Computational Economics
https://doi.org/10.1007/s10614-024-10770-7

Effective Convergence Trading of Sparse, Mean Reverting Portfolios

Attila Rácz[1] · **Norbert Fogarasi**[1]

Accepted: 22 October 2024

Abstract
This paper introduces an effective convergence trading algorithm for mean reverting portfolios using Long Short Term Memory (*LSTM*) neural networks. Utilizing known techniques for selection of sparse, mean reverting portfolios from asset dynamics following the *VAR*(1) model, we introduce a 2-step technique to effectively trade the optimal portfolio. Sequence-to-sequence (*Seq*2*Seq*) *LSTM* architecture is implemented to make longer term prediction of future portfolio values and establish a trading range. In addition, a simple *LSTM* network is applied to predict very precisely one time step ahead. Combining these two constructions, a sophisticated convergence trading algorithm is implemented which produced Sharpe ratios around 1.0 on optimal portfolios selected from historical *S*[*NONSPACE*]&*P*500 stocks during 2015 – 2022. This represents a very significant improvement compared to the previous convergence trading algorithms on the same set of portfolios by around 141% on average.

Keywords Mean reversion · Portfolio optimization · Sequence-to-sequence · LSTM · trading

1 Introduction

Finding accurate techniques for predicting financial time series is of extraordinary interest among researchers. There is extensive literature studying the use of deep learning methods to forecast financial time series. However, stock price forecasting is quite challenging because of the significant noise, non-linearity, and volatility. Extensive literature is based on deep-learning forecasting model. (Zaheer et al.

✉ Attila Rácz
faus2s@yahoo.com

Norbert Fogarasi
fogarasi@hit.bme.hu

[1] Department of Networked Systems and Services, Budapest University of Technology and Economics, Budapest, Műegyetem rakpart 3, 1111, Hungary

Published online: 23 November 2024

2023; Zhang 2022; Wu et al. 2020; Yang et al. 2017; Tah 2018; Narula 2018; Banerjee et al. 2008). Vanilla recurrent neural networks (*RNN*) are unable to learn long sequences due to their property of exploding or vanishing gradients during the back propagation process (Schmidhube and Hochreiter 1997). A successful step toward finding accurate techniques, which was originally developed for natural language processing, is the *LSTM* networks. These networks were developed to handle this disadvantage as the expression *long short term* refers to this (Liu et al. 2020; Yadav and Jha. 2020; Chen et al. 2021). *Seq*2*Seq* architecture (Sutskever et al. 2014) was applied with *LSTM* networks which are able to predict longer time ranges using our predicted data to estimate the trading range to set trigger points to sell or buy portfolios (Khalil and Pipa 2022). The predicted information provided by single and *Seq*2*Seq LSTM* networks is used to build a trading framework. The attention mechanism was also incorporated to the architecture which is about the utilization of the historical information in order into enhance precision of prediction (Bahdanau et al. 2014; Luong et al. 2015).

In this paper, we propose to apply the above described techniques to the problem of effective trading of sparse, mean reverting portfolios. These portfolios were created using *VAR*(1) model fitted on stock price data and the weights of the portfolio are coming from eigenvectors related to the maximal eigenvalue of the regression matrix using *Simulated Annealing* (see Fogarasi and Levendovszky 2011).

In our previous work (Racz and Fogarasi 2021) we tried to forecast sparse, mean reverting portfolios using single *LSTM* networks and built a trading strategy that utilizes this prediction. However, it was shown during the tests that in most cases the trading ranges were estimated inaccurately, so few trading events occurred. The aim of this article is to build a neural network that is able to predict a more accurate trading range rather than rely on the estimation of the long term mean. A more complex trading logic was built that incorporates both the *Seq*2*Seq LSTM* network which predicts the range and the simple *LSTM* which is able to estimate upcoming portfolio value very precisely

The structure of the paper is the following:

- In section 2 we briefly discuss the optimal portfolio selection
- In section 3 we briefly discuss the concept of *LSTM*.
- In section 4 we discuss the concept of the *Seq*2*Seq LSTM* neural network and their use on mean reverting time series.
- In section 5 we discuss the trading strategy based on the predictions.
- In section 6 we compare the performance of the different techniques.

2 Selecting Optimal Sparse, Mean Reverting Portfolios

This section briefly explains the mean reverting processes and the modeling of the financial time series with *VAR*(1). We call a stochastic process mean reverting when in the long term the value of the process oscillates around its average value. When the price is below its long term mean it will more likely increase rather than decrease and vice versa. This assists to build a simple trading strategy and to estimate the

trading range for the portfolio. The mathematical model of this is called the Ornstein-Uhlenbeck process. We denote the price of our portfolio by p_t at time t, and by s_t^i the price of the i^{th} stock at time t. Our mean reverting portfolio p_t is composed by the linear combination of s_t^i's. The stochastic differential equation that drives the *Ornstein − Uhlenbeck* process is

$$dp_t = \lambda\big(\mu - p_t\big)dt + \sigma dW_t \tag{1}$$

where W_t is a *Wiener* process, λ is the speed of *mean reversion* and μ is the long term mean to which the process is reverting. The deterministic part of the SDE represents the property that the magnitude of attraction to the long term mean is proportional to the distance from the mean. The solution of the stochastic differential equation is:

$$p(t) = p(0)e^{-\lambda t} + \mu\big(1 - e^{-\lambda t}\big) + \int_0^t \sigma e^{-\lambda(t-s)} dW(s) \tag{2}$$

The expected value of Eq. (2) is

$$\mathbb{E}\big[p(t)\big] = p(0)e^{-\lambda t} + \mu\big(1 - e^{-\lambda t}\big) \tag{3}$$

and the variance is

$$\mathbb{V}\big[p(t)\big] = \sigma^2 \int_0^t e^{-2\lambda(t-s)} ds = \sigma^2 \frac{1 - e^{-2\lambda t}}{2\lambda}. \tag{4}$$

Consequently in very long term, the expectation converges to

$$lim_{t\to\infty}\mathbb{E}\big[p(t)\big] = \mu, \tag{5}$$

while the variance is

$$lim_{t\to\infty}\mathbb{V}\big[p(t)\big] = \frac{\sigma^2}{2\lambda}, \tag{6}$$

Note the variance is inversely proportional to the speed of mean reversion. In (Box and Tiao 1977; d'Aspremont. 2011) the concept of predictability was introduced as follows:

$$\nu = \frac{\sigma_{t-1}^2}{\sigma_t^2} \tag{7}$$

where σ_t^2 is the variance of the time series. If the denominator in this equation is larger, p_t will be pure noise as t approaches infinity, making the time series completely unpredictable. Conversely, if the numerator is larger as t progresses, p_t will be perfectly predictable. Let the dynamics of the assets be modeled as discrete *vector autoregressive process* with parameter 1. As mentioned before $s_{i,t}$ denotes the price of the stock i at time t where $i = 1, \ldots, n$, where n is the number of stocks. The most general model of 8 is the non-stationary $VAR(1)$ model, which includes a

time-independent constant scalar shift term to describe drift or ensure positivity of the elements for all t:

$$s_{t+1} = c + As_t + W_t, \tag{8}$$

where A is an n by n real matrix constant over a certain time period, c is a time independent real scalar constant, W_t represents the noise or error term of the model with zero mean value, some constant variance, and uncorrelated across time. This can be rewritten in concise $VAR(1)$ notation by incorporating shift into the autoregression matrix:

$$s'_{t+1} = A's'_t + W'_t \tag{9}$$

extending the notations, (Racz and Fogarasi 2021)

$$\begin{bmatrix} s'^1_{t+1} \\ \vdots \\ s'^n_{t+1} \\ 1 \end{bmatrix} = \begin{bmatrix} a_{1,1} & \cdots & a_{1,n} & c_1 \\ \vdots & \ddots & \vdots & \\ a_{n,1} & \cdots & a_{n,n} & c_n \\ 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} s'^1_t \\ \vdots \\ s'^n_t \\ 1 \end{bmatrix} + \begin{bmatrix} W^1_t \\ \vdots \\ W^n_t \\ 0 \end{bmatrix} \tag{10}$$

where A' is an $(n+1) \times (n+1)$ matrix, the last column contains the constant shift c, the last row has 0's except the $(n+1)st$ element, which is strictly 1, x'_t is a vector with $n+1$ elements strictly 1 at the $(n+1)st$ element and W'_t represents the noise as before, with no noise for the $(n+1)st$ element. For simplicity, we will omitt the $'\prime'$ notation in this article. The autoregression matrix in Eq. (8) can be approximated using least squares regression as follows:

$$\hat{A} = \left(s^T_{t-1} s_{t-1}\right)^{-1} s^T_{t-1} s_t \tag{11}$$

Using this model a portfolio can be created with linear combinations of the assets. Let P be a real valued vector, representing the weights related to stocks. The time evolution of the portfolio's value can be written as

$$Ps_t = PAs_{t-1} + PW_t \tag{12}$$

Applying the definition of predictability (7) to the $VAR(1)$ model we get:

$$v(P) = \frac{var\left(P^T A s_{t-1}\right)}{var\left(P^T s_{t-1}\right)} = \frac{E\left(P^T A s_{t-1} s^T_{t-1} A^T P\right)}{E\left(P^T s_{t-1} s^T_{t-1} P\right)} \tag{13}$$

Maximizing predictability utimately becomes a generalized eigenvalue problem:

$$P_{opt} = argmax(v(P)) = argmax\left(\frac{P^T AGA^T P}{P^T GP}\right), \tag{14}$$

The argument of the $argmax$ operator is known as the Rayleigh quotient. Therefore

$$AGA^T P = \lambda \tag{15}$$

To keep the number of constituents low and minimize transaction costs while maintaining low portfolio complexity, the optimization problem now involves a trade-off between maximizing mean reversion speed and minimizing the number of stocks. Mathematically the Eq. 14 has an additional constraint, (Fogarasi and Levendovszky 2011)

$$P_{opt} = argmax(v(P)) = argmax\left(\frac{P^T AGA^T P}{P^T GP}\right), \quad subject\ to\ \mathbf{Card}(P) \leq k \tag{16}$$

Solving an optimization problem (16) can happen in several ways. The simplest way is the exhaustive method where the global minimum is found by sweeping through the whole configuration space. This is a brute force method, that runs through all possible configurations and selects the one with the highest generalized eigenvalue. The method is accurate with very high computational cost (d'Aspremont. 2011). An efficient alternative way to get near the optimal solution is the greedy search. The greedy method starts with the largest diagonal element in the matrix. Then it chooses a subspace such that the generalized eigenvalue is the highest among the others. This for sparsity $L = 1$ and $L = N$ should result in the same result as for the exhaustive method.

Simulated annealing (SA) is a probabilistic technique for approximating the global optimum of a given function. Specifically, it is metaheuristic to approximate global optimization in a large search space for an optimization problem. It is often used when the search space is discrete (e.g., the traveling salesman problem). For problems where finding an approximate global optimum is more important than finding a precise local optimum in a fixed amount of time, simulated annealing may be preferable to alternatives such as gradient descent.

This notion of slow cooling implemented in the simulated annealing algorithm is interpreted as a slow decrease in the probability of accepting worse solutions as the solution space is explored. Accepting worse solutions is a fundamental property of metaheuristics because it allows a more extensive search for the global optimal solution. In general, the simulated annealing algorithms work as follows. At each step, the algorithm randomly selects a solution that is close to the current one, it measures its quality and then it decides to move toward or to stay with the current solution based on either one of two probabilities between which it chooses on the basis of the fact that the new solution is better or worse than the current one. During the search, the temperature is progressively reduced from an initial positive value to 0 which affects the two probabilities: at each step, the probability of moving to a better new solution is either kept to 1 or changed to a positive value; on the other hand, the probability of moving to a worse new solution is progressively changed towards zero.

3 Architecture

3.1 Vanilla LSTM to Predict 1 Step Ahead

Recurrent neural networks are capable of predicting future prices of a portfolio using historical values. In particular, *LSTM* networks are useful for processing and forecasting time series (Schmidhube and Hochreiter 1997).

This specific network comprises a single hidden layer with 4 hidden units and a dense layer on top of the *LSTM*. The optimizer used is stochastic *Adam* (Kingma and Ba 2014), and the applied loss function is the *mean squared error*, (*MSE*). The network was trained to predict future values using four historical values.

As mentioned earlier, this single-layer vanilla *LSTM* network is solely used for predicting one time step ahead accurately. However, for more efficient trading strategies, estimation over longer time intervals is required.

3.2 Seq2Seq LSTM

An architecture developed originally in the field of natural language processing is called *Seq*2*Seq*. A special class of these problems is called a modeling problem, where the input as well as the output is a sequence. A typical *Seq*2*Seq* model has an encoder and a decoder part which are two different neural networks. The encoder network is trained to create a smaller dimensional representation from the input data. This representation is then forwarded to a decoder network which generates a sequence of its own that represents the output. A schematic representation of *Seq*2*Seq* architecture is shown Fig. 1.

Both the encoder and decoder were created using one single layer vanilla *LSTM* network with a *tanh* activation function. Parameters like the *number* of *hidden units*, *batch size* and the *learning rate* were optimized during the training process. On top of the decoder *LSTM* a *dense* layer was applied.

Figures 3, 4, 5 and 6 display the true and predicted data of the portfolio for several date ranges.

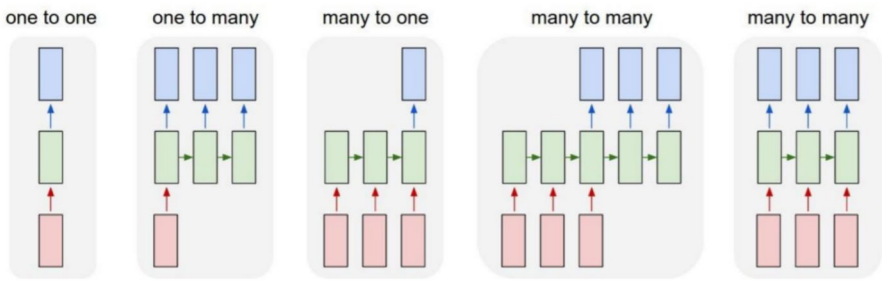


Fig. 1 Possible *seq*2*seq* constructions. Image source form Andrej Karpathy blog

Table 1 shows the weight of the portfolio constituents for a particular portfolio. The sign indicates long or short position for + and − respectively. Under the stock name column the *yahoo* tickers are displayed.

3.3 Attention Mechanism

The attention mechanism was introduced by (Bahdanau et al. 2014) to address the bottleneck problem that arises with the use of a fixed-length encoding vector, where the decoder would have limited access to the information provided by the input. This is thought to become especially problematic for long and/or complex sequences, where the dimensionality of their representation would be forced to be the same as for shorter or simpler sequences. One of the limitations of the simple *Seq2Seq* model is that only the last state of the encoder *RNN* is used as input to the decoder *RNN*. If the sequence is very long, the encoder will tend to have much weaker memory about earlier time steps. The attention mechanism can solve this problem. An attention layer is going to assign proper weight to each hidden state output from the encoder, and map them to an output sequence. Note that (see Bahdanau et al. 2014) the attention mechanism is divided into the step-by-step computations of the alignment scores, the weights, and the context vector:

Alignment scores: The alignment model takes the encoded hidden states, h_i, and the previous decoder output, s_{t-1}, to compute a score, $e_{t,i}$, that indicates how well the elements of the input sequence align with the current output at position t. The alignment model is represented by function, $a(.)$, which can be implemented by a feed-forward neural network:

$$e_{t,i} = a(s_{t-1}, h_i) \tag{17}$$

Weights: The weights, $\alpha_{t,i}$, are computed by applying a *softmax* operation to the previously computed alignment scores:

$$\alpha_{t,i} = softmax(e_{t,i}) \tag{18}$$

Context vector: A unique context vector, c_t, is fed into the decoder at each time step. It is computed by a weighted sum of all, T, encoder hidden states:

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_i \tag{19}$$

The *Luong* attention mechanism is an advanced method which was built on top of the *Seq2Seq LSTM* architecture, and uses *Dot* method to calculate alignment score.

The above detailed algorithms and methods were implemented in *Python* using *Keras* and *TensorFlow* packages.

To implement the attention mechanism we consider a weighted sum of the input from each time step of the encoder. The weights depend on the importance of that time step for the decoder to optimally generate the next value in the sequence.

The encoder compresses the sequential input and processes the input in the form of a context vector. We can introduce an attention mechanism to create a

shortcut between the entire input and the context vector where the weights of the shortcut connection can be changeable for every output.

Because of the connection between input and context vector, the context vector can have access to the entire input, and the problem of forgetting long sequences can be resolved to an extent Using the above-given information, the context vector will be more responsible for performing more accurately by reducing the bugs on the transformed data.

4 Trading Strategy

Let x_t be the actual value of the portfolio, p_t the predicted value, p_{min} and p_{max} the minimum and maximum of the time series predicted by the *Seq2Seq LSTM* neural network. The b_{max} and b_{min} are the upper and lower bounds of the trading range respectively, defined using p_{max} and p_{min} as:

$$b_{max} = p_{max} - R \tag{20}$$

$$b_{min} = p_{min} + R, \tag{21}$$

where

$$R = \alpha(p_{max} - p_{min}) \tag{22}$$

and α is a tolerance constant. For the purposes of our tests, we have fixed α at 0.05 based on visually observing asset price fluctuations. The single and *Seq2Seq LSTM* networks are applied in a two step trading strategy. In the first line of trigger the value of the portfolio hits the appropriate barrier of trading range defined by *Seq2Seq LSTM*. In the case of cash at hand, there is no event occuring when $b_{min} \leq x_t$, but an event is triggered when $b_{min} > x_t$. In the case of portfolio at hand, no action occurs when $x_t < b_{max}$ and the first trigger is hit when $b_{max} \geq x_t$. The second line of trigger eventually decides about trading event. Single *LSTM* has very precise prediction for one step ahead and we utilize this property when a trading event is triggered. Single *LSTM* checks if the predicted portfolio value moves away from trading range, (in case of cash at hand $x_t < p_{t+1}$) if so then it waits 1 time step to trade. Similar logic applies when there is a portfolio at hand $p_{t+1} \leq x_t$. The trading range is updated every 5 steps. This is a heuristic number. The lower and upper bounds of the trading range are symmetric to the mid of the predicted long term portfolio prices. In other words, at the very begining, one is being in state "Cash @ hand" and remains in this state until the first buying trigger (defined by the *seq2seq LSTM*) is activated by the portfolio value. The actual purchase occurs when the second level trigger(defined by the vanilla LSTM network) is activated by the portfolio value, transitioning to state "Portfolio @ hand". The logic for selling portfolio is basically similar to that for buying, except the portfolio must hit selling thresholds. The schematic diagram of the trading logic can be seen on Fig. 7.

Fig. 2 Example of *Seq2Seq LSTM* training and testing result. The red curve is the training input for the network, the blue is the testing data. The green curve is the fitted data for learning data set, the blue is the predicted based on the red input

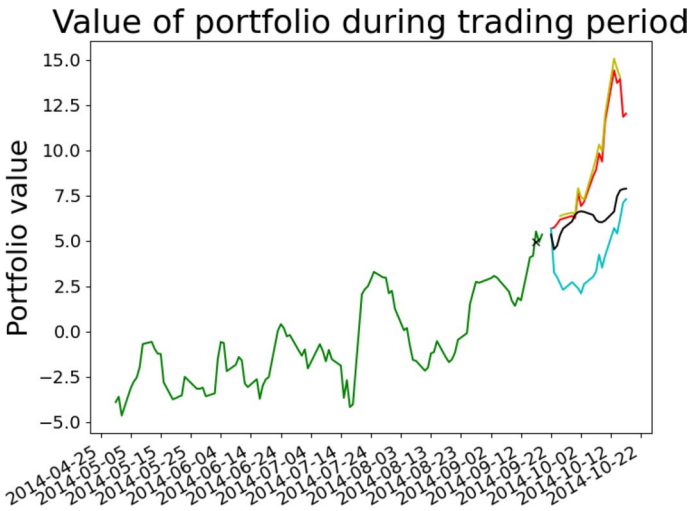


Fig. 3 Predicting optimized portfolio. The green curve is the calibration range, the red curve is the real value for calibration, the blue is the *VAR*(1) prediction, the black curve is the *Seq2Seq LSTM* predicted, the yellow is the online one-step *LSTM* prediction. The portfolio value is in arbitrary units as the optimization and prediction performed on the *log* of the real data

5 Performance Test

The network was trained on the same time range as used for *VAR*(1) calibration. The portfolio was created with the regression matrix and used as an input for the training. The number of *LSTM* layers is 4, batch size is 1, epochs is 100. We used 3 consecutive data points to predict. Figure 2 shows an example of the testing.

The performance testing was started with $\$10,000$ investment and the trading interval was 100 days. The number of tests were 500 and each used 400 day long calibration data set with different starting points (reasons for these parameters are in Racz and Fogarasi 2021). The tests were run on portfolios selecting

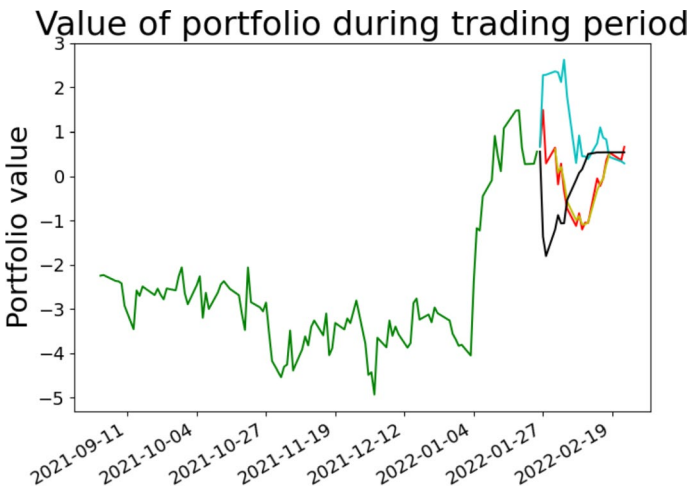


Fig. 4 Predicting optimized portfolio. The green curve is the calibration range, the red curve real value for calibration, the black curve is the *Seq2Seq LSTM* predicted, the yellow is the online one-step *LSTM* prediction. The portfolio value is in arbitrary units as the optimization and prediction performed on the *log* of the real data

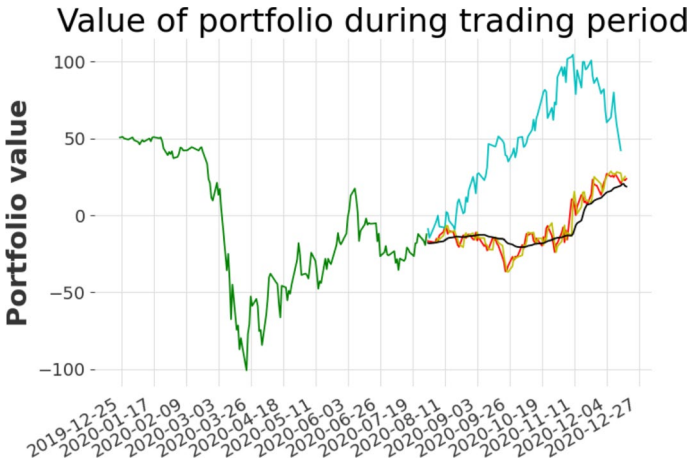


Fig. 5 Predicting optimized portfolio. The green curve is the calibration range, the red curve real value for calibration, the blue is the *VAR*(1) prediction, the black curve is the *Seq2Seq LSTM* predicted, the yellow is the online one-step *LSTM* prediction. The portfolio value is in arbitrary units as the optimization and prediction performed on the *log* of the real data

constituents from *S*[*NONSPACE*]&*P*500 pool from a time period from 1st of January 2015 to 31st of December 2022. Although this time period contains periods of rapid growth in the index this does not introduce a bias to our algorithm which is performed on a small subportfolio of the index, containing short positions also.

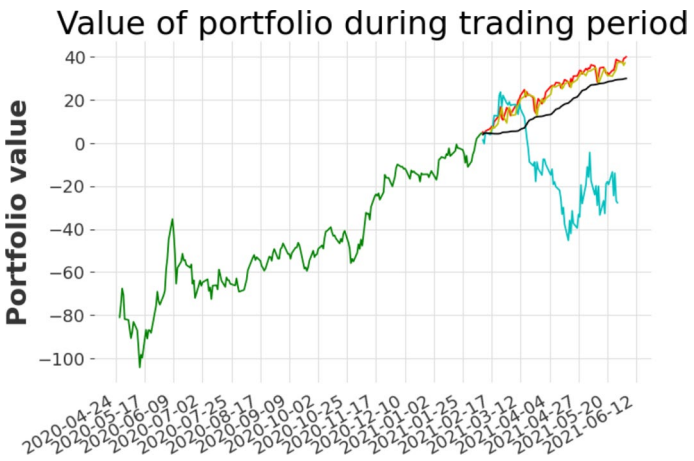


Fig. 6 Predicting optimized portfolio. The green curve is the calibration range, the red curve real value for calibration, the blue is the *VAR*(1) prediction, the black curve is the *Seq2Seq LSTM* predicted, the yellow is the online one-step *LSTM* prediction. The portfolio value is in arbitrary units as the optimization and prediction performed on the *log* of the real data

Table 1 Constituents of an example portfolio. The first column has the *Yahoo* tickers of stocks, the second shows the weights in portfolio and the third is the number of stocks

Stock Name	Weight	# Stocks in portfolio
PLD	−0.3247	−23
AMD	−0.0448	−31
AZO	0.383	1
MS	0.248	7
HD	0.24	−13
DE	−0.3807	51
WEC	0.1027	6
GPN	−0.1467	11
INTU	−0.6303	−22
VZ	0.203	11

The minus sign represents short position. The nominal value of the portfolio is $10*k*

The profit after the trading time range was calculated as the difference between the buying and selling price.

To see the impact on performance of the trading technique using the *Seq2Seq LSTM* neural network Fig. 8 shows histograms of returns of different techniques. The online single step *LSTM* trading strategies, the *Seq2Seq LSTM* without and the 100 days returns of the *S*[*NONSPACE*]&*P*500 portfolio on the same time period described above.

The mean, standard deviation and Sharpe ratio are calculated for two different sparsity values with and without the attention mechanism on Table 2.

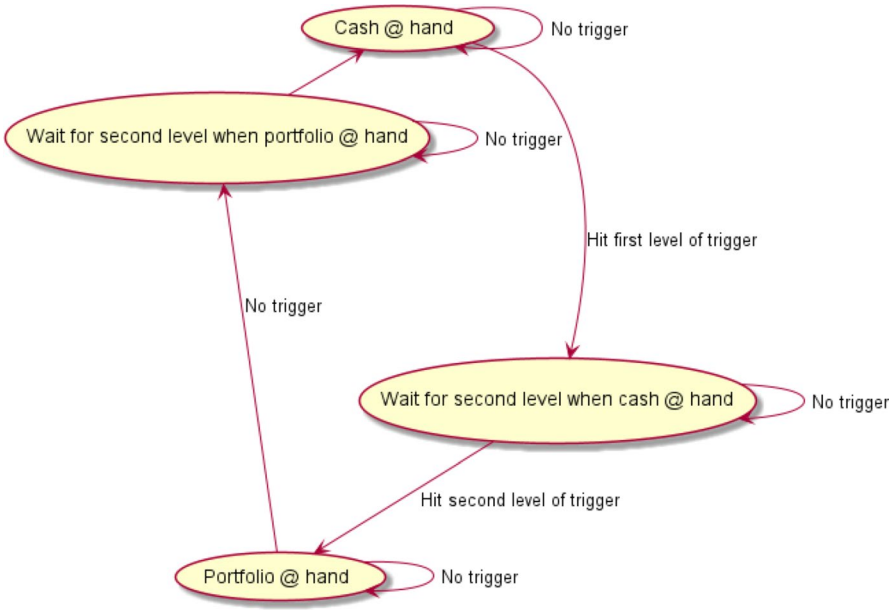


Fig. 7 Schematic diagram of trading logic. At the very beginning, one is being in state "Cash @ hand" and remains in this state until the first buying trigger (defined by the *seq2seq LSTM*) is activated by the portfolio value. The actual purchase occurs when the second level trigger(defined by the vanilla LSTM network) is activated by the portfolio value, transitioning to state "Portfolio @ hand". The logic for selling portfolio is basically similar to that for buying, except the portfolio must hit selling thresholds

The mean Shapre ratios, averaged across different sparsities are shown in Table 3.

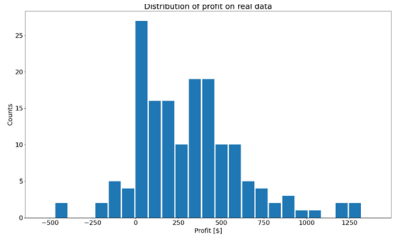
Figure 9 shows a summary comparison of the Sharpe ratios produced by the different methods.

The results of Table 3 are from (Racz and Fogarasi 2021) in which the definition of online and offline *Var* and *LSTM* strategies can be found. Table 3 shows poor trading performance as all the *Sharpe* ratios are below 0.5 and some of the techniques result in negative values. The method presented in this article provides *Sharpe* ratios around 1.0 as seen in Table 2 representing an average improvement of 141%.

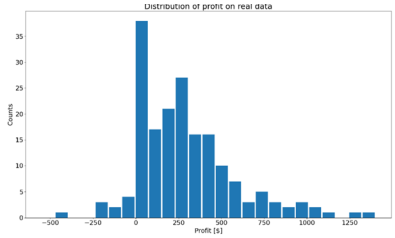
6 Conclusions and Directions for Future Works

We presented a two-level trading technique using two separate networks, a vanilla and a *Seq2Seq LSTM* network to be utilized on mean reverting portfolios. We tested the trading strategy on portfolios of $S[NONSPACE]\&P500$ stocks by fitting $VAR(1)$ model and optimizing predictability. Compared to our previous results we can see a significant increase in the *Sharpe* ratios from values below 0.5 to around 1.0. We did not observe a significant performance increase by adding an attention layer on top of the *Seq2Seq LSTM* network.

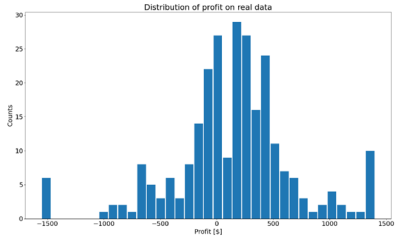
A future direction of work could be to create more complex prediction architectures such as stacked *LSTM* networks in both the encoder and the decoder.



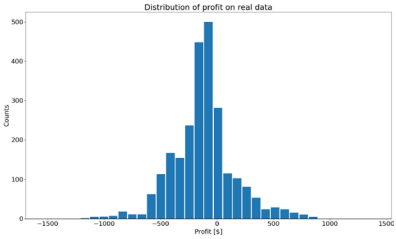
(a) The histogram of returns for regular network without attention.



(b) The histogram of returns for regular network with attention.



(c) The histogram of returns for Online LSTM.



(d) The histogram of returns calculated on S&P500 portfolio.

Fig. 8 Histrograms of returns of the investigated methods

Table 2 Mean, standard deviation and Sharpe ratio of the trading performance

Method	Mean	Std dev	Sharpe ratio	Expected shortfall
Luong average	287.96	293.31	0.982	−137.16
No att average	303.24	308.57	0.983	−153.57
S&P500 Buy&Hold	−119.4	268.19	−0.445	−226.79

Table 3 The Sharpe ratio of the previous methods averaged over the sparsities(number of stocks selected)

	Average sharpe ratios
Offline *VAR*	0.0132
Offline *LSTM*	−0.0556
Online *VAR*	0.166
Online *LSTM*	0.4068

Hyperparameter optimization for α, trading range and trading length. Further simulations and testing are required on different asset classes and markets to determine general applicability of our methodology outside of stock markets.

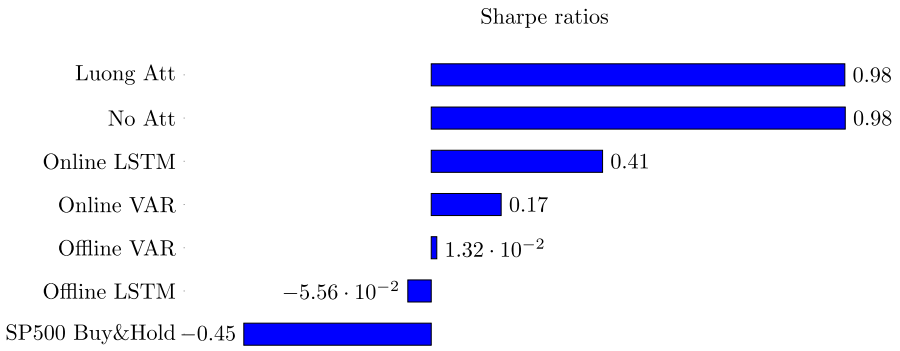


Fig. 9 The Sharpe ratio of the previous methods averaged over the sparsities

Funding Open access funding provided by Budapest University of Technology and Economics.

Declarations

Conflict of interest The authors have not disclosed any competing interests.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Analytic value function for a pairs trading strategy with a lévy-driven ornstein-uhlenbeck process. *Quantitative Finance, 20*(8), 1285–1306.

Banerjee, O., El Ghaoui, L., & d'Aspremont, A. (2008). Model selection through sparse maximum likelihood estimation. *The Journal of Machine Learning Research, 9*, 485–516.

Box, G. E., & Tiao, G. C. (1977). A canonical analysis of multiple time series. *Biometrika, 64*(2), 355.

Chen, Y., Fang, R., Liang, T., Sha, Z., Li, S., Yi, Y., Zhou, W., & Song, H. (2021). Stock price forecast based on CNN-BILSTM-ECA model. *Scientific Programming, 2021*(5), 1–20.

d'Aspremont, A. (2011). Identifying small mean reverting portfolios. *Quantitative Finance, 11*(3), 351–364.

Fogarasi, N., & Levendovszky, J. (2011). Sparse, mean reverting portfolio selection using simulated annealing. *Quantitative Finance, 11*(3), 351–364.

Khalil, F., & Pipa, G. (2022). Is deep-learning and natural language processing transcending the financial forecasting? investigation through lens of news analytic process. *Computational Economics, 60*(3), 147–171.

Kingma, P. D., Ba, L. J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980

Liu, C. M., Ta, V. D., & Tadesse, D. A. (2020). Portfolio optimization-based stock prediction using long-short term memory network in quantitative trading. *Applied Sciences, 10*(8), 437.

Luong, T., Pham, H., Manning, C.D. (2015). Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, (pp. 1412–1421)

Narula, I. (2018). Stock price randomness of brics nations. *International Journal of Public Sector Performance Management, 4*(2), 231–250.

Racz, A., & Fogarasi, N. (2021). Trading sparse, mean reverting portfolios using var(1) and LSTM prediction. *Acta Universitatis Sapientiae Informatica, 13*(2), 288–302.

Schmidhube, J., & Hochreiter, S. (1997). Long short-term memorys. *Neural Computation, 9*(8), 1735–1780.

Sutskever, I., Vinyals, O., Le, V. Q., Sha, Z., Li, S., Yi, Y., Zhou, W., Song, H. (2014). Sequence to sequence learning with neural networks. https://arxiv.org/abs/1409.3215

Tah, K. (2018). Random walk and structural break in exchange rates. *International Journal of Monetary Economics and Finance, 11*(4), 384–393.

Wu, L., Zang, X., & Zhao, H. (2020). Analytic value function for a pairs trading strategy with a lévy-driven ornstein-uhlenbeck process. *Quantitative Finance, 20*(8), 1285–1306.

Yadav, A. S. A., & Jha, C. K. (2020). Optimizing LSTM for time series prediction in Indian stock market. *Procedia Computer Science, 167*(3), 2091–2100.

Yang, X., Li, H., Zhang, Y., & He, J. (2017). Reversion strategy for online portfolio selection with transaction costs. *International Journal of Applied Decision Strategies, 11*(1), 79–99.

Zaheer, S., Anjum, N., Hussain, S., Algarni, A. D., Iqbal, J., Bourouis, S., & Ullah, S. S. (2023). A multi parameter forecasting for stock time series data using lstm and deep learning model. *Mathematics, 64*(3), 590.

Zhang, G. (2022). Pairs trading with general state space models. *Quantitative Finance, 21*(9), 1567–1587.