the online model and the target model.

The test dataset was constructed on all observations from January 1st, 2017 onwards. The test performance, as well as other trading information, are illustrated below:

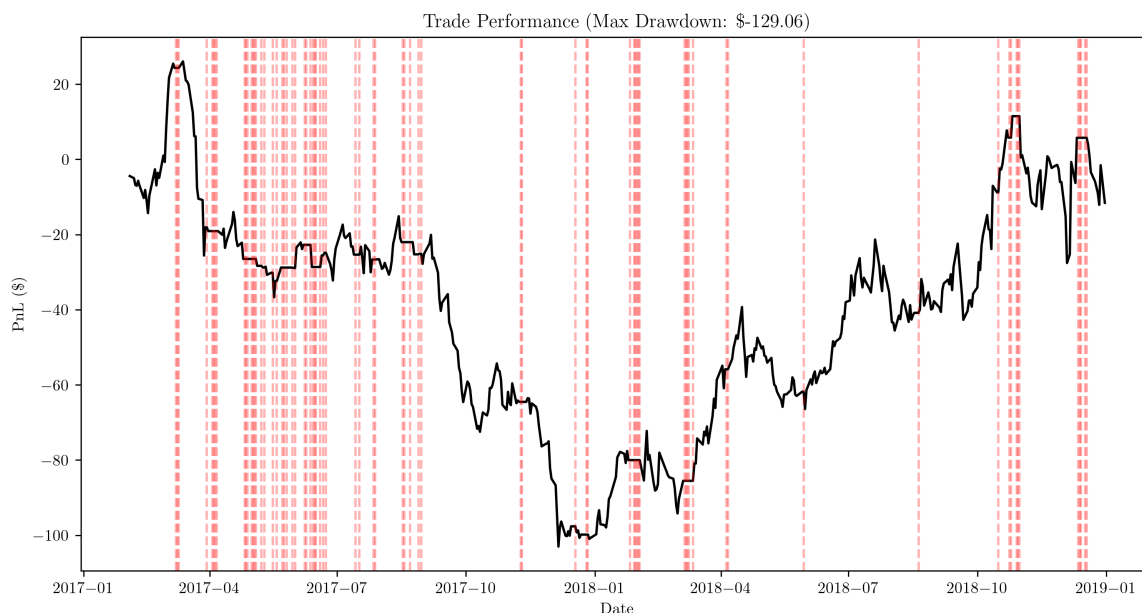


Figure 2: Cumulative PnL over the course of the test period. The vertical lines indicate dates on which the position was liquidated.

Interestingly, it appears as though this strategy is aggressively chasing momentum. The above graphic indicates that the entire position was liquidated every time trading performance was roughly flat.

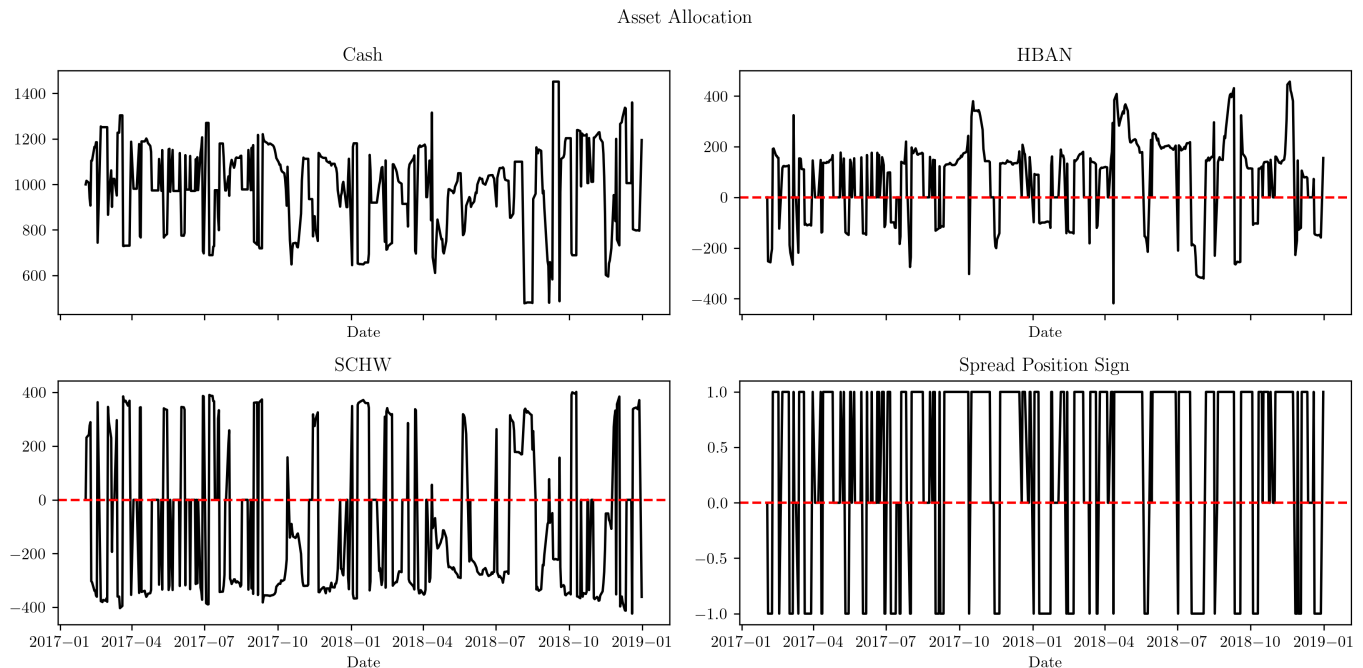


Figure 3: Asset allocation during the evaluation window.

We see a very high turnover, as indicated by the spread position sign chart in the bottom-right corner above. I'd like to draw some attention to GPT 5's argument that the momentum strategy softens short-term drawdown. The average draw-up length was calculated as 4 days, while the average draw-down length was calculated as 21 days. The magnitude of the drawdown each time was quite high, which is evident from the performance chart. Ultimately what we really observe is a model that, while efficiently reaping short-term high rewards, is highly susceptible to large-magnitude drawdown. With a total PnL of -\$11.57, the 2018 rebound was almost enough to recoup 2017's losses. Not quite there, though.