Deep Learning for Structural Break Prediction in Equities Pairs Trading

Harish Yerra, Harshavardhan Srijay, Rahul Prakash, Carol Wang, Gen Xu
Duke University, Durham, NC

1 Introduction

The stock market has been highly uncertain during the last year. Recent events like the Russian-Ukrainian War and the rise in inflation/spikes in interest rates have impacted financial markets. These external factors result in significant arbitrage opportunities and downsides due to rapid, unexpected market movements. In this project, we seek to build an effective pairs trading strategy robust to rapid market changes, namely structural breaks. We design a two-phase machine learning architecture to predict the likelihood of a structural break and use these predictions in a copula-based pairs trading strategy. We achieve promising results in both in and out-of-sample sets, indicating the potential for lightweight, deep learning-based methods for structural break prediction and their utility for risk control in constructing robust, market-neutral arbitrage strategies.

2 Data

2.1 Data Collection

Before fine-grain pair selection, we generated an extensive database of existing equity tickers. We used the Alpaca trading API in a Python environment to obtain a list of 4884 total equity assets listed on the US stock market. We refined this list by eliminating non-tradable, inactive, or non-shortable stocks to yield 4815 candidate assets. Next, we used the Yahoo Finance API to collect fundamental data for each candidate and assemble the metadata into a consolidated CSV file. We pulled over 20 features for each equity, including the market cap, industry sector, profit margins, revenue growth, earnings per share, free cash flow, etc.

Table 1: Equity Features Used in Clustering Analysis

Market Cap	Industry Sector	Profit Margins
Revenue Growth	Earnings Per Share	Free Cash Flow
Operating Cash Flow	Operating Margins	Market Beta
Trailing Earnings Per Share	Number of Full Time Employees	Gross Profits
Price Earnings to Growth Ratio	Debt to Equity Ratio	Total Cash

Once we selected a pair, we computed the spread, the average spread, the standard deviation of the spread, the z-scores, the trading volume, and the overall sentiment of the two stocks based on data provided by Alpaca. Given the closing price of two stocks s_1 and s_2, the spread was computed by the following: $\log(s_1) - \log(s_2)$. We then stored the result of the average spread and standard deviation of the spread over the past 90 days. Finally, given the log of the spread s, the average spread $\bar{x}$, and the standard deviation σ, we compute the z-score by the following formula: $\frac{s-\bar{x}}{\sigma}$. The trading volume of both stocks was stored directly based on the data provided by Alpaca. We also incorporated sentiment based on news data which we describe below.

2.1.1 News Sentiment Integration

To calculate the sentiment of the companies corresponding to the stock, we use the News API provided by Alpaca to obtain news articles related to each stock. Each news article affects the sentiment rating for one week after its publication. We then aggregate all of the news articles by concatenating the headline and summary of each article to obtain a long text of all of the news related to the given stock. Then we apply an industry standard sentiment model, NLTK's VADER model, to analyze the sentiment of the resulting text. Once the sentiment for each stock is determined, we compute the spread of the sentiment between both stocks and feed this as another feature.

2.2 Data Processing

We processed the generated fundamental data CSV for feature normalization and removal of missing data. All equities that had missing fundamental data in any feature vector were removed from the dataset entirely. This eliminated the need for robust data imputation, which could be challenging since the data in each row was not necessarily dependent on its surrounding neighbors. Next, the categorical features, such as the industry sector, were converted into "one-hot" representations such that each unique value was turned into a feature vector column of binary data. The rows for which that value was present would contain a 1 in this feature vector, and all other rows would contain zeros. Finally, we normalized each feature vector column in the data by subtracting the mean and dividing by the standard deviation. Such normalization allowed us to reduce sparsity and increase the homogeneity of similar equities in the clustering space.

2.3 Data Labeling

Data was labeled by looking at the z-scores of a pair. If the absolute value of the z-score went above a 3, we classified the data point as a structural break and assigned it the value 1. It remained a structural break until the absolute value of the z-score went down to 2.5. However, we also wanted to learn when a structural break was approaching to avoid trades that were likely to fall under a structural break. Thus, once we determined a structural break occurred, we labeled the past seven days with the following values: $\frac{1}{7}$, $\frac{2}{7}$, $\frac{3}{7}$, $\frac{4}{7}$, $\frac{5}{7}$, $\frac{6}{7}$, and $\frac{7}{7}$. We applied this approach to capture the growing intensity of diverging z-scores before a structural break occurred.

3 Strategy

3.1 Pairs selection

To implement our pairs trading strategy, we must first choose an appropriate corpus of pairs of equities, from which our downstream trading algorithm can dynamically choose from and trade. To do this, we implemented clustering-based pairs selection using OPTICS (Ordering points to identify the clustering structure), an algorithm to obtain density-based clusters in high dimensional spatial data.

3.1.1 Equity Clustering

After preprocessing, we filtered the candidate list of equities from 4884 to 2484 entries. We then used the normalized feature database in density-based clustering analysis. Initially, we deployed the DBSCAN algorithm for this analysis. However, DBSCAN failed to detect all meaningful clusters, since the density of equities varied throughout space. To address this limitation, we deployed the OPTICS algorithm. OPTICS was parametrized by ϵ, which is the maximum cluster radius that can be considered, and *mpts*, the minimum number of points permitted to form a cluster. Through trial and error search through the parameter space, we converged on using low epsilon values, ranging from 0.3-1.8 and 3 minimum points per cluster. A representative distribution for the size of the resulting clusters can be seen below for the top 100 largest companies by market capitalization.

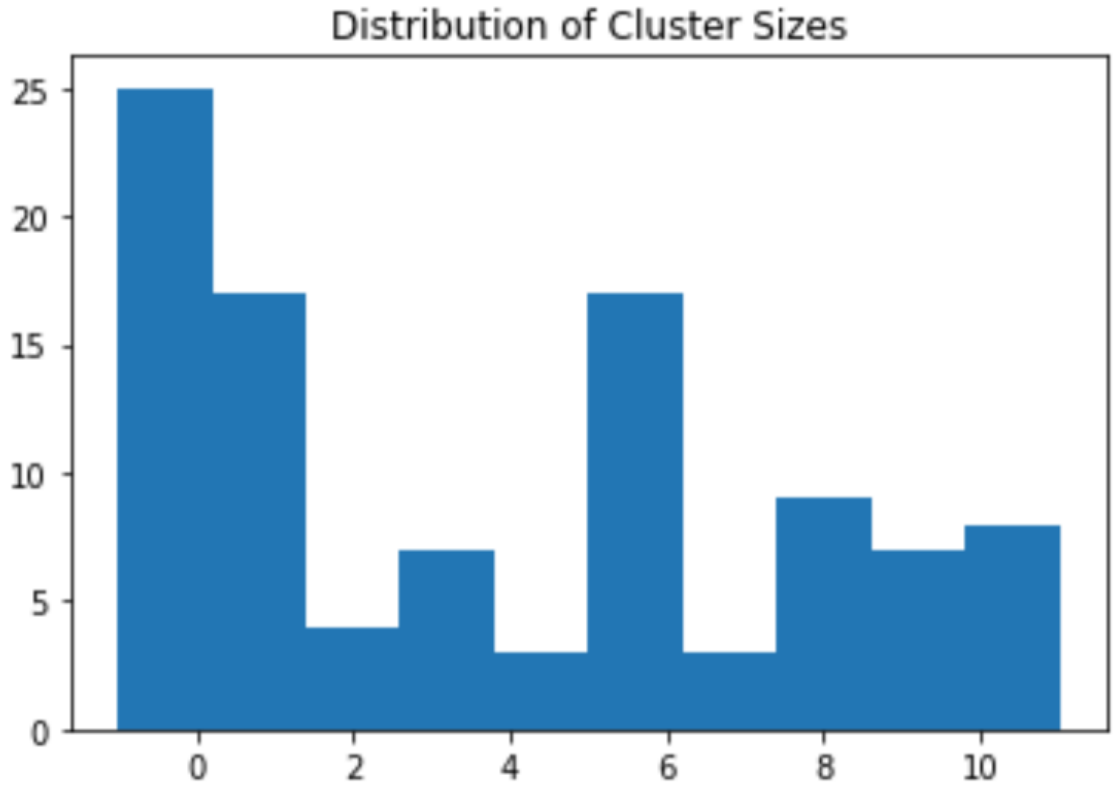


Figure 1: Representative distribution of cluster sizes for 100 largest companies (-1 indicates unclustered points)

A visualization of the spatial separation between clusters can be seen in the below 2D

projection of the multi-dimensional cluster space. The 20+ dimensional feature vectors were condensed into 2 dimensions using a PCA decomposition so that they can be plotted as shown below.

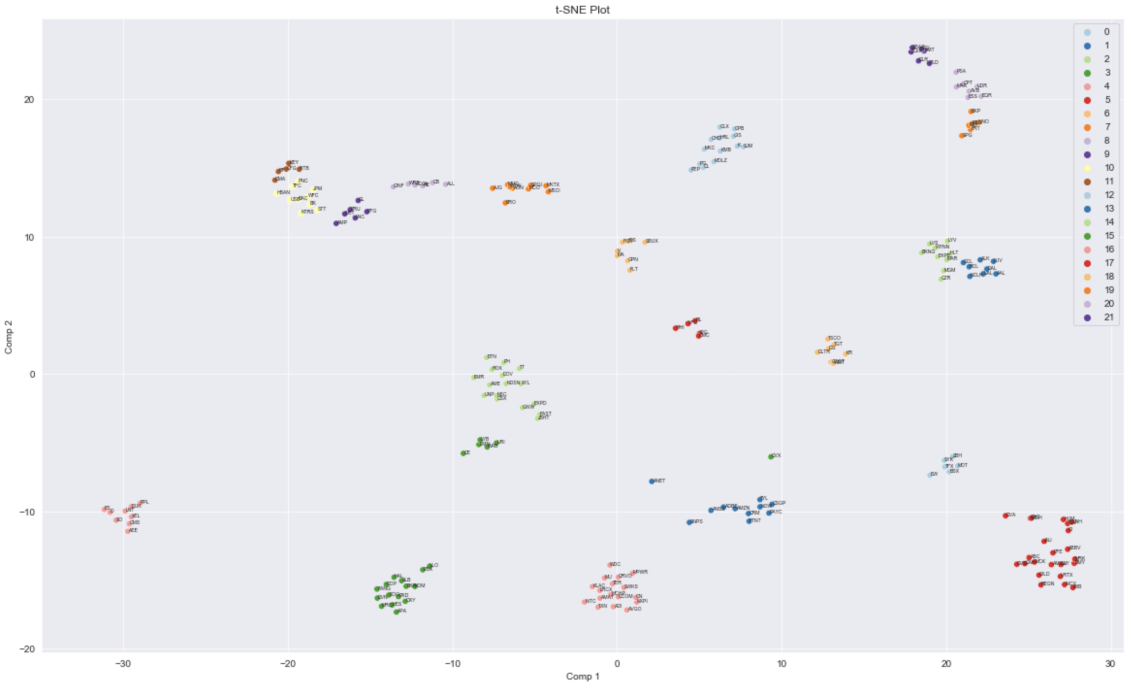


Figure 2: 2D projection of clusters

3.1.2 Pairs List Selection

During trading, a list of 25-50 candidate pairs is maintained by our algorithm, with a single one being traded at any given time. This list was generated by statistical analysis of each of the clusters formed in the previous step. For each member of a given cluster, its relationship with every other cluster member was analyzed to assign a quality metric to that potential pairing.

Johansen Test

The Johansen test was used for cointegration analysis between every set of possible pairs within a cluster. This test treats every asset as an independent variable and uses a vector error correction model to quantify levels of cointegration. To determine if the asset prices are statistically significantly when cointegrated, a critical value of $p = 0.05$ was used, with a correlation threshold of at least 0.9. All pairs which passed these two criteria when tested on historical data from the training period were stored as candidate pairs.

The final candidate pairs after filtering are as listed below. A single pair is then chosen using a rolling statistical analysis each month.

Pair Member 1	Pair Member 2
EPR	TGH
TRTN	TGH
TRTN	SPG
EPR	SPG
SPG	TGH
EPR	TRTN
SPG	EPR

Kendall Tau Rank Correlation
Kendall Tau rank correlation was used to eliminate entire clusters of pairs from candidacy as a tradable pair. For each cluster, a rolling analysis over the training period was conducted. For each month, 250 days of lookback data were considered and the highest possible Kendall tau rank correlation using all stock permutations within the cluster was recorded. The median value of the recorded Kendall tau correlations was used to sort clusters, with higher correlations implying a more suitable candidate cluster. All permutations from the top candidate clusters were analyzed by trial and error for maximal returns during the training period.

3.2 Structural Break Prediction

Pairs trading strategies typically assume that stationary spreads revert to the historical mean. Because of this, events when the spread unexpectedly becomes non-stationary, known as a structural break, can present significant challenges to implementing profitable pairs trading strategies. Due to the underlying assumptions of pairs trading strategies, such events may lead to significant losses, even with risk management measures like stop losses and/or trading boundaries, as the spread may not return to the historical mean after a structural break. To prevent such situations from negatively impacting our strategy's performance, we first train a model to predict the probability of a structural break occurring. These probabilities are then used as an important indicator in our pairs trading strategy, in order to better guide our algorithm's trading decisions, and become more robust to high-risk situations like structural breaks. To do this, we must use the time-series data, spreads, and other relevant data sources related to our chosen pairs, and learn an abstract representation that can help predict the incidence of a structural break during live trading. We identify two primary modalities of the data, from which we learn these representations: frequency and time.

3.2.1 Frequency-Domain Representation Learning

One common approach in signal processing applications of frequency-domain representation learning is using a Fourier transform. Many related studies have attempted to learn features for time-series prediction in the market using Fourier transforms and their variations. When considering the unique structural break prediction problem, it is clear that any method employed must have a high-temporal resolution in its frequency-domain representation. Otherwise, valuable information about when specific frequencies occur

will be lost. In predicting structural breaks, it is important to know the temporal patterns of frequencies from a signal's spectral decomposition, as opposed to a time-invariant power spectrum. Scalograms obtained from Fourier transforms, despite their robustness to noise and their ability to represent meaningful frequency representations of time-series data, contain no temporal resolution. In contrast to Fourier transforms; however, continuous wavelet transforms derive their frequency-domain representations of a signal by continuously changing, scaling, and shifting some wavelets, and CWTs have been shown to be robust to temporally-varying time-series data.

Because of this, we choose to learn CWT scalograms of spreads, using which we train a convolutional neural network (CNN) to extract frequency-domain features of time-series data.

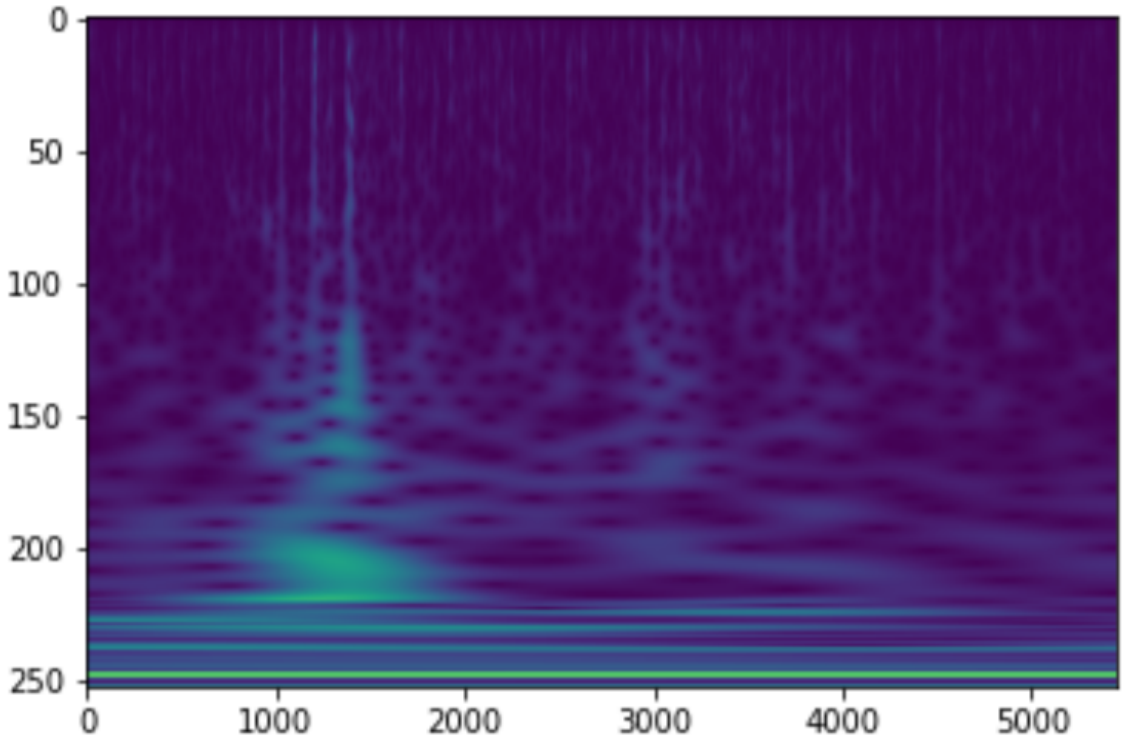


Figure 3: Example scalogram output of continuous wavelet transform on (SPG, EPR) pair

We train a CNN on these scalograms, as opposed to other models designed for time-series data, like recurrent neural networks (RNN) and/or long short-term memory networks (LSTM), primarily due to its more efficient abstractions of important spatial and/or temporal features in an image. Features unique to a CNN, like sparse connections, parameter sharing, and convolutions, significantly reduce the number of parameters and computational complexity of the model, thereby improving its generalization ability and out-of-sample (OOS) performance. Furthermore, simple, interpretable abstractions from scalograms are sufficient for the purposes of frequency-domain representation learning. In contrast, more complex models like RNNs or LSTMs should only be used when necessary due to their size, and potential for overfitting.

Furthermore, we considered training capsule networks on these scalograms, as they have been theoretically shown to be better at capturing spatial resolution in their abstract

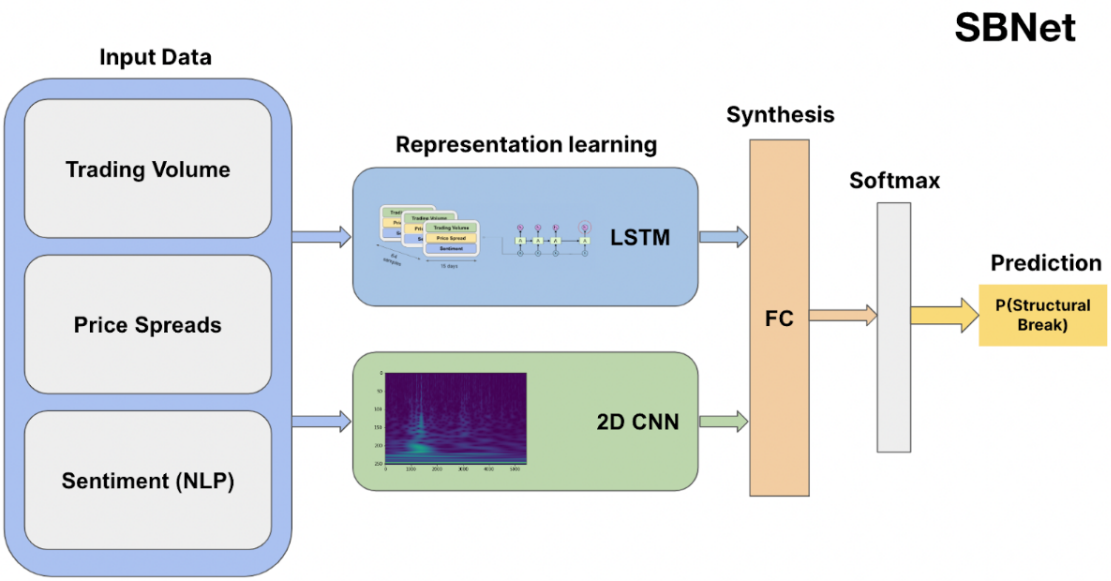
representations. Given the purpose of these networks is to train using scalograms that spatially represent temporal frequency patterns, having a spatial resolution is an essential feature of a network trained on such images. However, we found that the improved information captured in the learned features using capsule networks was not worth the significant increase in computational complexity that it poses relative to traditional CNNs, for the same reasons detailed above. Because of this, we chose to train a lightweight, robust CNN with only 2 convolutions and 2 max-pooling layers (to avoid overfitting and reduce computational effort), as it is especially important in trading applications for machine learning models to be explainable, simple, and generalizable. The input to this CNN is a partial scalogram, i.e., a fixed window size (d=9) of day-to-day time points within the full scalogram obtained from the entire training period (1/1/2017-1/1/2021). These partial scalograms represent the raw frequency-domain features from the given pair over every size-d daytime window (t to t+d, t+1 to t+1+d, etc.). To further reduce the likelihood of overfitting, we also implemented dropout regularization between the two convolution layers, with a probability parameter of 0.35 (i.e., a 35% chance for any node in the network to be ignored during training, simplifying the model).

3.2.2 Time-Domain Representation Learning

Next, we must learn time-domain features, as stock market data is inherently sequential and temporally rich. One of the components of our model to predict a structural break is an LSTM. LSTMs and RNNs are good models for learning time-domain representations because they feed the hidden states of the previous time step into the next time step, so they can "build up memory" of previous events. On the other hand, models like CNNs used for the feature-domain representation are better for detecting patterns in different areas of space (i.e., an image). Since we want to learn from a sequence of time-based events, using RNNs and LSTMs is the best choice. We choose to use an LSTM to avoid the Vanishing Gradient Problem in order to still learn from long-range dependencies. Our LSTM learns from the following features: z-score, trading volume, and sentiment. We pass in sequences of 15 entries to the LSTM (this corresponds to 15 days) for it to examine both short and long-term dependencies.

3.2.3 Joint SB Model (SBNet)

To train both time-domain and frequency-domain networks jointly, we concatenate the flattened output of the convolutional neural network with the final hidden state of the LSTM and apply a fully-connected linear layer followed by the sigmoid activation function to achieve a final result between 0 and 1, that indicates the predicted likelihood of a structural break. The final architecture for structural break prediction is shown here:



3.3 Copula-based Pairs Trading

3.3.1 Statistical Background

Copula is a statistical method to characterize a joint distribution. This section briefly describes the bivariate copulas used in pairs trading. For two random variables X and Y whose CDF are F and G respectively, define $U(X) = F(X)$, $V(Y) = G(Y)$, the marginal distribution of U and V are both uniform on $[0, 1]$. Let the joint CDF of U and V be $C(u, v) = P(U \leq u, V \leq v)$, where $C(\cdot, \cdot)$ is the copula function that incorporates the dependency structures between X and Y through that of U and V. Popular choices of the copula function include the Clayton, Gumbel and Frank copula formulas in Archimedean copulas family. All of these copula functions possess the computational and inferential advantage in that their parameter estimation can be easily carried out by estimating the Kendall rank correlation tau. In practice, choosing the appropriate copula function could be carried out in a frequentist model selection flavor by minimizing the AIC.

3.3.2 Application to Pairs Trading

For a given pair of stocks X and Y, we model their log return $\{(R_X, R_Y)_t\}$ with the bivariate copula. During the trading period, at each time point, the current log return (R_X, R_Y) is first transformed to (u, v) with their empirical CDF. Then, with the estimated copula function C, the fitted copula is used to determine the "mispriced index" MI, defined as $MI_{x|y} = \frac{\partial C(u,v)}{\partial v}$, $MI_{y|x} = \frac{\partial C(u,v)}{\partial u}$.

The mispriced index is an estimate of the conditional CDF, which represents the extent to which the current return of one stock deviates from its predictive distribution given the current return of the other. It has a similar interpretation as the p-value. For example, $MI_{x|y} = 0.01$ means that given the current return of stock Y, the probability that the return of X is smaller than the observed R_X is 1%, indicating the current R_X is on the left tail of its conditional distribution, hence X is undervalued. If $MI_{y|x}$ is extremely large at the same time, then Y is overvalued, and we should short the spread. Formally, we short the spread if $MI_{x|y} < 0.05$ AND $MI_{y|x} > 0.95$, and long the spread if $MI_{x|y} > 0.95$ AND

$MI_{y|x} < 0.05$.

3.3.3 Comparison with Conventional Approaches

Conventionally, the joint distribution of the returns for a given pair is modeled by a Gaussian distribution. However, a key detriment to this approach is its inability to model pertinent behaviors like tail dependence. Tail dependence measures the degree of dependence between the extreme values of two random variables. This behavior can describe the likelihood that two prices will move in the same direction even as the individual prices approach their respective distribution tails. Positive tail dependence indicates that the two stocks are more likely to move in the same direction as the values become extreme (convergence). Conversely, negative tail dependence occurs when the two stocks are less likely to move in the same direction as the values become extreme (divergence).

The idea of tail dependence is especially relevant to implement proper risk management, as it can help us understand the worst-case risks associated with extreme events and make more informed parametrizations of risk.

3.3.4 Our Implementation

Our pairs trading strategy implementation relies upon the predictions of our SBNet, the results of our clustering to determine which pairs to trade, and the Copula-based Pairs Trading algorithm.

The SBNet serves as an additional data point to help us determine whether or not a structural break is likely to happen, which can be extremely costly and lead to a high drawdown. If the likelihood of a structural break is at least 90%, even if other indicators recommend pairs-trading the equity, we do not trade the given pair to prevent big losses.

4 Risk management

We considered comprehensive risk management approaches throughout multiple stages of this trading project, from stock selection to position sizing and exiting conditions, etc.

4.1 Kelly's Principle

We implemented Kelly's principle in the algorithm that can be used to determine the weight of the pairs. Since the original Kelly's Principle is not for portfolio construction, we adopted the formula from Raposa (Aug 9th, 2021) to determine each pair's weight inside the portfolio. So that this calculation can also be effective when we expand the strategy to trade multiple pairs simultaneously in the future. Mean return, risk-free rate, and standard deviation are the three inputs of this formula. We calculate each pair's mean return and standard deviation with the last 20 days' data. The risk-free rate is set to be 3.1%. This number is the average US 10-year treasury bonds' return in the last 4 months.

$$\text{pair weight} = \frac{\text{mean return} - \text{risk free rate}}{\text{standard deviation}}$$

4.2 Stop-loss limit and kill switch

4.2.1 Daily stop-loss limit

For daily loss control, when the unrealized loss of a pair exceeds the daily loss limit (In our model, we set the limit as 20%), it gets liquidated. Since currently, our model only trades one pair at a time, the overall daily loss will be limited to 20% as well. The overall daily loss will decrease if we allow more pairs to trade simultaneously. Thus, the overall daily loss limit requirement is guaranteed. Moreover, as long as the pairs are not grossly dependent, there is no rationale to liquidate a pair simply due to the huge loss incurred by other pairs, hence such pair-level risk control in this approach is desirable.

4.2.2 Drawdown limit

In order to minimize the volatility of our portfolio, we control our drawdown to be lower than 20% in backtests by using the SetRiskManagement() function with the argument of MaximumDrawdownPercentPerSecurity() being passed into Quantconnect.

Besides, in our model, we took the 'kill switch' into consideration by adding stop loss and drawdown limits. The reason why we did not apply any sophisticated 'kill switch' strategies to our model is that our trading strategy would identify the structural breaks and buy/sell our holdings before 'kill switch' strategies. Thus, we only implemented basic 'kill switch' strategies in the risk management part. In addition to directly liquidating the holdings at the stop-loss limit, we took more sophisticated approaches that reduce the position size that depends on measurements of risk, as detailed in the following sections.

4.3 Trading cost limit

Given that the proposed strategy performs trading with daily resolutions, controlling the trading cost is crucial. Here we adopted an ad hoc approach that limits the number of trades of each pair to 10 per day (Potentially, one can treat this number as a tuning parameter and make the proposed model "cost aware" by directly incorporating the trading cost in the loss function.). However, the limit doesn't apply to the trade for liquidation or the weight adjustment determined by the Value-at-Risk. Since liquidation will happen only when the equities have positions. The limit for the Value-at-Risk is set by only allowing the algorithm to check the Value-at-Risk once per day. Thus, the algorithm can only make 1 adjustment per day to rebalance the Value-at-Risk of the portfolio.

4.4 Margin Call limit

In order to avoid causing warnings of insufficient capital, we add a margin call limit to our model, so that every time the algorithm tries to make a trade, it has to check whether the trading weights break the margin call limit or not.

4.5 Value-at-Risk

4.5.1 Scaling positions to risk

At the time t, we define the following quantities:

$$T(t) : \text{the total value of the portfolio, including cash}$$

$$P(t) : \text{the current position (the total value of the stocks)}$$

$$C : \text{the amount of cash in the portfolio. At } t = 0, \text{ we have } T(0) = C + P(0)$$

$$r(t) : \text{return relative to the previous time point}$$

At time $t' = t+1$, the return is $r(t') = \frac{P(t')+C}{P(t)+C} - 1$. We use the log return and this fractional return interchangeably in this project for mathematical and computational ease. For any given level α (usually 95%), based on historical return, we can calculate the Value-at-Risk statistic r_α, which is the α-percentile of the return at time t'

$$t' : Pr(r(t') \geq r_\alpha) = 1 - \alpha \tag{1}$$

which could be rewritten as

$$Pr(\frac{P(t') + C}{P(t) + C} - 1 \geq r) = 1 - \alpha \tag{2}$$

For risk management purposes, we set a lower bound r' of such percentile. If at time t, we have $r > r^*$, then we reduce our position to a fraction of the current position to make the α-percentile of the return at time t' equal to the limit r^*, while keeping the relative proportion of the pairs unchanged.

Mathematically, at time t, we change our position from $P(t)$ to $kP(t)$ so that we obtain $(1 - k)P(t)$ cash. Then at time t', we have

$$T(t') = kP(t') + (1 - k)P(t) + C \tag{3}$$

and we need to find the fraction k such that the return (after reducing the position) satisfies

$$Pr(\frac{kP'(t) + C + (1 - k)P(t)}{P(t) + C} - 1 \geq r^*) = 1 - \alpha \tag{4}$$

Assuming our reduction of position does not change the distribution of the random variable P(t'), combining (2) and (4) we have

$$(r + 1)(P(t) + C) - C = \frac{1}{k}(r^* + 1)(P(t) + C) - C - (1 - k)P(t)$$

which gives $k = \frac{r}{r^*}$.
In our implementation, the calculation of VaR and the adjustment of portfolio weights is carried out on a daily basis.

4.5.2 Inference strategies for VaR

We implemented two ways of evaluating the Value-at-Risk statistic r_α using historical returns, namely the Gaussian method and the historical method. Both methods assume that the distribution of $r(t')$ is the same as the distribution of historical returns. The Gaussian method posits the returns follow Gaussian distribution and estimates the mean and variance from historical data, then the VaR statistic is available analytically. The historical method uses the α-percentile of historical returns as an estimate of the Value-at-Risk statistic r_α. Theoretically, the historical method is more robust to extreme values but requires more data.

In our algorithm, we use a lookback period of 30 days for VaR calculations to capture the short-term distribution. A longer lookback period provides more data and thus a more stable inference on VaR if the returns are homogeneous during the lookback period. On the other hand, a shorter lookback period yields better inference in case of sudden changes in the distribution of returns (e.g., caused by drastic changes in market condition) since it is more responsive to the "local" data.

Even though there are limitations in the VaR inference methods and tradeoffs in the choice of lookback period as discussed, we note that the goal of incorporating VaR in the algorithm is risk management, so a reasonable inference strategy for VaR that controls the potential loss to some extent should suffice. A more sophisticated inference strategy of VaR is possible but not the primary focus of this project.

4.6 Limiting the Number of Trades

We limit the number of trades each day to 10 trades maximum. Performing many trades per day is risky and can amount to an increase in fees. Thus, we set a hard limit on the number of trades we perform.

4.7 Risk quantification for ML model

The proposed strategy is based on machine learning models and is overparameterized, making it hard to analytically obtain a lower bound on the performance in case of extreme market conditions. Thus we test the proposed algorithm under different market regimes including highly irregular conditions to obtain lower bounds on performance. We also used March 2020 as a stress testing period. Additionally, our usage of ML is to detect and avoid potentially risky trades, and we never enter a trade based on the results of our ML model. Thus, while we may miss out on a big gain due to the result of an ML model, we never directly lose money because of it. The testing period and results are detailed in the Results section.

5 Model Training

The time/frequency representations, as well as their linear combination, are optimized using the Adam optimizer, with a learning rate of 0.005, and a batch size of 64. We use binary cross entropy loss to train and optimize our model, as it is equivalent to fitting the

model using maximum likelihood estimation (i.e. minimizing the dissimilarity between the empirical data distribution and the distribution induced by the model). The data is randomly split so that around 80% comprises the training dataset and 20% comprises the testing dataset. After training for 300 epochs, we achieve an average loss of 0.075 on the testing dataset.

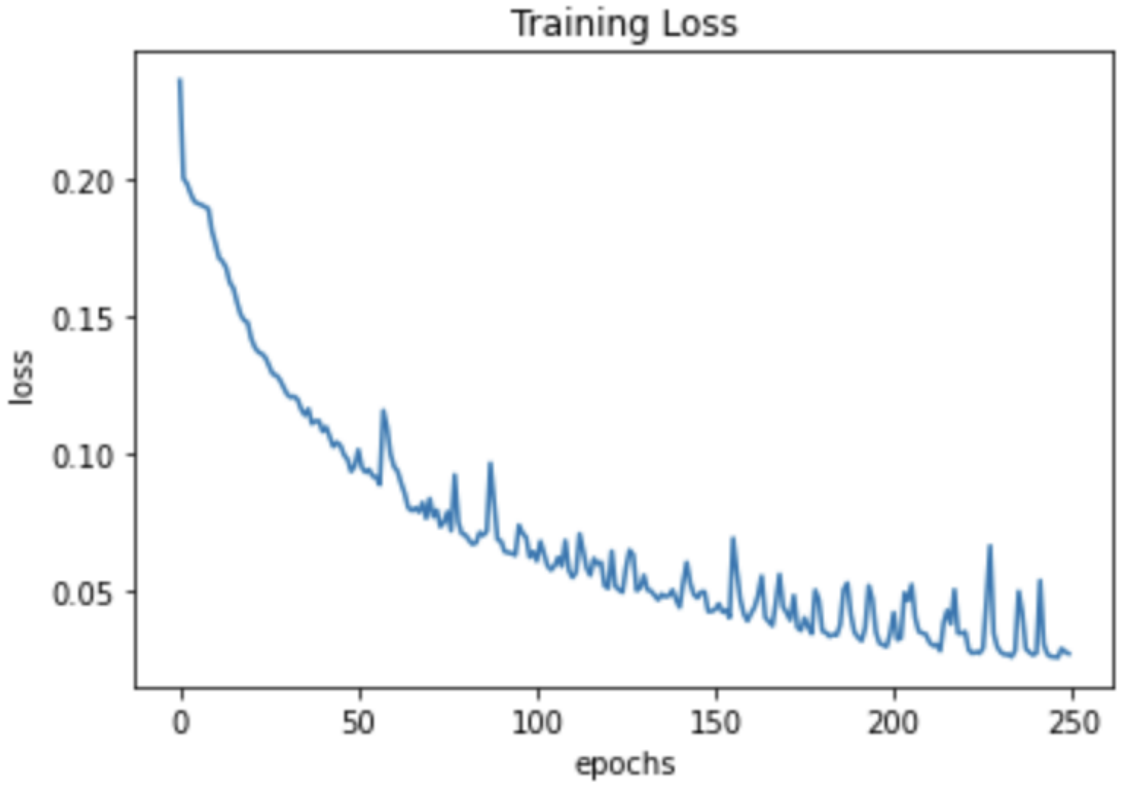


Figure 4: Average training loss after 250 epochs

6 Results

6.1 Backtesting Settings

Our backtesting period consists of many years of history when considering both the in-sample training period and the out-of-sample testing period. The training set is 1/1/2017 - 1/1/2021. Three testing periods are used to gain a more comprehensive evaluation of the performance of the proposed strategy under different market conditions:

1. In-Sample (Training) period: 1/1/2017 - 1/1/2021
2. Out-of-sample period A: 1/1/2022 - 11/1/2022
3. Out-of-sample period B: 1/1/2016 - 1/1/2017
4. Out-of-sample period C: 1/1/2010 - 1/1/2011
5. Stress Testing period: 3/1/2020 - 4/1/2020

6.2 Backtesting and Live Trading Results

We observe that our model does significantly better when we use our machine learning model to predict structural breaks with the incorporation of sentiment data. Here is the performance of our model when we do not avoid trades during a predicted structural break:

Table 1: Results WITHOUT SBNet

Period	Return	Sharpe ratio	Drawdown
IS	83.910%	0.725	19.400%
OOS A	26.227%	1.786	7.700%
OOS B	3.148%	0.262	10.200%
OOS C	8.125%	0.693	4.200%
Stress Test	0.848%	0.398	15.300%
Live trading	0.062%	6.944	0.000%

Here is the performance of our model when we avoid trades during a predicted structural break by using our machine learning model:

Table 2: Results WITH SBNet

Period	Return	Sharpe ratio	Drawdown
IS	141.73 %	1.082	18.400%
OOS A	20.03 %	1.367	7.700%
OOS B	18.62 %	1.352	5.800%
OOS C	6.90 %	0.627	4.200%
Stress Test	0.85 %	0.398	15.300%
Live trading	0.06 %	6.944	0.000%

* In the original design of the performance evaluation procedure, Nov 2022 is a live paper trading period. Here we substitute this with the first week of December without further parameter tuning or modifications on the strategy to mimic live trading due to waiting on an API provider for sentiment data.

Figure 5: Strategy Equity During In Sample Period

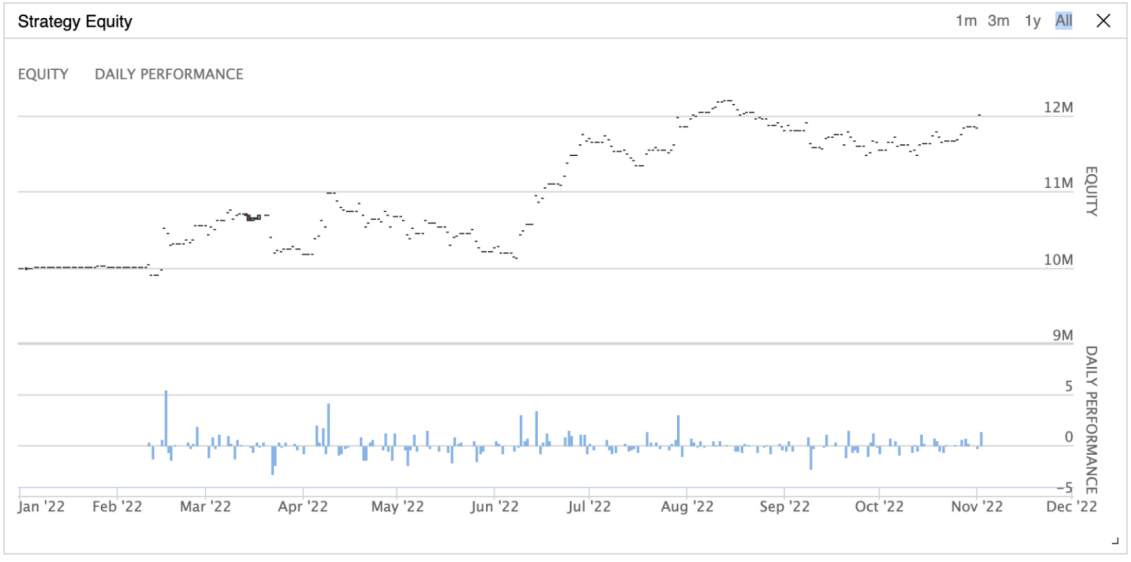


Figure 6: Strategy Equity During Out of Sample Period A

Figure 7: Strategy Equity During Out of Sample Period B

Figure 8: Strategy Equity During Out of Sample Period C

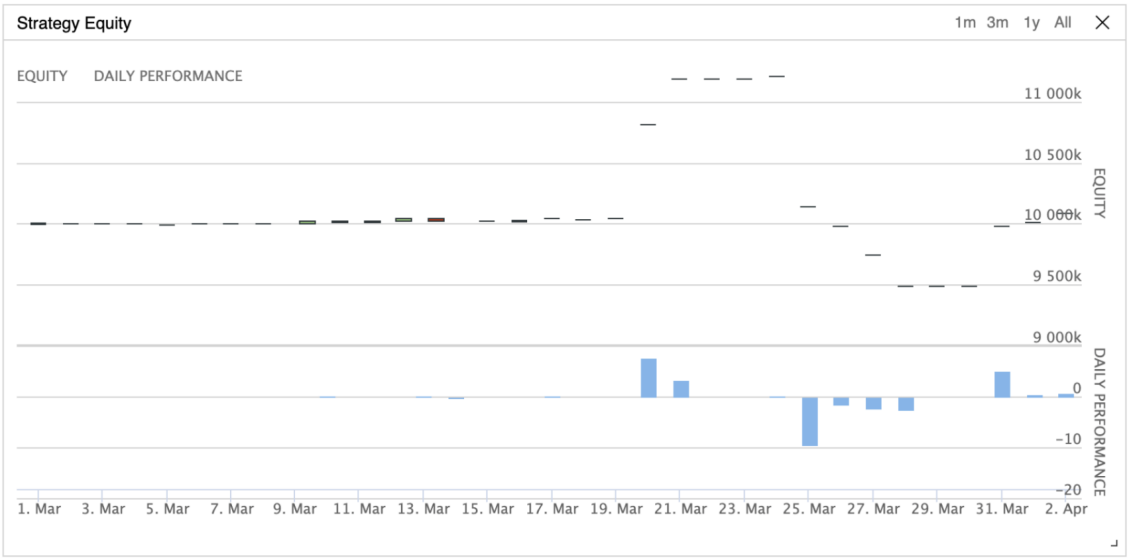


Figure 9: Strategy Equity During Stress Test

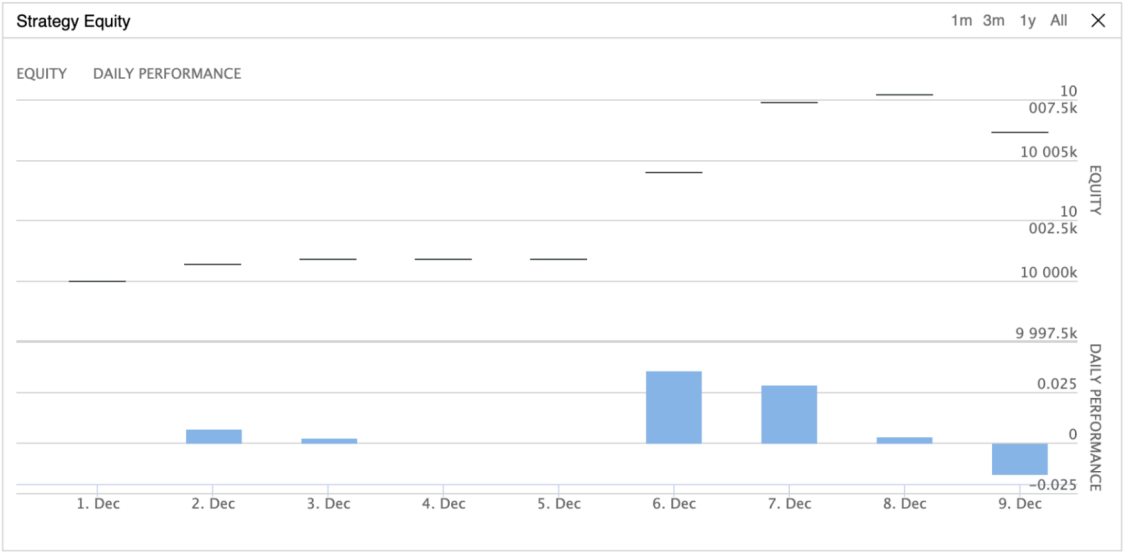


Figure 10: Strategy Equity During Live Trading

In the in-sample (IS) period, the high return of 141.73%. indicates strong performance, and the Sharpe ratio of 1.082 suggests that the returns were reasonable compared to the level of risk taken. However, the relatively high drawdown of 18.400% indicates that there is potential to reduce the requirements for what we consider a structural break in order for SBNet to recommend not trading more often.

In the first out-of-sample (OOS A) period, the return of 20.03% is lower than the IS period, but the Sharpe ratio of 1.367 is higher, indicating that the strategy was still performing well in terms of risk-adjusted returns. Additionally, the lower drawdown of 7.700% suggests that the portfolio volatility was better managed via structural break predictions during this period compared to the IS period. In the second out-of-sample (OOS B) period, the return of 18.62% is lower than the previous two periods, and the Sharpe ratio of 1.352 is also slightly lower, indicating that the strategy performed comparably to the OOS A period. The drawdown of 5.800% is actually lower than in the previous two periods, indicating that our model was able to manage structural breaks even more effectively. In the third out-of-sample (OOS C) period, the return of 6.900% and Sharpe Ratio of 0.627 is lower than the previous OOS periods. However, our drawdown of 4.200% is also lower than in the previous periods, indicating an abundance of structural-break opportunities and/or good detection of structural break environments. We note that while the performance is lower in OOS C, this might also be due to the time period being very different.

Under the stress test, the return of 0.85% is lower than in any of the previous periods, and the Sharpe ratio of 0.398 is also lower, with very minimal trading activity. The drawdown of 15.300% is higher than most of the previous periods, due to missed pair divergences.

During live trading, the return was 0.06%. During this small period of trading, not many structural-break opportunities were present. The Sharpe ratio of 6.944 is extremely positive which is nice but is not reflective of the algorithm's true performance due to the small sample duration. The drawdown is 0.000%, lower than in all preceding periods, indicating reduced volatility. Overall, the results from live trading are unclear since we need ample exposure to conditions with and without structural break conditions before our model becomes statistically viable.

7 Discussion

During in-sample and out-of-sample periods, our algorithm demonstrated relatively strong returns, underscoring the potential for lightweight, deep-learning-based methods for risk management in market-neutral arbitrage strategies. By incorporating stringent risk management techniques, like structural break prediction, drawdown limits (all drawdowns in IS/OOS periods $< 20\%$), and stop losses, our backtesting results demonstrate an overall healthy risk efficiency (> 0.6 Sharpe for all regular periods), with strong returns ($> 6.8\%$ compounding annual return for all regular periods).

During the stress testing period, our algorithm experienced limited trading signals due to

the high resemblance of market behavior to a structural break. This results in only a few trades being taken during the entire month, with a small positive profit. The drawdown was also manageable, at 15.1% which mainly occurred on a single day. This demonstrates the robustness of our algorithm in mitigating downside risk potential during "black-swan" or tail events in the market. The ability to detect structural breaks thus raises profitability by cutting down the magnitude and number of losing days within the trading period.

Overall, the ideas developed in this report demonstrate strong potential for the use of deep learning models for risk mitigation strategies in a regime-shifting environment like pairs trading of equities.

8 Future Work

8.1 Hyperparameter tuning

An integral part of machine learning is determining what hyperparameters you should use. Hyperparameters include things like the number of hidden layers in the LSTM or the learning rate, for example. In the literature, this is usually achieved by having a validation set and a test set. The validation set is used to help determine what values the hyperparameters should have, and the test set is used at the very end to give a final performance metric of how well the model does.

8.2 Integration of a Higher Quality Data Stream

A lot of our mid-cap and small-cap stocks had missing data, and even some data points were missing on our large-cap stocks such as AAPL and GOOGL too. This problem was made worse when we considered minute data. Given these limitations, we decided to use daily stock data on larger cap stocks to avoid the issue of missing data. If given a better data stream though, we might have been able to consider a broader portfolio of stocks and trade minutely.

8.3 Better Pair Selection

Choosing a good pair is at the heart of our strategy. If given more time, we could have dedicated even more time to looking through all of the possible pairs and choosing the one that has better performance. It would be especially helpful to choose a pair that is more liquid.

8.4 Trading Multiple Pairs at Once

While we do consider multiple pairs and switch between pairs at the start of every month, we do not trade multiple pairs at the same time. In the future, it would be helpful to trade multiple pairs for diversification and better risk management.

8.5 Risk analysis

The risk quantification approach could be further extended in three directions: (1) including backtesting on simulated datasets that have particular pathological conditions, for

example, frequent structural breaks and/or pairs of stocks with almost zero correlation; (2) sensitivity analysis: assessing the extent to which the estimated parameters change on different testing set and/or different conditions; (3) using more intensive experiment design and Gaussian process surrogates to identify factors that have a significant impact on performance.

9 Acknowledgments

We would like to thank Dr. David Ye for his extremely helpful advice and teachings throughout the semester. He provided key insights on how to improve our strategy and incorporate better risk management protocols. Furthermore, he enabled us to access a data source for sentiment data.

10 Appendix

Code relevant to this report can be found here.

References

[1] "QuantConnect Tutorials." Tutorials - Strategy Library - Pairs Trading-Copula vs Cointegration - QuantConnect.com, QuantConnect, https://www.quantconnect.com/tutorials/strategy-library/pairs-trading-copula-vs-cointegration.

[2] Raposa, Aug 9th, 2021, How To Improve Your Trading System With The Kelly Criterion. Retrieved from https://raposa.trade/blog/how-to-improve-your-trading-system-with-the-kelly-criterion/ in Aug 9th, 2021.

[3] Lu, JY., Lai, HC., Shih, WY. et al.(2022). Structural break-aware pairs trading strategy using deep reinforcement learning. J Supercomput 78, 3843–3882 .

[4] Sabour, S., Frosst, N., Hinton, G. E. (2017). Dynamic routing between capsules. Advances in neural information processing systems, 30.

[5] Liu, F., Cai, M., Wang, L. Lu, Y. (2019). An Ensemble Model Based on Adaptive Noise Reducer and Over-Fitting Prevention LSTM for Multivariate time-series Forecasting. IEEE Access, vol. 7, pp. 26102-26115.