Neural Computing and Applications (2020) 32:15249–15262
https://doi.org/10.1007/s00521-020-04877-9

ORIGINAL ARTICLE

Thresholded ConvNet ensembles: neural networks for technical forecasting

Sid Ghoshal[1] · Stephen Roberts[1]

Received: 22 February 2019 / Accepted: 20 March 2020 / Published online: 20 April 2020

Abstract
Much of modern practice in financial forecasting relies on *technicals*, an umbrella term for several heuristics applying visual pattern recognition to price charts. Despite its ubiquity in financial media, the reliability of its signals remains a contentious and highly subjective form of 'domain knowledge'. We investigate the predictive value of patterns in financial time series, applying machine learning and signal processing techniques to 22 years of US equity data. By reframing technical analysis as a poorly specified, arbitrarily preset feature-extractive layer in a deep neural network, we show that better convolutional filters can be learned directly from the data, and provide visual representations of the features being identified. We find that an ensemble of shallow, thresholded convolutional neural networks optimised over different resolutions achieves state-of-the-art performance on this domain, outperforming technical methods while retaining some of their interpretability.

Keywords Technical analysis · Machine learning · Deep neural networks

1 Introduction

In financial media, extensive attention is given to the study of charts and visual patterns. Known as *technical analysis* or *chartism*, this form of financial analysis relies solely on historical price and volume data to produce forecasts, on the assumption that specific graphical patterns hold predictive information for future asset price fluctuations [1]. Early research into genetic algorithms devised solely from technical data (as opposed to e.g. fundamentals or sentiment analysis) showed promising results, sustaining the view that there could be substance to the practice [2, 3].

The rising popularity of neural networks in the past decade, fuelled by advances in computational processing power and data availability, renewed interest in their applicability to the domain of finance. Krauss et al. [4] applied multilayer perceptrons (MLPs) to find patterns in the daily returns of the S&P500 stock market index. Dixon et al. [5] further demonstrated the effectiveness of neural nets on intraday data, deploying MLPs to classify returns on commodity and FX futures over discrete 5-min intervals. Architectures comprised of 4 dense hidden layers were sufficient to generate annualised Sharpe ratios in excess of 2.0 on their peak performers. In each instance, patterns were sought in the time series of returns rather than in the price process itself.

Seminal findings by Lo et al. [6] employed instead the visuals emerging from line charts of stock closing prices, relying on kernel regression to smooth out the price process and enable the detection of salient trading patterns. An equally common visual representation of price history in finance is the candlestick. Candlesticks encode the opening price, closing price, maximum price and minimum price over a discrete time interval, visually represented by a vertical bar with lines extending on either end. Much as with line charts, technical analysts believe that specific sequences of candlesticks reliably foreshadow impending price movements. A wide array of such patterns are commonly watched for [7], each with their own pictogram and associated colourful name ('inverted hammer', 'abandoned baby', etc).

✉ Sid Ghoshal
sid.ghoshal@shell.com

Stephen Roberts
sjrob@robots.ox.ac.uk

[1] Department of Engineering Science, Oxford-Man Institute of Quantitative Finance, University of Oxford, Oxford, UK

Though recurrent neural networks–and in particular Long Short-Term Memory (LSTM) models [8]—have been the most popular choice for deep learning on time series data [9–11], promising results have begun to appear from the application of convolutional neural networks to financial data. Neural networks have frequently been labelled as black boxes, limiting their deployment in domains where interpretability is sought. Convolutional neural networks partially overcome this, by extracting locally interpretable features in their early layers. Furthermore, recent research suggests that these models bear the capacity to generalise not merely across time but across assets as well, identifying universal features of stock market behaviour [12, 13].

The contributions of this paper are threefold: firstly, we rigorously evaluate the practice of candlestick chartism, and find little evidence to support it. The human-engineered features prescribed by technical analysis produce classifiers that barely outperform guesswork, unlike the patterns identified through deep learning. Secondly, we show that filters learned and tested on 22 years of S&P500 price data in a CNN architecture yield modest gains in accuracy over both technical methods and machine learning alternatives, including MLPs unsupported by the feature extraction capabilities of a convolutional layer. Thirdly, we demonstrate that considerable gains in forecasting capability are achievable through ensemble methods and thresholding based on model confidence.

This paper evaluates quantitatively the merits of candlestick-driven technical analysis before proposing an improved, data-driven approach to financial pattern recognition. Formally, we reframe candlestick patterns as a form of *feature engineering* intended by chartists to extract salient features, facilitating the classification of future returns with higher fidelity than the raw price process would otherwise allow. After a brief review of neural networks, we define the data used throughout the paper (Sect. 2) and motivate the pursuit of new, better visual heuristics for finance by assessing the predictiveness of candlestick formations (Sect. 3). Feeding candlestick data through a neural network involving separate filters for each technical pattern, we classify next-day returns with the filters implied by chartist doctrine (Sects. 4.1–4.2) and set this cross-correlational approach as a baseline to improve upon [14]. We then compare the model's accuracy when filters are not preset but instead learned by convolutional neural networks (CNNs) during their training phase (Sect. 4.3), and benchmark deep learning against alternative methods drawn from both traditional finance and machine learning (Sect. 4.4). We enhance the accuracy of CNNs through the addition of thresholding and ensembling (Sect. 4.5), and finish with two practically minded extensions: the backtested performance of the model (Sect. 4.6) and the visual interpretation of the features extracted by the CNN (Sect. 4.7).

2 Methodology overview

Over the last decade, neural networks have risen dramatically in popularity, propelled by the success of deep learning in a wide range of practical applications. Formally, neural networks map inputs to outputs through a collection of nonlinear computation nodes, called *neurons*, stacked into *hidden layers*. Inputs and outputs are connected by potentially many such hidden layers, leading to the so-called deep learning architectures. Neural networks can be interpreted as an ultra-parametric extension of linear regression models, wherein each neuron computes a weighted linear combination of its inputs, applies a nonlinear transformation to the newfound value and forwards its output to the next layer's neurons.

2.1 Multilayer perceptrons

The benefit of nonlinear transformation is particularly pronounced as architectures are extended in depth: without nonlinearity, additional layers would not confer any incremental value, as the linear combinations of linear combinations would themselves just be linear combinations with different weights. In other words, multiple hidden layers without nonlinear transformations would be equivalent to a single hidden layer with appropriately chosen weights. The inclusion of nonlinear transformations, termed *activation functions*, between the hidden layers allows neural networks to learn complex functional mappings. MLPs harness this potential by including multiple layers between input and output and allowing each layer to possess many neurons (Fig. 1). The activation functions employed are commonly the hyperbolic tangent function $\tanh(o)$, logistic function $\sigma(o)$ and rectified linear unit $\mathrm{ReLU}(o)$.

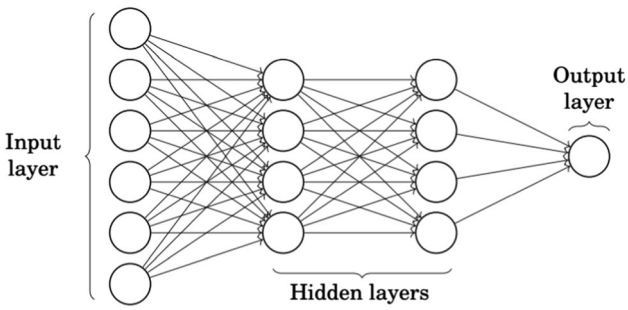

Fig. 1 A multilayer perceptron with 2 hidden layers. Each neuron within a layer computes a linear combination of its inputs followed by a nonlinear transformation

$$\tanh(o) = \frac{\sinh(o)}{\cosh(o)} = \frac{e^{o} - e^{-o}}{e^{o} + e^{-o}} \quad (1)$$

$$\sigma(o) = \frac{1}{1 + e^{-o}} \quad (2)$$

$$\text{ReLU}\ (o) = \max(0, o) \quad (3)$$

We adopt ReLu as our preferred activation function, a common choice in neural network architectures given its computational efficiency and performance [15].

2.2 Convolutional neural networks

Convolutional neural networks extend multilayer perceptrons, by adding one or several additional layers at the beginning of the architecture. These layers, termed *convolutional layers*, consist of a set of learned filters. These filters are typically much smaller than the input, and measure local similarity (calculated by sliding dot or Hadamard product). The output of a convolutional layer is a feature map, identifying regions where the input to the layer was similar to the learned filter. In effect, convolution functions as bespoke feature extractors for neural network architectures, enabling in the process vastly superior model performance (Fig. 2).

2.3 Definition of candlestick data

Both the financial time series data and the candlestick technical filters used by chartists take the same form. Asset price data for a discrete time interval is represented by four features: the opening price (price at the start of the interval), closing price (price at the end of the interval), high price (maximum over the interval) and low price (minimum over the interval). The candlestick visually encodes this information (Fig. 3): the bar's extremities denote the

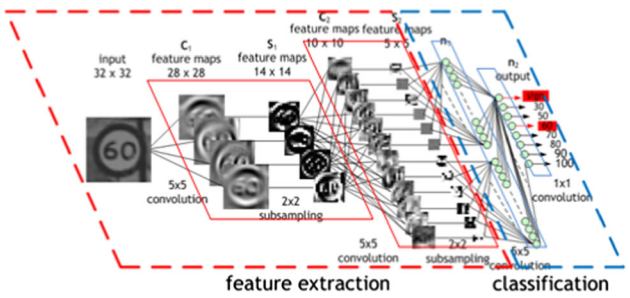

Fig. 2 A convolutional neural network with 2 convolutional layers. The red hashed outline contains the feature-extractive convolutional layers, and the blue hashed outline is effectively a single-layer perceptron. In this architecture, convolutional outputs over a local area are reduced to a single value via *subsampling* or *pooling*, an operation designed to improve the model's memory footprint and invariance to translations/rotations. Image from Nvidia (https://developer.nvidia.com/discover/convolutional-neural-network)

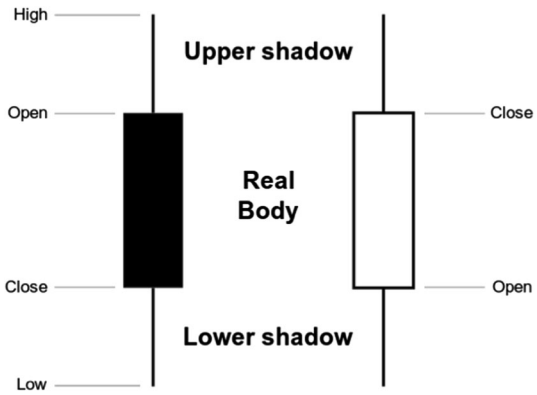

Fig. 3 Candlestick representation of financial time series data

open and close prices, and the lines protruding from the bar (the candle's 'wicks' or shadow) denote the extrema over the interval. The colour of the bar determines the relative ordering of the open and close prices: a white bar denotes a positive return over the interval (close price > open price) and a black or shaded bar denotes a negative return (close price < open price).

We can therefore summarise the candlestick representation of a financial time series of length n timesteps as a $4 \times n$ price signal matrix F capturing its four features. Throughout this paper, we rely on daily market data, but the methods can be extended to high-frequency pattern recognition on limit order books—an active area for current research [13].

2.4 Definitions of technical patterns

We include major candlestick patterns cited by practitioners of technical analysis at three timescales: 1-day, 2-day and 3-day patterns. The simple 1-day patterns include the hammer (normal and inverted), hanging man, shooting star, dragonfly doji, gravestone doji, and spinning tops (bullish and bearish, where bullish implies a positive future return and bearish implies a negative future return). Our 2-day patterns cover the engulfing (bullish and bearish), harami (bullish and bearish), piercing line, cloud cover, tweezer bottom and tweezer top. Finally our 3-day patterns cover some of the most cited cases in chartist practice: the abandoned baby (bullish and bearish), morning star, evening star, three white soldiers, three black crows, three inside up and three inside down. Figure 4 provides both the visual template associated with each pattern and the future price direction it is meant to presage. As before, we summarise a technical pattern P of length m timesteps as a $4 \times m$ matrix T_{P_m}, standardised for comparability to have zero mean and unit variance.

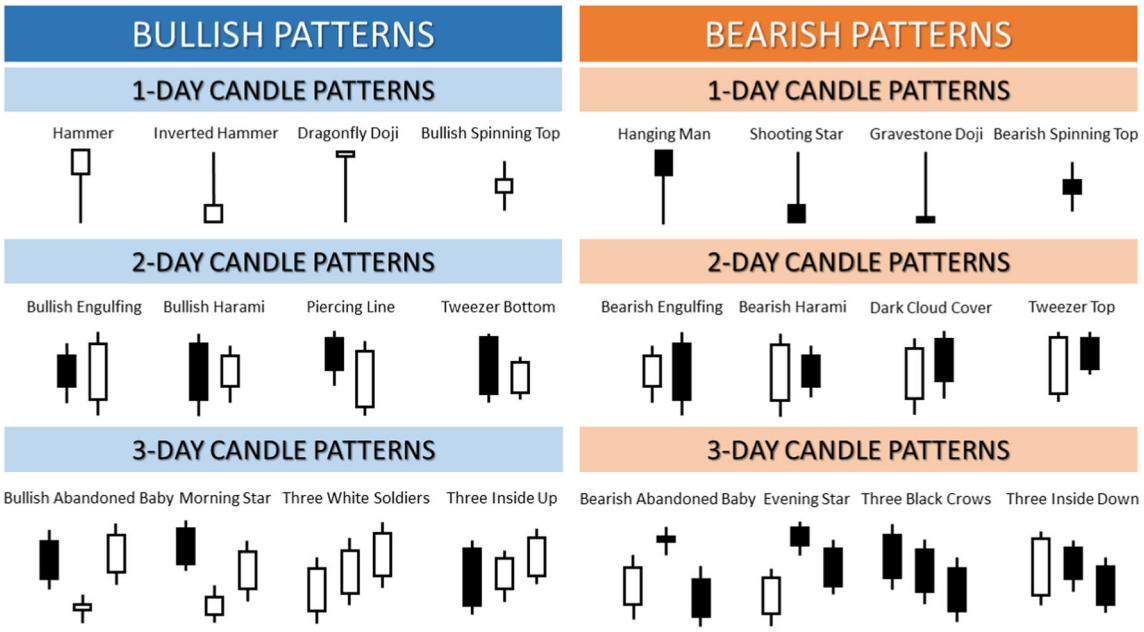

Fig. 4 For each timescale (1-day, 2-day and 3-day), we specify 8 chartist patterns and the future direction they predict ('bullish' for positive returns, 'bearish' for negative returns)

2.5 Empirical data

Throughout our work, we use daily technical (i.e. open, close, high and low price) data from the S&P500 stock market index constituents for the period January 1994–December 2015, corresponding to $n = 2{,}439{,}184$ entries of financial data in the price signal F.[1] This data set covers a representative cross section of US companies across a wide timeframe suitable for learning the patterns, if any, of both expansionary and recessionary periods in the stock market.

3 Evaluation of current tools

As a preliminary motivation for the adoption of machine learning for technical forecasts, we assess the merits of candlestick chartism in finance. We run several diagnostics to assess separately the informativeness and predictiveness for each technical pattern.

3.1 Conditioning returns on the presence of a pattern

The $4 \times m$ matrix representation T_{P_m} for pattern P of length m and equal-length, standardised rolling windows F_n of the full price signal F at timestep n can be cross-correlated together to generate a time series S_P measuring the degree of similarity between the price signal and the pattern. For a given pattern P, at each timestep n:

$$S_{P,n} = \left\langle \frac{T_{P_m}}{\|T_{P_m}\|}, \frac{F_n}{\|F_n\|}, \right\rangle \tag{4}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of the two matrices and $\| \cdot \|$ is the L^2 norm.

For each pattern P, we produce a conditional distribution of next-day returns by extracting the top quantile (in our study, decile and centile) of similarity scores S_P.

3.2 Informativeness

For our purposes, we define a technical pattern to be *informative* if its presence significantly alters the distribution of next-day returns in a Kolmogorov–Smirnov two-sample (K–S) test [16], comparing the unconditional distribution of all next-day returns to the distribution conditioned on having just witnessed the pattern. The K–S test is a non-parametric test which can be used to compare similarity between two empirical or analytical distributions. The two-sample K–S test is one of the most useful and general nonparametric methods, as it is sensitive to differences in both location and shape of the empirical cumulative distribution functions of the two samples. As the test is non-parametric, no assumptions are made regarding the nature of the sample distributions. Denoting by $\{R_{P\,t=1}^{n_1}\}$ the subset of returns conditioned on matching pattern P and $\{R_{t=1}^{n_2}\}$ the full set of unconditional returns, we compute their

[1] We include 500 individual stocks from the S&P500 index as of 31 December 2015. Each stock's daily open, close, high and low is recorded over a period of up to 22 years, with each year including 252 business days on average.

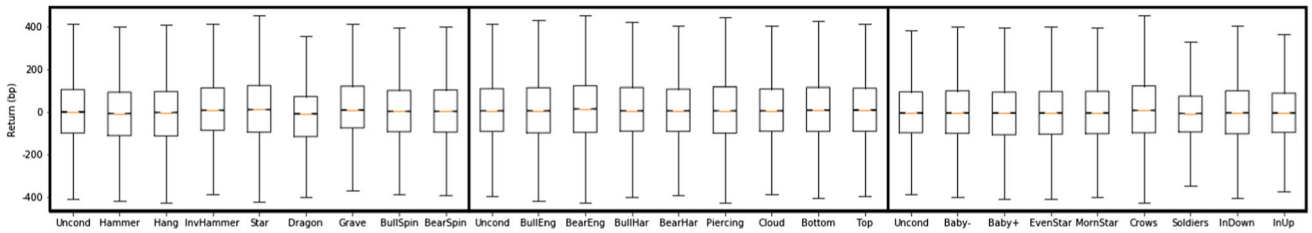

Fig. 5 Notched boxplots of the distributions of returns in basis points (100th of a percent), conditional on observing each of the technical patterns (similarity score S_P in its top centile). Whiskers cover twice the interquartile range. At a glance, none of the conditional distribution medians diverge substantively from the unconditional baseline of zero, and the distributions' deviations dwarf their medians by two orders of magnitude

empirical cumulative distribution functions $F_1(z)$ and $F_2(z)$. The K–S test evaluates the null hypothesis that the distributions generating both samples have identical cdfs, by computing the K–S statistic:

$$\gamma = \left(\frac{n_1 n_2}{n_1 + n_2}\right)^{1/2} \sup_{-\infty < z < \infty} |F_1(z) - F_2(z)| \tag{5}$$

The limiting distribution of γ provides percentile thresholds above which we reject the null hypothesis. When this occurs, we infer that conditioning on the pattern does materially alter the future returns distribution.

3.3 Predictiveness

While these patterns may bear some information, it does not follow that their information is actionable, or even aligns with the expectations prescribed by technical analysis. Notched boxplots of both unconditional returns and returns conditioned on each of the filters (Fig. 5) allow us to gauge whether the pattern's occurrence does in fact yield significant returns in the intended direction.

A closer examination suggests that several of the 1-day patterns are in fact relevant, but that the more elaborate 2-day and 3-day formations are not. Conditioning on 14 of the 16 multi-day patterns produces no significant alteration in the median of next-day returns distributions (Fig. 6): only the 'Bearish Engulfing' and 'Three Black Crows' patterns produce a conditional distribution for which the 95% confidence interval of the median (denoted by the notch) differs markedly from its unconditional counterpart.

3.4 Findings

We report the empirical results of the K–S goodness-of-fit tests and top decile and centile (Tables 1, 2, respectively) conditional distribution summary statistics, using daily stock data from the S&P500. Though several of the patterns do indeed bear information altering the distribution of future returns, their occurrence is neither a reliable predictor of price movements (high standard deviation relative to the mean) nor even, in many instances, an accurate classifier of direction. Elaborate multi-day patterns systematically perform worse than their single-day counterparts. Surprisingly, 6 of the 8 single-day patterns do in fact produce meaningful deviations from the unconditional baseline, with the dragonfly and gravestone doji standing out as significant outliers ($-$ 25.81 bp[2] and $+$ 22.41 bp, respectively, when conditioning on the top centile of similarity score, Table 2). But even in those instances, technical analysis forecasts incorrectly, as prices move in the direction opposite to chartism's predictions. McLean and Pontiff [18] showed that predictor variables lose on average 58% of their associated return, post-publication. In a similar vein, we hypothesise that these patterns may have once been predictive on a historical data set, but that their disclosure and subsequent overuse has long since negated their value. Conceptually, the notion of using filters in financial data to extract informative feature maps may bear merit—but the chartist filter layer is demonstrably an improper specification today.

4 Proposed improved approach

The approach of searching for informative intermediate feature maps in classification problems has seen widespread success in domains ranging from computer vision [15] to acoustic signal processing [19]. Where technical analysis uses filters that are arbitrarily pictographic in nature, we turn instead to *convolutional layers* to extract features from data. We evaluate the performance of passing the raw data both with and without chartist filters, and subsequently measure the incremental gain from learning optimal feature maps by convolution. The findings are then benchmarked against widely recognised approaches to time series forecasting including autoregression, recurrent neural networks, nearest neighbour classifiers, support vector machines (SVM) and random forests.

[2] A basis point (bp) corresponds to one hundredth of one percentage point.

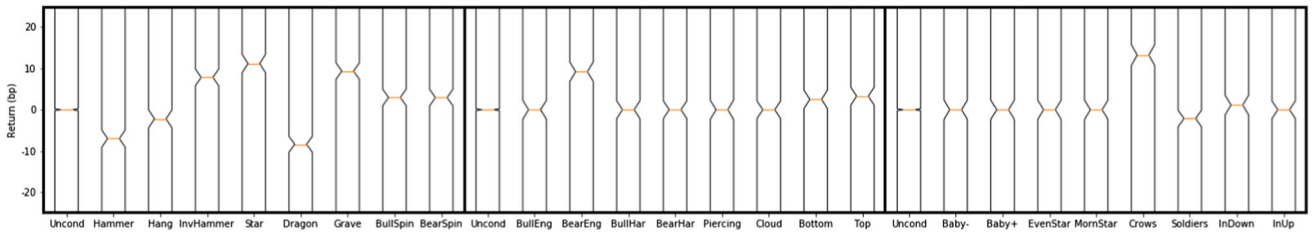

Fig. 6 Close-up of boxplot notches for the distributions of returns in basis points (100th of a percent), conditional on observing each of the technical patterns (similarity score S_P in its top centile). Absence of overlap between the boxplot notches of a conditional distribution and the unconditional distribution provides evidence at the 95% confidence threshold that the medians of the 2 distributions differ [17]. Surprisingly, several single-day patterns do in fact correlate with abnormal next-day returns. Almost all of the multi-day patterns exhibit notches that overlap with the unconditional distribution's, implying that the distribution medians are not meaningfully changed by conditioning. Only 'Bearish Engulfing' and 'Three Black Crows' seem to be significant—as harbingers of better times, despite their names

Table 1 Summary statistics for the next-day return distributions conditioned on matching technical patterns

Pattern	γ	μ (bp)	σ (bp)
Unconditional		4.26	229.40
Hammer	11.17	− 10.20	214.20
Inverted Hammer	8.37	+ 9.83	216.49
Hanging Man	9.84	− 11.67	220.59
Shooting Star	10.90	+ 9.99	222.52
Dragonfly Doji	10.65	− 11.37	217.57
Gravestone Doji	9.46	+ 10.53	220.37
Bullish Spinning Top	4.97	− 0.42	209.17
Bearish Spinning Top	5.22	− 1.47	213.24
Bullish Engulfing	2.78	+ 1.30	226.39
Bearish Engulfing	5.65	+ 0.83	228.06
Bullish Harami	2.65	+ 4.11	224.22
Bearish Harami	4.39	− 0.04	214.6
Piercing Line	2.28	+ 0.44	232.04
Cloud Cover	2.56	− 0.72	220.12
Tweezer Bottom	3.45	+ 3.26	230.10
Tweezer Top	2.78	+ 0.93	218.13
Abandoned Baby−	4.03	− 4.24	224.60
Abandoned Baby+	2.88	+4.87	227.16
Evening Star	2.53	− 2.32	223.24
Morning Star	2.79	+ 4.86	228.15
Three Black Crows	14.28	+ 5.62	265.14
Three White Soldiers	12.97	− 7.98	208.90
Three Inside Down	2.91	+ 0.45	231.62
Three Inside Up	3.27	+ 0.71	220.71

A match on pattern P is deemed to have occurred when the cross-correlational similarity score S_P is in its top *decile*. K–S statistics γ above 1.95 are significant at the 0.001 level. Mean return μ for each pattern is expressed as a difference from the unconditional baseline. The incremental mean returns are dwarfed by their standard deviation, and do not even always move in the direction prescribed by chartism

In the experimental results that follow, we split our S&P500 time series data into training and test sets corresponding to single stock prices from 1994–2004 and 2005–2015, respectively.[3] This specific train-test split of the data includes both expansionary and recessionary periods in each subset, to ensure the model is capable of learning the financial patterns that may emerge in different economic regimes.[4]

4.1 Multi-layer perceptron

To address issues of scale and stationarity, we process the original $4 \times n$ price signal matrix F into a new $80 \times n$ price signal matrix F^* where each column is a standardised encoding of 20 business days of price data. This encoding provides 4 weeks of price history, a context or 'image' within which neural network filters can scan for the occurrence of patterns and track their temporal evolution. We pass F^* through a multilayer perceptron (MLP) involving fully connected hidden layers. Preliminary five-fold cross-validation experiments with financial time series determined the network topology required for the model to

[3] Our classes are best be defined as 'negative return' and 'strictly positive return'. As zero return days occur (albeit infrequently) in assets with low denomination, we address the issue of class imbalance in two ways. Firstly, we add Gaussian noise with variance 10^{-6} to all returns, evenly spreading the zero return days across both classes. Secondly, we select the boundary value that separates our training set into two equally balanced classes (+ 0.0000005%) The resulting training set is perfectly balanced, against a very mild positive skew in the test set (51.1% strictly positive return days, 48.9% negative return days).

[4] The models of Sect. 4 were also tested on shorter, more recent intervals (training on data from 2010–2012, testing on data from 2013–2015). While still supportive of our findings, the results were less impressive—in part because deep learning models require substantial amounts of data to perform.

Table 2 Summary statistics for the next-day return distributions conditioned on matching technical patterns more stringently

Pattern	γ	μ (bp)	σ (bp)
Unconditional		4.26	229.40
Hammer	5.13	− 15.80	223.04
Inverted Hammer	5.00	+ 13.75	211.62
Hanging Man	3.71	− 14.92	222.06
Shooting Star	4.78	+ 12.01	232.42
Dragonfly Doji	14.73	− 25.81	219.99
Gravestone Doji	12.93	+ 22.41	223.57
Bullish Spinning Top	2.64	− 0.72	214.70
Bearish Spinning Top	1.67	+0.94	213.30
Bullish Engulfing	1.61	− 0.28	236.50
Bearish Engulfing	4.16	+ 5.75	238.47
Bullish Harami	1.07	+ 5.5	222.17
Bearish Harami	1.51	− 0.96	219.43
Piercing Line	2.29	+0.78	241.42
Cloud Cover	1.06	− 0.75	218.24
Tweezer Bottom	1.76	+4.19	223.23
Tweezer Top	1.22	+ 2.97	221.93
Abandoned Baby−	3.29	− 4.04	232.45
Abandoned Baby+	1.27	+ 2.94	232.28
Evening Star	2.89	− 0.27	231.76
Morning Star	1.80	+ 2.59	231.89
Three Black Crows	6.85	+ 13.09	229.40
Three White Soldiers	6.30	− 11.77	203.26
Three Inside Down	1.63	+ 2.72	233.12
Three Inside Up	2.50	+ 0.13	220.75

A match on pattern P is deemed to have occurred when the cross-correlational similarity score S_P is in its top *centile*

learn from its training data.[5] Insufficient height (neurons per hidden layer) and depth (number of hidden layers) led to models incapable of learning their training data. We settled on 2 fully connected layers of 64 neurons with ReLU activation functions, followed by a softmax output layer to classify positive and negative returns. Regularisation was achieved via the inclusion of dropout [21] in the fully connected layers of the network, limiting the model's propensity towards excessive co-adaptation across layers. A heavily regularised (dropout $= 0.5$) 2-layer MLP is already able to identify some structure in its data (out-of-sample accuracy of 50.6% after 100 epochs, Table 3). From our experiments with alternative dropout rates, we found that insufficient regularisation (dropout $= 0.3$) led to overfitted models with poor out-of-sample performance,

[5] For optimisation, we employed Adam, an extension of stochastic gradient descent that dynamically adapts per-parameter learning rates to deal with sparse and noisy data sets [20].

Table 3 Accuracy (%) obtained after training a 2-layer MLP on single stock data from the S&P500, using open-close-high-low price data

Epochs	In Sample	Out-of-Sample
1	50.3	50.3
5	50.7	50.4
10	51.4	50.5
50	52.2	50.6
100	52.5	50.6

Table 4 Accuracy (%) obtained after training a technically filtered MLP (filter length $m = 1$) on single stock data from the S&P500, using open-close-high-low price data

Epochs	In Sample	Out-of-Sample
1	50.0	50.2
5	50.2	49.8
10	50.2	49.7
50	50.4	49.9
100	50.5	49.8

Multi-day filters produced similarly lacklustre results. The technical analysis filters produce feature maps with less discernible structure than the original input

but that erring on the side of excessive regularisation (dropout $= 0.7$) was an acceptable choice, leading to similar generalisation to our base case at the expense of needing more epochs to converge.

4.2 Technically filtered MLP

Reframing technical patterns as pre-learned cross-correlational filters, we consider for each pattern length m the 8 pattern matrices T_{P_m} defined visually in Fig. 4. Each such formation, of form $4 \times m$, is stacked along the depth dimension, producing a $4 \times m \times 8$ tensor T whose inner product with standardised windows of the raw price signal F yields a new $8 \times n$ input matrix F_T,

$$F_T = \langle T, F \rangle. \tag{6}$$

This new input is the result of cross-correlating the raw price signal F with the technical analysis filter tensor T, and can be interpreted as the feature map generated by technical analysis. We now use F_T as the input to the same MLP as before and look for improvements in model forecasts. The results we find are consistent with Sect. 3: using technical analysis for feature extraction hinders the classifier, slightly degrading model performance (out-of-

Table 5 Details of the architecture for a CNN scanning patterns of length m

#	Layer	Units	Activation function	Dropout	Filter shape	Outgoing dimensions
1	Input	–	–	–	–	(input) [4 × 20]
2	Convolutional	8	ReLU	0.5	[4 × m]	[8 × 20]
3	FC	64	ReLU	0.5	–	[64]
4	FC	64	ReLU	0.5	–	[64]
5	FC	2	Softmax	–	–	(output, 2 classes) [2]

The number of filters in the convolution layer was deliberately kept low (8), and their dimensions (4 × m) match the technical patterns used in Sect. 4.2, to enable like-for-like comparability with the technical filter approach

Table 6 Accuracy (%) obtained In-Sample (IS) and Out-of-Sample (OoS) after training a deep neural network with a single convolutional layer learning 1-day, 2-day and 3-day patterns

Filter Length	1-day		2-day		3-day	
Epochs	IS	OoS	IS	OoS	IS	OoS
1	50.3	50.4	50.2	50.3	50.1	50.2
5	50.6	50.3	50.7	50.4	50.5	50.3
10	50.9	50.7	51.0	50.7	51.1	50.6
50	51.4	51.1	51.5	50.9	51.4	51.0
100	51.7	51.3	51.8	51.2	51.7	51.2

sample accuracy of 49.8% after 100 epochs using the 1-day patterns, Table 4).

4.3 Convolutional neural network

We now deepen the neural network by adding a single convolutional layer with 8 filters (so chosen to match the number of technical filters at each timescale, per Fig. 4) to our earlier MLP (architecture detailed in Table 5). Separate experiments are run for convolutional filters of size 4, 8 and 12, corresponding to scanning for 1-day, 2-day and 3-day patterns. Their performance is reported in Table 6. The CNN finds much greater structure in its training data than the MLP could, and generalises better. Accounting for the size of the test set (n = 1,332,395), the leap from the MLP's out-of-sample accuracy of 50.6% to the 1-day CNN's out-of-sample accuracy of 51.3% is considerable.

4.4 Model evaluation

To investigate whether the predictive performance of the neural network classifiers is not merely considerable but statistically significant, we derive the area under the curve (AUC) of each model's receiver operating characteristic curve (ROC), and exploit an equivalence between the AUC and Mann–Whitney–Wilcoxon test statistic U [22]:

$$\mathrm{AUC} = \frac{U}{n_P n_N} \tag{7}$$

where n_P and n_N are the number of positive and negative returns in the test set, respectively. In our binary classification setting, the Mann–Whitney–Wilcoxon test evaluates the null hypothesis that a randomly selected value from one sample (e.g. the subset of test data classified as positive next-day returns) is equally likely to be less than or greater than a randomly selected value from the complement sample (the remaining test data, classified as negative next-day returns). Informally, we are testing the null hypothesis that our models have classified at random. The test statistic U is approximately Gaussian for our sample size, so we compute each model's standardised Z-score and look for extreme values that would violate this null hypothesis.

$$Z = \frac{U - \mu_U}{\sigma_U} \tag{8}$$

where:

$$\mu_U = \frac{n_P n_N}{2} \tag{9}$$

and

$$\sigma_U = \sqrt{\frac{n_P n_N (n_P + n_N + 1)}{12}} \tag{10}$$

We benchmark our CNNs against traditional linear models popular in finance (AR(1) and AR(5) models), a buy-and-hold strategy and a range of machine learning alternatives detailed below.

4.4.1 Recurrent neural networks (RNN)

Deep learning for time series analysis has typically relied on recurrent architectures capable of learning temporal relations in the data. Long Short-Term Memory (LSTM) networks have achieved prominence for their ability to memorise patterns across vast spans of time by addressing the vanishing gradient problem. A thorough RNN architecture search [23] identified a small, but persistent gap in performance between LSTMs and the recently introduced

Table 7 Benchmark performance across a range of models trained on S&P500 technical data for January–December 1994 and tested on January 2005–December 2015

Model	Acc	Prec	Rec	F1	AUC	Z	Significance
MLP	50.6	49.7	49.6	49.6	51.1	23.766	<0.0001
Technical NN	49.8	49.1	49.3	49.2	49.9	− 1.878	–
1-day CNN	51.3	50.9	51.2	51.0	51.8	36.546	< 0.0001
2-day CNN	51.2	51.0	51.0	51.0	51.5	31.291	< 0.0001
3-day CNN	51.2	50.8	51.0	50.9	51.5	31.423	< 0.0001
RNN-LSTM	50.8	50.6	51.0	50.8	51.0	19.616	< 0.0001
RNN-GRU	50.9	50.3	50.8	50.6	51.2	24.880	< 0.0001
1-NN	50.0	50.0	50.0	50.0	50.1	1.087	0.1386
10-NN	49.9	49.9	49.9	49.9	49.8	− 3.317	–
100-NN	49.7	49.6	49.9	49.8	49.6	− 7.651	–
Linear SVM	49.9	49.9	49.8	49.8	49.8	− 0.962	–
RBF SVM	49.9	49.8	49.8	49.8	49.8	− 2.416	–
10-RF	50.0	49.9	49.8	49.9	50.0	0.256	0.3991
50-RF	49.8	49.9	49.8	49.8	49.7	− 5.986	–
100-RF	49.8	49.8	49.7	49.7	49.6	− 7.628	–
AR(1)	49.9	50.1	49.9	50.0	49.9	− 2.129	–
AR(5)	49.8	50.1	49.8	49.9	49.8	− 3.323	–
Buy-and-Hold	51.1	26.2	51.1	34.7	51.2	23.701	< 0.0001

Precision and recall are computed as weighted averages across both classes. Significance refers to the p value of the Mann–Whitney–Wilcoxon test for each model

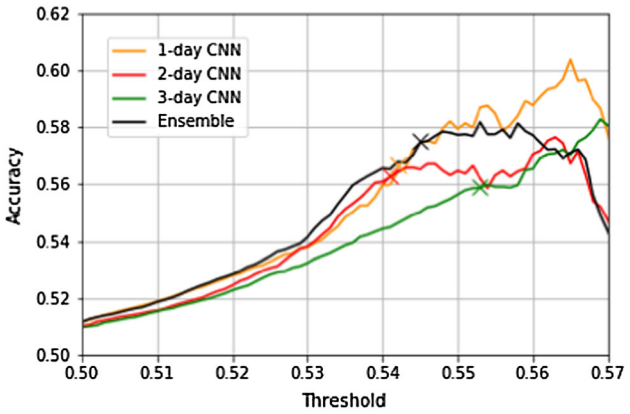

Fig. 7 Model accuracy as a function of softmax threshold α. For each model, we indicate by a cross the threshold level that retains the 1% of test data for which the model's output probabilities imply the highest confidence

Gated Recurrent Unit [24] on a range of synthetic and real-world data sets. Our benchmark RNNs involve a preliminary recurrent layer (LSTM and GRU, in separate experiments) of 8 neurons followed by 2 dense layers of 64 neurons with dropout, comparable in architectural complexity to the CNN models of Sect. 4.3.

4.4.2 k-Nearest Neighbours (k-NN)

We evaluate a range of nearest neighbour classifiers, labelling each day of the test set with the most frequently observed class label (positive or negative next-day return) in the k training points that were closest in Euclidean space.

4.4.3 Support vector machines (SVM)

SVMs have been applied to financial time series forecasting in prior literature, and achieved moderate success when the input features were not raw price data but hand-crafted arithmetic derivations like Moving Averages and MACD [25]. We report SVM performance under different kernel assumptions (linear and RBF), where the model hyperparameters (regularisation parameter C to penalise margin violations, RBF kernel coefficient γ to control sensitivity) were selected by cross-validation on a subset of the training data.

In their study of European financial markets, Ballings et al. [26] evaluated the classification accuracy of ensemble methods against single classifiers. Their empirical work highlighted the effectiveness of random forests in classifying stock price movements and motivates their inclusion in our list of benchmarks, under varying assumptions for the number of trees hyperparameter n.

4.4.4 Benchmark findings

Table 7 provides the AUC, Z-score and significance of each model, where significance measures the area of the distribution below Z. We disregard significance for

Table 8 Performance comparison between MLP, CNN and TCNN models trained on S&P500 technical data for January–December 1994 and tested on January 2005–December 2015. Precision and recall are computed as weighted averages across both classes

Model	Acc	Prec	Rec	F1	AUC	Z	Significance
MLP	50.6	49.7	49.6	49.6	51.1	23.766	<0.0001
Technical NN	49.8	49.1	49.3	49.2	49.9	-1.878	–
1-day CNN	51.3	50.9	51.2	51.0	51.8	36.546	<0.0001
2-day CNN	51.2	51.0	51.0	51.0	51.5	31.291	<0.0001
3-day CNN	51.2	50.8	51.0	50.9	51.5	31.423	<0.0001
CNN Ensemble	51.2	51.0	51.2	51.1	51.7	35.628	<0.0001
1-day TCNN	56.7	56.5	56.6	56.5	57.2	14.533	<0.0001
2-day TCNN	56.3	56.1	56.7	56.4	56.5	13.017	<0.0001
3-day TCNN	55.9	57.1	55.9	56.5	56.2	12.493	<0.0001
TCNN Ensemble	57.5	56.9	57.0	56.9	57.5	15.301	<0.0001

Significance refers to the p value of the Mann–Whitney–Wilcoxon test for each model

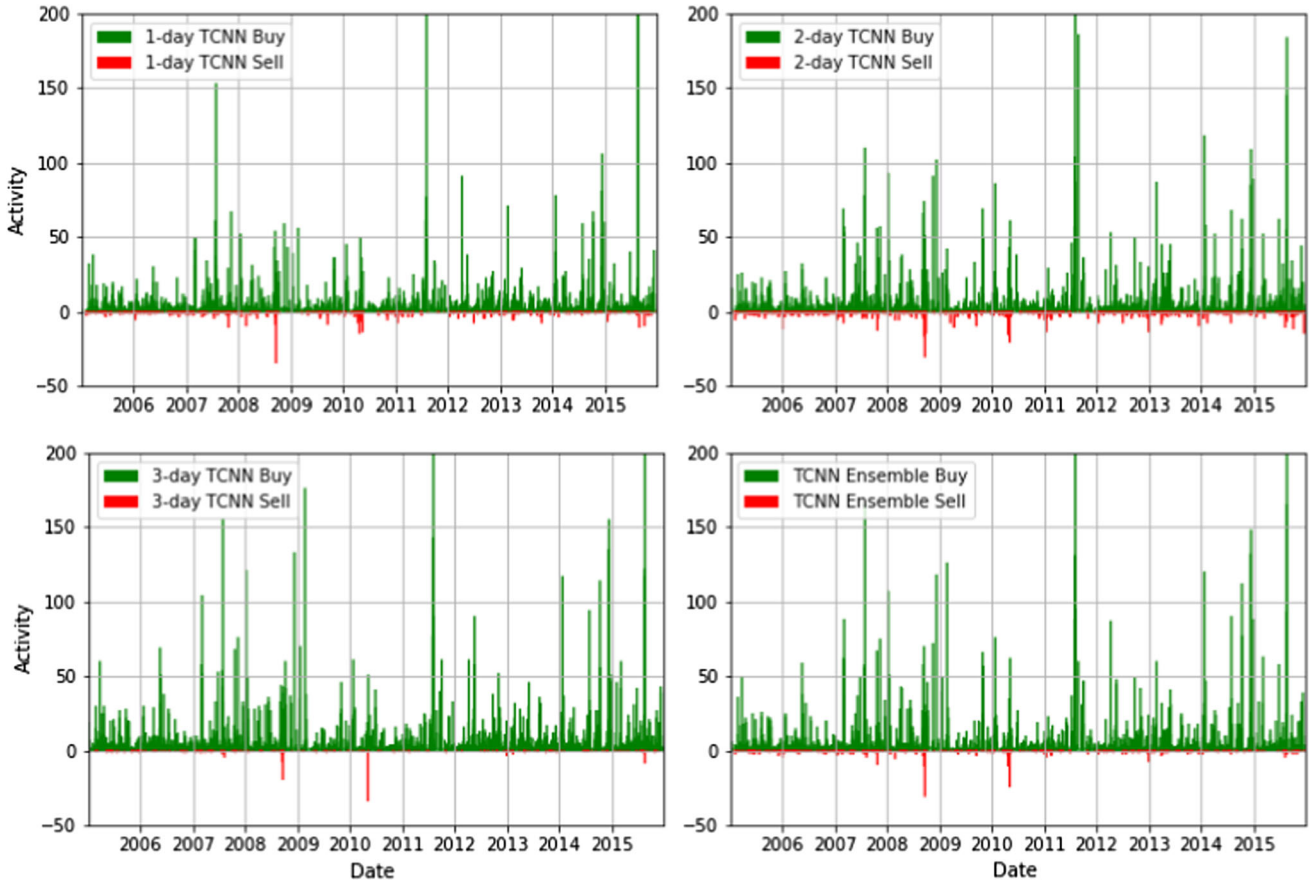

Fig. 8 Activity level of the various TCNN models through a 11-year period. As we retain the top centile from the 1,408,679 test points, each model is generating 14,087 trading decisions over 2868 business days, or on average 4.91 trades per day. Though the model is active throughout the window, discernible spikes in activity occur around major events, most notably the US debt ceiling crisis in August 2011

negative Z-scores (as is the case for the technically filtered neural network) as they imply classifiers that performed (significantly) worse than random chance. The results underscore the scale of the challenge for pattern recognition in finance: deep learning achieved the best results by a significant margin, and most alternative methods yielded accuracies that were not statistically distinguishable from guesswork.[6] Convolution also outperforms recurrence in

[6] The best alternative was also the simplest: buy-and-hold outperformed many systematic alternatives. This is in part a reflection of the test window (2005–2015) and the upwards bias of equity markets.

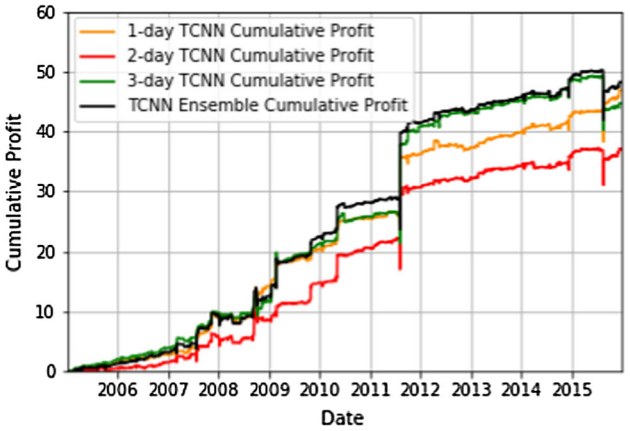

Fig. 9 Cumulative profit (as a multiple of starting wealth, per Table 7) generated by the various TCNN models between January 2005 and December 2015, in the absence of friction costs. The models are steadily profitable, with occasional spikes related to major events. Drawdowns are infrequent and of limited scale

our experiments, suggesting that a 20-day window may be sufficient to capture temporal dependencies in markets.

4.5 Methodological extensions to the ConvNet framework

Learning neural network filter specifications via convolution yields a significant boost to predictive prowess over the baseline model of Sect. 4.1 and technically filtered variant of Sect. 4.2. The CNNs' outperformance of autoregressive and machine learning techniques further confirms the aptitude of convolutional feature extraction on technical data, and spurs us to target domain-specific enhancements to our deep learning models.

4.5.1 Confidence thresholding

In contrast to mission-critical application domains like autonomous navigation, finance does not require an algorithmic agent to be accurate at all times. It is acceptable (and factoring in friction costs, preferable) for a model to be sparse in making decisions, only generating 'high conviction' calls, if this results in greater accuracy. Furthermore, and unlike several other common classifiers in machine learning like SVMs or Nearest Neighbours, the output values in the final layer of the CNN can be assigned a probabilistic interpretation, enabling a filtered, nuanced approach to classification. We replicate this by adding a confidence threshold α to the classification output of the final softmax layer of Table 5: test points where neither class is assigned a probability greater than α are deemed uncertain, and disregarded by the thresholded convolutional neural network (TCNN). For each model (1-, 2- and 3-day TCNN), the confidence threshold α is tuned through fivefold cross-validation. Accuracy as a function of confidence threshold α is presented in Fig. 7, and demonstrates in all 3 cases that a substantial increase in model prowess can be achieved by thresholding the softmax output to only consider class assignments with high certainty. We also highlight the α threshold which retains the top centile of test outputs, corresponding to the model's most confident assignments. These vary by model (54.2%, 54.1% and 55.3% for the 1-, 2- and 3-day TCNNs, respectively), but in each case form a reliable heuristic for balancing model confidence and sample size. A notable analogue to the study of technical analysis in Sect. 3: models searching for more elaborate multi-day patterns tend to underperform the single-day TCNN.

4.5.2 Ensembling TCNNs

An effective technique in image processing involves homogeneous ensembling of multiple copies of the same CNN architecture, averaging across the class assignments of the constituent models [15, 27]. Combining this probabilistic interpretation of the softmax layer with model averaging, we construct a heterogeneous ensemble out of our 1-day, 2-day and 3-day TCNNs. The ensemble benefits from learning patterns manifesting at different timescales, and achieves a higher accuracy (57.5%) on its top-confidence centile than any of the individual learners (56.7%, 56.3% and 55.9% for the 1-day, 2-day and 3-day TCNN, respectively, Fig. 7).

Performance metrics of both the TCNNs and TCNN ensemble are provided in Table 8. While the Z-scores of

Table 9 Compound Annual Growth Rate (CAGR, in %) and Sharpe ratio of the TCNN models under various assumptions for the cost of trading

Friction Cost Model	No friction			0.10% per transaction			0.25% per transaction		
	Profit	CAGR	Sharpe	Profit	CAGR	Sharpe	Profit	CAGR	Sharpe
1-day TCNN	46.9	42.15	**8.04**	32.8	37.72	**7.16**	11.7	25.98	**4.75**
2-day TCNN	36.9	39.16	7.81	22.8	33.41	6.61	1.7	9.52	1.65
3-day TCNN	44.6	41.50	5.95	30.5	36.84	5.59	9.4	23.71	3.49
TCNN Ensemble	**48.2**	**42.50**	6.57	**34.1**	**38.20**	5.86	**13.0**	**27.13**	4.08

The TCNN ensemble and 1-day TCNN are optimal choices (denoted in bold) for return and risk-adjusted return maximisation, respectively

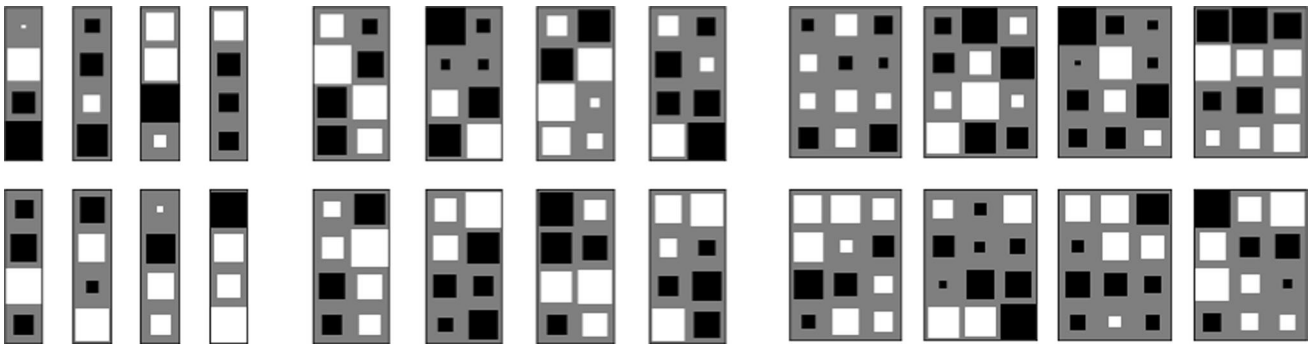

Fig. 10 Weight-space visualisation as Hinton diagrams for the 24 cross-correlational filters learned from the first layer of each CNN (8 per constituent model)

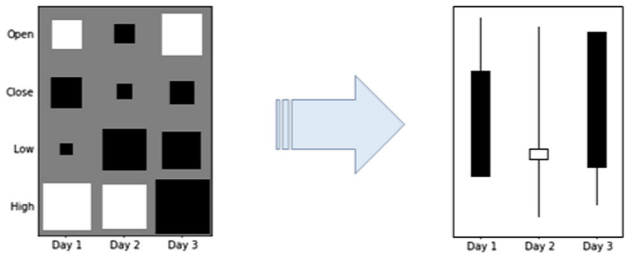

Fig. 11 Hinton diagram of the sixth cross-correlational filter learned in the first layer of the 3-day CNN. The relative values of the standardised open, close, low and high for each column in the filter define, in a chartist sense, a specific candlestick sequence (or patch thereof, in instances where the filter's open or close is incompatible with the high-low range) which the neural network extracted as informative for time series forecasting

the TCNN models are lower than those of unthresholded CNN models, this is primarily the consequence of sample size on statistical significance tests—AUC improves markedly under thresholding.

4.6 Practical implementation

Through thresholding, we enforce sparsity in the model's decision-making. In a real-world deployment, infrequent activity keeps friction costs low—a desirable outcome for trading algorithms. We track the activity level of the various models over time, as well as the cumulative profit they would generate over the 11-year test window. We assume the model fully captures the 1-day return associated with the top centile of its thresholded class assignments, additively for positive class predictions and subtractively for negative class predictions.

The models are heavily skewed towards buying activity, with accurately timed spikes centred around major world events (Fig. 8). The 2 largest single-day buy orders occurred on 9 August 2011 (328 buys), at the tail end of the US debt ceiling crisis which caused the S&P500 to drop 20% in 2 weeks, and on 24 August 2015 (241 buys), following a flash crash in which US markets erased 12% of their value before recovering. The largest sell volume occurred on 22 September 2008 (31 sells), a full week after the collapse of Lehman Brothers. This coincides with market-wide relief over Nomura's decision to buy Lehman's operations—and presented the last opportunity to sell before the nosedive of the Great Financial Crisis in late 2008. Despite having no information about world news in their technical data set, the models were capable of both inferring crucial moments in history, and timing trading decisions around them.

Figure 9 presents the model's profitability over time to highlight the relative steadiness of convolution's performance in identifying stock market patterns, when the decisions are generated by TCNNs and their ensemble. Table 9 translates this performance into compounded annual returns and Sharpe ratios under various assumptions for friction. Even in the absence of tight execution (average trading cost of 0.25% from the mid-market price), the models remain highly profitable. This sensitivity analysis does nevertheless highlight the importance of good execution in any real-world deployment of algorithmic trading: the TCNN ensemble can only just break even if the per-transaction cost rises to 0.35%.

4.7 Interpretable feature extraction

The convolutional filters learned by the network provide a basis for feature extraction. In particular, the convolutional layer's filters define patches whose cross-correlation with the original input data was informative in minimising both in-sample and out-of-sample categorical cross-entropy. We produce a mosaic of these filters as Hinton diagrams[7] (Fig. 10) and visualise them in the language of technical analysis as candlestick patterns (Figs. 11 and 12), cross-correlational templates whose occurrence is informative for

[7] Hinton diagrams provide a means of visualising numerical values in a matrix. The area occupied by a square is proportional to the value's magnitude, and the sign of the matrix entry is colour-coded (white for positive values, black for negative values).

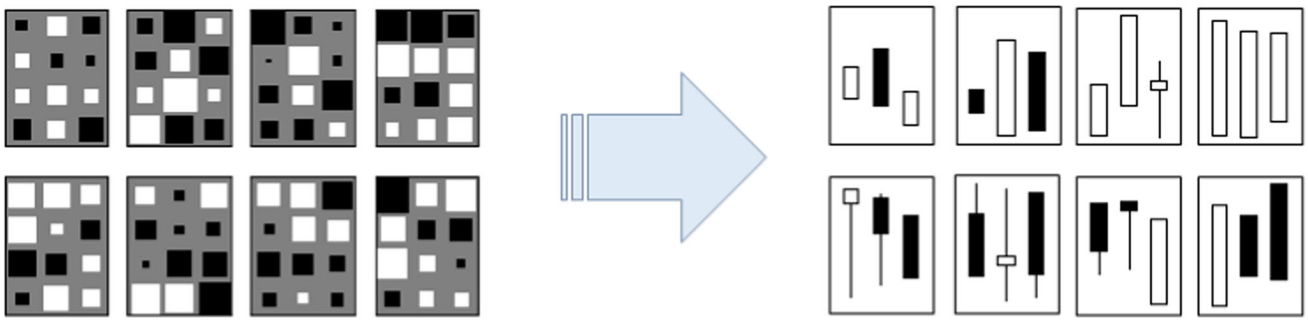

Fig. 12 Candlestick pattern translation of the cross-correlational filter mosaic for the 3-day CNN

financial time series forecasting. Unlike technical patterns, however, these templates have no set meaning: the purpose of individual neurons in a convolutional layer is not readily interpretable.

5 Conclusion

Our results present, to our knowledge, the first rigorous statistical evaluation of candlestick patterns in time series analysis, using normalised signal cross-correlation to identify pattern matches. We find little evidence of predictive prowess in any of the standard chartist pictograms, and suspect that the enduring quality of such practices owes much to their subjective and hitherto unverified nature. Nevertheless, it is not inconceivable that price history might contain predictive information, and much of quantitative finance practice relies on elements of technical pattern recognition (e.g. momentum-tracking) for its success. Through a deep learning lens, technical analysis is merely an arbitrary and incorrect specification of the feature-extractive early layers of a neural network. Within relatively shallow architectures, learning more effective filters from data improves accuracy significantly while also providing an interpretable replacement for chartism's visual aids. The simplicity of our architecture showcases the potential for deep learning to supplant technical analysis: we do not expect shallow convolution to be optimal. In the context of computer vision, accuracy improved significantly through the expansion of model depth.[8] We hypothesise that a thorough neural architecture search will yield similar incremental gains in our space. Hybrid neural architectures, including recurrent layers capable of learning long-term dependencies between the patterns identified by convolution, may further enhance the potential for deep learning in finance. Thresholding and deep ensembling of such models would form a robust framework for systematic decision-making in financial markets.

[8] AlexNet's 5 convolutional layers and 3 dense layers yielded a 16.4% error rate on ILSVRC-2012. VGG19's 16 convolutional layers and 3 dense layers yielded a 7.3% error rate on ILSVRC-2014. ResNet's 152 convolutional layers and 1 dense layer yielded a 3.6% error rate on ILSVRC-2015.

Funding This study was funded by the Engineering and Physical Sciences Research Council (Grant Number 1642370) and the Economic and Social Research Council (Grant Number 1535288).

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Blume L, Easley D, O'Hara M (1994) Market statistics and technical analysis: the role of volume. J Financ 49(1):153–181
2. Neely C, Weller P, Dittmar R (1997) Is technical analysis in the foreign exchange market profitable? A genetic programming approach. J Financ Quant Anal 32(4):405–426
3. Allen F, Karjalainen R (1999) Using genetic algorithms to find technical trading rules. J Financ Econ 51:245–271
4. Krauss C, Do XA, Huck N (2017) Deep neural networks, gradient-boosted trees, random forests: statistical Arbitrage on the S&P500. Eur J Oper Res 259(2):689–702
5. Dixon M, Klabjan D, Bang JH (2016) Classification-based financial markets prediction using deep neural networks. Algorithmic Finance 6(3–4):67–77
6. Lo AW, Mamaysky H, Wang J (2000) Foundations of technical analysis: computational algorithms, statistical inference, and empirical implementation. J Financ 55(5):1706–1765

7. Taylor MP, Allen H (1992) The use of technical analysis in the foreign exchange market. J Int Money Financ 11(3):304–314
8. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
9. Tsantekidis A, Passalis N, Tefas A, Kanniainen J, Gabbouj M, Iosifidis A (2017) Forecasting stock prices from the limit order book using convolutional neural networks. In: 19th IEEE conference on business informatics (CBI) vol 1, pp 7–12
10. Fischer T, Krauss C (2017) Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res 270(2):654–669
11. Dixon M (2018) Sequence classification of the limit order book using recurrent neural networks. J Comput Sci 24:277–286
12. Zhang Z, Zohren S, Roberts S (2018) DeepLOB: deep convolutional neural networks for limit order books. IEEE Trans Signal Process 67(11):3001–3012
13. Sirignano J, Cont R (2018) Universal features of price formation in financial markets: perspectives from deep learning. Available at SSRN: https://ssrn.com/abstract=3141294 or https://doi.org/10.2139/ssrn.3141294
14. Romaszko L (2015) Signal correlation prediction using convolutional neural networks. J Mach Learn Res Workshop Conf Proc 46:45–56
15. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1106–1114
16. Massey FJ (1951) The Kolmogorov–Smirnov test for goodness of fit. J Am Stat Assoc 46(253):66–78
17. Chambers JM, Cleveland WS, Kleiner B, Tukey PA (1983) Comparing data distributions. In: Graphical methods for data analysis, pp 60–63
18. McLean RD, Pontiff J (2016) Does academic research destroy stock return predictability? J Financ 71(1):5–32
19. Hinton GE, Deng L, Yu D, Dahl GE, Mohamed A, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN, Kingsbury B (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–97
20. Kingma D, Ba J (2015) Adam: a method for stochastic optimization. ICLR 2015
21. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958
22. Mason SJ, Graham NE (2002) Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: statistical significance and interpretation. Q J R Meteorol Soc 128:2145–2166
23. Jozefowicz R, Zaremba W, Sutskever I (2015) An empirical exploration of recurrent network architectures. J Mach Learn Res 37:2342–2350
24. Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 deep learning and representation learning workshop
25. Kim K (2003) Financial time series forecasting using support vector machines. Neurocomputing 55(1):307–319
26. Ballings M, Van den Poel D, Hespeels N, Gryp R (2015) Evaluating multiple classifiers for stock price direction prediction. Exp Syst Appl 42(20):7046–7056
27. Antipova G, Berrani SA, Dugelay JL (2016) Minimalistic CNN-based ensemble model for gender prediction from face images. Pattern Recognit Lett 70:59–65