A hybrid convolutional neural network with long short-term memory for statistical arbitrage

P. Eggebrecht & E. Lütkebohmert

To cite this article: P. Eggebrecht & E. Lütkebohmert (2023) A hybrid convolutional neural network with long short-term memory for statistical arbitrage, Quantitative Finance, 23:4, 595-613, DOI: 10.1080/14697688.2023.2181707

To link to this article: https://doi.org/10.1080/14697688.2023.2181707

Quantitative Finance, 2023
Vol. 23, No. 4, 595–613, https://doi.org/10.1080/14697688.2023.2181707

A hybrid convolutional neural network with long short-term memory for statistical arbitrage

P. EGGEBRECHT and E. LÜTKEBOHMERT*

Department of Quantitative Finance, Institute for Economic Research, University of Freiburg, Rempartstr. 16, 79098 Freiburg, Germany

(*Received 16 May 2022; accepted 10 February 2023; published online 10 March 2023*)

We propose a CNN-LSTM deep learning model, which has been trained to classify profitable from unprofitable spread sequences of cointegrated stocks, for a large scale market backtest ranging from January 1991 to December 2017. We show that the proposed model can achieve high levels of accuracy and successfully derives features from the market data. We formalize and implement a trading strategy based on the model output which generates significant risk-adjusted excess returns that are orthogonal to market risks. The generated out-of-sample Sharpe ratio and alpha coefficient significantly outperform the reference model, which is based on a standard deviation rule, even after accounting for transaction costs.

Keywords: Statistical arbitrage; Pairs trading; Deep learning; Convolutional neural network; Long short-term memory

JEL Codes: C32, C38, C41, C45, G11

1. Introduction

Statistical arbitrage strategies commonly refer to trades which have a neutral exposure to the market (i.e. the trading book has zero beta to the market) and for which the mechanism for generating risk-adjusted excess returns is of purely statistical nature. As such, statistical arbitrage strategies are prone to generating low-volatility return profiles that are uncorrelated with the market. In case of pairs trading, statistical arbitrage reduces to simultaneously buying and selling two cointegrated securities at times when a certain condition is met. In this context, a simple rule of thumb has often been used for timing the opening of a position: A trade is triggered when the price difference (spread) exceeds two standard deviations (2σ) estimated from the empirical distribution of the spread. This rule is applied, for example, in Gatev *et al.* (2006), Dunis *et al.* (2010), Caldeira and Moura (2013), Harlacher (2016). While being very simple, the question arises whether more effective trading rules can be established.

In this paper we develop and analyze a deep learning based framework for pairs trading. Therefore, two tasks have to be addressed: First, during the *formation period* a predefined universe of stocks is screened to identify suitable pairs. To this end, we apply a classic cointegration approach as first suggested in Vidyamurthy (2004). Second, during the *trading period* the spread between the selected pairs is monitored and short and long positions are opened in the respective stocks if a trading signal is triggered. Thus, for this step, we need to determine the correct timing when to open and close the respective pairs positions. For this task, we construct a deep neural network approach which is trained to learn how to correctly classify profitable entry periods from unprofitable periods. Our deep learning model is a CNN-LSTM hybrid model which combines a Convolutional Neural Network (CNN) as the first part in the processing chain to extract hidden features from the input time series with a Long Short-Term Memory (LSTM) model as a subsequent pattern recognition model to classify binary targets across time steps from the input sequences. Further, we propose an extended version of our original CNN-LSTM model, for which we extend the feature space by adding conditional probabilities that the spread will return to its long-term mean within k-steps given past and current spread data. Finally, we compare the performances of the baseline 2σ approach, the CNN-LSTM, and the extended CNN-LSTM model before and after transaction costs on a sample set ranging from January 1991 to December 2017 which includes all constituents of the S&P 500 index at daily frequency. Our research aims at precisely analyzing the benefits of substituting a single component of an established and well-researched financial model (the baseline

*Corresponding author. Email: eva.luetkebohmert@finance.uni-freiburg.de model) by a deep learning extended alternative and studying the effects of this substitution on the economic performance.

Our paper relates to the existing literature on statistical arbitrage and in particular on pairs trading. The concept of pairs trading has been introduced to the academic community with the seminal paper of Gatev *et al.* (2006), in which suitable pairs are identified based on a distance measure and trades are initiated based on the 2σ rule. Gatev *et al.* (2006) document an annualized excess return of up to 11% for their method over the period 1962 to 2002, which cannot be explained by factors such as momentum or mean-reversion. Following the framework of Gatev *et al.* (2006), subsequent research on statistical arbitrage, and pairs trading in particular, has shed light on how these strategies have earned investors excess risk-adjusted returns over recent decades. An early key contribution is Vidyamurthy (2004) who first introduced concepts of cointegration in the context of statistical arbitrage. Their univariate cointegration approach inspired other researchers in their studies, in particular the studies by Wilmott (2006), Dunis *et al.* (2010), Caldeira and Moura (2013), Huck and Afawubo (2015) or Harlacher (2016). Huck and Afawubo (2015) conduct a comparison study in which they analyze the effectiveness of the cointegration approach to pairs formation compared to other methods. They find that the performance of pairs trading strategies based on the cointegration approach is robust to standard risk factors and significantly outperforms other approaches to form pairs after considering risk loadings and transaction costs. For a comprehensive survey of more than 90 papers on statistical arbitrage see Krauss (2017). We modify the approach by Gatev *et al.* (2006) in selecting suitable pairs based on the cointegration method (while trading is still triggered through the 2σ rule) and use this approach as baseline reference model in our study. Pairs trading thresholds that are determined by modelling the spread between two stocks by an Ornstein-Uhlenbeck process are studied e.g. in Elliott *et al.* (2005), Cummins and Bucca (2012), Zeng and Lee (2014), Bai and Wu (2018), Suzuki (2018), and Zhang (2021). In particular, using a state space model approach, De Moura *et al.* (2016) calculate conditional probabilities for the spread to return to its long-term mean within a certain number of time steps. We build on this approach by incorporating these probabilities in our extended CNN-LSTM model.

More recently, machine learning approaches have been utilized for exploiting statistical arbitrage. Krauss *et al.* (2017) train multiple types of machine learning algorithms (multi-layer perceptron models, decision trees, random forest and a weighted average of the three) to estimate the probability of a stock outperforming the cross-sectional median of a security universe on the next trading day. The feature space is composed of simple return series from the last 750 days for each stock. The response variable is binary, indicating whether a return exceeded the next day's cross-sectional median return or not. During the trading phase which is set to 240 days the m stocks with the highest model output are bought and the m stocks with the lowest model output are sold short, thus creating a dollar neutral strategy. Fischer and Krauss (2018) extend the model range and introduce an LSTM model to solve the classification task. Even though model designs differ slightly between these two studies, results show that machine learning model type forecasts are able to generate large risk-adjusted returns. The LSTM and random forest models achieve Sharpe ratios of 5.82 and 5.00 before transaction costs over the sample period of 1992–2015. Results further suggest that models which can learn patterns from sequences, i.e. the LSTM model, are superior to models that do not explicitly model time both in terms of accuracy and generated risk-adjusted returns. Guijarro-Ordonez *et al.* (2021) implement a CNN-transformer model to extract time series signals from a set of unrestricted arbitrage factor portfolios. These time series signals are then used to train another deep neural network that learns to choose optimal long and short weights of the arbitrage portfolios in order to maximize risk-adjusted returns. The backtesting study shows that the CNN-transformer model combined with the weighting model generates time-stable Sharpe ratios above 3 and is robust to variations in the model settings and transaction costs. The authors show that their model successfully filters and extracts the relevant trends of the spread series which are then exploited by the proposed trading strategy. Motivated by these studies, we utilize a hybrid deep learning model which combines CNN and LSTM approaches for the detection of trading signals in pairs trading strategies.

Another stream of literature focuses on the application of machine learning algorithms to solve the task of finding related securities during the formation period. Avellaneda and Lee (2010) apply principal component analysis and develop a trading strategy based on eigenportfolios in a market neutral setting. Sarmento and Horta (2020) make use of an unsupervised learning algorithm called OPTICS (Ankerst *et al.* 1999) to identify suitable pairs from a set of commodity ETFs. Huck (2009) apply neural networks to forecasts spreads between stock returns at the end of a trading period. Pairs are then selected and traded using multi-criteria decision methods. Huck (2010) extends this study by using multi-step-ahead forecasts. Flori and Regoli (2021) apply an LSTM model to detect pairs trading opportunities. As our focus in this study lies on the development of a deep learning alternative to the optimal timing problem of opening a pairs trade during the trading period, we opt for a traditional cointegration-based methodology as our pairs selection mechanism. Leaving the selection methodology in its original form will allow to closely control for the impact of the newly introduced approach on economic performance.

The remainder of the paper is organized as follows. In Section 2 we explain the baseline statistical arbitrage model which uses the 2σ rule for triggering a trade and which serves as a reference for comparison with the deep learning methods in the empirical study. In Section 3 we introduce our hybrid CNN-LSTM model and its extended version. Model performance and backtesting of the trading strategies is analysed in Section 4. Section 5 summarizes and concludes.

2. Baseline statistical arbitrage model

In this section, we describe a simple pairs trading model that applies a rule of thumb based on standard deviations to trigger a trade. This model will serve as reference model for comparison with the deep learning approaches that we will introduce in Section 3. The selection of suitable pairs of stocks is based on the concept of cointegration.

Consider a pair of stocks with prices P_t^i and P_t^j. If their log-prices are cointegrated with cointegration coefficient β and mean level α, their spread

$$S_t^{i,j} = \ln(P_t^i) - \alpha - \beta \ln(P_t^j) \tag{1}$$

is stationary, implying that the spread fluctuates around its equilibrium value, and we obtain the long-run equilibrium relationship from OLS-regression. Therefore, in a first step, we estimate the coefficients $\hat{\alpha}$ and $\hat{\beta}$ and calculate the equilibrium spread value of log prices over time by

$$\hat{S}_t^{i,j} = \ln(P_t^i) - \hat{\alpha} - \hat{\beta} \ln(P_t^j). \tag{2}$$

Since it is not known in advance which price series describes the dependent and which the independent variable, we propose that the regression is run for both possible selections of the dependent variable, i.e. also for $\ln(P_t^j) = \alpha + \beta \ln(P_t^i) + S_t^{j,i}$, and opt for a model based on the obtained t-statistic. In the second step of the testing procedure, a unit root test is applied to the residual series $\hat{S}_t^{i,j}$ to determine whether it is stationary. For this analysis we use the *Augmented Dickey-Fuller (ADF)* test as proposed by Said and Dickey (1984).†

Given the large number of pairwise hypothesis tests which have to be conducted on a large test set, the problem of multiple testing arises. This increases the likelihood of incorrectly rejecting the null hypothesis, i.e. of getting a Type I error. A natural choice to reduce the probability of false positive results on an otherwise unrestricted data set is to define an effective prepartitioning methodology of the set to reduce the amount of possible combinations. In what follows, we opt for the sector classification according to the *MSCI Global Industry Classification Standard (GICS)*. Partition based on industry classifications is easily replicated and has an intuitive economic rationale. We thus allow two stocks to be tested for cointegration only if they are both part of the same industry classification according to GICS. The backtesting studies by Gatev *et al.* (2006) have shown that a larger number of pairs in the portfolio reduces the overall portfolio performance even before taking transaction costs into consideration. In the base case scenario, we therefore place a cap on the number of possible pairs per sector which leaves the portfolio with a maximum of 55 pairs per trading period (11 sectors partitioning the field of securities and $m = 5$ pairs per sector). Additionally, we present economic results for varying numbers m of pairs per sector in Section 4.3.6.

The data set is split into formation periods and trading periods. Each formation period of 252 trading days is followed by a trading period of 130 days. This interval length is also used in Gatev *et al.* (2006) and Harlacher (2016). During the formation period, security pairs are identified within sector groups by applying the ADF test. This is done by selecting the m pairs with the highest p-value for the ADF test. In the subsequent trading period the spread process (2) of each selected pair is monitored and a trading signal is generated when the spread crosses the $+/-$ 2 standard deviation boundaries which are calculated based on the empirical spread distribution throughout the formation period. Whenever a trade signal is generated, i.e. when the spread crosses the $2\hat{\sigma}$ boundaries, the relatively cheap asset is bought and the relative expensive asset is sold in exactly the same monetary amounts. As soon as prices start to change, the value of funds moves dynamically. Trading is executed at the next day's closing price, which technically corresponds to shifting the signal series by one day. In order to limit the risk of the strategy, a stop loss on the cumulative return process of each pair during each investment period is set. As in Gatev *et al.* (2006) and Harlacher (2016), the maximum drawdown is set to -20%, implying that a position gets closed automatically if the cumulative return from a pairs trade falls below this threshold from a prior high. In summary: After a trade has been opened the long and short positions are unwound if (i) the spread crosses the estimated long-term equilibrium value, (ii) the cumulative performance of the position falls below the specified maximum drawdown or (iii) the trading period ends. These trading rules conclude the baseline model. Algorithm 1 summarizes the model in pseudo code.

3. Deep learning statistical arbitrage model

Next, we introduce our deep neural network approach for the detection of trading signals in the statistical arbitrage strategy. We propose a hybrid CNN and LSTM deep learning model, henceforth CNN-LSTM model, which combines the method of extracting spatial features from the CNN model with the LSTM model to support sequence prediction by extracting time series features.‡ This allows us to extract the spatio-temporal correlation between multiple spread-relevant factors and the observed spread series of the chosen pairs. Figure 1 summarizes our hybrid CNN-LSTM model for signal detection. We refer to Appendix 1 for a short review of the two advanced machine learning tools which we combine in our CNN-LSTM approach. A similar model has been applied in Eggebrecht and Lütkebohmert (2023) to construct a deep trend following strategy for equity markets.

3.1. Feature generation and network architecture

In the base case set-up, the 3-feature input matrix that will be passed to the input layer contains the spread series $S_t^{i,j}$ of the log prices of an eligible pair of the respective formation/training period as well as the discrete return series r_t^i, r_t^j of the stocks i and j that form the pair. We split the input

† Harlacher (2016) finds that differences between the ADF test and other unit root tests such as the Phillips–Perron or the Phillips–Ouliaris test are not significant.

‡ CNN-LSTM architectures have been found to achieve state of the art performance in time series forecasting tasks related, e.g. to heart rate signals Swapna *et al.* (2018), rainfall intensity Shi *et al.* (2017), particulate matter Huang and Kuo (2018), waterworks operations Cao *et al.* (2018), or the gold price Livieris *et al.* (2020).

Algorithm 1 Statistical arbitrage: baseline model

```
1:  Formation Period:
2:  for each Formation Period (FP) do
3:      $U_{FP}$ = Set of stocks that are part of the index by FP's year end
4:      $S_{U_{FP}}$ = Set of unique industry sectors of all stocks $P$ in $U_{FP}$
5:      for each sector $I$ in $S_{U_{FP}}$ do
6:          for stocks $P^i, P^j$ in sector $I$ and for all $i \neq j$ do
7:              ▷ regress $S^{i,j}_{FP} = ln(P^i) - \alpha_{i,j} - \beta_{i,j}\, ln(P^j)$
8:              ▷ apply $ADF(\hat{S}^{i,j}_{FP})$
9:              ▷ save results in list $R_{FP,I} = [i, j, \hat{\alpha}_{i,j}, \hat{\beta}_{i,j}, \text{Std}(S^{i,j}_{FP}), \text{p-value}(t^{i,j}_{DF})]$
10:         end for
11:         ▷ Sort $R_{FP,I}$ by p-value in ascending order
12:         ▷ Select the top $m$ pairs
13:     end for
14: end for
15: Trading Period:
16: for each FP's consecutive Trading Period (TP) do
17:     for each sector $I$ in $S_{U_{FP}}$ do
18:         for each top $m$ pair $P^i, P^j$ in $R_{FP,I}$ do
19:             Compute spread $S^{i,j}_{TP} = \ln(P^i_{TP}) - \hat{\alpha}_{i,j} - \hat{\beta}_{i,j}\ln(P^j_{TP})$
20:             if $S^{i,j}_{TP} > +2\text{Std}(S^{i,j}_{FP})$ or $S^{i,j}_{TP} < -2\text{Std}(S^{i,j}_{FP})$ then
21:                 ▷ open pairs trade
22:                 if $S^{i,j}_{TP}$ crossing zero line or return $< -20\%$ from top then
23:                     ▷ close pairs trade
24:                 end if
25:             end if
26:             ▷ Calculate sector performance from pairs within sector $I$
27:         end for
28:         ▷ Calculate portfolio performance from performances of all sectors $I$ in $S_{U_{FP}}$
29:     end for
30:     ▷ Append each TP's portfolio performance
31: end for
```

matrix into sequences of 10, i.e. of approximately two trading weeks. There are few reference models in the academic literature for choosing the sequence length hyperparameter in a CNN-LSTM set-up for financial prediction tasks. However, there are indications that shorter time periods are more suitable, e.g. Livieris *et al.* (2020) and Lu *et al.* (2020) use a sequence length of 10 days, Widiputra *et al.* (2021) take an input sequence length of 4 days. We also found that sequence lengths of 20 and more data points decrease model performance while shorter lengths like 5 or 10 days generate better optimization results. Further, we reshape the data into two sub-sequences of 5 time steps. We found that by passing two stacked 5-day-sequences (i.e. two 3×5 tensors) to the input layer instead of one 10-day-sequence (i.e. one 3×10 tensor), training time was reduced and model out-of-sample performance increased.

The CNN part of the model contains two 1D convolutional layers with 16 filters each, ReLU activation and a kernel size of 2. Before being flattened, the 1D convolution enters a pooling layer.[†] The output of the flattening layer then serves as the input to the LSTM layer which has 10 memory blocks with tanh activation.[‡] The LSTM model was trained in stateless mode as we found equally accurate but more stable results in stateless mode compared to training the network in stateful mode. This observation is in line with results from Yadav *et al.* (2020), Koudjonou and Rout (2020) and Angerbauer *et al.* (2022) for training LSTM models. Finally, the 3 fully-connected layers with 10 and 5 neurons and ReLU activation and the single output neuron with sigmoid activation produce the final estimation value.[§]

Hence, we generate 242 samples of 10-day sequences from the three features of the input matrix per stock and training period. More precisely, we follow the methodology of Fischer and Krauss (2018): We first sort the input vector by feature (return $\mathbf{r}^i$, return $\mathbf{r}^j$, spread $\mathbf{S}^{i,j}$) and date t in ascending order. For each feature $k \in \{1, 2, 3\}$ and each $t \geq 5$ we construct a sequence of the form $\{x^k_{t-4}, x^k_{t-3}, \ldots, x^k_t\}$. We evaluate the out-of-sample performance of each model on a validation set that comprises 20% of the complete data set per training period. This means that we randomly choose $0.2 * 11 * 5$ pairs to

[†] Hyperparameters are inspired by the choices in Livieris *et al.* (2020), except for the number of filters, where we found better optimization results for a lower number of filters compared to 32 and 64 filters used in Livieris *et al.* (2020).

[‡] We tested the following hyperparameters: hidden LSTM layers $\in$ [1,2] and LSTM cells per layer $\in$ [2, 5, 10, 15, 20] on 10 randomly selected pairs. We found that a single layer with 10 cells returned the most accurate results.

[§] The total number of trainable parameters of the model is 1,891.

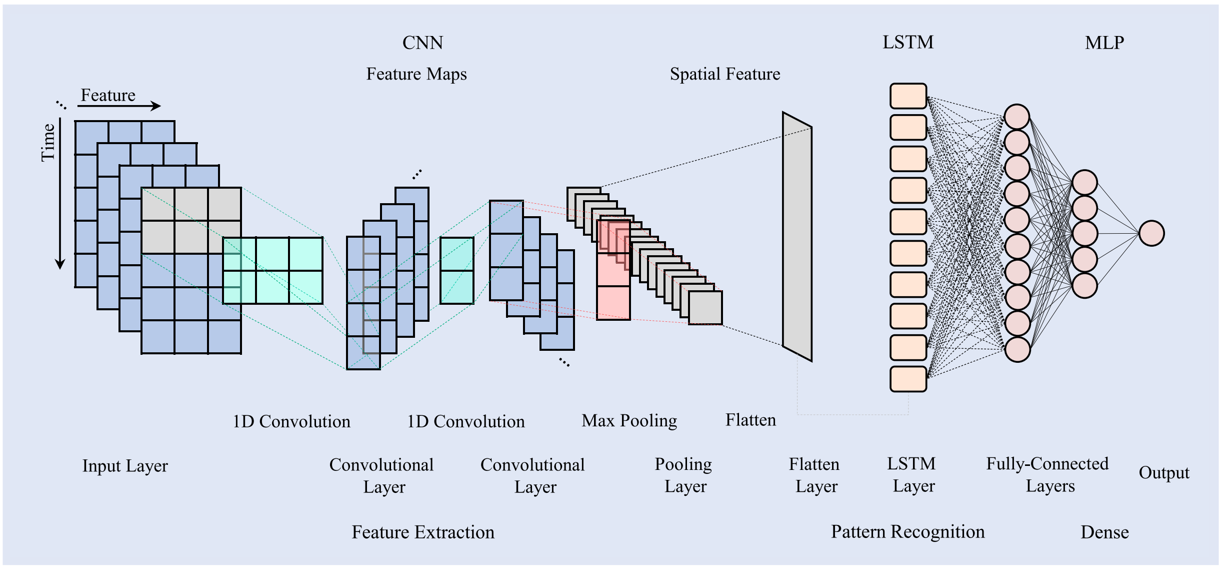


Figure 1. CNN-LSTM model architecture. The CNN-LSTM model contains two 1D convolutional layers, a pooling layer, a flatten layer, an LSTM layer with 10 memory blocks and two fully-connected layers, the first with 10 and the second with 5 neurons, which produces the final estimation.

compute the validation loss. In total, each training period consists of 13,310 individual samples (242 sequences times 11 sectors times 5 pairs per sector) of which 2,662 samples (i.e. 20% of the sequences) are used for out-of-sample predictions. For batch size we use 20 to apply faster convergence with stochastic gradient descent. To further improve generalization we shuffle batches. As a pre-processing variant we standardize the elements of the input vectors by subtracting the mean and dividing by the standard deviation of the respective training feature.† To optimize the model weights, we apply the Adam algorithm as proposed by Kingma and Ba (2014)—an optimized version of stochastic gradient descent. To ensure that the model is not overfitting to the training data we apply multiple regularization methods. First, the number of epochs is determined by the early stopping mechanism: During the optimization, the loss computed on the validation set is monitored. If the validation loss did not decrease for 15 epochs, the model with the lowest validation loss until that point (the model from 15 epochs ago) is saved for later use. Second, we apply a random weight dropout of 20% between the second convolutional layer and the pooling layer, between the pooling and the flattening layer, as well as between the LSTM and the first dense layer.

3.2. *Target generation*

As the studies of Leung *et al.* (2000) and Enke and Thawornwong (2005) suggest, classification models perform better than regression models in predicting financial market data. We therefore adopt the approach of Fischer and Krauss (2018) to formalize the task of classifying profitable from less- or unprofitable positioning on the long and short side and translate it into the context of pairs trading. From the available time series of the formation period, this is conducted as follows: Given the trading rules in Section 2, the best point in time to open the pairs trade is when $S_t^{i,j}$ reaches its largest absolute value after crossing the zero line. We define a possible entry time as any time where $|S_t^{i,j}|$ is different from zero but returns to zero within the formation period. The time at which $|S_t^{i,j}|$ is the largest is classified as an optimal entry point with time index t_{opt}. Further, denote the point in time, where the spread crossed the equilibrium value either from above or below, by t_{out} (with $t_{out} > t_{opt}$). We classify days that are located between t_{opt} and t_{out} as target days. The main motivation behind this target construction methodology is the assumption that one can train a neural network to identify the characteristics of these mean-reverting parts of the sequence which maximize a pairs trader's profit over the training period (given the exit rule at the equilibrium value). If generalization of the task can be sufficiently well done, a systematic trading strategy should be more efficient/profitable compared to a static standard deviation rule. To remove noise from the training data, we remove sequences from the target vector that would not have been profitable after deducting trading costs. Recall that $\boldsymbol{S}^{i,j}$ is the fitted residual series of log prices and by this a non-linear representation of differences in direct price levels. To reduce the noise of the target series, we remove target sequences that are shorter than 4 days. Accounting for these two aspects will leave the target vector only with the most profitable entry signals that are followed by a reversion to the equilibrium value lasting more than 4 days. For the m selected pairs per sector we derive all spread series of length 252. From each of these $S_t^{i,j}$ series and the corresponding return series r_t^i and r_t^j of the pair i, j we construct a corresponding target vector $\boldsymbol{y}$. Designed as a binary classification problem, the elements of target vector $\boldsymbol{y}$ can take either values 0 or 1 which follows the experimental design from Fischer and Krauss (2018). y_t is 1 if $t \in (t_{opt}, t_{out})$ and 0 otherwise. Figure 2 shows the construction of the target vector from the given spread series over the complete training period. Figure 3 provides a more in-depth illustration of the split of

† For example, each element of the input vector $\mathbf{r}^i$ that is passed to the outermost layer is standardized according to $\tilde{r}_t^i = (r_t^i - \text{mean}(\mathbf{r}^i))/\text{stddev}(\mathbf{r}^i)$.

the single sequences for a particular part of the sequence of figure 2 and the corresponding entries on target vector $\boldsymbol{y}$. Note that the day after the crossing of the equilibrium line is also included in the target classification to have the confirmation included that the spread needs to cross the zero-line before the trade is unwound.

3.3. Specification of trading signal

In the trading period, the network receives a 10-day sequence and, based on the classification learned in the training period, returns a value $\hat{y}$ between 0 and 1, which is ensured due to the sigmoid function wrapping the output layer. The closer the value is to 1, the higher is the probability that the last day of the specific sequence can be regarded as an optimal entry point. We use the model output during the out-of-sample trading period directly as an indicator function to generate signals for opening a pairs trade. We can specify an arbitrary threshold value v for scalar model output $\hat{y}$ to reach in order to trigger a pairs trade. As we constructed a binary estimation task, opting for the value $v = 0.5$ is a natural choice. In other words: As soon as the value $\hat{y}$ exceeds 0.5 (which implies that based on the given input sequence, the model estimates a higher probability of the current state of the spread to be in profitable territory) a trade on the respective pair is opened. Given a pairs trade has been opened, the same rules as in the baseline model apply. By this, the only part of Algorithm 1 that is changed according to the new deep learning strategy is line 20 which would be re-written as 'If $\hat{y} > 0.5$ then...'.

3.4. Extensions to the feature space

Next, we extend our CNN-LSTM framework by including additional features in the feature space. As shown in Halevy *et al.* (2009), adding more data to the model helps to improve optimization results on a wide range of different learning algorithms. To this end, we add two additional features (two additional columns) to the input matrix and closely control for the effects on the optimization process and economic performance. More specifically, we build on De Moura *et al.* (2016), who develop a new methodology for calculating conditional probabilities P_{up} and P_{down} that the spread will return to its long-term mean within the next k-steps given past and actual spread data.[†] Appendix 2 shows a more detailed derivation of the state space model and the corresponding Kalman filter equations. We choose this model as the conditional probabilities of the spread to revert to its long-run mean, calculated within a fully mean-reverting model framework, can be regarded as a single input of high explanatory power. We enlarge the feature space of our CNN-LSTM model by including the probabilities P_{up} and P_{down} for each time step t. Here, we choose an AR(1) model and we opt for $k = 5$ steps-ahead predictions. In the original study, the authors propose a value of $k = 25$ but we find that smaller values better suit the dynamics of the stocks in the sample universe used in this study. The equilibrium value is set to 0 by default. The two additional time series are added to the input data set during the training and testing periods. In the sequel, we refer to the model with the feature space extended by the conditional probabilities from the state space model in De Moura *et al.* (2016) as the 'extended CNN-LSTM'. We deliberately limit ourselves to this low-dimensional but sophisticated extension as the subsequent analysis and comparison to the reference model without the additional input will be straightforward.

[†] According to the classification in Krauss (2017), this model represents a stochastic control approach.

4. Performance and backtesting

In this section, we analyse the model performance of the suggested CNN-LSTM deep learning models for pairs trading. We first provide a short description of the data set and then present the empirical results in two stages. First, we analyze the model performance and generalization progress of the optimization over the training and testing set. Second, we backtest the implemented pairs trading strategy and contrast the performance prior to and after transaction costs to the baseline model.

4.1. Data

We obtain daily closing prices of all constituents of the S&P 500 index starting on Jan 1, 1990, and ending on Dec 31, 2017. This corresponds to a total of 7056 daily observations for the S&P 500 index and 53 trading and formation periods. Data is sourced from Bloomberg. To eliminate the survivor bias, we only consider stocks that have been part of the index by the last day of the corresponding year. From this data set we are able to approximately resemble the S&P 500 index at any given point in time between January 1990 and December 2017. If a stock price series is not available for the full formation or trading period it is ignored. Summary statistics are provided in table 1 and are based on simple monthly returns. Following Fischer and Krauss (2018) and Clegg and Krauss (2018), we present statistics grouped by GICS sectors and based on equal-weighted portfolios per sector which are generated monthly, and constrained to index constituency of the S&P 500 index.

We implemented our model in Python 3.6. In addition, we use the packages NumPy (Van Der Walt *et al.* 2011) and Pandas (McKinney 2010) for data management and the Bloomberg Python API for price feed from Bloomberg. The deep learning LSTM network is implemented using the prebuilt environment from Keras[‡]. Moreover, sci-kit learn (Pedregosa *et al.* 2011) is used for preprocessing the data. The neural network is trained on a NVIDIA GeForce GTX 1060 6GB and on an Intel Core i7 CPU 16 GB. Model training on the full sample took approximately 36 hours.

4.2. Model performance

We start by analyzing the learning curves to evaluate the optimization and generalization development throughout the

[‡] See https://github.com/fchollet/keras

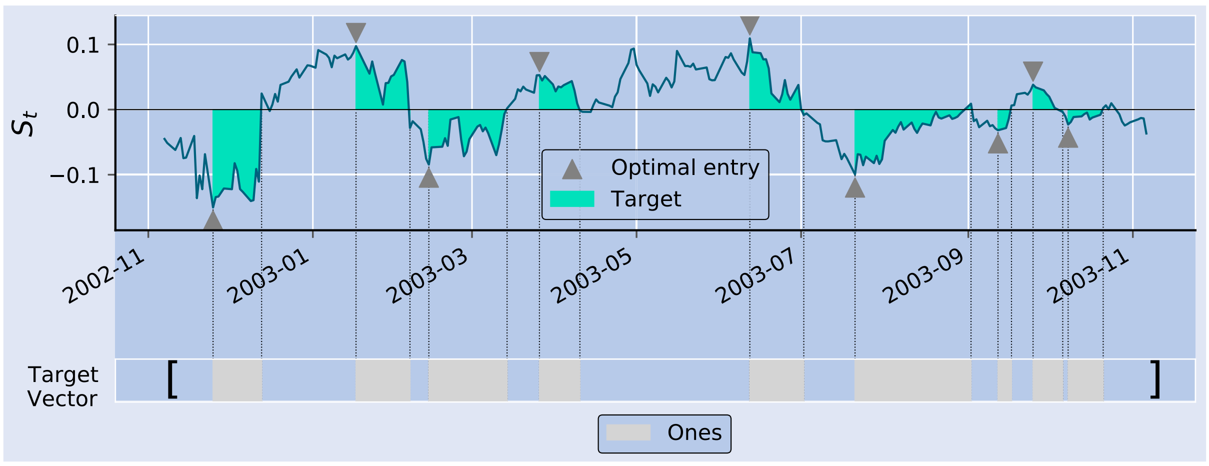
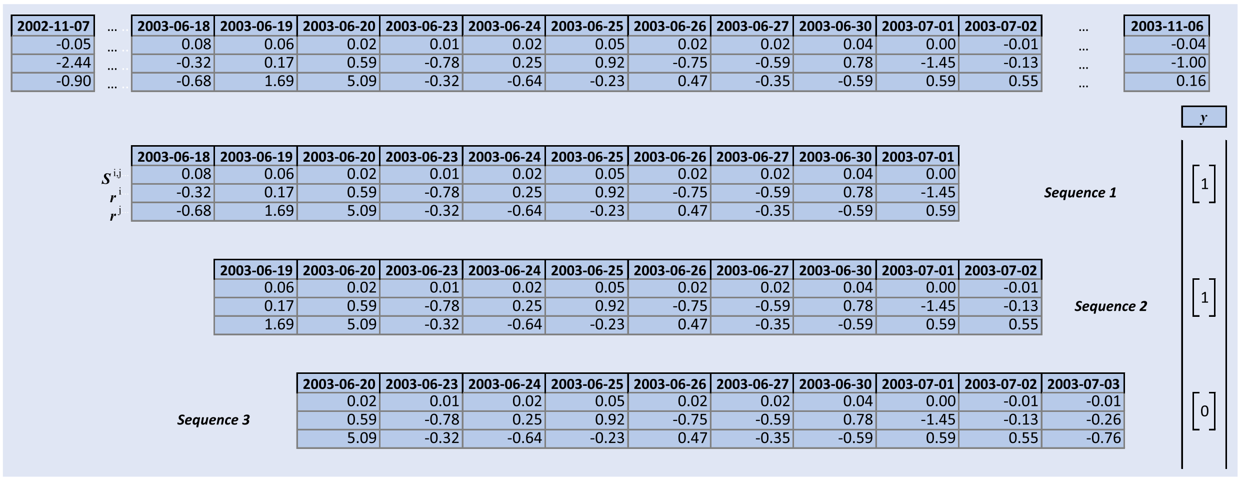


Figure 2. Generating the target vector. The plot shows the spread generated by the pair TXU and Pacific Gas and Electric for the training period 2002-11-07 to 2003-11-06. The shaded areas mark the target periods which are encoded as ones on the target vector.



2002-11-07	…	2003-06-18	2003-06-19	2003-06-20	2003-06-23	2003-06-24	2003-06-25	2003-06-26	2003-06-27	2003-06-30	2003-07-01	2003-07-02	…	2003-11-06
-0.05	…	0.08	0.06	0.02	0.01	0.02	0.05	0.02	0.02	0.04	0.00	-0.01	…	-0.04
-2.44	…	-0.32	0.17	0.59	-0.78	0.25	0.92	-0.75	-0.59	0.78	-1.45	-0.13	…	-1.00
-0.90	…	-0.68	1.69	5.09	-0.32	-0.64	-0.23	0.47	-0.35	-0.59	0.59	0.55	…	0.16

y

Sequence 1 — $y = [1]$

	2003-06-18	2003-06-19	2003-06-20	2003-06-23	2003-06-24	2003-06-25	2003-06-26	2003-06-27	2003-06-30	2003-07-01
$S^{i,j}$	0.08	0.06	0.02	0.01	0.02	0.05	0.02	0.02	0.04	0.00
r^{i}	-0.32	0.17	0.59	-0.78	0.25	0.92	-0.75	-0.59	0.78	-1.45
r^{j}	-0.68	1.69	5.09	-0.32	-0.64	-0.23	0.47	-0.35	-0.59	0.59

Sequence 2 — $y = [1]$

2003-06-19	2003-06-20	2003-06-23	2003-06-24	2003-06-25	2003-06-26	2003-06-27	2003-06-30	2003-07-01	2003-07-02
0.06	0.02	0.01	0.02	0.05	0.02	0.02	0.04	0.00	-0.01
0.17	0.59	-0.78	0.25	0.92	-0.75	-0.59	0.78	-1.45	-0.13
1.69	5.09	-0.32	-0.64	-0.23	0.47	-0.35	-0.59	0.59	0.55

Sequence 3 — $y = [0]$

2003-06-20	2003-06-23	2003-06-24	2003-06-25	2003-06-26	2003-06-27	2003-06-30	2003-07-01	2003-07-02	2003-07-03
0.02	0.01	0.02	0.05	0.02	0.02	0.04	0.00	-0.01	-0.01
0.59	-0.78	0.25	0.92	-0.75	-0.59	0.78	-1.45	-0.13	-0.26
5.09	-0.32	-0.64	-0.23	0.47	-0.35	-0.59	0.59	0.55	-0.76

Figure 3. Construction of input sequences for the artificial neural networks. The first panel shows the input matrix for the pair TXU and Pacific Gas and Electric for the training period 2002-11-07 to 2003-11-06. It is constituted by the spread series $\mathbf{S}^{i,j}$ generated by the pairs and the standardized returns $\mathbf{r}^{i}$ and $\mathbf{r}^{j}$ of securities i, j. The following panels show the sequences as provided to the input layer. The right column vector shows the corresponding target vector for the illustrated sequences.

iterative gradient descent process. Figure 4 plots the in-sample accuracies (a) and validation accuracies (b) over the respective epochs. To get a better impression of the average model progress, the averages of each diagnostic learning curve of all 53 optimizations are shown.† The CNN-LSTM model with extended input from the state space model of De Moura *et al.* (2016), labeled ‘CNN-LSTM extended’, optimizes the model parameters on average for 43 epochs before getting stopped by the early stopping criterion and the CNN-LSTM model optimizes on average for 59 epochs. The figures therefore show the corresponding curve values over the first 50 epochs of training as a suitable average optimization lengths. We observe a smooth improvement of the learning rate (increase of model accuracy) over time. The learning curves further reveal that the extended CNN-LSTM model generates higher accuracy levels on the in-sample training set as well as on the held-out validation set than the CNN-LSTM model.

The average in-sample accuracy levels start from an initial value of 0.5 and reach levels of around 0.7 for the extended CNN-LSTM model compared to a slightly lower value of the CNN-LSTM reference model. At epoch 50, the CNN-LSTM model returns an average accuracy level of 0.68 compared to 0.71 of the extended model for the in-sample training set. This result does not come as a surprise as the additional feature in the extended model can be regarded as being of high explanatory power for the learning task and therefore helps to achieve better optimization results than the reference CNN-LSTM model. For the test set, this difference shrinks and the CNN-LSTM model achieves an out-of-sample accuracy of 0.67 compared to a marginally more accurate extended CNN-LSTM model with a testing accuracy of 0.68. For a classification task in a financial market context, these levels of accuracy are very promising. As shown in Fischer and Krauss (2018), out-of-sample estimation accuracies of around 0.55 for a binary classification in a statistical arbitrage context can result in significant financial excess returns.

To verify that the accuracy results are not biased by a possibly imbalanced data set, we analyse Precision and F1 scores

† We note that we did not notice any problems related to vanishing or exploding gradients during the training of the models.

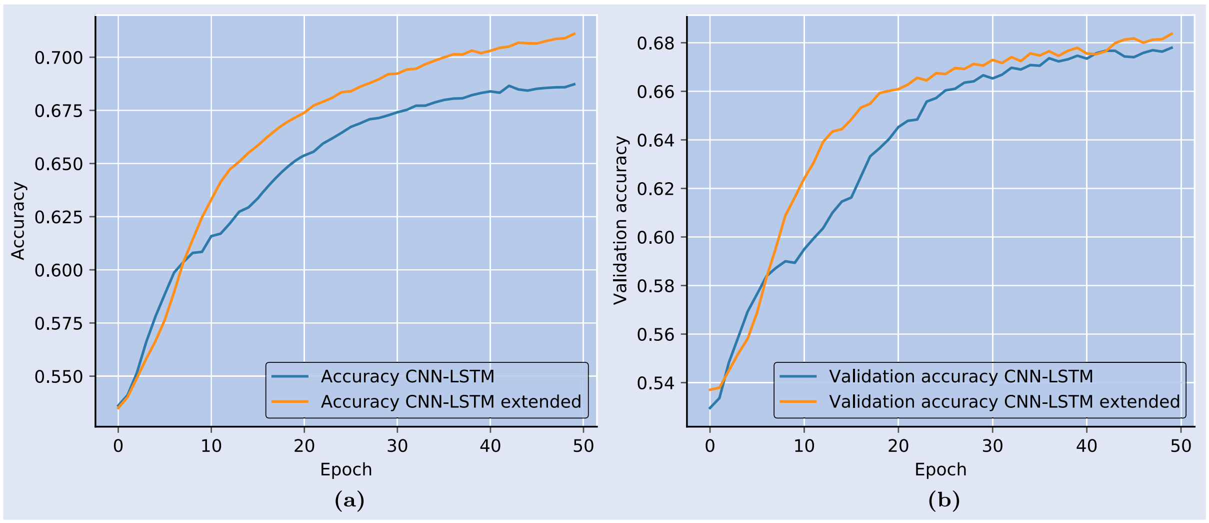


Figure 4. Learning curves from statistical arbitrage experiment. The plot shows the diagnostic learning curves over the first 50 epochs. The values represent the average over all 53 optimizations at the respective epoch. (a) Accuracy (b) Validation accuracy.

Table 1. Summary statistics of S&P 500 index by GICS sectors, 01/1991–12/2017.

	Mean return	Standard deviation	Skewness	Kurtosis
Consumer Staples	1.1415	3.8849	− 0.3950	2.0660
Energy	0.8938	7.5798	− 0.0244	0.9069
Materials	1.0084	6.0420	− 0.0298	2.1612
Consumer Discretionary	1.2275	6.3639	0.0622	3.7611
Financials	1.1272	6.3150	− 0.0646	2.7398
Industrials	1.3729	5.9081	− 0.1456	1.5991
Utilities	0.6122	4.2883	− 0.4008	1.2169
Communication Services	1.1646	6.3232	0.0790	2.3154
Information Technology	2.0282	8.2721	0.1289	1.2537
Health Care	1.6614	4.8736	− 0.4540	1.3748
Real Estate	1.0677	5.4607	− 0.1867	8.6526
All	1.2096	5.9374	− 0.1301	2.5498

Note: Summary statistics for S&P 500 index constituents from 01/1991–12/2017 grouped by industry. Values are calculated from equal-weighted portfolios for each industry as defined by the GICS methodology, formed on a monthly basis. Returns and standard deviation are denoted in percent.

as standard metrics used for classification tasks.† Figure 5 visualizes the out-of-sample validation values for Precision and F1 for both models during the optimization process over the epochs. As can be seen, on absolute levels, validation F1 scores of both models start from low levels below 0.1 and converge to a value of 0.64 at epoch 50. For the majority of the epochs, the extended CNN-LSTM model returns higher values for the plotted Precision and F1 scores. Towards the end of the optimization periods, both models are very close in value which is in line with what we observed for the validation accuracy values.

From these results we can conclude that the model parameters are well-optimized to solve the task with a high level of accuracy on the in-sample training set and that the model generalizes sufficiently well to new unseen data. Results are not skewed by the distribution of the training data as shown by the F1 scores. Finally, to rule out the possibility that the results are dominated by outliers, we analyze the distribution of the learning curves per epoch. Therefore, figure 6 shows the box plots obtained from the distribution of the 53 data points per epoch over the first 50 optimization epochs for the extended CNN-LSTM model. The plots confirm that the variation in the different paths is not extreme. Upper and lower quartiles being close in value along the epochs and narrowing to the end indicate a similar optimization success across the different training periods. We note one path on which optimization results are outside of the box plot. This single relatively bad optimization result out of the 53 samples is an acceptable outlier for the extended model.

Until now, we discussed the results based on how the average model has performed over the respective epochs. To further examine how the optimization success, as measured by the average model's accuracy on the training set, develops over time, we plot the final accuracy values along their corresponding training period on the *x*-axis. Figure 7 shows the corresponding model results for the two models under consideration over the respective formation period. We further perform a linear regression to capture the trend of model performance over time. We note that the CNN-LSTM model's regression line on the in-sample model results decreases at − 0.0412% per formation period (distinguishable from zero at the 97.5% confidence interval with a p-value of 0.022). The extended CNN-LSTM model on the other hand shows a slightly positive slope parameter of 0.002% per formation period (not distinguishable from zero with a p-value of 0.914.) which reveals that the additional input from the state space model of De Moura *et al.* (2016) counteracts a decrease of accuracy over time. A possible explanation for the decreasing trend in the CNN-LSTM model's estimation accuracy is the increasing awareness of market participants of those arbitrage strategies. Given only market data (returns and spread), the CNN-LSTM model looses the edge of successfully filtering

† Precision is defined as $\frac{TP}{TP+FP}$ and F1 score is defined as $\frac{TP}{TP+\frac{1}{2}(FP+FN)}$ where TP, FP, and FN refer to true positives, false positives and false negatives, respectively.

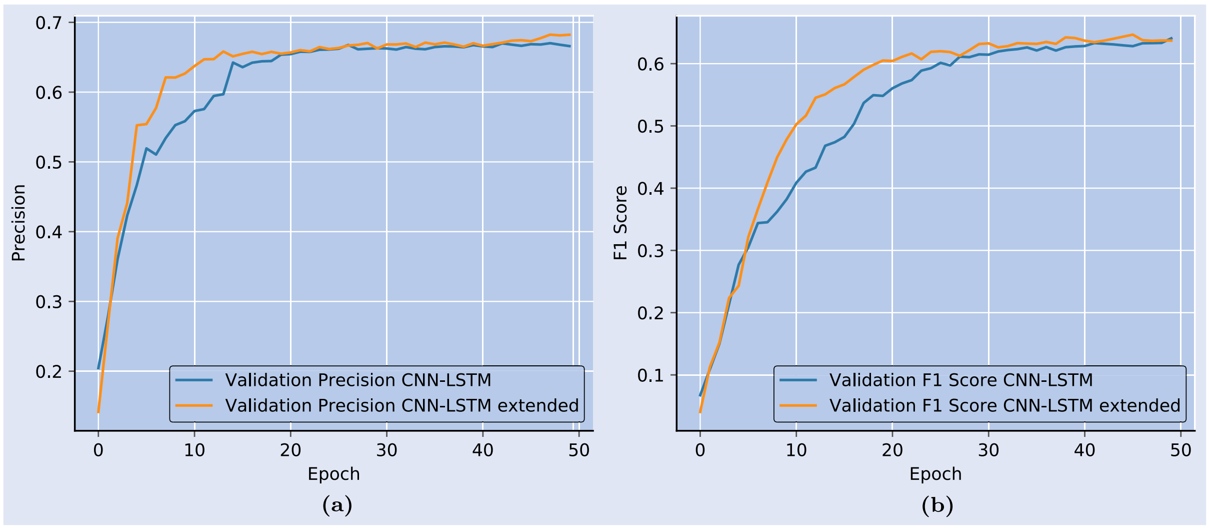


Figure 5. Precision and F1 scores from statistical arbitrage experiment. The plot shows the Precision and F1 score curves over the first 50 epochs. The values represent the average over all 53 optimizations at the respective iteration. (a) Validation Precision (b) Validation F1.

out enough relevant information from the data to keep accuracy levels seen during the 1990s (formation period 19 marks the start of the year 2000). Spread divergences are quickly arbitraged away by increasingly informed market participants and only additional information that is not easily accessible by the average arbitrageur can reverse this trend as seen from the results obtained from the extended CNN-LSTM model.

4.3. Economic performance

Next, we compare the economic performance of the baseline model (with 2σ boundary rule), the CNN-LSTM model and the extended CNN-LSTM model. As a further reference model, we include the backtest results from a less advanced model, i.e. a plain LSTM model with two hidden layers and 10 LSTM cells each.

As a first indication of how the different strategies performed economically over the sample period of January 1991 to December 2017, figure 8 plots the cumulative returns of each strategy over time. As can be seen, the baseline model and the LSTM model yield the least profitable strategies, followed by the CNN-LSTM model. The extended CNN-LSTM model marks the highest absolute return over the time frame outperforming the other models by a large margin. On an absolute basis, periods of high profitability and periods of low or negative profitability seem similar across the three strategies. This is as expected as all strategies derive their opening signal from the same spread series. From a first inspection of the cumulative return series, however, we can already tell that the extended CNN-LSTM model yields the most profitable strategy.†

Table 2 provides a comparison between the four model alternatives prior to and after deducting transaction costs. Following Avellaneda and Lee (2010) and Fischer and Krauss (2018), we assume a slippage/transaction cost of 0.05% or 5 basis points (bps) per trade (a round trip transaction cost of 10 bps) which we consider a conservative estimate given the highly liquid stock universe. We separate the table into four horizontal panels. Panel A shows statistical measures derived from daily return series generated by the respective trading systems. Panel B provides insights about the perceived risk of the strategies. Panel C depicts annualized return metrics and panel D shows statistics related to the CAPM model.‡ In the following, we will first discuss the results before transaction costs (compare column 2) and then proceed by analysing the effects of transaction costs (compare column 3).

4.3.1. Economic performance before transaction costs.

Starting with panel A we note that the baseline model and the deep learning models exhibit similar statistical characteristics for the daily return series. All strategies generate statistically significant returns. Daily average returns of the suggested models are well below the one of the general market index. Quartiles as well as minimum and maximum values suggest that results are not distorted by outliers. The extended CNN-LSTM model has a higher standard deviation of the daily return distribution compared to the baseline model and the reference CNN-LSTM model which implies that the returns of the extended CNN-LSTM model come at the price of higher volatility. As expected, all models generate much less volatile return series compared to the reference index.

Panel B, which shows various risk measures, confirms the picture from the statistical summary. The conditional Value at Risk (cVaR) model at the 99% confidence level, the most conservative metric shown in the panel, returns a value of -1.023% for the extended CNN-LSTM model which, compared to the market that returns a value of -4.3882% for the same time span, confirms a strong reduction in absolute risk

† We did a comparison for different values of extrapolation parameter k and obtained the most promising results for $k = 5$. We found that the superior performance of the $k = 5$-variant can be attributed to the better out-of-sample classification accuracy compared to the alternatives for $k = 10$ or 20 days. Final average out-of-sample accuracies are 68.5% for $k = 5$, 66.5% for $k = 10$, 67.1% for $k = 20$.

‡ Alpha and beta coefficients relate to the one-factor model regression. We discuss further dependencies on risk factors in Section 4.3.3.

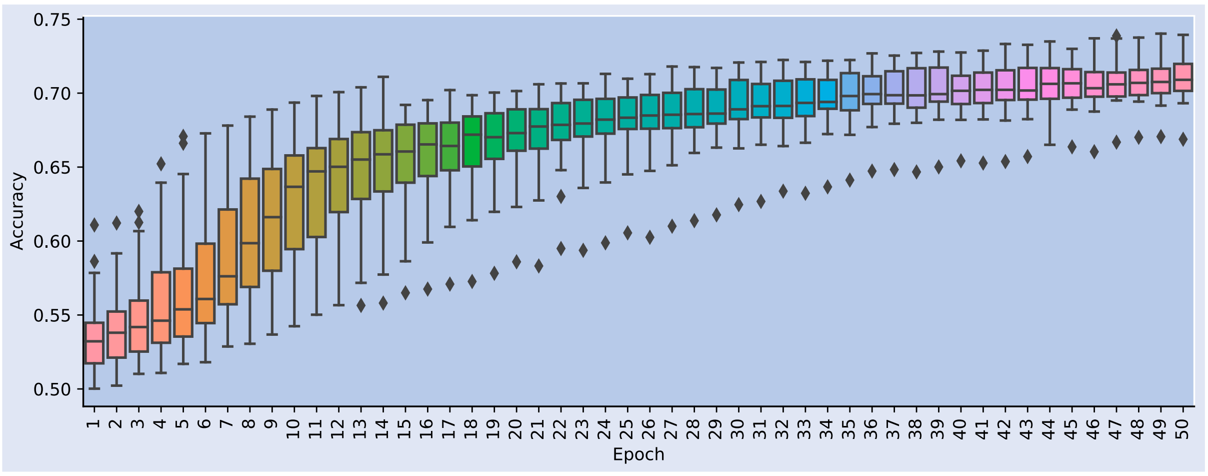


Figure 6. Box plots of accuracy learning curves from statistical arbitrage experiment. Box plots of the distribution of observations per epoch.

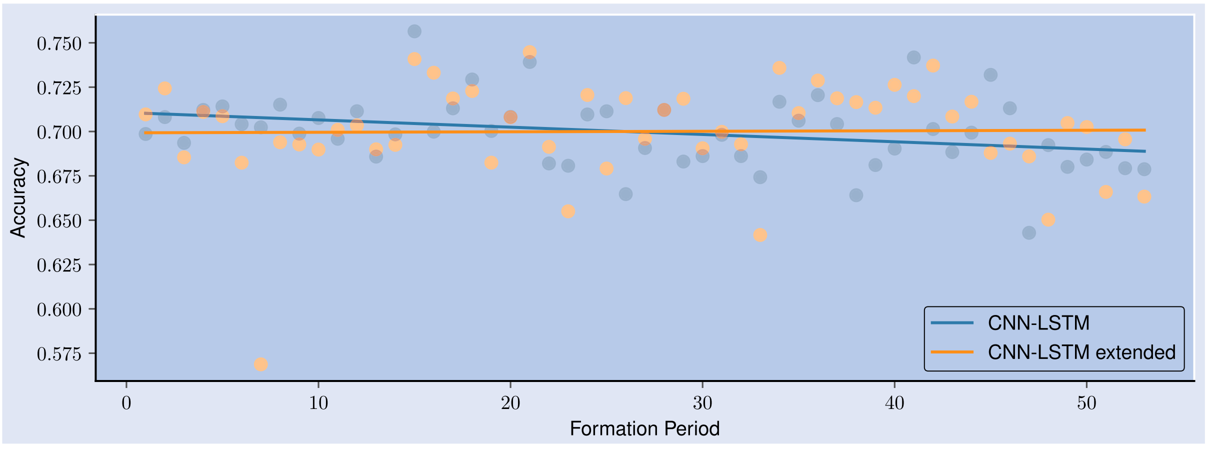


Figure 7. Model performance over time.

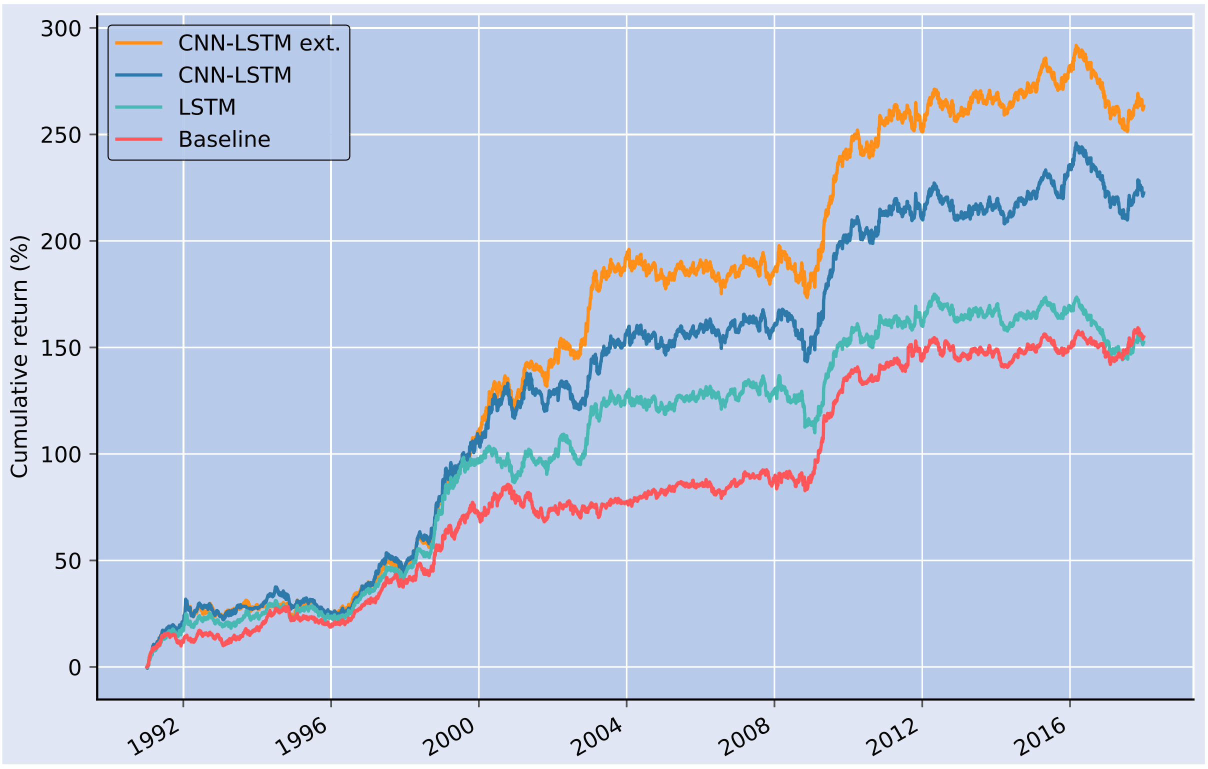


Figure 8. Cumulative returns of statistical arbitrage strategies.

Table 2. Results from statistical arbitrage experiment.

		Before transaction costs				After transaction costs				
		Baseline	LSTM	CNN-LSTM	CNN-LSTM extended	Baseline	LSTM	CNN-LSTM	CNN-LSTM extended	Market
A	No. of observations	6811	6811	6811	6811	6811	6811	6811	6811	6811
	Mean return daily (%)	0.0142***	0.0141***	0.0177***	0.0195***	0.0129***	0.0125**	0.01440**	0.0166***	0.0456
	Mean return monthly (%)	0.2992***	0.2982***	0.3751***	0.4120***	0.2728***	0.2654**	0.3048***	0.3521***	0.9132***
	Months positive (%)	59.8765	58.0247	59.2593	59.2593	59.2593	58.0247	57.0988	57.0988	66.6667
	Minimum daily (%)	– 2.1536	– 2.8101	– 2.2392	– 2.3183	– 2.1554	– 2.8759	– 2.2443	– 2.3229	– 9.0257
	Minimum monthly (%)	– 4.9353	– 6.2678	– 4.8865	– 4.5673	– 4.9571	– 6.7264	– 4.9565	– 4.6302	– 16.7951
	Quartile 1 (%)	– 0.1401	– 0.1719	– 0.1710	– 0.1710	– 0.1410	– 0.1732	– 0.1742	– 0.1746	– 0.4198
	Median (%)	0.0041	0.0088	0.0091	0.0117	0.0033	0.0078	0.0060	0.0080	0.0565
	Quartile 3 (%)	0.1691	0.19050	0.1959	0.2060	0.1672	0.1906	0.1918	0.2026	0.5621
	Maximum daily (%)	3.3024	2.3263	2.3202	2.1805	3.2970	2.3019	2.3245	2.1837	11.5800
	Maximum monthly (%)	6.8788	7.6417	8.5627	8.4293	6.8168	7.6911	8.3663	8.2199	11.4314
	Standard deviation (%)	0.2902	0.31970	0.3294	0.3354	0.2898	0.3246	0.3289	0.3346	1.1108
	Skewness	0.2830	0.2416	0.1806	0.2149	0.2845	0.1712	0.1784	0.2095	– 0.0685
	Kurtosis	5.5268	3.3790	3.0416	2.6409	5.5365	3.2990	3.0393	2.6490	9.3123
B	VaR 99 (%)	– 0.7234	– 0.7470	– 0.7761	– 0.8077	– 0.7234	– 0.7697	– 0.7778	– 0.8123	– 3.0452
	cVaR 99 (%)	– 0.9522	– 0.9629	– 1.0518	– 1.0230	– 0.9528	– 1.0099	– 1.0551	– 1.0254	– 4.3882
	VaR 95 (%)	– 0.4403	– 0.4869	– 0.4962	– 0.5049	– 0.4408	– 0.4967	– 0.4984	– 0.5087	– 1.6803
	cVaR 95 (%)	– 0.6239	– 0.6673	– 0.7021	– 0.7044	– 0.6228	– 0.6857	– 0.7043	– 0.7077	– 2.6237
	Max. drawdown (%)	– 9.4840	– 11.3424	– 10.4688	– 10.1187	– 9.9260	– 14.6949	– 11.4809	– 11.1206	– 55.2171
C	Return p.a. (%)	3.5276	3.4850	4.4320	5.0741	3.2010	3.0703	3.5593	4.1408	10.4253
	Standard deviation p.a. (%)	4.6094	5.0756	5.2314	5.3264	4.6029	5.1530	5.2245	5.3149	17.6424
	Downside deviation p.a. (%)	2.1261	3.2135	2.3352	2.3319	2.1315	3.3098	2.3499	2.3483	9.2157
	Sharpe ratio p.a.	0.7653	0.6866	0.8472	0.9526	0.6954	0.5958	0.6813	0.7791	0.5909
	Sortino ratio p.a.	1.6592	1.0845	1.8979	2.1759	1.5018	0.9276	1.5146	1.7633	1.1313
D	Alpha (%)	0.0131***	0.0140***	0.0173***	0.0192***	0.0119***	0.0123**	0.0140***	0.0164***	0.0
	Beta (%)	2.2686***	0.1944	0.8479*	0.5805	2.2490***	0.3965	0.8658*	0.6161	100.0
	Information ratio	– 38.6572	– 37.8300	– 32.8263	– 29.1964	– 40.4839	– 40.1220	– 37.6165	– 34.3019	NaN

Notes: The four panels illustrate performance characteristics of the $m = 5$ portfolios, before and after transaction costs of 5bps per half-trip for the baseline model, the CNN-LSTM model, the extended CNN-LSTM model and the S&P 500 index (market) from 01/1991–12/2017. The table shows daily return characteristics (Panel A), daily risk characteristics (Panel B), annualized risk-return metrics (Panel C), and metrics related to the CAPM model (Panel D). ***$p < 0.001$, **$p < 0.01$, *$p < 0.05$.

for the returns generated by the models. Both, cVaR and VaR are showing the 1-day-ahead estimates. The maximum drawdowns for all three models, are at low levels. This is a direct consequence of the hard-coded maximum loss of 20% on each pair that is trading in the market and stands in stark contrast to the value of -55.2171% as measured in the general market.

Panel C shows annualized risk/return metrics. The CNN-LSTM model and the extended CNN-LSTM model return 4.4320% and 5.0741% per annum before transaction costs. This is to be compared to an annualized return of 3.5276% from the baseline model. The LSTM model fails to outperform the simple standard deviation rule over the complete sample time span. An inspection of figure 8 shows, that indeed the LSTM generated higher returns compared to the baseline model until the mid 2000s. From then on it generated weaker returns. A detailed discussion on profitability over time can be found in Section 4.3.4. As seen from panel A, the deep learning models' returns are achieved at slightly higher levels of volatility compared to the baseline model. However, the annualized Sharpe ratio of 0.9526 for the extended CNN-LSTM model and 0.8472 for the CNN-LSTM model compared to a corresponding value of 0.7653 for the baseline model tells that the investor gets compensated with a higher risk-adjusted return from the machine learning models compared to applying the simple two standard deviation rule. Also the annualized Sortino ratio (returns adjusted by downside deviation) shows an impressive value of 2.1759 for the extended CNN-LSTM model compared to 1.6592 for the baseline model, indicating a more profitable and at the same time conservative risk profile of the deep learning model. All models show a superior risk-adjusted return profile compared to the market. Panel D shows that these risk-adjusted excess returns are achieved with very low exposure to market risk as beta parameter values from the single-factor model regression are non-significantly different from zero for all models except for the baseline model which shows a relatively high beta coefficient. All strategies generate significantly positive alpha. The extended CNN-LSTM model yields the highest alpha coefficient of all strategies.

4.3.2. Economic performance after transaction costs.

The third column of table 2 depicts the same metrics after deducting 5 bps per trade and security. As can be seen, daily statistics and risk measures are not much affected by the introduction of transaction costs. Mean returns stay significantly positive for all strategies. However, on the CNN-LSTM strategies, we note a reduction of positive return months from 59% to 57%. This is not an extreme change, however, the impact is less pronounced for the baseline model. Annualized return measures confirm that transaction costs indeed have a notable effect on profitability for all strategies. Annualized returns are reduced by an absolute of 0.3266% for the baseline and 0.9333% for the extended CNN-LSTM model. This results in a decrease in Sharpe ratio from 0.9526 to 0.7791 for the most profitable machine learning model which still is above the corresponding value of 0.5909 for the long-only market investment. Alpha coefficients from the single market risk factor model stay significantly positive from zero. Only the LSTM model is not significant at the 0.001 confidence level anymore. Figure 9 further visualizes the annualized returns and Sharpe ratios of the three trading strategies before and after transaction costs. The results of the baseline model are in line with the findings of Harlacher (2016). From these results, we note that the CNN-LSTM model without additional input after deducting transaction costs, does not exceed the baseline model in terms of Sharpe ratio. The reference LSTM model fails to beat the baseline model before and after transaction costs. As returns are linearly decreasing in transaction costs, we note that transaction costs of more than 0.12% per round trip would make the baseline model more profitable than the extended CNN-LSTM model and transaction costs of more than 0.3% would lead to negative total returns for the extended CNN-LSTM model. The numbers, however, confirm that the extended CNN-LSTM model states the most profitable strategy even after accounting for trading costs and delivers a superior risk-adjusted return profile compared to a long-only market investment as well as to the here presented alternative strategies. This indicates that the CNN-LSTM deep neural network is able to capture significant information from the time series of the spread values and the corresponding return series, which exceeds the information contained in the 2σ threshold rule from the baseline model.

4.3.3. Exposure to risk factors.

To evaluate the strategies' exposure to systematic sources of risk, we use the three factor model as introduced in Fama and French (1996).† In addition to the three factors which include a market factor ('Mkt'), a market capitalization factor ('SMB' for Small Minus Big) and a book-to-market factor ('HML' for High Minus Low), we follow Fischer and Krauss (2018), and add a momentum ('Mom') as well as a long-term and a short-term reversal factor to the multivariate regression.‡ This choice is also motivated by findings in Chen *et al.* (2017) who show that both the short-term reversal and the pairs momentum can explain profits from pairs trading strategies to a large degree. All results are summarized in table 3. We first note that the overall model fit is not very high but in relative terms, the baseline model yields the highest R^2. For the most profitable model, the extended CNN-LSTM model, only 2.1% of the variance in the data can be explained by the regression model. Looking at the single factor loadings, the market-neutrality of all considered strategies is reflected in the non-significant market factor of the two CNN models. The LSTM and the baseline model however, show a significant linear dependence to the market. In contrast to the findings of Fischer and Krauss (2018), the SMB factor is significantly positive which indicates that the models have a preference toward small-cap stocks.§ On the other hand the HML factor is non-significant and close to zero across strategies so there is no indication that the model portfolio is tilted towards value stocks. For the additional factors,

† We use the statsmodels library (Seabold and Perktold 2010) in Python with default parameters for the linear regression.
‡ The authors thank Kenneth French for allowing to source all data from his website: https://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html
§ Note that Chen and Bassett (2014) show that due to the self-financing nature of these factor portfolios and the market capitalization structure, this interpretation is not necessarily true.

Table 3. OLS regression results.

	CNN-LSTM ext.	CNN-LSTM	LSTM	Baseline
Intercept	0.00019***	0.00017***	0.00013***	0.00012***
	(0.00004)	(0.00004)	(0.00004)	(0.00003)
Mkt	− 0.00611	− 0.00418	− 0.00976**	0.01089**
	(0.00389)	(0.00387)	(0.00376)	(0.00339)
SMB	0.02858***	0.02561***	0.02984***	0.02701***
	(0.00727)	(0.00723)	(0.00703)	(0.00634)
HML	− 0.00238	− 0.00861	0.00354	− 0.00687
	(0.00817)	(0.00813)	(0.00789)	(0.00712)
Mom	− 0.04763***	− 0.05152***	− 0.04341***	− 0.04283***
	(0.00516)	(0.00514)	(0.00499)	(0.00450)
LT_Rev	0.01918	0.01410	0.01477	0.01048
	(0.00985)	(0.00980)	(0.00952)	(0.00859)
ST_Rev	0.02107***	0.02104***	0.02346***	0.02625***
	(0.00517)	(0.00515)	(0.00500)	(0.00452)
R^2	0.021233	0.022501	0.021232	0.032106
Adj. R^2	0.020368	0.021638	0.020368	0.031251

Note: ***p $<$ 0.001, **p $<$ 0.01, *p $<$ 0.05. Standard errors in parentheses.

we see that the momentum factor is negative and significant and the short-term reversal factor is positive and significant which confirms that the strategies tend to sell recent outperformers and buy recent underperformers. Most importantly, daily returns cannot be sufficiently explained by the applied model and the generated alpha coefficients (Intercept) of the two CNN models stay unchanged compared to the daily mean returns (see table 2) and are still statistically significant.

In addition to the discussed factors, there is no statistical evidence that the day of the week has a significant impact on returns. Running a linear regression with weekdays as exogenous variables on the daily returns of the discussed models shows that the day of the week does not have a significant impact on the returns generated by the respective strategies. R^2 values are very low across the 4 models and the alpha (intercept) remains significantly different from zero. We omit the results here to keep the paper compact.

4.3.4. Performance over time. Looking separately at the three decades of the sample period allows to derive conclusions about the development of the profitability of the strategies over time. Figure 10 illustrates cumulative returns over time, i.e. from January 1991 to December 2017. The early period of Jan 1991 to Dec 1999 is characterized by a highly profitable economic performance for all models with the deep learning models being superior to the baseline model with respect to total return in almost all years. Average annualized returns over this first decade for the extended CNN-LSTM model are as high as 9.1% compared to 6.2% for the baseline model.†

The second decade from Jan 2000 to Dec 2009 (b) is characterized by diminishing profitability compared to the 1990s. Over the whole period, all models are able to generate positive returns but the absolute magnitude decreased compared to the 1990s. The LSTM model underperforms the baseline model. The period of the burst of the tech bubble from 2000 to 2001 and the subsequent market recovery in 2003 marks a period of very high profitability for the CNN-LSTM models. Especially, the extended CNN-LSTM model achieves very high returns, which stands in stark contrast to the performance of the returns generated by the baseline model as well as of the market over the same period. During the great financial crisis 2008/2009, however, the baseline model performs better compared to the deep learning models. It is possible that the diminishing profitability during this period is caused by the increasing popularity of these strategies among market participants which causes these arbitrage opportunities to persist shorter. The reason for the CNN-LSTM strategies to perform worse than the fixed 2σ threshold rule during times of extreme market turmoil might be that the machine learning model gets detracted by noise and outliers causing false/unprofitable signals for the deep model.

The final period from Jan 2010 to Dec 2017 (c) marks a phase of slightly positive to zero profitability. The edge of the presented machine learning methods as well as of the baseline model seems to be arbitraged away in the market. This development is also observed by other research studies that focus on systematic trading rules; see, for instance, Bogomolov (2013), Hsu *et al.* (2016), Harlacher (2016), and particularly Fischer and Krauss (2018) who report similar results in their statistical arbitrage experiment.

We tested alternative formation and trading period length of 3 and 1 year, resp., and found that compared to our results in figure 8 of the paper, which are based on the one year formation and six months trading period, the cumulative return for the alternative choice of 3 and 1 year periods is much lower. Moreover, the chosen deep learning methods also outperform the baseline model on the alternative period set-up and the order of economic performance along models is consistent with the results for our baseline choice of one year formation period and 6 months trading period. With regard to performance over time, results are very similar to the ones presented above.

4.3.5. Transaction analysis. Next, we analyse the individual transaction histories of the different trading models to derive insights in the economic rationals for the dominance of certain model specifications.

Selected trade statistics per trading period: Figure 11 provides an overview of various relevant trade statistics over each of the 53 trading periods. Subfigure (a) reveals that, with the exception of a few periods, the three machine learning based strategies trade more frequently. Subfigure (b) entails the most important information. The plot shows that the relative share of trades that generated a positive return after a trade has been closed (sometimes referred to as 'Hit rate') is $>$ 70% for the machine learning models on average (with the extended CNN-LSTM model marking the highest Hit rate and the LSTM the lowest) compared to an average of slightly below 60%

† It is important to note that deep learning techniques such as LSTM or CNN models have been introduced in the late 1990s. As such, the high risk-adjusted returns in the 1990s need to be seen against the backdrop that neither the theory nor the necessary technology for this strategy has been available for the majority of market participants.

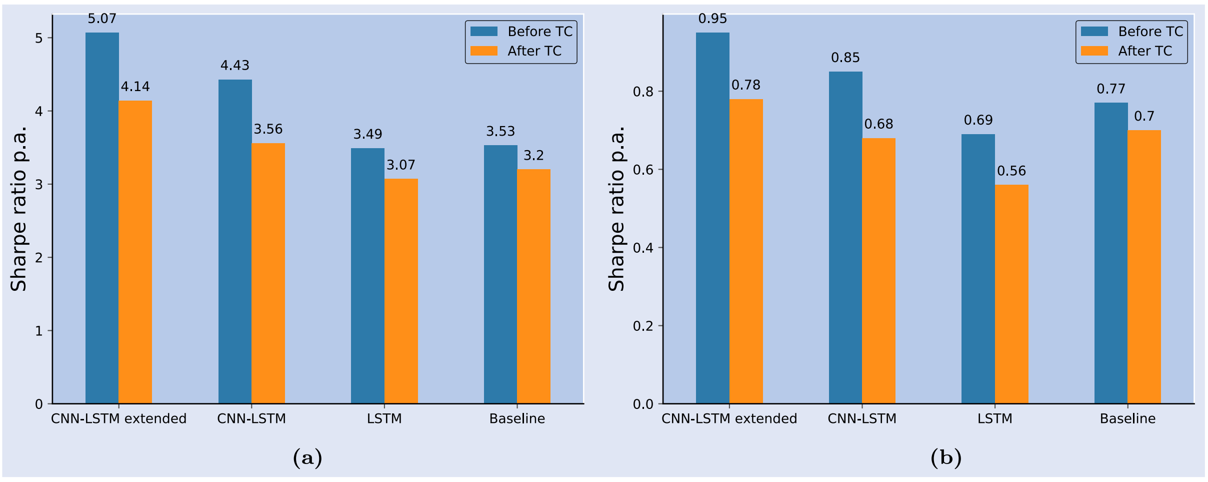


Figure 9. Impact of transaction costs. (a) Annualized return (b) Sharpe ratio.

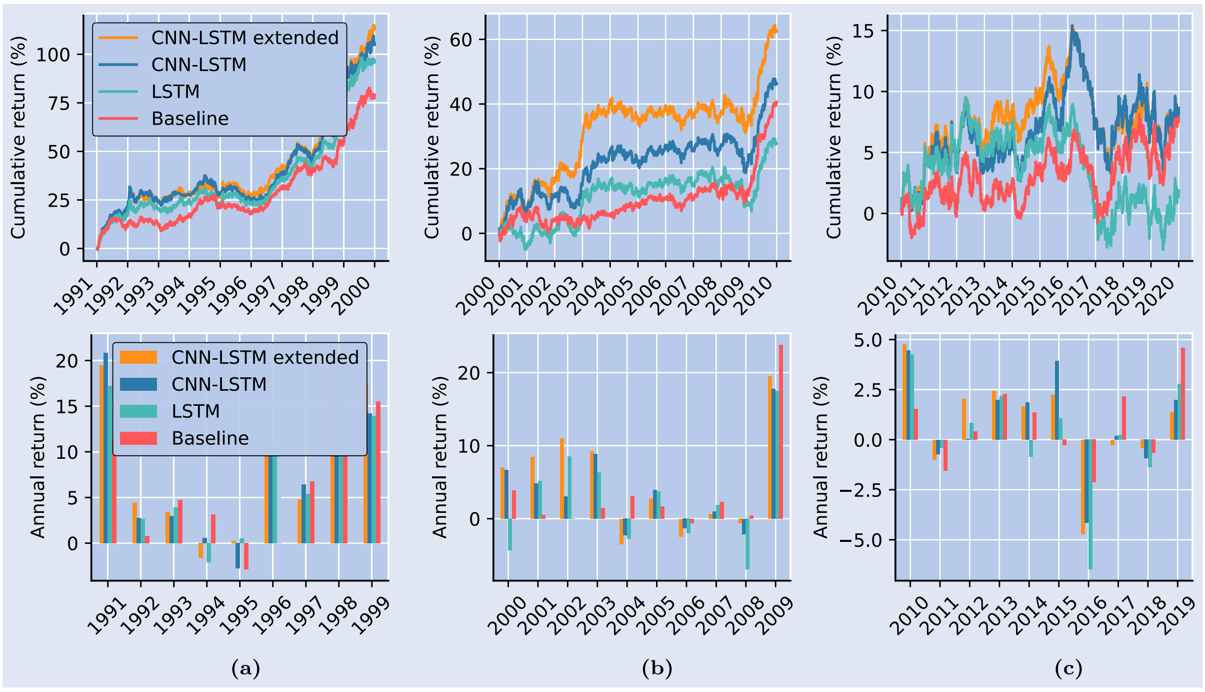


Figure 10. Cumulative returns of the strategies over three consecutive decades. (a) 1991–1999 (b) 2000–2009 (c) 2010–2017.

for the baseline strategy. As expected, the relative share of profitable trades is higher in the first trading periods compared to the later periods, which is intuitively plausible given the insights from Section 4.3.4. The reference LSTM model yields the lowest Hit rate, trades less frequent and marks the lowest average return among the three machine learning alternatives. Finally, Subfigure (c) shows that the average duration of trades opened by the machine learning strategies is much shorter compared to those of the baseline model. Overall the plots reveal that, from a trading analysis perspective, the profitability of the machine learning strategies (CNN-LSTM and extended CNN-LSTM) is based on a relatively high frequency of trades that are being closed in profit after only a short period of time.

Types of trade closings: To further shed light on the strategies' types of trade closings over time, figure 12 provides a transition plot of the relative share of the three possible different types of trade closings over the trading periods for the extended CNN-LSTM model and the baseline model. We note that the relative number of trades that have been closed due to reversion of the spread to the equilibrium value (i.e. closed by strategy) is overall higher for the deep learning strategy. Further, the standard deviation rule closes trades more often due to the ending of the trading period which is intuitive given that trades are generally longer in duration for the baseline strategy as seen in figure 11(c). Further, the stop loss threshold of -20% is more frequently hit by following the baseline strategy.

Duration and profitability: As figure 11(c) shows, the average duration of trades from the machine learning strategies is lower compared to that of the baseline model. We further note that to the end of the sample period, the average duration of trades is increasing. Given the decrease in profitability of all tested alternatives over time (see Section 4.3.4), we test if trade duration and profitability are linearly dependent. Figure 13 shows the results for the machine learning models

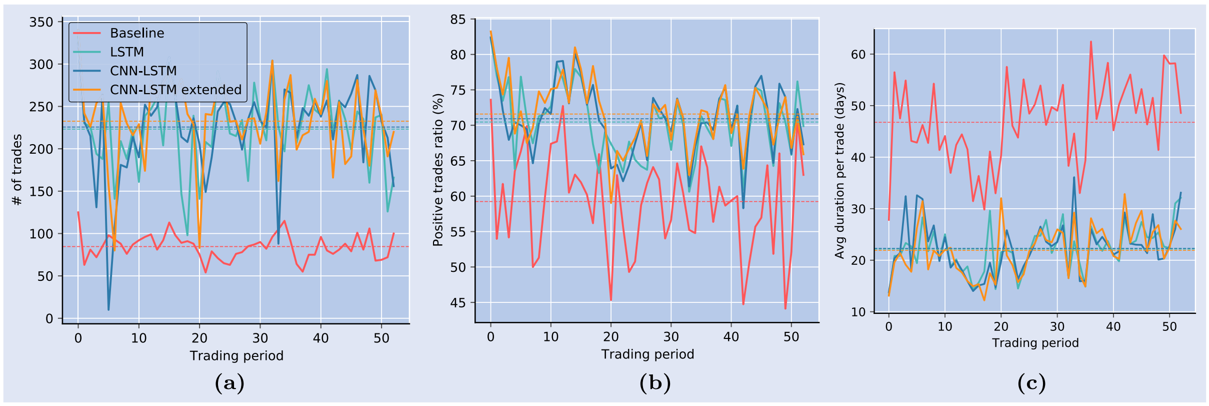


Figure 11. Trade statistics. Horizontal dotted lines indicate average values over the 53 samples. (a) Number of trades (b) Positive trades (c) Average duration.

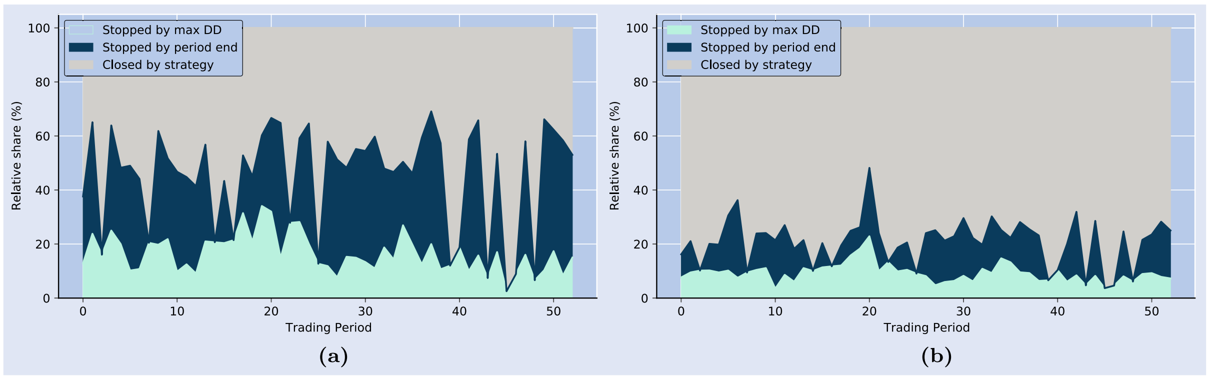


Figure 12. Regression of profit on duration per trade. (a) Baseline (b) CNN-LSTM ext.

and confirms that profits are indeed decreasing in duration. More precisely, an additional day of keeping a trade open decreases expected profits by -0.0641% for the extended CNN-LSTM model. Coefficients are similar for the other models and all are statistically significant at the 99.9% level.

It should therefore be possible to increase the risk-return profile by setting a further condition on closing the trade. As a simple additional rule we can set a limit of n business days for each trade to remain open before getting closed by default. Figure 14 and Table 4 provide a summary of the results for various values of n. Indeed, the risk-adjusted returns for $n =$ 10, 20, and 30 are superior to the default case, we tested so far with no condition on n (None). In terms of Sharpe ratio, there is a sweet spot at closing the trade after 10 days which yields a Sharpe ratio of 1.3 and shows an impressively low maximum drawdown of only -4.6% over the sample set.

4.3.6. Varying strategy configurations. Since we have based many strategy parameters on related experiments from the literature, the question arises whether we could further increase the Sharpe ratio by reconsidering the default choices. As suggested in the study of Harlacher (2016), we took the top $m = 5$ pairs from each sector to be eligible for opening a

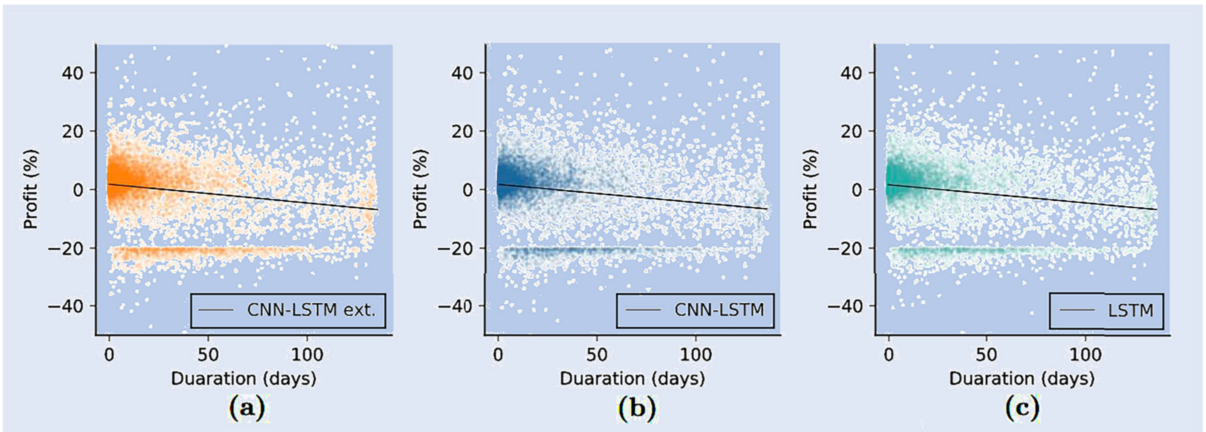


Figure 13. Regression of profitability on duration. (a) CNN-LSTM ext. (b) CNN-LSTM (c) LSTM.

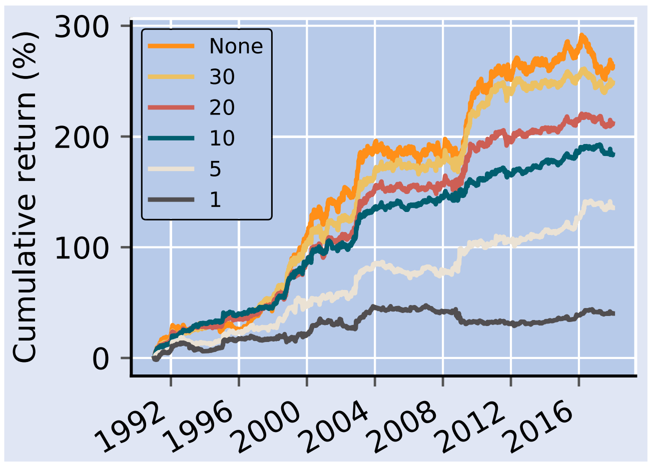


Figure 14. Cumulative performance of different closing day alternatives.

Table 4. Risk/return measures of different closing day alternatives.

n	Ann. Return (%)	Sharpe Ratio	Max DD (%)
None	5.0741	0.9526	− 10.1187
30	4.7275	1.122	− 6.8157
20	4.2958	1.1257	− 6.2918
10	3.9323	1.2837	− 4.6125
5	3.22	0.7668	− 7.7758
1	1.2563	0.4186	− 12.5707

pairs trade during the corresponding trading period. Table 5 shows the key economic performance measures for the two CNN-LSTM models and for the baseline strategy conducted with the range of $m = \{1, 2, 3, 4, 5\}$ pairs per sector. For $m = 1$ we note the highest final cumulative return for all strategies. However, this comes at the cost of very high standard deviations and pronounced drawdowns. The Sharpe ratios and Sortino ratios increase in m. For $m = 5$ we note the most attractive risk/return trade-off, which underlines the advantages of diversification even within the scope of this market neutral strategy.

We further apply a range of different pairs per sector m from 1 to 20 and a range of values for the maximum drawdown on each pair and re-run the backtest with the different settings.† Table 6 shows the resulting Sharpe ratios for the extended CNN-LSTM model. We note that the combination of $m = 5$ and a maximum drawdown of -20% indeed is a good choice with respect to risk-adjusted return. A further increase in the Sharpe ratio could have only been achieved by taking $m = 15$ pairs per sector. However, the improvement would have only been marginal. We can therefore conclude that the strategy parameters applied in this analysis have been a good choice in terms of economic performance and are rather robust given the neighboring values in table 6.

† Note that for the backtest with $m > 5$ we need to optimize the extended CNN-LSTM models each trading period again based on the enlarged data set, i.e. on $m = 20$. The results for $m = \{10, 15, 20\}$ are therefore based on newly trained models.

Table 5. Economic performance for different values of m.

	Baseline					CNN-LSTM					CNN-LSTM extended				
m	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Return p.a. (%)	5.2	3.9	3.1	3.0	3.5	5.9	4.1	3.7	3.7	4.4	7.2	5.0	4.1	4.2	5.1
Standard deviation p.a. (%)	9.1	6.7	5.6	5.0	4.6	9.9	7.4	6.3	5.6	5.2	10.0	7.4	6.4	5.7	5.3
Sharpe ratio p.a.	0.6	0.6	0.6	0.6	0.8	0.6	0.6	0.6	0.7	0.8	0.7	0.7	0.6	0.7	1.0
Sortino ratio p.a.	1.3	1.3	1.2	1.3	1.7	1.3	1.2	1.3	1.5	1.9	1.6	1.5	1.4	1.7	2.2
Max. drawdown (%)	− 18.4	− 15.3	− 16.1	− 12.9	−9.5	−21.3	− 24.8	− 18.7	− 12.9	− 10.5	− 18.2	− 24.7	− 20.0	− 12.3	− 10.1

Table 6. Sharpe ratios of different strategy settings.

		Maximum Drawdown					
		− 1%	− 5%	− 10%	− 20%	− 50%	− 100%
Pairs *m*	1	0.39	0.41	0.60	0.72	0.62	0.60
	5	0.50	0.67	0.70	0.95	0.78	0.74
	10	0.43	0.56	0.63	0.89	0.78	0.74
	15	0.47	0.62	0.76	0.98	0.82	0.78
	20	− 0.04	0.15	0.35	0.68	0.68	0.63

5. Conclusion

We train a deep CNN-LSTM network to solve the task of classifying profitable and non-profitable spread sequences of cointegrated US large cap stocks. The proposed model achieves high in- and out-of-sample accuracies and F1 scores around 0.7. Accuracies, however, are decreasing over time for a simple 3-feature input. This trend is reversed when we extend the feature space by including conditional probabilities for the spread to return to its long-run mean as calculated in the state space model of De Moura *et al.* (2016). Using the deep model estimations as a dynamic indicator function to trigger a pairs trade generates higher risk-adjusted returns over the sample period 1991 to 2017 compared to a static rule based on standard deviations. The inclusion of additional features increased economic and model performance. The study shows that deep learning can successfully be applied in a statistical arbitrage context and lays the foundation for further research on this topic. We conclude that the profitability of pairs trading should be reconsidered in light of its potential for improvement through deep learning. In further research, it might be of value to add more features to the data set. We found that adding two additional features already greatly increases economic performance and it would certainly be interesting to evaluate how much more alpha there is to be generated from other relevant features.

Acknowledgements

We thank the editor and two anonymous referees for carefully reading the manuscript and for several constructive and detailed comments that helped to improve our paper.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

E. Lütkebohmert http://orcid.org/0000-0002-5510-5091

References

Angerbauer, M., Grill, M. and Kulzer, A. Long short-term memory networks for the prediction of fuel cell voltage and efficiency. EasyChair Preprint no. 9126, 2022.

Ankerst, M., Breunig, M.M., Kriegel, H.-P. and Sander, J., Optics: Ordering points to identify the clustering structure. *ACM Sigmod Record*, 1999, **28**(2), 49–60.

Avellaneda, M. and Lee, J.-H., Statistical arbitrage in the US equities market. *Quant. Finance*, 2010, **10**(7), 761–782.

Bai, Y. and Wu, L., Analytic value function for optimal regime-switching pairs trading rules. *Quant. Finance*, 2018, **18**(4), 637–654.

Bogomolov, T., Pairs trading based on statistical variability of the spread process. *Quant. Finance*, 2013, **13**(9), 1411–1430.

Caldeira, J. and Moura, G.V., Selection of a portfolio of pairs based on cointegration: A statistical arbitrage strategy, 2013. Available at https://ssrn.com/abstract=2196391.

Cao, K., Kim, H., Hwang, C. and Jung, H., CNN-LSTM coupled model for prediction of waterworks operation data. *J. Inform. Processing Syst.*, 2018, **14**(6), 1508–1520.

Chen, H., Chen, S., Chen, Z. and Li, F., Empirical investigation of an equity pairs trading strategy. *Manage. Sci.*, 2017, **65**(1), 370–389.

Chen, H.-I. and Bassett, G., What does $\beta_{\text{smb}} > 0$ really mean? *J. Financ. Res.*, 2014, **37**(4), 543–552.

Clegg, M. and Krauss, C., Pairs trading with partial cointegration. *Quant. Finance*, 2018, **18**(1), 121–138.

Cummins, M. and Bucca, A., Quantitative spread trading on crude oil and refined products markets. *Quant. Finance*, 2012, **12**(12), 1857–1875.

De Moura, C.E., Pizzinga, A. and Zubelli, J., A pairs trading strategy based on linear state space models and the Kalman filter. *Quant. Finance*, 2016, **16**(10), 1559–1573.

Dunis, C.L., Giorgioni, G., Laws, J. and Rudy, J., Statistical arbitrage and high-frequency data with an application to Eurostoxx 50 equities, 2010. Available at https://ssrn.com/abstract=2272605.

Durbin, J. and Koopman, S.J., *Time Series Analysis by State Space Methods*, 2nd ed., 2012 (Oxford University Press: Oxford).

Eggebrecht, P. and Lütkebohmert, E., A deep trend following trading strategy for equity markets. *J. Financ. Data Sci.*, 2023, **5**(2), forthcoming.

Elliott, R.J., Van Der Hoek, J. and Malcolm, W.P., Pairs trading. *Quant. Finance*, 2005, **5**(3), 271–276.

Enke, D. and Thawornwong, S., The use of data mining and neural networks for forecasting stock market returns. *Expert. Syst. Appl.*, 2005, **29**(4), 927–940.

Fama, E.F. and French, K.R., Multifactor explanations of asset pricing anomalies. *J. Finance*, 1996, **51**(1), 55–84.

Fischer, T. and Krauss, C., Deep learning with long short-term memory networks for financial market predictions. *Eur. J. Oper. Res.*, 2018, **270**(2), 654–669.

Flori, A. and Regoli, D., Revealing pairs-trading opportunities with long short-term memory networks. *Eur. J. Oper. Res.*, 2021, **295**(2), 772–791.

Gatev, E., Goetzmann, W.N. and Rouwenhorst, K.G., Pairs trading: Performance of a relative-value arbitrage rule. *Rev. Financ. Stud.*, 2006, **19**(3), 797–827.

Gers, F.A., Schmidhuber, J. and Cummins, F., Learning to forget: Continual prediction with LSTM. *Neural. Comput.*, 2000, **12**(10), 2451–2471.

Guijarro-Ordonez, J., Pelger, M. and Zanotti, G., Deep learning statistical arbitrage, 2021. Available at https://arxiv.org/abs/2106.04028.

Halevy, A., Norvig, P. and Pereira, F., The unreasonable effectiveness of data. *IEEE. Intell. Syst.*, 2009, **24**(2), 8–12.

Harlacher, M., Cointegration based algorithmic pairs trading. PhD Thesis, University of St. Gallen, 2016. Available at https://www.e-helvetica.nb.admin.ch/api/download/urn%3Anbn%3Ach%3Abel-983813%3ADis4600.pdf/Dis4600.pdf.

Hochreiter, S. and Schmidhuber, J., Long short-term memory. *Neural. Comput.*, 1997, **9**(8), 1735–1780.

Hsu, P.-H., Taylor, M.P. and Wang, Z., Technical trading: Is it still beating the foreign exchange market? *J. Int. Econ.*, 2016, **102**, 188–208.

Huang, C.-J. and Kuo, P.-H., A deep CNN-LSTM model for particulate matter (pm$_{2.5}$) forecasting in smart cities. *Sensors*, 2018, **18**(7), 2220.

Huck, N., Pairs selection and outranking: An application to the S&P 100 index. *Eur. J. Oper. Res.*, 2009, **196**(2), 819–825.

Huck, N., Pairs selection and outranking: The multi-step-ahead forecasting case. *Eur. J. Oper. Res.*, 2010, **207**, 1702–1716.

Huck, N. and Afawubo, K., Pairs trading and selection methods: Is cointegration superior? *Appl. Econ.*, 2015, **47**(6), 599–613.

Kingma, D.P. and Ba, J., Adam: A method for stochastic optimization, 2014. Available at https://arxiv.org/abs/1412.6980.

Koudjonou, K.M. and Rout, M., A stateless deep learning framework to predict net asset value. *Neural Comput. Appl.*, 2020, **32**(14), 1–19.

Krauss, C., Statistical arbitrage pairs trading strategies: Review and outlook. *J. Econ. Surv.*, 2017, **31**(2), 513–545.

Krauss, C., Do, X.A. and Huck, N., Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *Eur. J. Oper. Res.*, 2017, **259**(2), 689–702.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P., Gradient-based learning applied to document recognition. *Proc. IEEE*, 1998, **86**(11), 2278–2324.

Leung, M.T., Daouk, H. and Chen, A.-S., Forecasting stock indices: A comparison of classification and level estimation models. *Int. J. Forecast.*, 2000, **16**(2), 173–190.

Livieris, I., Pintelas, E.G. and Pintelas, P., A CNN-LSTM model for gold price time-series forecasting. *Neural Comput. Appl.*, 2020, **32**, 17351–17360.

Lu, W., Li, J., Li, Y., Sun, A. and Wang, J., A CNN-LSTM-based model to forecast stock prices. *Complexity*, 2020, **2020**, 6622927.

McKinney, W., Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, edited by S. van der Walt and J, Millman, Vol. 445, pp. 56–61, 2010 (Austin).

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J., Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 2011, **12**, 2825–2830.

Petersen, P.C., Neural network theory. University of Vienna, 2020. Available at http://www.pc-petersen.eu/Neural_Network_Theory.pdf.

Said, S.E. and Dickey, D.A., Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 1984, **71**(3), 599–607.

Sarmento, S.M. and Horta, N., Enhancing a pairs trading strategy with the application of machine learning. *Expert. Syst. Appl.*, 2020, **158**, 113490.

Seabold, S. and Perktold, J., Statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference*, edited by S. van der Walt and J, Millman, pp. 92–96, 2010 (Austin).

Shi, X., Gao, Z., Lausen, L., Wang, H., Yeung, D.-Y., Wong, W.-k. and WOO, W.-C., Deep learning for precipitation nowcasting: A benchmark and a new model. *Adv. Neural. Inf. Process. Syst.*, 2017, **30**, 5617–5627.

Suzuki, K., Optimal pair-trading strategy over long/short/square positions—empirical study. *Quant. Finance*, 2018, **18**(1), 97–119.

Swapna, G., Kp, S. and Vinayakumar, R., Automated detection of diabetes using CNN and CNN-LSTM network and heart rate signals. *Procedia. Comput. Sci.*, 2018, **132**, 1253–1262.

Van Der Walt, S., Colbert, S.C. and Varoquaux, G., The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.*, 2011, **13**(2), 22–30.

Vidyamurthy, G., *Pairs Trading: Quantitative Methods and Analysis*, 2004 (John Wiley & Sons: Hoboken).

Widiputra, H., Mailangkay, A. and Gautama, E., Multivariate CNN-LSTM model for multiple parallel financial time-series prediction. *Complexity*, 2021, **2021**, 1–14.

Wilmott, P., *Paul Wilmott on Quantitative Finance*, 2006 (John Wiley & Sons: Chichester).

Yadav, A., Jha, C. and Sharan, A., Optimizing lstm for time series prediction in indian stock market. *Procedia. Comput. Sci.*, 2020, **167**, 2091–2100.

Zeng, Z. and Lee, C.-G., Pairs trading: Optimal thresholds and profitability. *Quant. Finance*, 2014, **14**(11), 1881–1893.

Zhang, G., Pairs trading with general state space models. *Quant. Finance*, 2021, **21**(9), 1567–1587.

Appendices

Appendix 1. Hybrid CNN-LSTM model for signal detection

Our proposed model is a combination of two advanced machine learning methods: *Long Short-Term Memory (LSTM)* and *Convolutional Neural Network (CNN)* models. Both models are built upon artificial neural networks which are formally defined as follows.

A *Multi-Layer Perceptron (MLP) model* with input dimension $d_{\text{in}} \in \mathbb{N}$, output dimension $d_{\text{out}} \in \mathbb{N}$, and $l \in \mathbb{N}$ layers is a function from $\mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ of the form

$$\boldsymbol{x} \mapsto \phi_l \circ A_l \circ \phi_{l-1} \circ A_{l-1} \circ \cdots \circ \phi_1 \circ A_1(\boldsymbol{x}),$$

where $(A_i)_{i=1,\ldots,l}$ are affine functions $A_i : \mathbb{R}^{h_{i-1}} \to \mathbb{R}^{h_i}$ and $h_i \in \mathbb{N}$ denotes the *number of neurons* of layer i with $h_0 = d_{\text{in}}$ and $h_l = d_{\text{out}}$. The (non-constant) *activation functions* $\phi_i : \mathbb{R} \to \mathbb{R}$ are applied component-wise on the output of the affine function A_i. The affine function A_i of layer i with h^i neurons can be described by the tensor $\Theta_i = (\boldsymbol{b}_i, \boldsymbol{W}_i)$ with weight matrix $\boldsymbol{W}_i \in \mathbb{R}^{h_i \times h_{i-1}}$ and bias vector $\boldsymbol{b}_i \in \mathbb{R}^{h_i}$. Then, for an input vector $\boldsymbol{x}$, the output of layer i is given by

$$Q(\boldsymbol{x}, \Theta_i, \phi) = \phi(\boldsymbol{b}_i + \boldsymbol{W}_i^\top \boldsymbol{x}). \tag{A1}$$

The neural network is called *deep* if there is at least one hidden layer, i.e. if $l \geq 2$.[†]

Long Short-Term Memory (LSTM) networks as introduced in Hochreiter and Schmidhuber (1997) are a special form of recurrent neural networks that allow to capture long-term temporal dependencies in sequential data by introducing a complex artificial neuron called *LSTM memory block*. The distinct features of LSTM blocks are gates[‡], which control the flow of data through the memory block and enable the cell to keep its activation over long time periods. This is achieved by 'protecting' the activation of the memory block, referred to as the *cell state*, by the three gates which learn when to let activation pass forward through the network and when to stop it. First, the forget gate $\boldsymbol{f}_t = \phi_g(\boldsymbol{W}_{f,h}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{f,x}\boldsymbol{x}_t + \boldsymbol{b}_f)$ determines which information to let pass or block from the previous cell state $\boldsymbol{c}_{t-1}$.[§] Due to the sigmoid activation ϕ_g, all input values get scaled into the range between 0 (completely forget) and 1 (completely remember). The input gate $\boldsymbol{i}_t = \phi_g(\boldsymbol{W}_{i,h}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{i,x}\boldsymbol{x}_t + \boldsymbol{b}_i)$ then selectively determines what new information will be added to the cell state. Next, a vector of candidate values $\tilde{\boldsymbol{c}}_t = \phi_c(\boldsymbol{W}_{\tilde{c},h}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{\tilde{c},x}\boldsymbol{x}_t + \boldsymbol{b}_{\tilde{c}})$ with tanh activation ϕ_c is created and added to the adjusted cell state to form the new cell state $\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t$, where $\odot$ denotes the Hadamard (elementwise) product. As a final step, the new hidden state $\boldsymbol{h}_t = \boldsymbol{o}_t \odot \phi_c(\boldsymbol{c}_t)$ is computed by elementwise multiplication of the activation of the updated cell state $\boldsymbol{c}_t$ and

[†] We refer to Petersen (2020) for a detailed mathematical study on neural networks.

[‡] We will refer to the LSTM model as established by Gers *et al.* (2000), who modified the original LSTM of Hochreiter and Schmidhuber (1997) and proposed a total of three gates named according to their functions: *input*, *output* and *forget gate*.

[§] Subscripts are expressing the to-from-relationships, i.e. $\boldsymbol{W}_{f,h}$ denotes the recurrent weight connection from the previous time step's hidden state $\boldsymbol{h}_{t-1}$ to the current time step's forget gate $\boldsymbol{f}_t$.

the activation of the output gate $\boldsymbol{o}_t = \phi_g(\boldsymbol{W}_{o,h}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{o,x}\boldsymbol{x}_t + \boldsymbol{b}_o)$. Recalling equation (A1) reveals that each gate in the LSTM memory block is a trainable non-linear simple neural network itself. The input to these internal networks is always a combination of the model's previous hidden state $\boldsymbol{h}_{t-1}$ and the current input $\boldsymbol{x}_t$. The LSTM memory block can therefore be described as a stacked model of multiple neural networks working together, with each individual network contributing to the final combined output.

Convolutional Neural Networks (CNN) models (sometimes *ConvNet*) as introduced in LeCun *et al.* (1998) are artificial neural network models where the input dimension is enlarged by replacing each input x_i by an input matrix $\boldsymbol{X}_i$ and each single connecting weight ω_i by a matrix $\boldsymbol{W}_i$. We then obtain a *convolutional unit* of the form

$$c(\boldsymbol{X}, \Theta, \phi) = \phi\left(b + \sum_i \boldsymbol{W}_i \boldsymbol{X}_i\right). \tag{A2}$$

Multiple convolutional units receiving the same input constitute a *convolutional layer*:

$$C(\boldsymbol{X}, \Theta, \phi)_j = \phi\left(\boldsymbol{b}_j + \sum_i \boldsymbol{W}_{i,j} \boldsymbol{X}_i\right). \tag{A3}$$

Input tensors $\boldsymbol{X}$ are referred to as *feature maps* and the weight tensor $\boldsymbol{W}$ is commonly called the *filter* or *kernel*. Typically, *pooling layers* are added after the convolutional layer to reduce the feature dimension by shrinking the feature maps. The last part of a CNN model for most tasks is constructed as one (or more) simple fully-connected layer(s), that integrates information across all spatial locations of the convolutional layers and produces the final estimation. CNNs are receptive to the ordering of the input. For time series, this means that specific patterns are learned as local features of the input.

Appendix 2. Calculation of conditional probabilities

In this appendix, we briefly review the state space approach by De Moura *et al.* (2016) for the calculation of conditional probabilities for the spread to return to its long-run mean. Following Elliott *et al.* (2005) the authors suggest a Gaussian linear model:

$$\text{Measurement equation:}\quad S_t = Z_t x_t + d_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, H_t) \tag{A4}$$

$$\text{State equation:}\quad x_{t+1} = T_t x_t + c_t + R_t \eta_t, \quad \eta_t \sim \mathcal{N}(0, Q_t), \tag{A5}$$

and $x_1 \sim \mathcal{N}(a_1, P_1)$, with system matrices $Z_t, d_t, H_t, T - t, c_t, R_t$ and Q_t that are deterministic (or only depend on the past value of S_t) and where the error terms ϵ and η are independent (of each other and in time). Here S_t denotes the observed spread and x_t the unobserved mean-reverting state variable. For any time instants $t, j \in \{1, 2, \ldots, n\}$ with $j < t$, denote the sigma-field $\mathcal{F}_j = \sigma(S_1, \ldots, S_j)$ and the conditional moments by $a_{t\,|\,j} = \mathrm{E}[x_t \,|\, \mathcal{F}_j]$ and $P_{t\,|\,j} = \mathrm{Var}(x_t \,|\, \mathcal{F}_j)$. As shown in detail in Durbin and Koopman (2012), the Kalman filter formulae for one-day ahead predictions then become:

$$\begin{aligned} &v_t = S_t - Z_t a_{t\,|\,t-1} - d_t && F_t = Z_t P_{t|t-1} Z_t^\top + H_t \\ &K_t = T_t P_{t|t-1} Z_t^\top F_t^{-1} && L_t = T_t - K_t Z_t, \quad t = 1, \ldots, n \\ &a_{t+1\,|\,t} = T_t a_{t\,|\,t-1} + c_t + K_t v_t && P_{t+1\,|\,t} = T_t P_{t|t-1} L_t^\top + R_t Q_t R_t^\top \end{aligned} \tag{A6}$$

where v_t are called the model 'innovations'. If S_t follows an ARMA(p,q) model, then it can be formulated according to (A6) due to the mean-reverting property of the spread (compare De Moura *et al.* (2016), Proposition 1).

The probabilities of the spread to return to its equilibrium value c within the next k time steps conditional on the current information contained in the σ-algebra $\mathcal{F}_t$ can be expressed as $P_{up} = 1 - F_{S_{t+1}, S_{t+2}, \ldots, S_{t+k} | \mathcal{F}_t}(c, c, \ldots, c)$ for up movements of the spread and as $P_{down} = 1 - F_{-S_{t+1}, -S_{t+2}, \ldots, -S_{t+k} | \mathcal{F}_t}(-c, -c, \ldots, -c)$ for down movements of the spread, where F denotes the cumulative distribution function (for details see De Moura *et al.* (2016) equations 5–9). In the Gaussian linear state space model considered here, we have

$$S_{t,k} \equiv (S_{t+1}, \ldots, S_{t+k}) \,|\, \mathcal{F}_t \sim N(\mu_{t,k}, \Sigma_{t,k}),$$

where $\mu_{t,k} \equiv \mathbb{E}[S_{t,k} \,|\, \mathcal{F}_t]$ and $\Sigma_{t,k} \equiv Var[S_{t,k} \,|\, \mathcal{F}_t]$. For any $k > 1$ the available one-step-ahead estimations $x_{t+1\,|\,t}$ and $P_{t+1\,|\,t}$ are not sufficient to compute these conditional probabilities. Instead, the conditional moments need to be iteratively evaluated for each time step t. This involves using an augmented state space form equivalent to a given time series model (i.e. models (A4)–(A5)) estimated using the spread data. To achieve this, De Moura *et al.* (2016) propose to construct a k-times extended state space model to obtain first and second order conditional moments of $[x_{t+1}, \ldots, x_{t+k}]'$ from the following system:

$$S_t = \begin{bmatrix} Z_t & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_t \\ x_{t-1} \\ \vdots \\ x_{t-k} \end{bmatrix} + d_t + \varepsilon_t$$

$$\begin{bmatrix} x_{t+1} \\ x_t \\ \vdots \\ x_{t-(k-1)} \end{bmatrix} = \begin{bmatrix} T_k & 0 & \cdots & 0 \\ I & 0 & \cdots & 0 \\ \vdots & \ddots & \cdots & \vdots \\ 0 & \cdots & I & 0 \end{bmatrix} \begin{bmatrix} x_t \\ x_{t-1} \\ \vdots \\ x_{t-k} \end{bmatrix} + \begin{bmatrix} c_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} R_t \\ 0 \\ \vdots \\ 0 \end{bmatrix} \eta_t, \tag{A7}$$

where Z_t, d_t, T_t, R_t and c_t are the system matrices of the original model (A4) and (A5). The first- and second-order conditional moments of $[x_{t+1}, x_t, \ldots, x_{t+k}]^\top$ allow to obtain

$$\begin{aligned} \mathbb{E}[S_{t+i} \,|\, \mathcal{F}_t] &= Z_{t+i}\mathbb{E}[x_{t+i} \,|\, \mathcal{F}_t] + d_{t+i} + \mathbb{E}[\epsilon_{t+i}] \\ &= Z_{t+i} a_{t+i\,|\,t} + d_{t+i} \\ Var[S_{t+i} \,|\, \mathcal{F}_t] &= Z_{t+i} P_{t+i\,|\,t} Z_{t+i}^\top + H_{t+i} \\ Cov[S_{t+i}, S_{t+j} | \mathcal{F}_t] &= Z_{t+i} Cov[x_{t+i}, x_{t+j} \,|\, \mathcal{F}_t] Z_{t+j}^\top \\ &\equiv Z_{t+i} P_{t+i, t+j\,|\,t} Z_{t+j}^\top \end{aligned}$$

for $i, j = 1, 2, \ldots, k$ from the original model as there are no changes in the maximum likelihood estimation when considering the augmented model (A7) (see De Moura *et al.* (2016), Proposition 2). Based on the extended state space model, the k-step-ahead estimates for the conditional probabilities P_{up} and P_{down} are then evaluated using standard numerical multiple integration algorithms for multivariate normal distributions.†

† See https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.multivariate_normal.html.