Pairs Trading Using Unsupervised Clustering And Reinforcement Learning

Abstract

Pairs trading strategies are conventionally based on concepts like mean reversion and stationary stochastic processes, where pairs are assumed to have a linear relationship even though the consensus in literature is that interactions of asset prices in the real world are non-linear, more often than not. To tackle this issue, we explore pairs trading from a different perspective, by first clustering equity indices and then use a Reinforcement Learning based trading strategy on pairs selected from the clusters. Using a combination of fundamental and technical features we extract a set of 10 latent risk factors using a convolutional autoencoder and create 10 clusters of indices using these risk factors. We verify our approach on a universe of 96 equity indices and sample 13 pairs to train and test our trading strategy and compare our performance against the S&P 500 and Dow Jones Industrial Average Index (DJIA). Between the period starting from April 2017 to Dec 2022, our best performing strategy consisting of the NASDAQ Composite and MSCI World IT Index, earns an annualized return of 21.86% with a Sharpe ratio of 1.15, while generating an alpha of 20%. Additionally, all 13 pairs outperform the benchmark indices in terms of risk-adjusted return measured using Sharpe ratio.

1 Introduction

Conventional pairs trading strategy follows two steps. First, we identify two securities, generally based on distance/co-integration method, conveying that there are some underlying risk factors which both the securities respond to in a similar fashion. This results in a spread which is assumed to be stationary and mean reverting. Second, we use different strategies based on the spread and take opposite positions in the securities, expecting the spread to revert to the mean. This general approach imposes an underlying assumption of linearity. However, the present consensus is that the two stock returns are rarely linearly correlated as shown by [1].

In this research, we adapt the method of unsupervised learning (agglomerative clustering) to select pairs based on fundamental and technical indicators demonstrated by [2]. As we move away from linearity, we expand on the definition of pairs trading by allowing positions to not necessarily have net zero market exposure. We implement a Reinforcement Learning based trading strategy utilizing a Proximal Policy Optimization learning algorithm to trade on the selected indices.

The research is organized as follows, in Section 2, we present a review of existing pairs trading strategies and literature on machine learning based pairs trading. Section 3 describes the clustering and trading methodology, Section 4 details the implementation, Section 5 presents the empirical results and Section 6 concludes.

2 Literature Review

2.1 Existing Pairs Trading Strategies

[3] summarizes pairs trading mainly into four different groups. On the linear front, they discover that the linear strategy based on a distance approach produce low variance spreads with limited profit potential and substantial divergence risk. They also indicate that although the strategy has low exposure to systematic risk, the profitability declines over time and can partially be explained by information diffusion, pair visibility and liquidity factors. In addition, they also conclude that the co-integration approach produces better results than the distance approach as the identification of pairs is based on more robust equilibrium relationships.

[1] argue that the dependence between two stock prices is rarely linear and the traditional hypothesis of multivariate gaussianity is inadequate. To capture the multivariate distribution of the movement in stock prices, [4] propose a copula based pairs trading approach. [5] furthers the study but conclude that the copula approach underperforms the linear distance and co-integration approach by 31 and 28 bps respectively. [6] build on the approach by using mixed copulas to capture linear and non-linear relations covering different dependency structures. They achieve positive results but the excess return when compared to the distance approach is limited to the top 5 traded pairs in their stock universe when evaluated using committed capital as the criteria. However, the strategy outperforms distance method when considering returns based on fully invested performance evaluation as laid out by [7].

2.2 Machine Learning in Pairs Trading

The primary approach for pairs trading using machine learning has been to use different algorithms to predict the spread or individual prices of the securities. [8] takes an approach of forecasting the returns, determining undervalued and overvalued stocks and trading based on these rankings. [9] propose a filterbank CNN framework and combine fundamental data to improve predictions and profitability.

[2] deviates from the convention and implements unsupervised learning (PCA followed by agglomerative clustering) for

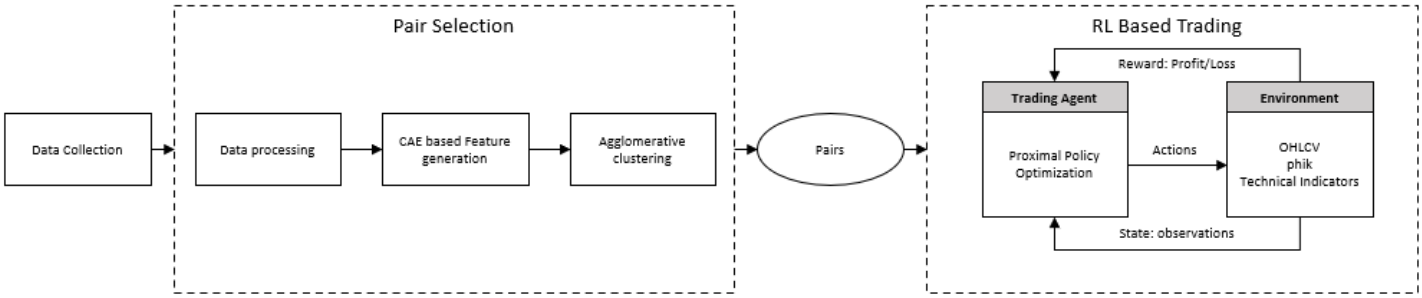

Figure 1: Pair Trading Strategy Workflow

pair selection based on past performance and firm characteristics. They then proceed to classify stocks based on past one month returns as undervalued or overvalued within the cluster and take positions accordingly. The authors are able to generate a strategy producing 23.8% annualized mean excess returns over a 40 year period with a sharpe ratio of 2.69 and a maximum drawdown of 12.3% lasting for 2 months.

[10] use Reinforcement Learning (RL) and Deep Q-Networks (DQN) to take simplistic actions to predict open and stop loss timings but generate limited profitability and Brim expands on the work by using a double DQN with three actions (hold, buy and sell) though a low win rate limits the applicability of their model. [11] use a two staged approach that determines the optimal open and stop loss thresholds using a multiscale ResNet with a Restructuring Labeling Mechanism, removing the unprofitable pairs using a second multistage ResNet and are able to enhance profit by up to 40%.

3 Methodology

We approach equity index pair trading by first clustering the equity indices for pair selection and subsequently implement a RL based pair trading strategy. Using raw features directly for unsupervised clustering can dilute the information contained in important features [2]. We modify the approach used by [2] for feature generation using PCA by implementing a non-linear dimensionality reduction algorithm. Convolutional AutoEncoders (CAE) have shown to be effective for unsupervised clustering in [12], which we use to generate a latent representation of 10 features. We interpret these features as representative of the underlying risk factors [13]. We cluster indices based on the 10 underlying risk factors using agglomerative clustering and further sample pairs within the clusters to test our methodology. As we approach pairs trading from a non linear perspective, we take a RL based approach which is demonstrated to perform well in complex environments (financial markets being one) [14]. We use a Long Short Term Memory (LSTM) network for the policy and Q functions to incorporate hidden states of historical observations.

3.1 Pairs Selection

There are several non-linear dimensionality reduction algorithms including but not limited to Kernel-PCA, t-Distributed Stochastic Neighbor Embedding (t-SNE), Multidimensional scaling (MDS), and Isometric mapping (Isomap). In our approach, we have chosen CAE and we describe our rationale in the following sub-section.

3.1.1 Dimensionality Reduction Using Convolutional AutoEncoder (CAE)

CAE has been shown to perform well for clustering applications in [12]. Further, CAEs have been found effective in dealing with time series data [15] which motivates us to use it in our clustering architecture. A conventional autoencoder is generally composed of two layers, corresponding to encoder $f_W(.)$ and decoder $g_U(.)$ respectively. It aims to find a representation for each input sample by minimizing the reconstruction error computed as mean squared errors (MSE) between its input and output over all samples (n), i.e.

$$\left(\frac{1}{n}\right)\sum_{i=1}^{n}||g_U(f_W(x_i)) - x_i||^2 \tag{1}$$

We use Convolutional AutoEncoders (CAE) instead of Stacked AutoEncoders (SAE) since they do not need tedious layer-wise pretraining, as shown in Figure 2.

First, convolutional layers are stacked on the input data set to extract hierarchical features. We then flatten all units in the last convolutional layer to form a vector, followed by a fully connected layer with only 10 units which is called an embedding layer (h). The input original feature set (x) is thus transformed into a 10-dimensional feature space. To train it in an unsupervised manner, we use a fully connected layer and convolutional transpose layers to transform embedded features back to the original feature set. The parameters of the encoder and decoder are updated by minimizing the reconstruction error defined above.

The key factor of the proposed CAE is the aggressive constraint on the dimension of the embedding layer. If the embedding layer is large enough, the network may be able to copy its input to output, learning useless features. The intuitive

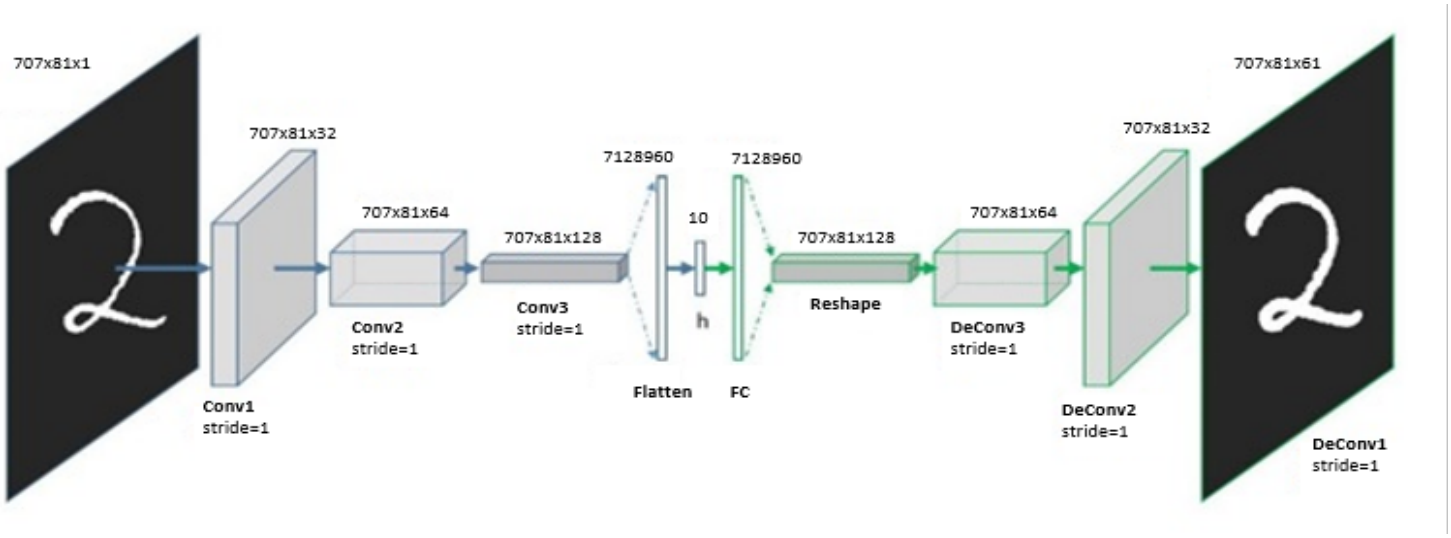

Figure 2: The structure of proposed Convolutional AutoEncoders (CAE) for feature extraction. In the middle there is a fully connected autoencoder whose embedding layer is composed of only 10 neurons. The rest are convolutional layers and convolutional transpose layers (some work refers to it as the Deconvolutional layer). The network can be trained directly in an end-to-end manner.

way of avoiding identity mapping is to control the dimension of latent features h lower than input data x. Learning such incomplete representations forces the autoencoder to capture the most salient features of the data. Thus, we force the dimension of the embedded space to be equal to the number of clusters of the dataset. In this way, the network can be trained directly in an end-to-end manner even without any regularization like Dropout or Batch Normalization. The learned compact representations are proven to be effective for clustering [12].

3.1.2 Agglomerative Clustering Using Embedding Layer Features

In [2] , they have compared the performance of k-means clustering, density-based spatial clustering of applications with noise (DBSCAN), and agglomerative clustering. Agglomerative clustering performed the best for pairs trading strategy based on various factors including but not limited to Sharpe ratio, maximum drawdown, and robustness through various market conditions. Hence we use agglomerative clustering for the pair selection process.

Agglomerative clustering is a hierarchical clustering method and starts by treating individual data points as a cluster on its own before merging the clusters step by step until termination criteria are met [16]. It requires the user to specify one of two hyperparameters: the number of clusters K or the maximum distance for clusters to be merged, known as linkage distance.

3.2 Reinforcement Learning Based Pairs Trading

Due to the non-equilibrium and mean-reversion constraints of conventional pairs trading strategies, we decided to look at our goal from a Reinforcement Learning perspective. RL is a branch of machine learning that is based on training an agent to operate in an environment based on a system of rewards. We choose this approach since the pairs trading problem is well suited for a RL framework, given the goal of maximizing returns for a well-defined and simple reward structure in a stochastic and non-stationary environment like the stock market. Another reason for choosing an RL based approach is because in conventional pairs trading strategies, it is a challenge to set appropriate thresholds to generate trading signals based on statistical arbitrage principles, which have a significant impact on profitability [11]. Using RL, we eliminate the need for strict stationarity assumptions and to generate trading signals altogether, and instead train an agent to learn an optimal trading strategy which maximizes returns in a complex and dynamic market. The optimal strategy learned by the agent is dynamic and adapts to a non-stationary market.

[14] use Deep Reinforcement Learning (DRL) to find the optimal strategy for a portfolio of stocks using various actor-critic based learning algorithms to train the RL agent's policy (trading strategy) and Q (reward) functions. DRL uses neural networks as function approximators to represent the policy and Q functions, which are trained during the learning stage.

To train the RL agent, we use an actor-critic algorithm to simultaneously update the actor network that represents the policy and the critic network that represents the value function. The critic estimates the value function, while the actor updates the policy probability distribution guided by the critic with policy gradients. Over time, the actor learns to take better actions and the critic gets better at evaluating those actions. The actor-critic approach allows us to have continuous action and state spaces which are useful since asset prices are continuous. Moreover, the actor-critic approach has been shown to learn and adapt to large and complex environments, showing

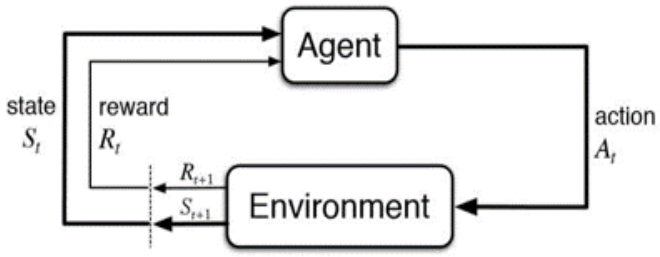

Figure 3: Reinforcement Learning Framework

promising results in video games such as Doom and StarCraft II [17].

We decided to use the PPO algorithm [18] for trading since it is stable, fast and relatively simpler to implement and tune. The PPO algorithm is used to control the policy gradient update and ensure that the new policy will not be too different from the previous one. PPO has shown to improve the stability of policy networks during training by restricting the policy update at each training step. We assume the asset prices in the market follow a Partially Observable Markov Decision Process (POMDP) and use recurrent LSTM networks for the policy and Q functions to incorporate a hidden state of historical observations with the idea that knowledge of previous observations would give the model better information to make optimal decisions and calculate the Q-value at every state.

We provide a description of how we formulate the pairs trading problem and the design of our custom trading environment in the following sections:

3.2.1 Markov Decision Process (MDP) Model for Trading

To model the stochastic nature of the dynamic stock market, we employ a MDP as follows:

- State $s = [p, f, c]$, a vector of asset price data p (OHLCV), technical features generated using the price data f, and c denotes the ϕ_K correlation [19] between the two indices.. $S \in R^{100}$, where the size of the $[p, f, c]$ vector for each equity index vector is 50. The technical indicators used have been listed in Table 1.

- Action a : A $1 - D$ vector of actions over the 2 indices. The action space is a tuple of values, both continuous and in the range $[-1, 1]$ and represents the portfolio allocation at every timestep. We normalize the vector a, to calculate portfolio weights and rebalance the portfolio. Positive, negative or 0 values represent long, short or no positions in the respective asset.

- Reward $r(s, a, s')$: The direct reward of taking action a at state s and arriving at the new state s'. We use the cumulative portfolio return at each timestep as the reward.

- Policy π(s) : The trading strategy, which is a probability distribution over actions at state s

- Q-value $Q_\pi(s, a)$: the expected reward of taking action a at state s following policy π.

At time t, we normalize the action vector a to generate portfolio weights and calculate the portfolio allocation at time $t+1$. We use the difference between the portfolio weights at $t+1$ and t, to calculate and take relevant actions in the market to rebalance the portfolio. The reward is calculated as the cumulative portfolio return at $t + 1$.

3.2.2 Incorporating Trading Constraints

We incorporate the following assumptions and constraints to reflect the concerns for practice :

- Transaction Costs : Although transaction costs greatly affect the profitability of daily trading strategies in practice, we ignore any transaction costs in our current implementation of the trading environment.

- Market Liquidity : The orders can be rapidly executed at the close price. We assume the market will not be affected by the actions of the RL agent.

- Margin Trading : The agent starts off with an initial amount in the cash account at $t = 0$ and subsequently takes actions in the market. The proceeds from previous transactions are deposited in the cash account, therefore the total value of assets at any time t, is the sum of the mark-to-market portfolio value and money in the cash account. We use the total value of assets as margin to finance our actions in the market. We also assume that the cash account itself does not generate any returns.

- Risk Aversion : We use a stop-loss level at 50% of the initial capital to control downside risk. If the total value of assets at any timestep t goes below the stop-loss level, the agent liquidates all positions and exits the market.

4 Implementation

4.1 Data Sources

We select the top 250 most tracked equity indices on Bloomberg as our index universe. We further reduce our universe size from 250 to 96 by omitting indices with unavailable data. All data has been sourced from Bloomberg.

4.2 Data Processing for Pair Selection

Fundamental characteristics have proven to be an important source of information in identifying pairs, therefore, we use a set of 18 such metrics for indices which we list in Table 1. In addition, we include 1 to 48 months of lagged returns [2] and a set of technical indicators. We perform feature wise rolling window min-max scaling on our entire feature set.

Fundamental Features	Technical features for Clustering	Technical features for Pairs Trading
Price to Book Ratio	Relative Strength Index	Simple Moving Average
Total Current Liabilities	Money Flow Index	Bollinger Bands
Current Market Cap	Average Directional Index	Money Flow Index
Number of Employees	On Balance Volume	Relative Strength Index (RSI)
Return on Assets	Average True Range	True Range
Profit Margin	Bollinger Bands	Stochastic RSI
Return on Common Equity	Exponential Moving Average	MACD
Current Ratio	MACD	ϕ_K correlation
Dividend Payout Ratio		
Price to Earnings Ratio		
Capital Expenditures		
Total Debt to Total Assets		
Operating Margin		
Return on Capital		
Bloomberg Estimated EBITDA		
Bloomberg Estimated Sales		
Short and Long Term Debt		
Cash & ST Investment / Share		

Table 1: List of metrics and indicators used

4.3 Pair Selection Using Unsupervised Clustering

We train the CAE on data from September 2020 to January 2023. For agglomerative clustering, we specify the number of clusters (K) rather than the maximum distance. We implement agglomerative clustering with K = 10, which is equal to the size of the CAE embedding layer. We sample 13 pairs from the universe of 497 possible pairs using 4 metrics - mutual information score, quantile regression R^2, ϕ_k non-linear correlation [19], and co-integration to test our strategy

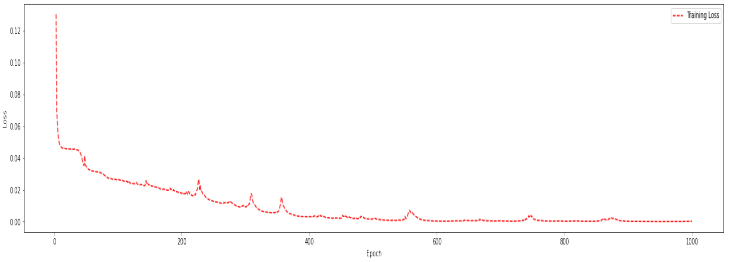

Figure 4: CAE Reconstruction Error v/s Training Epochs

4.4 Data Processing for RL

We use OHLCV data, 1-10 day lagged returns and technical indicators highlighted in Table 1 as input (state space) and use a 30-day rolling window min-max scaling.

4.5 Reinforcement Learning Based Pairs Trading

We train our RL agent with an initial capital of $\$1,000,000$ on each of the 13 sample pairs of indices on training data from 02-18-2010 to 04-23-2017, for 500,000 timesteps using PPO learning algorithm and recurrent actor-critic networks. We test the trained model on out-of-sample periods from 04-24-2017 to 12-22-2022.

We do not test our strategy against the conventional linear pairs trading strategy as the sampled pairs need not satisfy the assumptions of stationarity and mean reversion which will give us invalid and uncomparable results.

5 Empirical Results

5.1 Unsupervised Clustering and Pair Selection Results

In Fig 4, we observe that the MSE loss of the CAE reduces consistently and reaches a plateau at 600 epochs. By reducing the reconstruction error to almost 0, the CAE learns an accurate feature representation in the embedding layer to reproduce the input with minimal error. This validates our assumption that an embedding layer of size 10 is appropriate to learn the latent

representation of the index dataset. Table 2 lists the 10 clusters formed using agglomerative clustering.

A key observation from the clustering results is that the indices with similar characteristics (for eg. NDQ Index and MXWO0IT Index, SAX Index and OMXS30B Index, etc) are clustered together, reinforcing the hypothesis that the 10 embedded layers are representative of the underlying risks associated with the indices.

Cluster No.	Indices
1	SXXR Index, SGX Index, SVX Index, M1WD Index, SPTR500N Index, SXTP Index, SMIM Index, MXWO000G Index, SET50 Index, MXEM Index, NSEIT Index, MXBR Index
2	SET Index, SXPP Index, DJI Index, RAY Index, S5INFT Index, RLG Index, MXUS Index, MDAX Index, IPSA Index, MXAP Index, JALSH Index, SXNP Index, S5HLTH Index, SXRP Index, OSEBX Index, SX6P Index, BE500 Index, SXKP Index, IBOVE Index, S5TELS Index
3	NMX Index, SXOP Index, SAX Index, PSI20 Index, SPIEX Index, IMOEX Index, TPREAL Index, OMXS30B Index
4	DAX Index, NDX Index, IBOV Index, CAC Index, MXEF Index, MXWD Index, SPXL1 Index, OMX Index, MEXBOL Index, SPI Index
5	NYA Index, NDQ Index, SX3P Index, OEX Index, MXLA Index, SBF120 Index, SCXP Index, MXWOU Index, MXWO0IT Index, SPR Index, SBX Index
6	MXWO0HC Index, MXWO0EN Index, NZSE50FG Index, AMZ Index, MSDEE15N Index, SXQP Index
7	INDU Index, UKX Index, SMI Index, SPXT Index, TWSE Index, SXXE Index, ASX Index, SXEP Index, RLV Index, SPTR Index
8	RIY Index, TOP40 Index, SXDP Index, S5INDU Index, S5CONS Index, S5UTIL Index, SPXL2 Index, TAMSCI Index
9	SX4P Index, NDDUUS Index, MXWDU Index, OBX Index, MSPE Index
10	SPX Index, SXXP Index, MXWO Index, MXEU Index, PCOMP Index, MXEA Index

Table 2: Clusters using embedded features

5.2 Trading Performance

Table 3 shows the performance comparison for the 13 sampled pairs and the benchmark (S&P 500 Index), sorted from the best performing to the least in terms of annualized returns. The results show that 12 out of our 13 sampled pairs outperform the benchmark annualized returns and maximum drawdown. The top 3 performing pairs have an annualized return of 21.86%, 18.61%, and 17.61% respectively compared to 8.77% for the benchmark. All 13 sample pairs outperform the benchmark Sharpe ratio and annualized volatility. The top 3 performing pairs have a Sharpe ratio of 1.15, 0.94, 1.02 respectively compared to 0.40 for the benchmark.

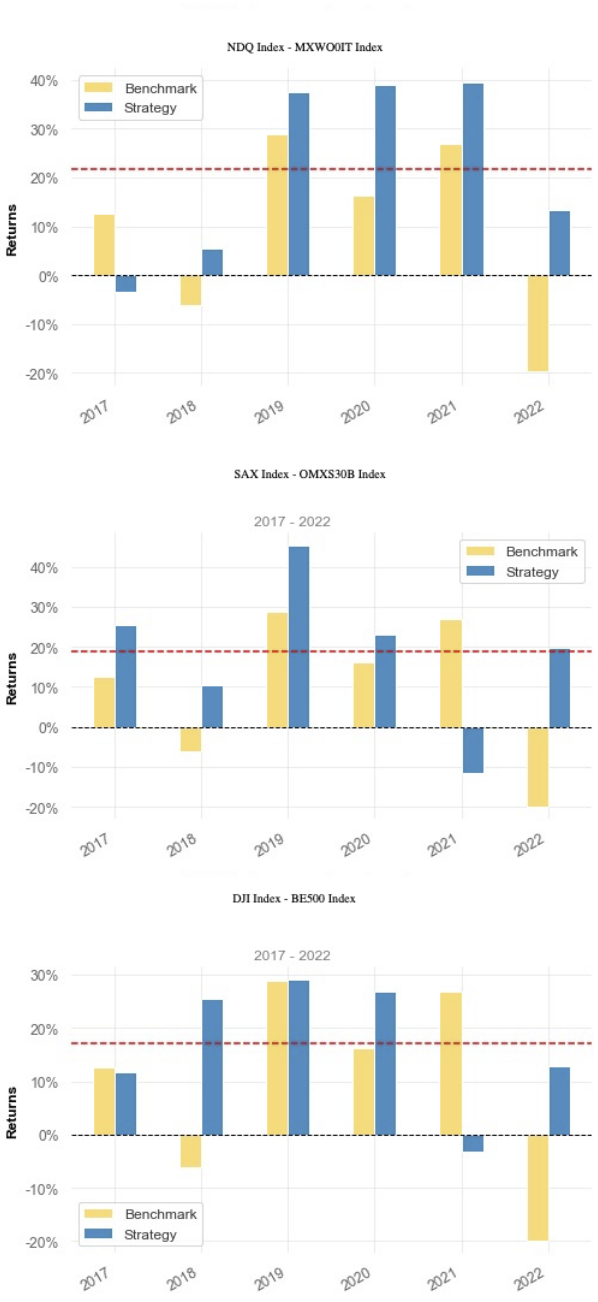

Figure 5: YoY Returns between the top three performing pairs and benchmark S&P500

In addition, a key characteristic observed in the results is that the proposed framework generates a beta-neutral strategy, indicating market-neutrality, which is characteristic of traditional statistical arbitrage strategies. This is further consolidated by

Index 1	Index 2	Cummulative Returns	Annualized Returns	Annualized Volatility	Sharpe Ratio	Max Drawdown	Alpha	Beta
NDQ	MXWO0IT	206.47%	**21.86**%	15.96%	1.15	-18.11%	0.20	-0.01
SAX	OMXS30B	163.00%	**18.61**%	16.91%	0.94	-22.54%	0.18	-0.01
DJI	BE500	150.65%	**17.61**%	14.38%	1.02	-23.50%	0.17	0.02
SXOP	SAX	110.50%	**14.04**%	16.63%	0.73	-20.43%	0.14	<0.01
RAY	RLG	101.44%	**13.16**%	13.71%	0.79	-18.11%	0.13	0.02
OEX	SPR	88.45%	**11.83**%	18.30%	0.57	-21.36%	0.12	0.03
SGX	MXWO000G	86.56%	**11.63**%	12.79%	0.74	-19.72%	0.11	0.01
SXXP	MXEU	83.36%	**11.29**%	13.47%	0.69	-19.44%	0.11	-0.02
RAY	MXUS	80.88%	**11.03**%	17.00%	0.56	-19.68%	0.11	0.01
RIY	SPXL2	80.82%	**11.02**%	16.40%	0.58	-19.86%	0.11	0.01
IPSA	S5TELS	71.84%	**10.03**%	12.40%	0.65	-16.16%	0.10	<0.01
DAX	OMX	61.74%	**8.86**%	19.76%	0.41	-38.32%	0.10	<0.01
M1WD	MXWO000G	54.99%	8.04%	10.96%	0.56	-17.17%	0.08	0.02
Benchmark S&P 500		61.00%	8.77%	20.28%	0.40	-33.92%	-	-

Table 3: Performance Analysis & Benchmarking (April 2017 to Dec 2022)

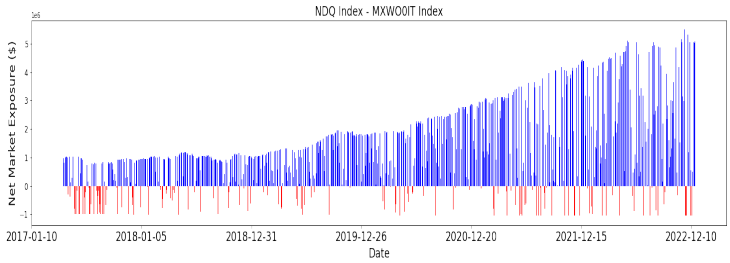

Figure 6: Daily net market exposure for NDQ Index - MXWO0IT Index pair strategy

examining the cumulative returns chart during the COVID-19 market crash in March 2020, where our strategy continues to generate stable returns in a volatile market. Furthermore, the beta-neutrality allows the strategy to generate alpha consistently.

Figure 5 shows the year-over-year (YoY) returns for the top 3 performing pairs compared with the benchmark S&P500 Index. Analyzing the YoY charts, it is clear that the strategy outperforms the benchmark consistently in almost every year in the testing period, with at most one year of negative returns for each pair. The strategy is therefore robust to different market conditions as can also be seen in the cumulative returns chart in Figure 7, where the strategy continues to generate stable returns through periods of market ups and downs.

6 Conclusion

In this paper, we have explored the potential of pairs trading based on non-linear relationships between equity indices. Our strategy is based on a dynamic RL-based framework with no assumptions of stationarity or mean reversion as opposed to linear pairs trading or a copula based approach. Pairs trading is well suited for a RL framework, given the goal of maximizing returns for a well-defined and simple reward structure in an ever-changing stock market. The optimal strategy learned by the agent is dynamic and more adaptable to changes in the underlying market conditions than conventional pairs trading strategies. Therefore, it does not need ad-hoc retraining of the model even during extreme market conditions. This can be observed during the market crash in March 2020 due to the COVID-19 pandemic in Figure 7. Although the RL agent was trained on data which did not include any such periods of high volatility or market turbulence, it adapts itself well to changes in the market conditions and continues to generate stable returns YoY. Therefore, the policy and Q networks of the RL model can be retrained periodically, as needed.

Additionally, a traditional statistical arbitrage strategy is market neutral as the net exposure to the market is almost zero. Although the net market exposure is not always 0 in our proposed approach [Figure 6], the strategy is still market neutral as the beta is close to zero. Hence, the strategy is able to generate alpha consistently for all sample pairs. We believe our approach to be a more robust pairs trading strategy as we do not have any assumptions of stationarity and generate beta neutral returns. Moreoever, we have defined the cummulative portfolio returns as the reward to be maximized in our RL framework, which can be easily redefined to be the Sharpe ratio, downsized risk-adjusted returns etc, to generate trading strategies with specified risk profiles.

A future work of our research is to perform further robustness tests of the proposed workflow by training the CAE for a period before the start of the testing period. We have assumed the latent features of the indices to be time independent, but further work is needed to validate our assumption. Secondly, we would also like to explore ensembles of RL algorithms [14] to further

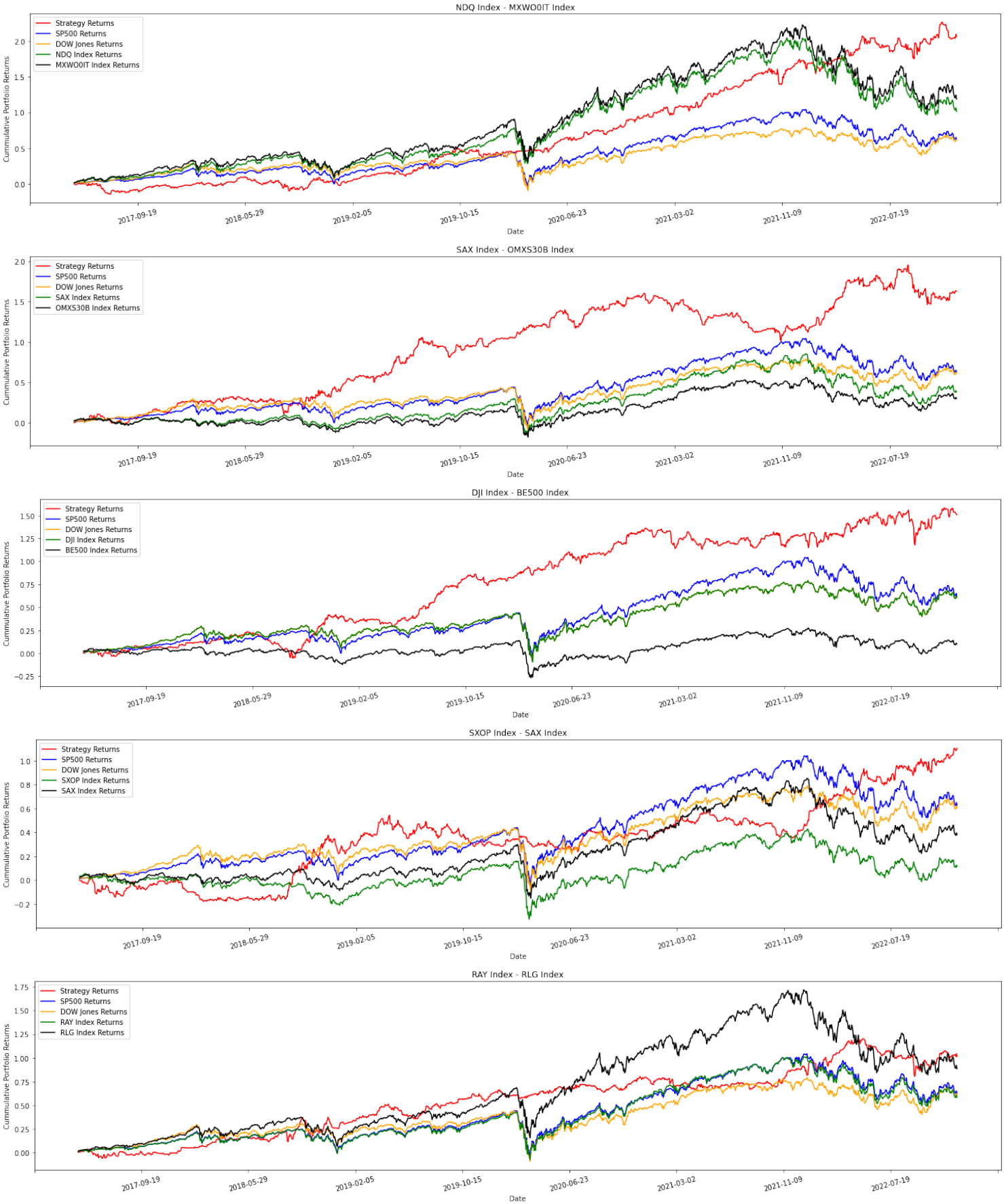

Figure 7: Cumulative Returns for top 5 performing pairs, S&P 500, DJIA and component indices (April 2017 to Dec 2022)

increase the robustness of our model and include a validation period to tune the hyperparameters of the framework. Lastly, we would like to test the strategy on more clustered pairs in addition to the 13 we sampled.

References

[1] Thierry Ane and Cécile Kharoubi. Dependence structure and risk measure. *The journal of business*, 76(3):411–438, 2003.

[2] Chulwoo Han, Zhaodong He, and Alenson Jun Wei Toh. Pairs trading via unsupervised learning. *European Journal of Operational Research*, 307(2):929–947, 2023.

[3] Christopher Krauss. Statistical arbitrage pairs trading strategies: Review and outlook. *Journal of Economic Surveys*, 31(2):513–545, 2017.

[4] Rong Qi Liew and Yuan Wu. Pairs trading: A copula approach. *Journal of Derivatives & Hedge Funds*, 19:12–30, 2013.

[5] Hossein Rad, Rand Kwong Yew Low, and Robert Faff. The profitability of pairs trading strategies: distance, cointegration and copula methods. *Quantitative Finance*, 16(10):1541–1558, 2016.

[6] Fernando AB Sabino da Silva, Flavio A Ziegelmann, and João F Caldeira. A pairs trading strategy based on mixed copulas. *The Quarterly Review of Economics and Finance*, 87:16–34, 2023.

[7] Evan Gatev, William N Goetzmann, and K Geert Rouwenhorst. Pairs trading: Performance of a relative-value arbitrage rule. *The Review of Financial Studies*, 19(3):797–827, 2006.

[8] Nicolas Huck and Komivi Afawubo. Pairs trading and selection methods: is cointegration superior? *Applied Economics*, 47(6):599–613, 2015.

[9] Yu-Ying Chen, Wei-Lun Chen, and Szu-Hao Huang. Developing arbitrage strategy in high-frequency pairs trading with filterbank cnn algorithm. In *2018 IEEE International Conference on Agents (ICA)*, pages 113–116. IEEE, 2018.

[10] Taewook Kim and Ha Young Kim. Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries. *Complexity*, 2019:1–20, 2019.

[11] Wei-Lun Kuo, Wei-Che Chang, Tian-Shyr Dai, Ying-Ping Chen, and Hao-Han Chang. Improving pairs trading strategies using two-stage deep learning methods and analyses of time (in) variant inputs for trading performance. *IEEE Access*, 10:97030–97046, 2022.

[12] Xifeng Guo, Xinwang Liu, En Zhu, and Jianping Yin. Deep clustering with convolutional autoencoders. In *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14-18, 2017, Proceedings, Part II 24*, pages 373–382. Springer, 2017.

[13] Marco Avellaneda and Jeong-Hyun Lee. Statistical arbitrage in the us equities market. *Quantitative Finance*, 10(7):761–782, 2010.

[14] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*, pages 1–8, 2020.

[15] Tingting Chen, Xueping Liu, Bizhong Xia, Wei Wang, and Yongzhi Lai. Unsupervised anomaly detection of industrial robots using sliding-window convolutional variational autoencoder. *IEEE Access*, 8:47072–47081, 2020.

[16] K Chidananda Gowda and TV Ravi. Divisive clustering of symbolic objects using the concepts of both similarity and dissimilarity. *Pattern recognition*, 28(8):1277–1282, 1995.

[17] Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *International Conference on Learning Representations*, 2017.

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[19] M Baak, R Koopman, H Snoek, and Sander Klous. A new correlation coefficient between categorical, ordinal and interval variables with pearson characteristics. *Computational Statistics & Data Analysis*, 152:107043, 2020.