Physica A 510 (2018) 587–609

Contents lists available at ScienceDirect

Physica A

journal homepage: www.elsevier.com/locate/physa

Bitcoin technical trading with artificial neural network

Masafumi Nakano, Akihiko Takahashi[*], Soichiro Takahashi

Graduate School of Economics, University of Tokyo, 7-3-1 Hongo Bunkyo-ku, Tokyo, 113-0033, Japan

HIGHLIGHTS

- Research of high frequency trading in a new financial market, Bitcoin.
- ANN models with technical indicator inputs are applied for return prediction.
- Numerical experiments show our proposal outperforms a buy-and-hold strategy.
- Reasonable trading costs are considered in our trading simulation.
- Performance comparison among different ANN model specifications.

ARTICLE INFO

Article history:
Received 28 March 2018
Received in revised form 19 May 2018
Available online xxxx

Keywords:
Bitcoin
Artificial neural network
Deep learning
High frequency data
Technical trading

ABSTRACT

This paper explores Bitcoin intraday technical trading based on artificial neural networks for the return prediction. In particular, our deep learning method successfully discovers trading signals through a seven layered neural network structure for given input data of technical indicators, which are calculated by the past time-series data over every 15 min. Under feasible settings of execution costs, the numerical experiments demonstrate that our approach significantly improves the performance of a buy-and-hold strategy. Especially, our model performs well for a challenging period from December 2017 to January 2018, during which Bitcoin suffers from substantial minus returns. Furthermore, various sensitivity analysis is implemented for the change of the number of layers, activation functions, input data and output classification to confirm the robustness of our approach.

1. Introduction

Cryptocurrencies, alternatives to traditional centralized currencies, are now rapidly emerging as new financial instruments. Especially, the market capital of Bitcoin, which is the first decentralized cryptocurrency introduced by [1], has grown up to be about 170 billion US dollars. Along with the market growth, researchers have also begun their works on cryptocurrency investment, though there are still a few papers (e.g., [2–6]).

With regard to the profitability of Bitcoin trading as an alternative to the Buy-and-Hold (B&H) strategy, which is a main topic of this paper, the Efficient Market Hypothesis (e.g., [7,8]) provides an important theoretical framework in financial economics [9]. That is, it tells us that if the market is informationally efficient, there should not exist any profitable strategy to outperform the market (B&H strategy) after taking into account execution costs. In other words, informationally inefficiency can lead us to beat the market, as shown in ANN-based empirical researches on Brazilian stock markets by [10] and [11], for instance.

As pointed out in [3], in terms of a financial asset, Bitcoin equips with features as a commodity in addition to a currency. That is, since Bitcoin "mining" is to solve very complex encryptions by brute force solutions, a large amount of computer

[*] Corresponding author.
E-mail addresses: msfmnakano325@gmail.com (M. Nakano), akihikot@e.u-tokyo.ac.jp (A. Takahashi), s7takahashi1255@gmail.com (S. Takahashi).

https://doi.org/10.1016/j.physa.2018.07.017
0378-4371/© 2018 Elsevier B.V. All rights reserved.

resources and electronics are required, which results in the limited supply of new Bitcoin as well as commodities such as gold. Then, combined with its finite total supply of 21 million units, Bitcoin obtains the scarcity, that is a feature of commodities.

In addition, related with the unique feature, the price dynamics of Bitcoin has shown some peculiarity. For instance, [12, 13] and [14] investigate the statistical properties of Bitcoin market including Hurst exponent of the time-series of returns to find some inefficiency. As well, through detection of power-law behavior and studies of scaling exponents, [15] has illustrated that Bitcoin returns, in addition to being more volatile, also exhibit heavier tails than stocks. Also, from the view of an alternative asset, Bitcoin shows extremely high return, high volatility and low correlation to traditional assets, as reported in [2]. That is, except for high volatility as well as exogenous restrictions including legal regulations, these characteristics seem quite attractive to investors, which implies that it is an important task to create trading schemes for the risk reduction.

In that sense, machine learning (ML) are promising technologies, which have been actively used and advanced for asset price/return prediction in recent years. Since the financial time-series are non-stationary with large noise, it seems necessary for their forecasting to incorporate non-linearity into prediction models. However, this theme is not seriously taken into consideration in the most empirical literatures of financial economics, as pointed out in [16]. In contrast to traditional linear statistical models such as ARMA [17], the artificial intelligence (AI) approach enables us to capture the non-linear property. Among the various AI models, artificial neural networks (ANNs) with deep learning (DL) algorithm are known to be one of the most thriving approaches due to its high predictive abilities.

Initiated by a pioneering work of [18], deep learning technique has been developed in a tremendous speed, which realizes fast and precise learning of multi-layered ANNs. In general, ANNs consist of input, hidden and output layers, where each layer has multiple information processing units called neurons. As reported in [19], this complex structure makes it possible to approximate non-linear functions with high accuracy.[1] Naturally, there are a number of previous literatures on the applications of ANNs to financial investment problems (e.g., [22–25]).

Thus, this paper explores Bitcoin investment based on ANNs, which extracts trading signals from the past time-series data of Bitcoin over every 15 min. In particular, we firstly construct seven-layered ANNs for multi-class classification, where the input information is technical indicators and historical returns available at the current time. Then, deep learning methods are applied to the ANN models for supervised training data. Finally, we decide to take long/neutral/short positions based on the prediction of our trained ANNs.

As a result, the numerical out-of-sample experiment demonstrates that under reasonable execution cost assumptions, our approach successfully improves the Bitcoin trading performance, especially for a period from December 2017 to January 2018, during which Bitcoin suffers from significant drawdown.

The contribution of this paper is to incorporate ML techniques into Bitcoin trading. In particular, to the best of our knowledge, this is the first work to properly apply ANN-based return prediction to Bitcoin high frequency technical trading under reasonable execution costs to obtain well performance. That is, although a lot of papers employ ML for solving financial problems (e.g., [26–30]), there are still few works that focus on ML-based cryptocurrency trading. Besides, as one of previous researches related to cryptocurrency, [31] tackles with volatility prediction of cryptocurrencies based on Support Vector Regression GARCH (SVR-GARCH) model, which is reported to be superior to traditional GARCH models.

However, differently from [31], the current work attempts return prediction based on the ANN models, which is more essential for the applications to financial investment. In addition, the detailed comparison among different ANN models is also a feature of our work. That is, since ANN performance may largely depends on model specification and datasets, we closely analyze what are key factors to the performance improvement by preparing multiple ANN models and datasets.

The remainder of the paper is organized as follows. Section 2 explains the concrete structure of our ANN models for the Bitcoin return classification with trading strategies. Section 3 summarizes the numerical results for ANN models. Finally, Section 4 concludes. Appendix provides some additional information.

2. Methodology

This section concretely explains our methodology: Section 2.1 introduces our time-series data for Bitcoin, and then Section 2.2 describes our ANN models with training data. Finally, Section 2.3 shows our trading strategies based on the ANN.

Our ANN models are equipped with multi-layered neural network architecture to effectively acquire price movement pattern based on deep learning technique. Then, by deep learning applications for a training dataset, our ANN predicts the price direction (up and down) in the next period from the input data (e.g. technical indicators). Thus, we implement simple investment strategies to buy (sell) when the price is likely to go up (down).

[1] Interestingly, recent researches (e.g. [20,21]) have illustrated that this functional approximation ability of ANNs is also quite effective to numerically solve high-dimensional partial differential equations (PDEs) and backward stochastic differential equations (BSDEs) with substantially lower computational times than the existing approaches.

2.1. Data

We obtain the high frequency price and volume data of Bitcoin from a cryptocurrency exchange, Poloniex, through its official Application Programming Interface (API),[2] where a time interval of the data is 15 min over the period from 2016/07/31 15:00 (GMT) to 2018/01/24 07:30 (GMT). With regard to price data, we use closing, high and low prices, which are denoted (P_t), (P_t^h), (P_t^ℓ), respectively. Here, the closing, high and low prices stand for the last, maximum and minimum (traded) prices over each 15 min, respectively. Although these price data are actually traded ones, it is reasonable to assume some costs for executing trading in practice, which will be discussed later. Let us remark that in the following simulation, we suppose our trading is contracted by closing prices.

Our numerical analysis also utilize the return data $(R_t)_t$ calculated from the closing prices $(P_t)_t$ as $R_t := P_t/P_{t-1} - 1$ $(t = 1, \dots, 52002)$, where t denotes a time index. Note that $t = 0$ and $t = 52002$ correspond with 2016/07/31 15:00 (GMT) and 2018/01/24 07:30 (GMT), respectively. Now let us define a set of times $\mathbb{T} = \{t;\ t = 1, \dots, 52002\}$, and divide it into two sets, $\mathbb{T}_1$ and $\mathbb{T}_2$. Here, we prepare $\mathbb{T}_1 = \{t;\ t = 1, \dots, 38100\}$ for the ANN training and $\mathbb{T}_2 = \{t;\ t = 38101, \dots, 52002\}$ for trading simulation. Let us remark that the trading simulation is implemented over a test period from 2017/09/01 12:30 (GMT) to 2018/01/24 07:30 (GMT).

2.2. Artificial neural network

This section introduces our base case ANN model for Bitcoin return classification, which consists of specific input, hidden and output layers with activation functions. Although the description is limited to the base case in this section, various type of alternatives will be tested in the next Section 3 to analyze the effectiveness of our methodology.

2.2.1. Network architecture

In our base case, the ANN model consists of 7 layers, i.e., input, hidden and output layers, where the number of nodes (units) for the ℓ-th layer $(\ell = 1, \dots, 7)$, denoted by $N^{[\ell]}$, is specified as follows:

$$(N^{[\ell]})_{\ell=1,\dots,7} = (12, 40, 30, 20, 10, 5, 4). \tag{1}$$

In the following, the number of layers denotes $L\,(=7)$ to preserve the generality of notation. Then, our ANN is described as the following nonlinear mapping $\boldsymbol{f} : \boldsymbol{i}_t \in \mathbb{R}^{N^{[1]}} \mapsto \boldsymbol{o}_t \in (0,1)^{N^{[L]}}$ $(L = 7)$.

$$\begin{aligned} \boldsymbol{x}^{[1]} &= \boldsymbol{i}_t, \\ \boldsymbol{x}^{[\ell]} &= \boldsymbol{h}^{[\ell]}(\boldsymbol{y}^{[\ell]}), \quad \boldsymbol{y}^{[\ell]} = \boldsymbol{W}^{[\ell]}\boldsymbol{x}^{[\ell-1]} + \boldsymbol{b}^{[\ell]}, \quad \ell = 2, \dots, L, \\ \boldsymbol{o}_t &= \boldsymbol{x}^{[L]}, \end{aligned} \tag{2}$$

where

- $\boldsymbol{y}^{[\ell]} \in \mathbb{R}^{N^{[\ell]}}$, $\boldsymbol{W}^{[\ell]} \in \mathbb{R}^{N^{[\ell]}\times N^{[\ell-1]}}$, $\boldsymbol{b}^{[\ell]} \in \mathbb{R}^{N^{[\ell]}}$, $\boldsymbol{x}^{[\ell-1]} \in \mathbb{R}^{N^{[\ell-1]}}$, $\ell = 2, \dots, L$,
- $\boldsymbol{h}^{[\ell]} : \mathbb{R}^{N^{[\ell]}} \to \mathbb{R}^{N^{[\ell]}}$, $\ell = 2, \dots, L-1$,
- $\boldsymbol{h}^{[L]} : \mathbb{R}^{N^{[L]}} \to (0,1)^{N^{[L]}}$.

Besides, the activation functions of the hidden layers (the ℓ-th layers, $\ell = 2, \dots, L-1$) and the output layer (the L-th layer), i.e. $\boldsymbol{h}^{[\ell]}$ $(\ell = 2, \dots, L-1)$ and $\boldsymbol{h}^{[L]}$, are the following rectified linear (ReLU), introduced by [32], and softmax functions, respectively.

$$\boldsymbol{h}^{[\ell]}(\boldsymbol{y}^{[\ell]}) = \begin{cases} \left(\max\{y_i^{[\ell]}, 0\}\right)_{i=1,\dots,N^{[\ell]}} & , \ \ell = 2, \dots, L-1, \\ \left(\dfrac{\exp(y_i^{[\ell]})}{\sum_{j=1}^{N^{[\ell]}} \exp(y_j^{[\ell]})}\right)_{i=1,\dots,N^{[\ell]}} & , \ \ell = L, \end{cases} \tag{3}$$

where $\boldsymbol{y}^{[\ell]} = (y_i^{[\ell]})_{i=1,\dots,N^{[\ell]}}$.

Here, we briefly explain the reasons to use ReLU for activation functions in hidden layers. That is, the main reason is that ReLU enables to escape the well-known gradient vanishing problem without the so-called pretraining step, as shown in [32]. Let us remark that ANN learning with sigmoid or other activation functions often needs some proper initialization called pretraining [33]. In addition, ReLU requires less computational cost in forward and backward propagation than other activation functions, due to its functional simplicity. Although the use of ReLU causes the "dying ReLU" problem as reported in [34], we discuss its effect and solution later in Section 3.3.

With regard to the output layer, this softmax specification indicates that our ANN $\boldsymbol{f}; \boldsymbol{i}_t \in \mathbb{R}^{N^{[1]}} \mapsto \boldsymbol{o}_t \in (0,1)^{N^{[L]}}$ is designed for the multi-class ($N^{[L]}$-class) classification. Since the sum of the output $\boldsymbol{o}_t$ equals to one and its element is positive, the output $\boldsymbol{o}_t$ can be interpreted as probabilities that the next return will belong to each class. Especially, it classifies the next 15 minutes' return into $N^{[L]}$ classes, as concretely shown in the next Section 2.2.2.

[2] https://poloniex.com/support/api/

Moreover, the error function is assumed to be the following cross entropy one for a given (supervised) training dataset $\boldsymbol{d} = (\boldsymbol{i}_t, \boldsymbol{s}_t)_{t\in\tilde{\mathbb{T}}_1}$, where $(\boldsymbol{s}_t)$ denote supervisory data and $\tilde{\mathbb{T}}_1 \subset \mathbb{T}_1$ will be specified in the next Section 2.2.2.

$$E(\boldsymbol{\Theta}; \boldsymbol{d}) = -\sum_{t\in\tilde{\mathbb{T}}_1} \sum_{i=1}^{N^{[L]}} s_{i,t} \log o_{i,t}, \tag{4}$$

where

- $\boldsymbol{\Theta} = (\boldsymbol{W}^{[\ell]}, \boldsymbol{b}^{[\ell]})_{\ell=2,\ldots,L}$,
- $(o_{i,t})_{i=1,\ldots,N^{[L]}} = \boldsymbol{o}_t = \boldsymbol{f}(\boldsymbol{i}_t; \boldsymbol{\Theta})$,
- $(s_{i,t})_{i=1,\ldots,N^{[L]}} = \boldsymbol{s}_t$.

Let us remark that our ANN $\boldsymbol{f}$ is a nonlinear function with the parameters $\boldsymbol{\Theta} = (\boldsymbol{W}^{[\ell]}, \boldsymbol{b}^{[\ell]})_{\ell=2,\ldots,L}$. With regard to the (deep) learning of these parameters, we employ a stochastic gradient descent (SGD) method, the so-called "Adam" [35], where minibatches with size 500 are introduced. Also, we set the number of epochs (i.e. the number of training times) to be 400.

2.2.2. *Preparation of training dataset*

Now let us prepare (supervised) training data $\boldsymbol{d}$, i.e. pairs of input and supervisory data $\boldsymbol{d} = (\boldsymbol{i}_t, \boldsymbol{s}_t)_t$. In the base case ANN model, we exploit "normalized" return data $(r_t)_{t\in\mathbb{T}_1}$ defined as follows:

$$r_t = \frac{R_t - \mathrm{Avg}((R_t))}{\mathrm{Std}((R_t)_t)}, \tag{5}$$

where Avg() and Std() are functions that calculate sample average and standard deviation for a given dataset. Let us remember that $(R_t)_t$ are return data calculated from closing prices $(P_t)_t$.

Now, we first construct the following input data $(\boldsymbol{i}_t)_t$, that is,

$$\boldsymbol{i}_t = ((EMA_{M,t})_M, (EMSD_{M,t})_M, (RSI_{K,t})_K, r_{t-1}) \in \mathbb{R}^{12}, \tag{6}$$

where

- Exponential moving average & standard deviation (EMA, EMSD):
$$\begin{aligned} EMA_{M,t} &= \alpha_M r_{t-1} + (1-\alpha_M) EMA_{M,t-1}, \\ EMSD^2_{M,t} &= \alpha_M (r_{t-1} - EMA_{M,t-1})^2 + (1-\alpha_M) EMSD^2_{M,t-1}. \end{aligned} \tag{7}$$
Here, the parameter α_M is specified as $\alpha_M = 2/(1+M)$ for $M = 2, 4, 12, 24$, which is usually referred as M-period EMA/EMSD, i.e. 30 min, 1 h, 3 h, 6 h EMA/EMSD. In practice, this indicator is often used in trend-following trading.
- Relative strength index (RSI):
$$\begin{aligned} &RSI_{K,t} = PR_{K,t}/(PR_{K,t} + NR_{K,t}), \\ &PR_{K,t} = \sum_{k=1}^{K} \max\{r_{t-k}, 0\}, \quad NR_{K,t} = \sum_{k=1}^{K} \max\{-r_{t-k}, 0\}, \end{aligned} \tag{8}$$
where the parameter K is tested for $K = 12, 24, 48$. In practice, this indicator, introduced by [36], is often employed in range trading. Specifically, if $RSI > 0.7$ (< 0.3), the asset is interpreted to be overbought (oversold).

Let us notice that $RSI_{K,t}$ is defined on $\{t \in \mathbb{T}_1;\ t > K\}$. Further, it is well-known that the values $EMA_{M,t}$ and $EMSD_{M,t}$ over the first several periods depend on the determination of their initial values. Thereby, for the ANN training, we use the data $(\boldsymbol{i}_t, \boldsymbol{s}_t)_{t\in\tilde{\mathbb{T}}_1}$, $\tilde{\mathbb{T}}_1 = \mathbb{T}_1 \setminus \{1, \ldots, 100\}$.

Next, we construct supervisory datasets $\boldsymbol{s}_t \in \{0, 1\}^4$ $(N^{[L]} = 4,\ L = 7)$ by using 100β percentiles q_β of the normalized return data $(r_t)_{t\in\tilde{\mathbb{T}}_1}$.

$$\boldsymbol{s}_t = \begin{cases} (1, 0, 0, 0) & , q_{3/4} \le r_t, \\ (0, 1, 0, 0) & , q_{1/2} \le r_t < q_{3/4}, \\ (0, 0, 1, 0) & , q_{1/4} \le r_t < q_{1/2}, \\ (0, 0, 0, 1) & , r_t \le q_{1/4}. \end{cases} \tag{9}$$

Thus, we have constructed the supervised training datasets $\boldsymbol{d} = (\boldsymbol{i}_t, \boldsymbol{s}_t)_{t\in\tilde{\mathbb{T}}_1}$ for the base case ANN model. For this training datasets $\boldsymbol{d}$, we implement deep learning (SGD) to the ANN $\boldsymbol{f}(\boldsymbol{i}_t; \boldsymbol{\Theta})$ $(t \in \tilde{\mathbb{T}}_1)$, which gives us the parameters $\hat{\boldsymbol{\Theta}}$ by minimizing the error function $E(\boldsymbol{\Theta}; \boldsymbol{d})$. Then, for the test data $(\boldsymbol{i}_t)_{t\in\mathbb{T}_2}$, the ANN $\boldsymbol{f}(\boldsymbol{i}_t; \hat{\boldsymbol{\Theta}})$ gives the outputs $(\boldsymbol{o}_t)_{t\in\mathbb{T}_2}$, where $\boldsymbol{o}_t = (o_{i,t})_{i=1,\ldots,N^{[L]}}$, $(N^{[L]} = 4)$. As explained in Section 2.2, $o_{i,t} \in (0, 1)$ can be interpreted as a probability that the next return will belong to each class.

Importantly, let us remark that ANN learning results generally depend on seeds of random numbers for parameter initialization, which has effects on the resulting model performance. Then, we employ a model averaging technique, where the average of 100 patterns of ANN outputs calculated from different random seeds is used as a final output for trading simulation.[3]

2.3. Trading strategy

As mentioned in Section 2.1, our investment simulation is implemented over the time period $\mathbb{T}_2$. Nonetheless, for notational simplicity, let us use a generic time period $\{0, 1, \ldots, T\}$ in this section to define an asset value process and trading strategies.

Firstly, for performance evaluation, we introduce an asset value process $(V_t)_{t=0,\ldots,T}$, which is also called as cumulative returns. For simplicity, the initial asset value is normalized to be one, i.e. $V_0 = 1$. Then, a trading strategy at time t, denoted by ω_t, indicates a proportion of money invested in BTC to the total investment principal at time t. Thus, by collecting ω_t for all times $t = 0, \ldots, T-1$, a trading strategy $\boldsymbol{\omega} = (\omega_t)_{t=0,\ldots,T-1}$ is defined.

In the subsequent investment simulations, we adopt the following three trading strategies, i.e. $\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}, \boldsymbol{\omega}^{(3)}$, for each classification result $\boldsymbol{o}_t$:

- Strategy (1), denoted by $\boldsymbol{\omega}^{(1)}$, takes a long position when ANN "strongly" predicts positive return. That is, we take a long position only when the next return is predicted to be in the 1st class:

$$\omega_t^{(1)} = \begin{cases} 1 & , \quad \text{if } \max\left\{(o_{i,t})_{i=1,\ldots,N^{[L]}}\right\} = o_{1,t}, \\ 0 & , \quad \text{otherwise}. \end{cases} \tag{10}$$

- Strategy (2), denoted by $\boldsymbol{\omega}^{(2)}$, takes long and short positions when ANN predicts positive and negative returns, respectively. That is, we take a long (short) position when the next return is in the 1st or 2nd ($N^{[L]}$-1th or $N^{[L]}$th) class:

$$\omega_t^{(2)} = \begin{cases} 1 & , \quad \text{if } \max\left\{(o_{i,t})_{i=1,\ldots,N^{[L]}}\right\} = o_{1,t} \text{ or } o_{2,t}, \\ -1 & , \quad \text{if } \max\left\{(o_{i,t})_{i=1,\ldots,N^{[L]}}\right\} = o_{N^{[L]},t} \text{ or } o_{N^{[L]}-1,t}, \\ 0 & , \quad \text{otherwise}. \end{cases} \tag{11}$$

- Strategy (3), denoted by $\boldsymbol{\omega}^{(3)}$, takes long and short positions when ANN "strongly" predicts positive and negative returns, respectively. That is, we take long and short positions only when the next return is in the 1st and $N^{[L]}$th classes, respectively:

$$\omega_t^{(3)} = \begin{cases} 1 & , \quad \text{if } \max\left\{(o_{i,t})_{i=1,\ldots,N^{[L]}}\right\} = o_{1,t}, \\ -1 & , \quad \text{if } \max\left\{(o_{i,t})_{i=1,\ldots,N^{[L]}}\right\} = o_{N^{[L]},t}, \\ 0 & , \quad \text{otherwise}. \end{cases} \tag{12}$$

Let us remember that $o_{i,t}$ is interpreted as a probability that the next return gets in the ith class ($i = 1, \ldots, N^{[L]}$).

Although it is difficult to take a short selling position (if possible, it is often required to pay expensive borrowing costs) in usual cryptocurrency exchanges, we have employed strategy (2) and (3) due to the expected development of cryptocurrency markets including futures markets.

As for time transition of asset values $(V_t)_{t=0,\ldots,T}$, we assume investment principal at time t equals to the time-t asset value V_t, which implies that $\omega_t V_t$ represents the amount of money invested in BTC at time t. That is, investment returns are compounded as follows.

$$\begin{aligned} V_{t+1} &= (V_t - C_t)\left(1 + \omega_t R_{t+1}\right), \quad V_0 = 1, \\ C_t &= \frac{c}{2}|\omega_t V_t - \omega_{t-1} V_{t-1}(1 + R_t)|, \quad C_0 = \frac{c}{2}|\omega_0 V_0|, \end{aligned} \tag{13}$$

where C_t denotes execution cost arising from a bid–ask spread c at time t. In the numerical experiments, we test the cases $c = 0/2.5/5/10$ bps. Let us remark that according to data.bitcoinity.org,[4] the average bid–ask spread during this test period is 2.37 *bps* in Bitfinex, that is one of the biggest cryptocurrency exchanges in the world.

Although the above-defined asset value process supposes compound investment returns, we will also treat with the case of simple returns in Section 3.9 for implementing more realistic simulations based on futures market data.

3. Numerical experiment

This section summarizes the results of our investment simulations: First of all, for several bid–ask spread assumptions, Section 3.1 shows the base case ANN results, which is also compared to a benchmark Buy-and-Hold (B&H) strategy of BTC

3 More precisely, we train ANN with different random seeds at 150 times due to the "dying ReLU" problem explained in later Section 3.3, and select 100 properly trained ANNs for the model averaging.

4 https://data.bitcoinity.org/markets/spread/6m/USD?c=e&f=m10&r=day&st=log&t=l

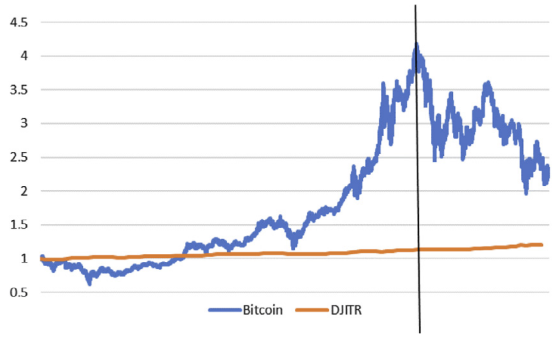


Fig. 1. Transition of BTC and DJIA.

as well as Dow Jones Industrial Average (DJIA), one of the most well-known US stock market indexes. Then, we attempt various sensitivity analysis in the remaining sections: Section 3.2 first implements shallower and deeper layer ANN models. Next, Section 3.3 treats with different activation function, the so-called leaky ReLU, to confirm dying ReLU problems. In addition, Section 3.4 checks different output layers, that is, 3 and 5-class classifications in addition to the base case 4-class classification. Also, Sections 3.5 and 3.6 focus on the input layers: In Section 3.5, time-series of returns, instead of technical indicators, are applied to the ANN input, while Section 3.6 employs additional technical indicators. Moreover, Section 3.7 attempts to construct the more advanced trading strategies. Further, Section 3.8 simulates the base case ANN model-based investment under the dynamically changing bid–ask spreads. Finally, in Section 3.9, we execute the simulation with BTC futures market data.

Now, let us remark that each of the numerical experiments presented below is implemented in Python using TensorFlow on Microsoft Windows 10 Pro with Intel®Xeon®Processor E5-2603 v4, 1.70 GHz. We remark that Table C.1 in Appendix shows the computational time to train one ANN for each variation under this environment.

3.1. Base case ANN

First of all, let us check investment performance of the B&H strategy of BTC in Fig. 1 and Table 1, where the movement of Dow Jones Industrial Average (DJIA) index is also displayed to confirm BTC market characteristics compared to the traditional financial markets.[5]

That is, for the test period, DJIA shows a steady uptrend price change, which is reflected in a fine risk-return balance such as high Sharpe ratio, 9.58, with low maximum drawdown, 1%. On the other hand, although BTC attains significantly high return, i.e. about 700% in annual basis, it also repeats serious drawdowns more than 30% within such a short-term test period as 3–4 months. Actually, BTC records 53% of maximum drawdown in the period when world financial markets are under stable conditions as shown in the performance of DJIA, i.e., one of the most familiar US stock market indexes.

In other words, regardless of potential high return, BTC market is usually exposed to significant risk as DJIA has experienced at Lehman shock, which implies standard risk averse investors hesitate to employ a buy-and-hold (B&H) strategy of BTC, compared to the cases of traditional financial instruments. In that sense, there is a practical necessity to develop alternative investment strategies to reduce its downside risk. Our ANN-based strategy is expected to take this role, due to its high representative ability for financial time-series data with non-linearity and high volatility, as pointed out in Section 1.

Table 2 and Fig. 2 show performance comparison between the buy-and-hold (B&H) strategy and our base case ANN model for various bid–ask spreads $c = 0/2.5/5/10$ bps. The detailed definition of the performance measures appearing in Table 2 is described in Appendix.

Importantly, it can be observed that all of the strategies (1), (2) and (3) are superior to the B&H strategy except for the widest bid–ask spread case ($c = 10$ bps). Moreover, it is remarkable that strategies (1)–(3) remain satisfactory performances over the period from December 2017 to January 2018, when BTC performance substantially deteriorates. Let us note that strategy (3) performs better than strategy (2), which implies that precise prediction of the middle classes, i.e. $q_{1/4} \leq r_t < q_{3/4}$ in Eq. (9), seems difficult for our ANN model.

In addition, it is also clear that as the bid–ask spread gets larger, the performances of all the strategies seem to be exponentially worse. Actually, in the case of $c = 10$ bps, our models are defeated by the B&H strategy, though it seems to be a higher cost setting than those in practice. Let us remember that the average bid–ask spread level over this period is below $c = 2.5$ bps, as explained in Section 2.3. Hereafter, we employ the cost level $c = 5$ bps as the base case, because it is the most conservative level among the candidates $c = 0/2.5/5$ bps, under which the base case ANN model performs better than the benchmark B&H strategy.

[5] Due to our limited accessibility to intraday data of DJIA, this results are calculated from daily return data. The data source of DJIA is Bloomberg. We remark that in Table 1 the performance measures of DJIA are represented in annual basis except for maximum drawdown, as well as those of BTC.

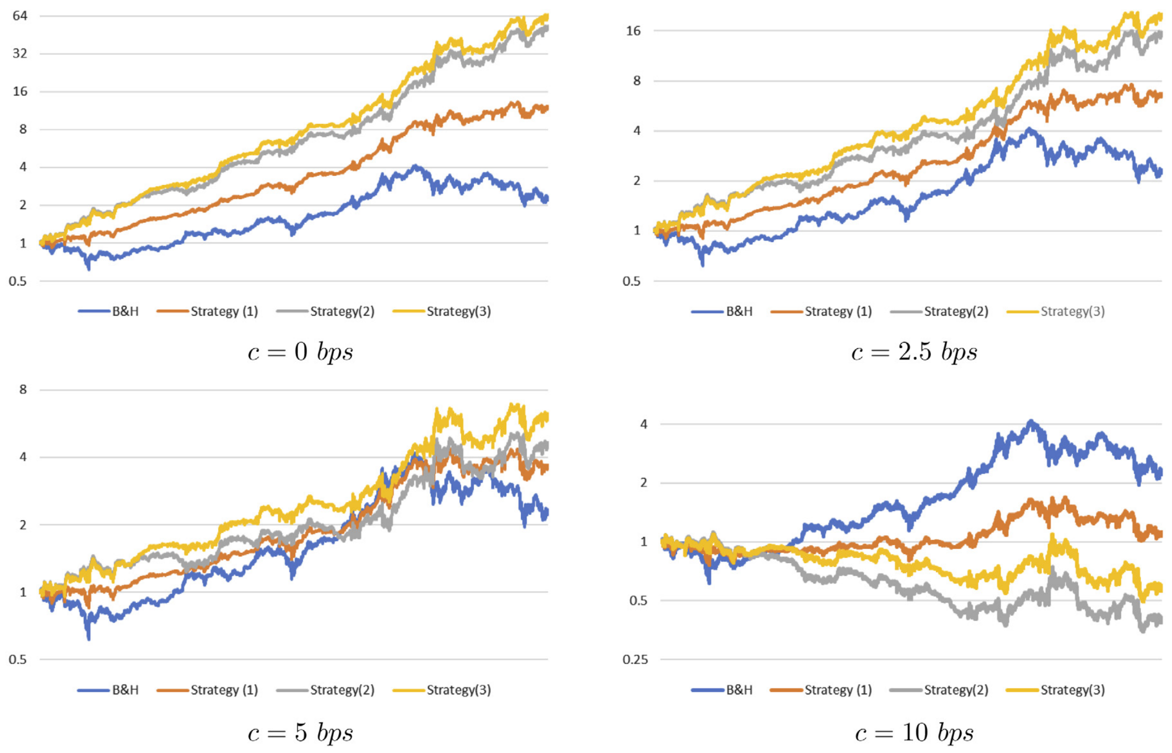


Fig. 2. Transition of asset value V_t for various bid–ask spreads c (base-2 log graph).

Table 1
Performance measure: BTC and DJIA.

	BTC	DJIA
Final value V_T	2.28	1.20
Compound return	7.01	0.63
Standard deviation	1.49	0.07
Downside deviation	1.05	0.03
Maximum drawdown	0.53	0.01
Sharpe ratio	4.69	9.58
Sortino ratio	6.66	21.73

Finally, let us focus on the performance during a period with important price increase, i.e. from the beginning of the test period, 2017/9/1 12:30:00 (GMT), to 2017/12/17 12:00:00 (GMT) shown by the vertical line in Fig. 1. Table 3 shows the performance comparison between the B&H strategy and our ANN-based strategies for $c = 5$ bps. Let us remark that by extracting an uptrend period, the B&H strategy of BTC turns out to outperform that of DJIA in Table 1 for risk-return profile. Even under such a substantial uptrend phase, our ANN approaches are not necessarily inferior to the B&H strategy. Actually, our strategy (1) and (3) achieve the higher risk-return profiles, though the strategy (2) cannot outperform the B&H strategy except for maximum drawdown, whose results are satisfactory for our purpose to take BTC returns with protection against frequent substantial drawdowns.

3.2. ANN with different layers

In this section, we test the sensitivity analysis to the number of layers in the ANNs. Since the combination of nodes among multiple layers improves ANN representative ability, deep layer models can offer both computational and statistical advantage against shallow layer ones to attain the same performance, as argued by [37]. In fact, some existing researches have proven that according to the increase of the number of layers, the efficiency of ANN representation exponentially improves for certain function classes (e.g. [38,39]).

In particular, we newly design ANNs with shallower and deeper layers, i.e. four and ten layers, respectively:

- Shallower case: $(N^{[\ell]})_{\ell=1,\dots,4} = (12, 30, 10, 4)$,
- Deeper case: $(N^{[\ell]})_{\ell=1,\dots,10} = (12, 40, 35, 30, 25, 20, 15, 10, 5, 4)$.

Table 2
Result of investment: The base case ANN model.

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
$c = 0$ bps				
Final value V_T	2.28	12.14	52.21	64.46
Compound return	7.02	540.00	21375.15	36367.81
Standard deviation	1.49	1.21	1.49	1.48
Downside deviation	1.05	0.84	1.04	1.03
Maximum drawdown	0.53	0.26	0.28	0.28
Sharpe ratio	4.70	447.95	14318.04	24601.06
Sortino ratio	6.66	642.20	20545.45	35348.43
$c = 2.5$ bps				
Final value V_T	2.28	6.68	15.49	19.89
Compound return	7.02	118.92	997.67	1876.51
Standard deviation	1.49	1.21	1.49	1.48
Downside deviation	1.05	0.84	1.05	1.04
Maximum drawdown	0.53	0.26	0.30	0.30
Sharpe ratio	4.70	98.64	668.14	1269.14
Sortino ratio	6.66	141.05	952.29	1811.52
$c = 5$ bps				
Final value V_T	2.28	3.67	4.59	6.14
Compound return	7.01	25.57	45.59	95.80
Standard deviation	1.49	1.21	1.49	1.48
Downside deviation	1.05	0.85	1.06	1.04
Maximum drawdown	0.53	0.28	0.34	0.34
Sharpe ratio	4.69	21.21	30.52	64.77
Sortino ratio	6.66	30.25	43.21	91.83
$c = 10$ bps				
Final value V_T	2.28	1.11	0.40	0.58
Compound return	7.01	0.30	−0.90	−0.74
Standard deviation	1.49	1.21	1.50	1.48
Downside deviation	1.05	0.85	1.07	1.06
Maximum drawdown	0.53	0.40	0.69	0.54
Sharpe ratio	4.69	0.25	−0.60	−0.50
Sortino ratio	6.65	0.36	−0.84	−0.70

Table 3
Result of investment: Period of price increase ($c = 5$ bps).

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
Final value V_T	4.18	3.92	3.26	4.50
Compound return	31.24	26.67	16.62	37.58
Standard deviation	1.07	0.84	1.07	1.05
Downside deviation	0.75	0.58	0.75	0.74
Maximum drawdown	0.41	0.24	0.25	0.25
Sharpe ratio	29.33	31.88	15.60	35.95
Sortino ratio	41.89	45.80	22.10	51.09

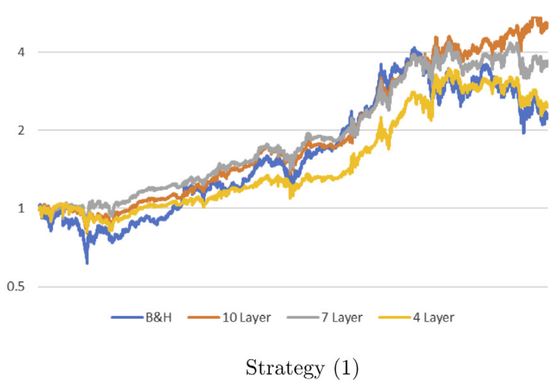
Please remind of the base case layers and nodes $(N^{[\ell]})_{\ell=1,\ldots,7} = (12, 40, 30, 20, 10, 5, 4)$, as shown in Section 2.2.1. The other settings such as activation functions and training dataset are the same as the base case. Then, the resulting performance under the base case setting of execution cost $c = 5$ bps is summarized in the following Table 4 and Fig. 3.

From these results, it is clear that the performance largely depends on the number of layers. Especially, although the ANN with four layers ($L = 4$) performs worse than B&H strategy, the ANN with ten layers ($L = 10$) outperforms the base case ($L = 7$) for almost all of the criteria. In this particular experiment, the performance gets better as the number of layers increases, which is consistent to the previous researches such as [37].

However, it is well-known that the introduction of more hidden layers may also cause troubles concerning the training of the neural network, such as the gradient vanishing problem, or over-fitting to in-sample data, even when applying regularization techniques. In general, there is a turning point in which more layers may actually hinder out-of-sample prediction. Also, the number of nodes in each layer may impact on the results. In fact, more often than not, parsimony is more worthwhile than complexity. Accordingly, we need to be cautious about applying positive effects of the number of layers and nodes on the performance in our specific example to other cases, since neural network methods tend to be quite sensitive to the architecture of the network itself.

3.3. *ANN with different activation function*

Although the base case ANN adopts ReLU as the activation function in the hidden layers, ReLU sometimes leads to "dying ReLU" problems, as mentioned in Section 2.2.1. The dying ReLU problem is that some nodes come to return zero for any input in a training process, which is often caused by their bias terms trained to be large negative. Importantly, once a node gets



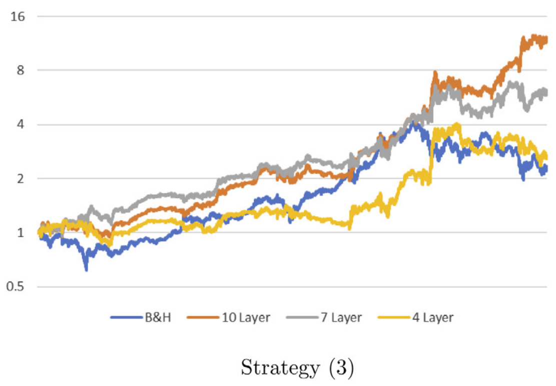


Fig. 3. Comparison of asset value V_t ($c = 5$ *bps*, base-2 log graph).

Table 4
Result of investment: The different number of layers L ($c = 5$ bps).

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
$L = 10$				
Final value V_T	2.28	5.15	8.82	12.03
Compound return	7.01	61.21	240.73	527.88
Standard deviation	1.49	1.22	1.49	1.48
Downside deviation	1.05	0.85	1.04	1.03
Maximum drawdown	0.53	0.23	0.28	0.28
Sharpe ratio	4.69	50.08	161.18	357.17
Sortino ratio	6.66	72.27	231.78	514.67
$L = 4$				
Final value V_T	2.28	2.50	1.78	2.65
Compound return	7.01	9.04	3.27	10.63
Standard deviation	1.49	1.20	1.49	1.48
Downside deviation	1.05	0.84	1.06	1.05
Maximum drawdown	0.53	0.34	0.42	0.42
Sharpe ratio	4.69	7.56	2.19	7.19
Sortino ratio	6.66	10.76	3.08	10.15

stuck in this state, it is unlikely to recover, because the zero gradient in the negative region of ReLU implies that the weights and biases will be never updated. Therefore, if all the nodes in a layer fall into this dying ReLU problems, the gradient-based learning process actually terminates, whose resulting ANN always outputs the same values regardless of input data.

Our model averaging procedure, which is introduced to omit the dependence on random seed choice in Section 2.2.2, is also one of the most primitive ways to avoid this problem. That is, as explained in Section 2.2.2, after excluding the ANNs that always return the same values from 150 candidates, we randomly select 100 ANNs for the model averaging. Remark that since the learning results in the training period tell us the failure of ANN training, our simulation remains to be out-of-sample. However, it is obvious that training extra ANNs is time-consuming, and there is a more sophisticated solution to this problem.

That is, [34] has introduced the following leaky ReLU function: For $\ell = 2, \dots, 6$,

$$\boldsymbol{h}^{[\ell]}(\boldsymbol{y}^{[\ell]}) = \left(\max\{y_i^{[\ell]},\ \alpha y_i^{[\ell]}\}\right)_{i=1,\dots,N^{[\ell]}} \tag{14}$$

where α is a positive constant less than 1. Leaky ReLU enables to escape the dying ReLU problem, due to its non-zero gradient over the entire domain. Moreover, the previous researches such as [34,40] has reported that the use of leaky ReLU improves the performance of ANN compared to ReLU. Thus, in this section, we construct an ANN with leaky ReLU in hidden layers to check the effect of "dying ReLU" problem and the sensitivity to the activation function. Let us notice that the setting other than the activation functions in hidden layers remains to be the same with the base case model.

Firstly, Table 5 compares the number of ANNs failed to train, i.e. ANNs that outputs the same value regardless of the inputs, where we also show its percentage to all the 150 candidates. From this table, it is clear that the number largely decreases by the use of leaky ReLU as expected. This result implies that if the investment performance of leaky ReLU is similar to or better than that of ReLU, leaky ReLU is recommended in the point of computational cost. We remark that the value of α used in this experiment is 0.2, which is the default value in Tensorflow package.

Then, we analyze the results of investment performance based on leaky ReLU in Table 6. Compared to the results of ReLU in Table 2, the performance does not change so much. After all, the use of leaky ReLU is better in the point of computational cost.

Table 5
Leaky ReLU vs ReLU.

	Leaky ReLU	ReLU
the number of ANNs failed to train (out of 150)	8	17
the percentage of ANNs failed to train	5.3%	11.3%

Table 6
Result of investment: Leaky ReLU ($c = 5$ bps).

	Strategy (1)	Strategy (2)	Strategy (3)
Final value V_T	3.68	4.39	6.08
Compound return	25.78	40.60	93.56
Standard deviation	1.21	1.49	1.48
Downside deviation	0.84	1.06	1.04
Maximum drawdown	0.26	0.29	0.29
Sharpe ratio	21.39	27.17	63.25
Sortino ratio	30.51	38.46	89.67

Table 7
Result of investment: Different output layers ($c = 5$ bps).

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
3-class classification				
Final value V_T	2.28	4.06	–	6.89
Compound return	7.01	33.14	–	128.46
Standard deviation	1.49	1.25	–	1.48
Downside deviation	1.05	0.88	–	1.05
Maximum drawdown	0.53	0.274	–	0.30
Sharpe ratio	4.69	26.54	–	86.53
Sortino ratio	6.66	37.84	–	122.75
5-class classification				
Final value V_T	2.28	3.17	4.24	3.86
Compound return	7.01	17.34	37.04	29.16
Standard deviation	1.49	1.23	1.49	1.49
Downside deviation	1.05	0.86	1.06	1.05
Maximum drawdown	0.53	0.24	0.29	0.29
Sharpe ratio	4.69	14.14	24.81	19.58
Sortino ratio	6.66	20.13	35.09	27.70

3.4. *ANN with different output*

In this section, we also test our ANN for 3 and 5-class classification to check the robustness of our approach. Concretely, we construct the following two types of supervisory datasets:

- 3-class classification ($N^{[7]} = 3$)

$$\boldsymbol{s}_t = \begin{cases} (1, 0, 0) & , \ q_{2/3} \leq r_t, \\ (0, 1, 0) & , \ q_{1/3} \leq r_t < q_{2/3}, \\ (0, 0, 1) & , \ r_t < q_{1/3}. \end{cases} \tag{15}$$

- 5-class classification ($N^{[7]} = 5$)

$$\boldsymbol{s}_t = \begin{cases} (1, 0, 0, 0, 0) & , \ q_{3/4} \leq r_t, \\ (0, 1, 0, 0, 0) & , \ c/2 \leq r_t < q_{3/4}, \\ (0, 0, 1, 0, 0) & , \ -c/2 \leq r_t < c/2, \\ (0, 0, 0, 1, 0) & , \ q_{1/4} \leq r_t < -c/2, \\ (0, 0, 0, 0, 1) & , \ r_t < q_{1/4}. \end{cases} \tag{16}$$

In the 5-class classification, the constant parameter c is related to potential bid–ask spreads. As stated in Section 2.1, although the downloaded data are the traded prices in a cryptocurrency exchange, it is unrealistic to assume no bid–ask spreads in trading simulations. Therefore, in our numerical experiments, bid–ask spreads are taken into account by the way concretely explained in Section 2.3. Then, if expected return in the next 15 min is less than the costs arising from the trading execution, its trading seems nonsense, which implies the possible effectiveness of the 5-class classification.

As shown in Table 7 and Fig. 4, there are some performance difference among the three models. Specifically, it is observed from this table that the 3-class classification outperforms the base case model in terms of performance measures, while the

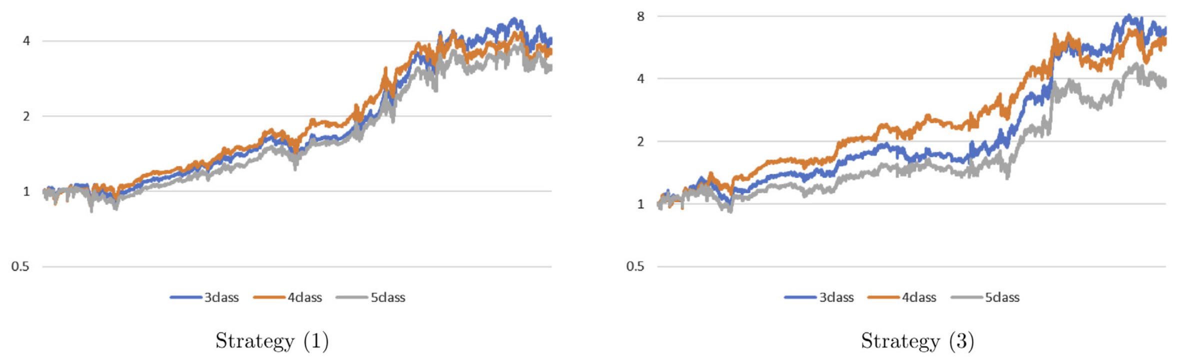


Fig. 4. Comparison among the different number of classification ($c = 5$ *bps*, base-2 log graph).

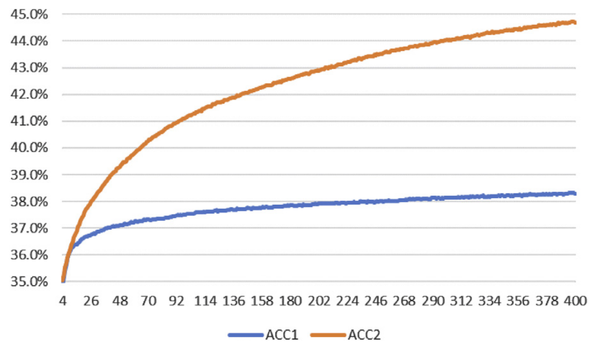


Fig. 5. Accuracy for training data (4-class): Technical indicator ACC$_1$ vs Return ACC$_2$.

5-class one performs the worst. In other words, as the number of classes in classification gets less, our ANN-based investment performs better. One of the possible interpretations of this result is that the number of our training data, i.e. about 38,000, is not enough to learn more than 5-class classification without over-fitting. In spite, a closer observation in Fig. 4 shows that the 3-class case is not necessarily superior to the base case model for the whole trading period. That is, in the first-half period, the base case seems to outperform the 3-class one.

3.5. ANN with different inputs

In this section, we confirm the effectiveness of technical indicators as ANN inputs. Especially, we implement ANN models with input of the past return data for comparison:

$$\boldsymbol{i}_t = (r_{t-u})_{u=1,\ldots,48} \tag{17}$$

For output layers, in addition to the base case 4-class classification ($N^{[7]} = 4$), we also test the cases of 3 and 5-classes ($N^{[7]} = 3,\ 5$), introduced in the previous Section 3.4 to confirm the robustness of results.

In order to clarify the relative effectiveness of technical indicators to historical time-series of returns as ANN inputs, we first analyze the learning history. Particularly, we focus on the model accuracy, because the investment performance does not directly link to the models' prediction performances. Let us remark that the following model accuracy (ACC) stands for the percentage of the correct predictions from all predictions.

$$\mathrm{ACC} = \frac{\#\left\{t \in \hat{\mathbb{T}} \,\middle|\, \arg\max_{i=1,\ldots,N^{[7]}} o_{i,t} = \arg\max_{i=1,\ldots,N^{[7]}} s_{i,t}\right\}}{\#\left\{t \in \hat{\mathbb{T}}\right\}}, \tag{18}$$

where $\hat{\mathbb{T}}$ denotes a set of time indexes. Besides, for a given set A, $\#A$ is its counting measure, i.e. $\#A := \sum_{t\in A} 1$. That is, the numerator of Eq. (18) denotes the number of correct predictions (classifications), i.e. the number of $t \in \hat{\mathbb{T}}$ such that

$$\arg\max_{i=1,\ldots,N^{[7]}} o_{i,t} = \arg\max_{i=1,\ldots,N^{[7]}} s_{i,t}. \tag{19}$$

As mentioned in Section 2.2.2, we have employed model averaging for ANN learning. Then, we also take averages of ACCs calculated from the 100 trials, whose results are shown in Figs. 5–10.

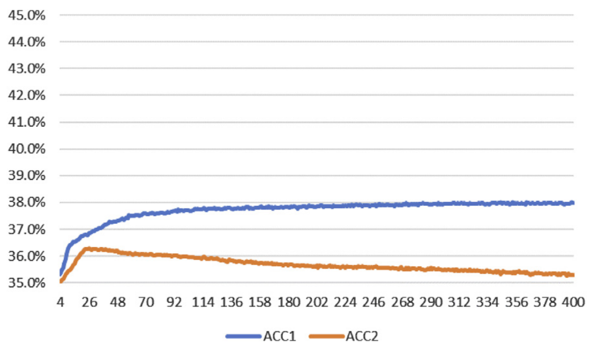


Fig. 6. Accuracy for test data (4-class): Technical indicator ACC_1 vs Return ACC_2.

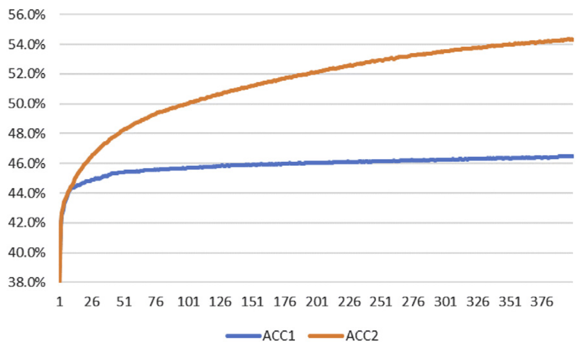


Fig. 7. Accuracy for training data (3-class): Technical indicator ACC_1 vs Return ACC_2.

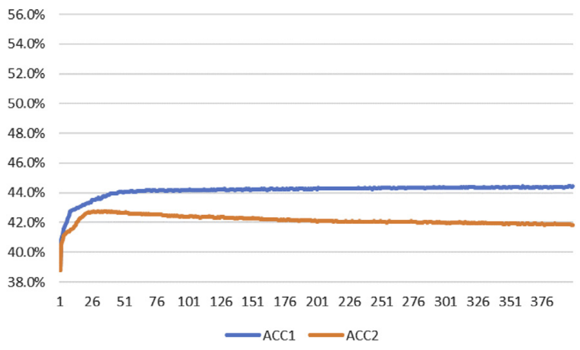


Fig. 8. Accuracy for test data (3-class): Technical indicator ACC_1 vs Return ACC_2.

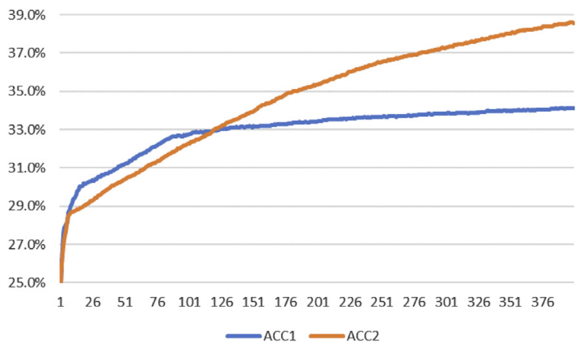


Fig. 9. Accuracy for training data (5-class): Technical indicator ACC_1 vs Return ACC_2.

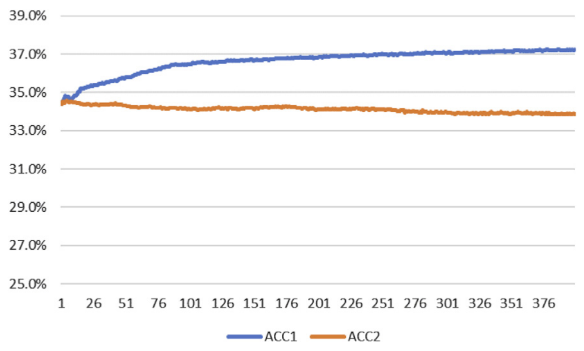


Fig. 10. Accuracy for test data (5-class): Technical indicator ACC_1 vs Return ACC_2.

Table 8
Result of investment: Return input ($c = 5$ bps).

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
4-class classification				
Final value V_T	2.28	1.12	0.40	0.57
Compound return	7.01	0.34	−0.90	−0.76
Standard deviation	1.49	1.23	1.50	1.48
Downside deviation	1.05	0.88	1.08	1.07
Maximum drawdown	0.53	0.47	0.65	0.56
Sharpe ratio	4.69	0.28	−0.60	−0.51
Sortino ratio	6.66	0.39	−0.84	−0.71
3-class classification				
Final value V_T	2.28	2.15	–	1.75
Compound return	7.01	5.88	–	3.09
Standard deviation	1.49	1.22	–	1.49
Downside deviation	1.05	0.86	–	1.06
Maximum drawdown	0.53	0.36	–	0.43
Sharpe ratio	4.69	4.83	–	2.07
Sortino ratio	6.66	6.84	–	2.92
5-class classification				
Final value V_T	2.28	2.37	2.35	2.12
Compound return	7.01	7.76	7.59	5.67
Standard deviation	1.49	1.23	1.49	1.49
Downside deviation	1.05	0.87	1.06	1.06
Maximum drawdown	0.53	0.33	0.37	0.36
Sharpe ratio	4.69	6.31	5.08	3.80
Sortino ratio	6.66	8.95	7.16	5.35

These figures display the historical records of model accuracy in training and test periods, respectively. We can see that the accuracies based on technical indicator inputs are higher than those of return inputs in the test period, while vise versa in the training period. Clearly, we can escape over-fitting by the use of technical indicators, which implies that those transformation of return data may enable to extract important information and cut off extra noises.

Now, let us show the investment performances for different input data in Table 8 and Fig. 11. As expected from the results of accuracy in the test period, the performance based on the ANN model with technical indicator inputs is much higher than that with time-series of return inputs. Particularly, the ANN with return inputs cannot improve the investment performance compared to the simple B&H. In other words, some pre-processing of input data seems effective to prevent over-fitting in the case of non-stationary financial time-series data. Let us remark that although the simple technical trading only based on the indicators (EMAs) does not perform well, as shown in Appendix, the combination of ANNs and technical indicators is quite effective.

3.6. ANN with additional technical indicators

In the previous section, we confirm the relative effectiveness of technical indicators as ANN input to simple time-series of returns. Then, this section further explores their possibility by adding technical indicators, i.e. the so-called MACD, Stochastic Oscillators, and On-Balance Volume. Before showing the definitions, we introduce normalized closing, high and low prices, $(p_t)_{t\in\mathbb{T}_1}, (p_t^h)_{t\in\mathbb{T}_1}, (p_t^\ell)_{t\in\mathbb{T}_1}$:

$$p_t := \frac{P_t - \mathrm{Avg}((P_t)_t)}{\mathrm{Std}((P_t)_t)}, \quad p_t^h := \frac{P_t^h - \mathrm{Avg}((P_t)_t)}{\mathrm{Std}((P_t)_t)}, \quad p_t^\ell := \frac{P_t^\ell - \mathrm{Avg}((P_t)_t)}{\mathrm{Std}((P_t)_t)}, \tag{20}$$

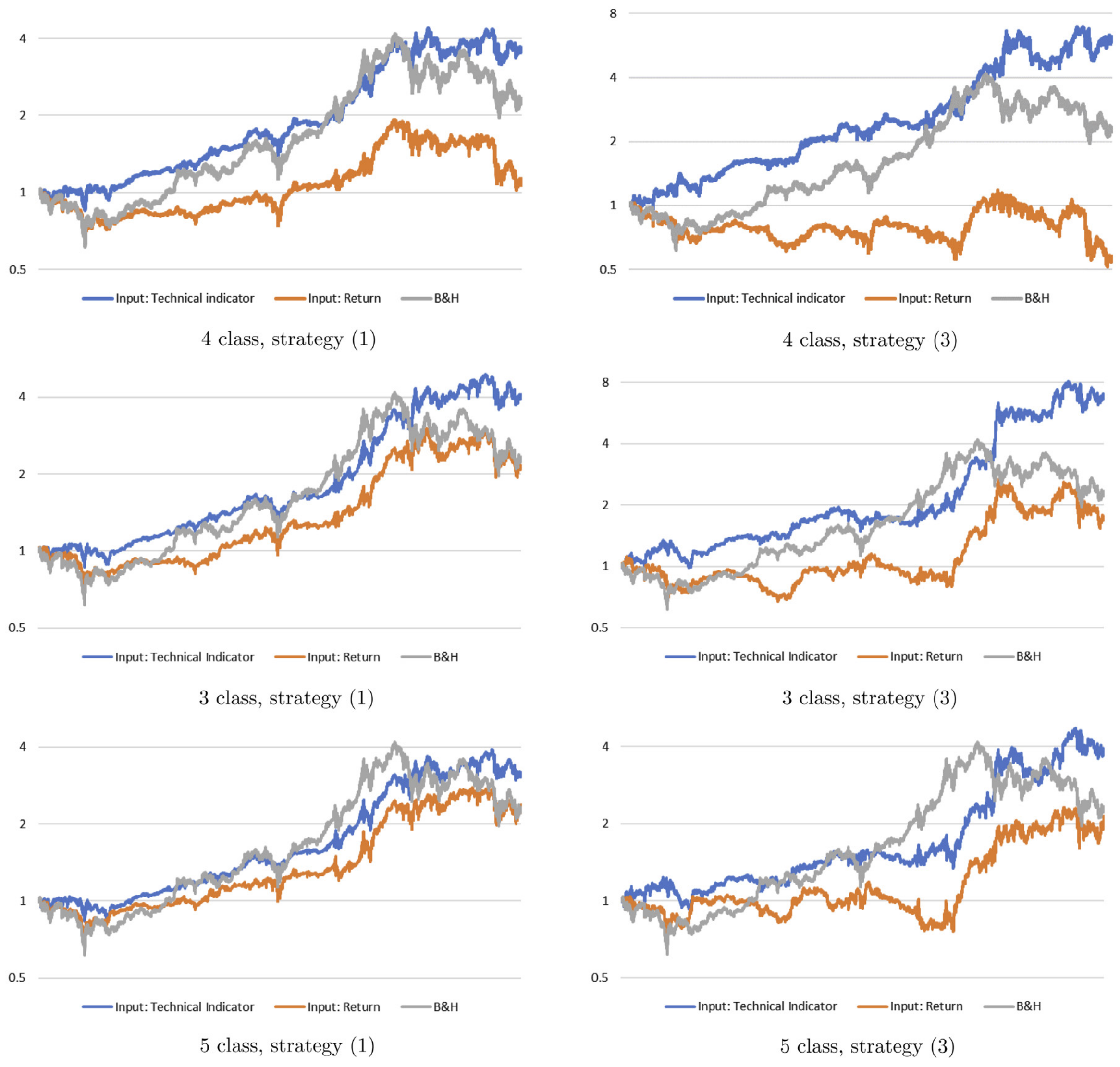


Fig. 11. Comparison of asset value V_t ($c = 5$ *bps*, base-2 log graph).

where P_t, P_t^h, P_t^ℓ denote raw closing, high and low prices as shown in Section 2.1, and Avg() and Std() represent sample average and standard deviation, respectively. Let us notice that in the definition of p_t^h and p_t^ℓ, we take averages of closing prices $(P_t)_t$ to preserve the relation of $p_t^\ell \leq p_t \leq p_t^h$.

- Moving Average Convergence Divergence (MACD):

$$MACD_{M_1,M_2,t} = EMA_{M_1,t} - EMA_{M_2,t}, \quad M_1 < M_2 \tag{21}$$

  MACD is a trend-following momentum indicator, which is defined as difference between short and long term EMA. Please recall that the definition of EMA is shown in Eq. (7), Section 2.2.2. Here, we test the cases $(M_1, M_2) = (2, 12), (2, 24), (4, 24)$.
- Stochastic Oscillator (SO):

$$SO_{M_3,M_4,t} = \%K_{M_3,t} - \%D_{M_3,M_4,t},$$
$$\%K_{M_3,t} = \frac{p_{t-1} - \min_{t-M_3-1<s\leq t-1}\left\{p_s^\ell\right\}}{\max_{t-M_3-1<s\leq t-1}\left\{p_s^h\right\} - \min_{t-M_3-1<s\leq t-1}\left\{p_s^\ell\right\}}, \quad \%D_{M_3,M_4,t} = \frac{1}{M_4}\sum_{s=t-M_4}^{t-1} \%K_{s,M_3}. \tag{22}$$

  Stochastic Oscillator is a momentum indicator that refers to where the last closing price locates in the price range over a certain period. Here, we test the cases $(M_3, M_4) = (12, 12), (24, 12), (48, 12)$.

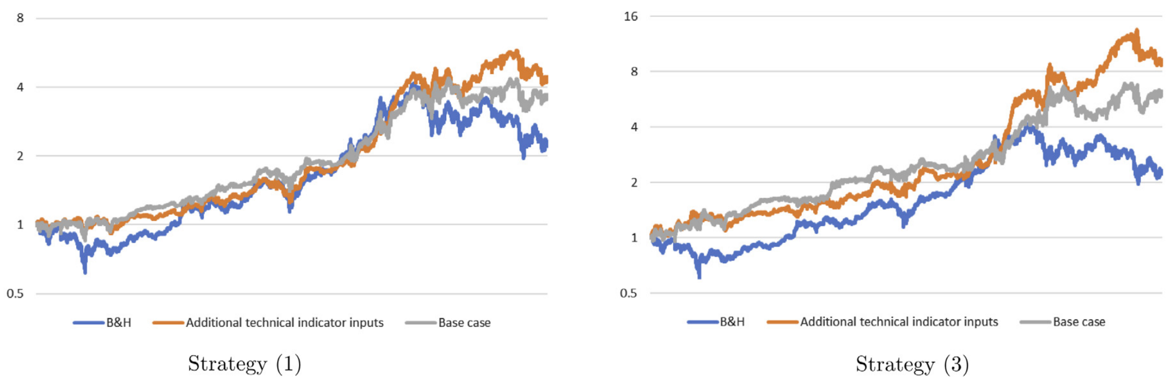


Fig. 12. Comparison of asset value V_t ($c = 5$ *bps*, base-2 log graph).

Table 9
Result of investment: Additional technical indicators input ($c = 5$ bps).

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
Final value V_T	2.28	4.40	6.43	9.23
Compound return	7.01	40.99	108.03	269.75
Standard deviation	1.49	1.26	1.49	1.48
Downside deviation	1.05	0.88	1.05	1.04
Maximum drawdown	0.53	0.30	0.36	0.36
Sharpe ratio	4.69	32.51	72.34	181.81
Sortino ratio	6.66	46.60	103.24	259.82

- On-Balance Volume (OBV):

$$OBV_t = \begin{cases} OBV_{t-1} + U_{t-1} & , \quad \text{if } p_{t-1} > p_{t-2}, \\ OBV_{t-1} - U_{t-1} & , \quad \text{if } p_{t-1} < p_{t-2}, \\ OBV_{t-1} & , \quad \text{if } p_{t-1} = p_{t-2}, \end{cases} \tag{23}$$

where U_t denotes trading volume at time t. On-Balance Volume is a momentum indicator based on the idea that when volume increases (decreases) sharply without significant price changes, the price will eventually jump upward (downward). Let us notice that (OBV_t) are not calculated from normalized data differently from the other indicators, because volume data $(U_t)_t$ should be positive for the meaningful definition of On-Balance Volume. Then, we normalize $(OBV_t)_{t\in\mathbb{T}_1}$ themselves instead of $(U_t)_t$ as follows:

$$obv_t := \frac{OBV_t - \mathrm{Avg}((OBV_t)_t)}{\mathrm{Std}((OBV_t)_t)}. \tag{24}$$

Now, we implement the ANN model with the following input:

$$\boldsymbol{i}_t = \Big((EMA_{M,t})_M, (EMSD_{M,t})_M, (RSI_{K,t})_K, r_{t-1}, (MACD_{M_1,M_2,t})_{M_1,M_2}, (SO_{M_3,M_4,t})_{M_3,M_4}, obv_t \Big) \in \mathbb{R}^{19}, \tag{25}$$

The other parts of ANN model, e.g. output and hidden layers, nodes and activation functions, are the same as the base case. Then, the resulting performance is summarized in Table 9.

From Table 9 and Fig. 12, it is found out that the ANN model with additional technical indicators outperforms the base case model, as well as the B&H strategy. In other words, through incorporating new technical indicators into the model, effective patterns for time-series prediction are additionally recognized in neural networks. This result strengthens the conclusion in the previous Section 3.5.

3.7. Further extension of trading strategy

Now, let us consider some extension of the simple trading strategies (1)–(3) for the base case ANN model. As illustrated in Section 3.1, since our trading interval is relatively short, i.e. fifteen minutes, the trading costs especially have a crucial impact on investment performances. Therefore, one of the promising extensions of trading strategies is to introduce more conservative investment decisions, which leads to reduce the number of trades, i.e. execution costs. That is, we impose the following restriction to execute trading on the strategies (1)–(3) by using $p^u_t := o_{1,t} + o_{2,t}$ and $p^d_t := o_{3,t} + o_{4,t} = 1 - p^u_t$, which basically correspond with probabilities that the next return is positive and negative, respectively. We remark that a median of raw returns $(R_t)_t$ equals to zero in our dataset.

Table 10
Result of hit rate with restriction (training period).

b	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45
Long	51.2%	53.9%	56.2%	61.0%	61.2%	66.0%	68.3%	70.3%	69.5%	91.3%
Short	51.0%	53.1%	53.6%	59.2%	61.6%	65.1%	67.0%	69.2%	71.8%	75.3%
b	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Long	71.4%	100.0%	100.0%	NA	NA	NA	NA	NA	NA	NA
Short	78.3%	85.4%	79.9%	85.7%	NA	NA	100.0%	NA	NA	NA

Table 11
Result of investment: New strategies (1')–(3') ($c = 5$ bps).

	B & H	Strategy (1')	Strategy (2')	Strategy (3')
Final value V_T	2.28	4.54	7.10	8.16
Compound return	7.01	44.40	138.64	197.85
Standard deviation	1.49	0.71	0.85	0.85
Downside deviation	1.05	0.47	0.57	0.56
Maximum drawdown	0.53	0.16	0.27	0.20
Sharpe ratio	4.69	62.28	162.19	232.94
Sortino ratio	6.66	94.87	245.31	353.30

- Strategy (1'):

$$\omega_t^{(1')} = \begin{cases} 1 & , \quad \text{if} \, (\max \left\{(o_{i,t})_i\right\} = o_{1,t}) \,\&(\, p_t^u - p_t^d \geq b), \\ 0 & , \quad \text{otherwise.} \end{cases} \tag{26}$$

- Strategy (2'):

$$\omega_t^{(2')} = \begin{cases} 1 & , \quad \text{if} \, (\max \left\{(o_{i,t})_i\right\} = o_{1,t} \text{ or } o_{2,t}) \,\&\, (p_t^u - p_t^d \geq b), \\ -1 & , \quad \text{if} \, (\max \left\{(o_{i,t})_i\right\} = o_{4,t} \text{ or } o_{3,t}) \,\&\, (p_t^d - p_t^u \geq b), \\ 0 & , \quad \text{otherwise.} \end{cases} \tag{27}$$

- Strategy (3'):

$$\omega_t^{(3')} = \begin{cases} 1 & , \quad \text{if} \, (\max \left\{(o_{i,t})_i\right\} = o_{1,t}) \,\&\, (p_t^u - p_t^d \geq b), \\ -1 & , \quad \text{if} \, (\max \left\{(o_{i,t})_i\right\} = o_{4,t}) \,\&\, (p_t^d - p_t^u \geq b), \\ 0 & , \quad \text{otherwise.} \end{cases} \tag{28}$$

In other words, under those strategies (1')–(3'), we take a long (short) position only if the upward probability p_t^u minus the downward probability p_t^d is more (less) than a constant threshold value $b \in \mathbb{R}$.

To decide the threshold value b, let us introduce the following hit rates HR^u and HR^d based on the learning results of the 1st ANN model.

$$\text{HR}^u = \frac{\#\left\{t \in \tilde{\mathbb{T}}_1 \,\middle|\, \text{arg max}_{i=1,\dots,4}\, o_{i,t} \in \{1,2\},\ \text{arg max}_{i=1,\dots,4}\, s_{i,t} \in \{1,2\}\right\}}{\#\left\{t \in \tilde{\mathbb{T}}_1 \,\middle|\, \text{arg max}_{i=1,\dots,4}\, o_{i,t} = \{1,2\}\right\}},$$
$$\text{HR}^d = \frac{\#\left\{t \in \tilde{\mathbb{T}}_1 \,\middle|\, \text{arg max}_{i=1,\dots,4}\, o_{i,t} \in \{3,4\},\ \text{arg max}_{i=1,\dots,4}\, s_{i,t} \in \{3,4\}\right\}}{\#\left\{t \in \tilde{\mathbb{T}}_1 \,\middle|\, \text{arg max}_{i=1,\dots,4}\, o_{i,t} \in \{3,4\}\right\}}. \tag{29}$$

Let us remark that HR^u (HR^d) stands for the ratio that the next return is actually positive (negative) when our ANN predicts the positive (negative) return.

Importantly, a possible effective threshold level b is required to increase these hit rates through adding the conditions $|p_t^u - p_t^d| \geq b$. Then, in Table 10, we have checked the hit rates with restrictions HR_b^u and HR_b^d for various ranges of the probability difference $|p_t^u - p_t^d|$, i.e. $|p_t^u - p_t^d| \in [b, b + 0.05)$ for $b = 0, 0.05, 0.1, \dots, 0.9, 0.95$, where HR_b^u and HR_b^d are defined as follows:

$$\text{HR}_b^u = \frac{\#\left\{t \in \tilde{\mathbb{T}}_1 \,\middle|\, \text{arg max}_{i=1,\dots,4}\, o_{i,t} \in \{1,2\},\ \text{arg max}_{i=1,\dots,4}\, s_{i,t} \in \{1,2\},\ b \leq p_t^u - p_t^d < b + 0.05\right\}}{\#\left\{t \in \tilde{\mathbb{T}}_1 \,\middle|\, \text{arg max}_{i=1,\dots,4}\, o_{i,t} \in \{1,2\},\ b \leq p_t^u - p_t^d < b + 0.05\right\}},$$
$$\text{HR}_b^d = \frac{\#\left\{t \in \tilde{\mathbb{T}}_1 \,\middle|\, \text{arg max}_{i=1,\dots,4}\, o_{i,t} \in \{3,4\},\ \text{arg max}_{i=1,\dots,4}\, s_{i,t} \in \{3,4\},\ b \leq p_t^d - p_t^u < b + 0.05\right\}}{\#\left\{t \in \tilde{\mathbb{T}}_1 \,\middle|\, \text{arg max}_{i=1,\dots,4}\, o_{i,t} \in \{3,4\},\ b \leq p_t^d - p_t^u < b + 0.05\right\}}. \tag{30}$$

where # denotes a counting measure, as concretely explained in the below of Eq. (18) (Section 3.5).

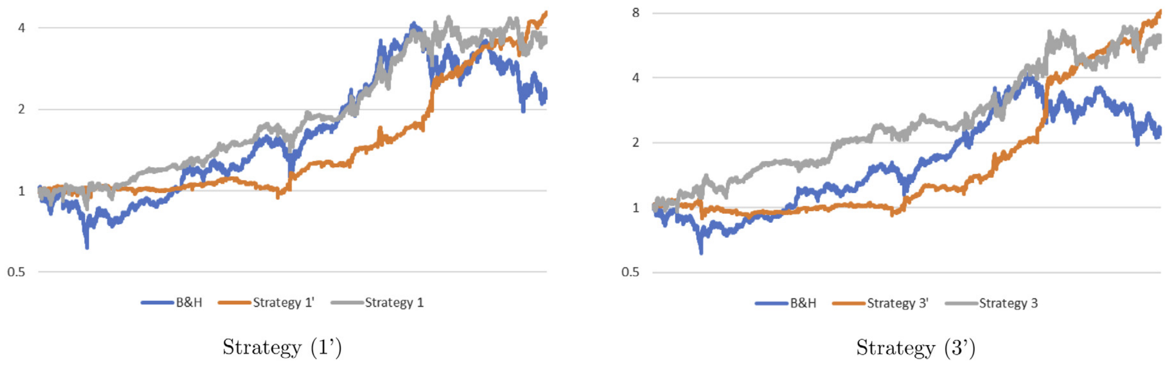


Fig. 13. Comparison of asset value V_t ($c = 5$ *bps*, base-2 log graph).

Table 12
Result of investment: Time-varying bid–ask spread.

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
Final value V_T	2.28	6.79	16.13	20.46
Compound return	7.02	124.19	1105.02	2014.87
Standard deviation	1.49	1.21	1.49	1.48
Downside deviation	1.05	0.84	1.05	1.04
Maximum drawdown	0.53	0.27	0.30	0.30
Sharpe ratio	4.70	103.02	740.06	1362.73
Sortino ratio	6.66	147.24	1053.73	1942.99

Here, NA in Table 10 means that the denominators of HR^u_b and HR^d_b are zero.

Table 10 presents that the hit rates in the cases $b = 0.0, \ldots, 0.15$ are lower than those without any restriction, $\mathrm{HR}^u = 60.4\%$ and $\mathrm{HR}^d = 59.7\%$. Therefore, we implement the new strategies (1')–(3') by setting the threshold value b to be 0.2, whose results are summarized in Table 11 and Fig. 13.

Table 11 shows that our new strategies (1')–(3') successfully outperform the strategies (1)–(3), whose performances are reported in Table 2 ($c = 5$ bps), for overall measures. Especially, as also observed in Fig. 13, risk performances including maximum drawdown are substantially improved by introducing more conservative investment decision, which leads to better risk adjusted returns (Sharpe and Sortino ratios).

3.8. Dynamic change of bid–ask spread

As explained in Sections 2.3 and 3.1, our base case cost setting $c = 5$ bps seems conservative enough to implement realistic simulation. Let us remind that the average bid–ask spread is 2.37 bps in Bitfinex, one of the biggest cryptocurrency exchanges. Nevertheless, it is an important aspect that its level can change through time, which is a topic discussed in this section.

Now, let us remark that under the assumption of exchange trading, there do not arise the transaction costs paid to the miners, because exchange trades do not get access to BTC blockchain (e.g. [41]). That is, in trading among users of the same platform, actual coins remain in the wallet of the exchange. Therefore, a user is required to pay miner fees only in withdrawing (depositing) his coins from (to) the exchange account to (from) his own wallet. In other words, transaction cost payment for miners does not occur at each execution in the exchange.[6]

Alternatively, the changes of transaction costs to miners seem to be reflected into the dynamics of bid–ask spreads. Especially, in December 2017, the transaction costs have tremendously increased, as observed in Fig. 14, which displays daily total transaction costs paid to miners,[7] as well as daily bid–ask spreads. At the same moment, Fig. 14 shows that the (daily) bid–ask spreads have also elevated to more than $c = 7$ bps, which is beyond our base case cost setting $c = 5$ bps.[8]

Thus, we include the time transition of bid–ask spreads into our simulation, whose results with the base case ANN model are summarized in Table 12 and Fig. 15. From these results, dynamic changes of bid–ask spreads do not deteriorate the advantage of our ANN-based approach.

Additionally, let us notice that bid–ask spread levels shown in Fig. 14 are below our base case $c = 5$ bps over almost all of the test period. Then, to reflect the execution costs more conservatively, we also implement the simulation with "twice"

6 On the contrary, for trading among individuals, the transaction costs paid to miners may have significant impact. As for the development of these transaction costs, see [42], for example.

7 Data source: https://blockchain.info/ja/charts/transaction-fees

8 It was difficult for us to get access to intraday bid–ask spread data.

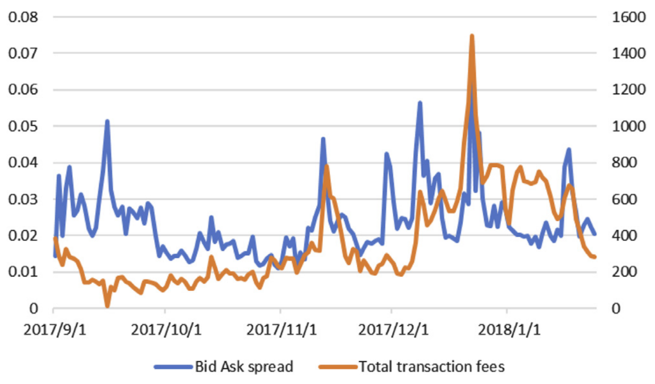


Fig. 14. Transaction cost to miner (right axis, USD) and bid–ask spreads (left axis, %).

Table 13
Result of investment: Time-varying bid–ask spread (doubled case).

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
Final value V_T	2.28	3.80	4.98	6.49
Compound return	7.01	27.96	56.14	110.57
Standard deviation	1.49	1.21	1.49	1.48
Downside deviation	1.05	0.85	1.06	1.05
Maximum drawdown	0.53	0.28	0.35	0.35
Sharpe ratio	4.69	23.19	37.58	74.75
Sortino ratio	6.66	33.04	53.10	105.76

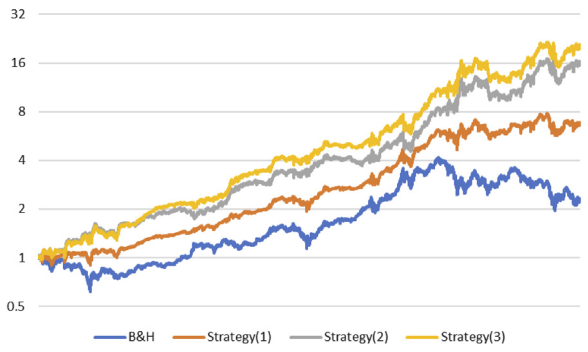


Fig. 15. Asset value with time-varying bid–ask spreads.

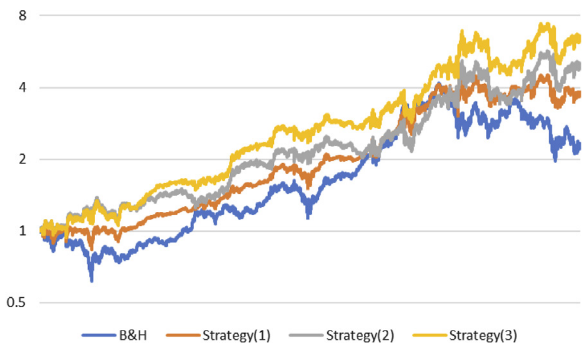


Fig. 16. Asset value with time-varying bid–ask spreads (doubled case).

levels of these bid–ask costs. Let us remember that the average bid–ask spread is 2.37 bps over this period. Thereby, these "doubled" costs setting assumes that in usual time the trading requires around our base case level of cost $c = 5$ bps, though it requires nearby 15 bps under illiquid market phases such as December 2017. As shown in Table 13 and Fig. 16, even for those conservative setting, our proposal remains superior to the B&H strategy.

Table 14
Result of investment: Futures market data (base case model).

	B & H	Strategy (1)	Strategy (2)	Strategy (3)
$c = 0$ bps				
Final value V_T	0.47	1.00	1.53	1.52
Compound return	−1.00	0.02	553.74	491.67
Standard deviation	2.25	1.39	1.38	1.38
Downside deviation	1.63	0.99	0.96	0.97
Maximum drawdown	0.58	0.21	0.21	0.21
Sharpe ratio	–	0.01	402.60	356.73
Sortino ratio	–	0.02	575.06	509.49
$c = 2.5$ bps				
Final value V_T	0.47	0.90	1.33	1.32
Compound return	−1.00	−0.78	69.71	60.87
Standard deviation	2.25	1.46	1.49	1.49
Downside deviation	1.63	1.05	1.05	1.05
Maximum drawdown	0.58	0.26	0.28	0.28
Sharpe ratio	–	–	46.82	40.77
Sortino ratio	–	–	66.43	57.84
$c = 5$ bps				
Final value V_T	0.47	0.80	1.13	1.12
Compound return	−1.00	−0.96	5.46	4.54
Standard deviation	2.25	1.56	1.64	1.64
Downside deviation	1.63	1.12	1.16	1.17
Maximum drawdown	0.58	0.32	0.37	0.37
Sharpe ratio	–	–	3.34	2.76
Sortino ratio	–	–	4.70	3.90
$c = 10$ bps				
Final value V_T	0.47	0.60	0.74	0.73
Compound return	−1.00	−1.00	−0.99	−0.99
Standard deviation	2.25	1.81	2.16	2.17
Downside deviation	1.63	1.31	1.55	1.56
Maximum drawdown	0.58	0.46	0.58	0.59
Sharpe ratio	–	–	–	–
Sortino ratio	–	–	–	–

3.9. Application to futures market data

Although we have implemented strategy (2) and (3) so far, it is difficult to take a short selling position in actual spot exchange markets. However, since 2017/12/10, Bitcoin futures have been available on the CBOE Futures Exchange, which makes it possible to take a short position. Therefore, we also demonstrate the investment performances based on the futures market data.

More concretely, we first download intraday Bitcoin futures data $(\hat{R}_t)_t$ from Bloomberg over the period from 2017/12/10 22:30 (GMT) to 2018/1/17 15:00 (GMT). Next, new inputs $\boldsymbol{i}_t$ for our trained ANN $\boldsymbol{f}(\cdot;\hat{\boldsymbol{\Theta}})$ are calculated from the data as the same way in Section 2.2.2. Finally, the strategies (1), (2), and (3) are applied to the new outputs $\boldsymbol{o}_t = \boldsymbol{f}(\boldsymbol{i}_t;\hat{\boldsymbol{\Theta}})$. The investment performances are shown in Table 14 and Fig. 17 for the bid–ask slippage levels $c = 0/2.5/5/10$ bps.

Here, we redefine the asset value process $(V_t)_t$, so that its returns cumulate in a simple form, not a compound form, in order to reflect the futures trading practice, where its profit and loss realize in a margin account at each moment:

$$V_{t+1} = V_t + \omega_t \hat{R}_{t+1} - C_t, \quad C_t = \frac{c}{2}|\omega_t - \omega_{t-1}|, \quad C_0 = \frac{c}{2}|\omega_0|, \tag{31}$$

As discussed in Section 2.3, a trading strategy ω_t at time t denotes a proportion of amount of money invested in BTC to investment principal at time t. Then, let us notice that since the principal always equals to one in the case of simple return, the amount of money invested in BTC at time t is ω_t, not $\omega_t V_t$, which is different from the previous sections, i.e. the case of compound return.

First of all, Table 14 and Fig. 17 show that the B&H strategy in this test period, i.e. from 2017/12/10 22:30 (GMT) to 2018/1/17 15:00 (GMT), is minus return −53% (= 1 − 0.47). Under such an underperforming market, strategy (2) and (3) attain the positive returns for $c = 0/2.5/5$ bps. As for the long-only strategy (1), it significantly reduces the loss and risk compared with the B&H. In addition, it is observed from Fig. 17 that strategy (2) and (3) successfully monetize the downside trends of BTC returns by utilizing short positions, which implies that our ANN model properly predicts the price movement directions. As well, the asset value transition of strategy (1) seems much more stable than the simple B&H strategy, though it implements frequent trades with execution costs.

Moreover, let us note that differently from the results in Section 3.4, the performance of strategy (3) is very close to that of strategy (2). In fact, the position of strategy (3) differs from strategy (2) only 7 times out of 2361 trading times. Since we use the parameters trained by the spot exchange market data, it seems more difficult to predict middle classes in futures BTC return classification.

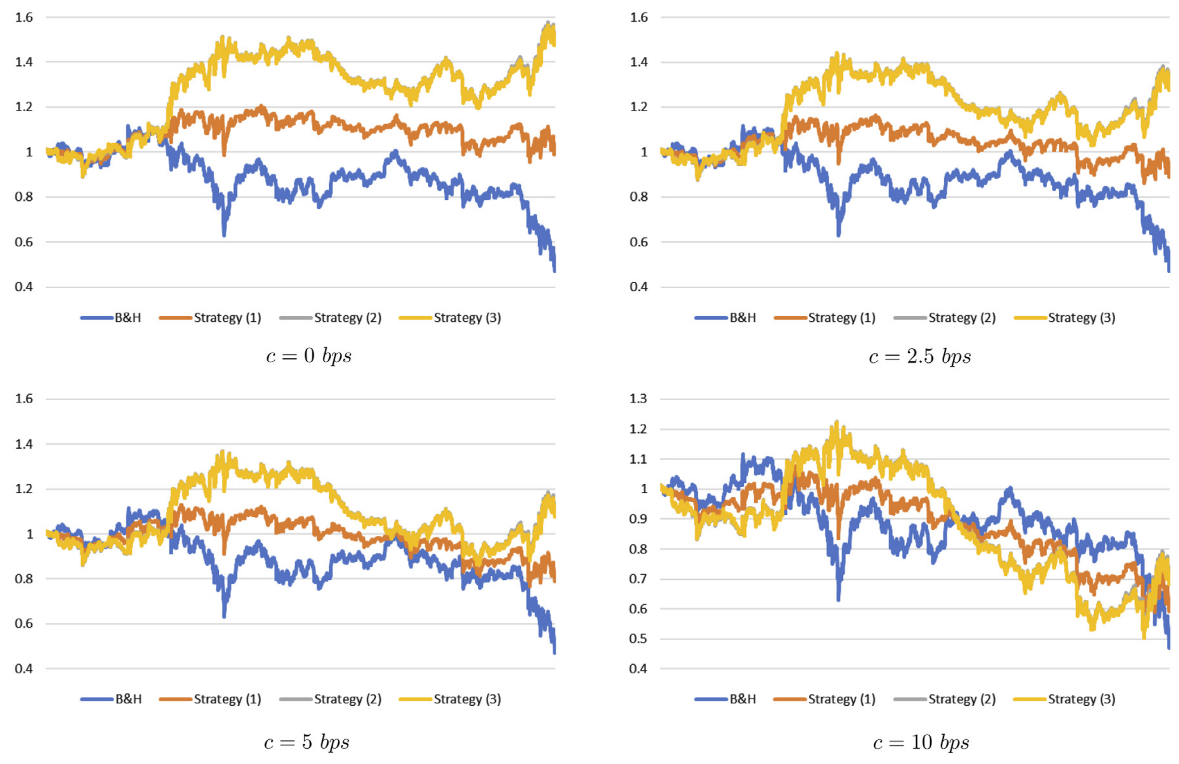


Fig. 17. Transition of asset value: Futures trading.

4. Conclusion

This paper has investigated effective trading strategies of Bitcoin, which is a new financial alternative asset with both features of currency and commodity. In particular, we have designed artificial neural network (ANN) models for classification to extract the meaningful trading signals from the input technical indicators calculated from the time-series return data at 15 min time intervals.

As a result, our approach has turned out to attain higher performances than the buy-and-hold and primitive technical trading strategies under realistic assumptions of execution costs. In addition, by implementing wide range of sensitivity analysis, we have found important implications on the relationship of ANN-based model performance with the execution costs, the number of layers, the determination of activation functions, the number of classes in the output layer, and the input data. Especially, our numerical results have shown that the use of various technical indicators possibly prevents over-fitting in the classification of non-stationary financial time-series data, which enhances trading performance.

Although we have focused on the single asset investment with Bitcoin, it seems valuable to construct a multi-asset portfolio from various cryptocurrencies, which is one of our future research topics.

Acknowledgments

The authors are very grateful to an editor-in-chief, Professor Eugene Stanley and five anonymous referees for their precious comments and suggestions, which substantially improve the previous version of the paper. The authors are also grateful to Mr. Daisuke Murayama at bitFlyer Inc. for valuable comments. This research is supported by CARF, Japan (Center for Advanced Research in Finance). Also, this work is supported by JSPS, Japan KAKENHI Grant Numbers JP17J09046 and JP17J09127.

Appendix A. Definition of performance measure

Here, we briefly describe the definition of the performance measures used in Section 3.

Table B.1
Result of investment: EMA-based strategy.

	B & H	EMA_2	EMA_4	EMA_{12}	EMA_{24}
Strategy (4)					
Final value V_T	2.28	0.11	0.13	0.33	0.35
Compound return	7.01	−1.00	−0.99	−0.94	−0.93
Standard deviation	1.49	1.02	1.01	0.97	0.97
Downside deviation	1.05	0.72	0.72	0.69	0.69
Maximum drawdown	0.53	0.91	0.88	0.73	0.73
Sharpe ratio	4.69	–	–	–	–
Sortino ratio	6.66	–	–	–	–
Strategy (5)					
Final value V_T	2.28	0.00	0.00	0.03	0.03
Compound return	7.01	−1.00	−1.00	−1.00	−1.00
Standard deviation	1.49	1.49	1.49	1.49	1.49
Downside deviation	1.05	1.07	1.07	1.07	1.08
Maximum drawdown	0.53	1.00	1.00	0.98	0.98
Sharpe ratio	4.69	–	–	–	–
Sortino ratio	6.66	–	–	–	–

- Compound Return (CR): We define CR as the annualized geometric average of the investment returns $IR_t = V_t/V_{t-1} - 1$, which is a standard measure of investment returns.

$$CR \equiv \left\{ \prod_{t=1}^{T} (1 + IR_t) \right\}^{\tilde{T}/T} - 1, \tag{A.1}$$

where $\tilde{T} = 365 \times 24 \times 4$, that is the total number of trading times in one year.
- Standard Deviation (SD), Downside Deviation (DD): SD is a well-known investment risk measure defined as the annualized standard deviation of $\{IR_t\}$, while DD only regards negative returns as risk.

$$SD \equiv \left\{ \frac{\tilde{T}}{T} \sum_{t=1}^{T} (IR_t - \bar{R})^2 \right\}^{1/2}, \quad DD \equiv \left\{ \frac{\tilde{T}}{T} \sum_{t=1}^{T} \min(0, IR_t)^2 \right\}^{1/2}, \quad \overline{IR} \equiv \frac{1}{T} \sum_{t=1}^{T} IR_t. \tag{A.2}$$

- Maximum Drawdown (MDD):

$$MDD \equiv \max_{1 \le t \le T} \frac{M_t - V_t}{M_t}, \quad M_t \equiv \max_{0 \le s \le t} V_s. \tag{A.3}$$

The drawdown is the decline from the past peak value M_t to the present value V_t. In general, investment performance depends on the investment timing. The MDD contributes to the performance analysis because it is independent of the investment timing given the horizon $[0, T]$.
- Sharpe Ratio (ShR): ShR is usually defined as excess average returns divided by SD. Since interest rates on cash are assumed to be zero, we define ShR as follows.

$$ShR \equiv AR/SD, \quad AR \equiv \tilde{T}\overline{IR}. \tag{A.4}$$

Here, AR denotes the annualized arithmetic average of $\{IR_t\}$, which corresponds to a simple return.
- Sortino Ratio (SoR): SoR does not regard upside volatility as a risk while ShR penalizes both upside and downside volatility, which is often pointed out as a weakness of ShR.

$$SoR \equiv AR/DD. \tag{A.5}$$

Appendix B. Effectiveness of EMA-based investment

In this paper, we have introduced some technical indicators to improve investment performance of ANNs. Among the technical indicators, EMA is often used in prediction of financial time-series [43]. Then, we implement EMA-based investment strategies, which will illustrate the effectiveness of our ANN methodology. In particular, we prepare the following strategy (4) and (5), slight modifications of strategy (1) and (2), respectively. Let us remark that in the following analysis we use EMAs made from raw return data (R_t) for simplicity.

- Strategy (4):

$$\omega_t^{(4)} = \begin{cases} 1 & , \text{ if } EMA_{M,t} > 0, \\ 0 & , \text{ otherwise.} \end{cases} \tag{B.1}$$

Table C.1
Computational time for each case.

Model	Section	Seconds
Base case	3.1	163
Shallower layer	3.2	94
Deeper layer	3.2	247
Leaky ReLU	3.3	165
3-class output	3.4	160
5-class output	3.4	171
Return input and 4-class output	3.5	188
Return input and 3-class output	3.5	188
Return input and 5-class output	3.5	196
Additional technical input	3.6	166

- Strategy (5):

$$\omega_t^{(5)} = \begin{cases} 1 & \text{, if } EMA_{M,t} > 0, \\ -1 & \text{, otherwise.} \end{cases} \tag{B.2}$$

The investment performances are summarized in Table B.1. As shown in these tables, all the performances are extremely worse than the B&H strategy. Particularly, the asset value V_T reaches to nearly zero in strategy (5). Hence, it is obvious that our ANN approach substantially outperforms EMAs. In the case of intraday cryptocurrency data, EMAs do not seem to have the predictive ability by themselves.

Appendix C. Computational time

Table C.1 shows the computational time to train one ANN for each case appearing in Section 3.

References

[1] Satoshi Nakamoto, Bitcoin: A peer-to-peer electronic cash system, 2008.
[2] Marie Brière, Kim Oosterlinck, Ariane Szafarz, Virtual currency, tangible return: Portfolio diversification with bitcoin, J. Asset Manag. 16 (6) (2015) 365–373.
[3] Anne Haubo Dyhrberg, Bitcoin, gold and the dollar–A GARCH volatility analysis, Finance Res. Lett. 16 (2016) 85–92.
[4] KiHoon Hong, Bitcoin as an alternative investment vehicle, Inf. Technol. Manag. 18 (4) (2017) 265–275.
[5] Young Bin Kim, Jun Gi Kim, Wook Kim, Jae Ho Im, Tae Hyeong Kim, Shin Jin Kang, Chang Hun Kim, Predicting fluctuations in cryptocurrency transactions based on user comments and replies, PLoS One 11 (8) (2016) e0161197.
[6] David Kuo Chuen Lee, Li Guo, Yu Wang, Cryptocurrency: A new investment opportunity? J. Altern. Invest. 20 (3) (2017).
[7] Eugene F. Fama, The behavior of stock-market prices, J. Bus. 38 (1) (1965) 34–105.
[8] Eugene F. Fama, Efficient capital markets: A review of theory and empirical work, J. Finance 25 (2) (1970) 383–417.
[9] G. William Schwert, Anomalies and market efficiency, Handb. Econ. Finance 1 (2003) 939–974.
[10] Martín Iglesias Caride, Aurelio F. Bariviera, Laura Lanzarini, Stock returns forecast: An examination by means of artificial neural networks, in: Complex Systems: Solutions and Challenges in Economics, Management and Engineering, Springer, 2018, pp. 399–410.
[11] Laura Lanzarini, Juan Martın Iglesias Caride, Aurelio Fernández Bariviera, Are Technical Trading Rules Useful to Beat the Market? Evidence from the Brazilian Stock Market.
[12] Aurelio F Bariviera, María José Basgall, Waldo Hasperué, Marcelo Naiouf, Some stylized facts of the bitcoin market, Physica A 484 (2017) 82–90.
[13] Aurelio F. Bariviera, The inefficiency of bitcoin revisited: a dynamic approach, Econom. Lett. 161 (2017) 1–4.
[14] J. Alvarez-Ramirez, E. Rodriguez, C. Ibarra-Valdez, Long-range correlations and asymmetry in the bitcoin market, Physica A 492 (2018) 948–955.
[15] Stjepan Begušić, Zvonko Kostanjčar, H. Eugene Stanley, Boris Podobnik, Scaling properties of extreme price fluctuations in bitcoin markets, arXiv preprint arXiv:1803.08405.
[16] Ming-Wei Hsu, Stefan Lessmann, Ming-Chien Sung, Tiejun Ma, Johnnie E.V. Johnson, Bridging the divide in financial market forecasting: machine learners vs. financial economists, Expert Syst. Appl. 61 (2016) 215–234.
[17] George E.P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, Greta M. Ljung, Time Series Analysis: Forecasting and Control, John Wiley & Sons, 2015.
[18] Geoffrey E. Hinton, Simon Osindero, Yee-Whye Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[19] George Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314.
[20] Masaaki Fujii, Akihiko Takahashi, Masayuki Takahashi, Asymptotic Expansion as Prior Knowledge in Deep Learning Method for high dimensional BSDEs, 2017.
[21] E. Weinan, Jiequn Han, Arnulf Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (4) (2017) 349–380.
[22] Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, Ruben Gryp, Evaluating multiple classifiers for stock price direction prediction, Expert Syst. Appl. 42 (20) (2015) 7046–7056.
[23] Eunsuk Chong, Chulwoo Han, Frank C. Park, Deep learning networks for stock market analysis and prediction: methodology, data representations, and case studies, Expert Syst. Appl. 83 (2017) 187–205.
[24] Nicolas Huck, Pairs trading and outranking: The multi-step-ahead forecasting case, European J. Oper. Res. 207 (3) (2010) 1702–1716.
[25] Fagner A. de Oliveira, Cristiane N. Nobre, Luis E. Zárate, Applying Artificial Neural Networks to prediction of stock price and improvement of the directional prediction index–Case study of PETR4, Petrobras, Brazil, Expert Syst. Appl. 40 (18) (2013) 7596–7606.
[26] Rubén Aguilar-Rivera, Manuel Valenzuela-Rendón, J.J. Rodríguez-Ortiz, Genetic algorithms and Darwinian approaches in financial applications: A survey, Expert Syst. Appl. 42 (21) (2015) 7684–7697.
[27] Håvard Kvamme, Nikolai Sellereite, Kjersti Aas, Steffen Sjursen, Predicting mortgage default using convolutional neural networks, Expert Syst. Appl. 102 (2018) 207–217.

[28] Ming-Chi Lee, Using support vector machine with a hybrid feature selection method to the stock trend prediction, Expert Syst. Appl. 36 (8) (2009) 10896–10904.
[29] Masafumi Nakano, Akihiko Takahashi, Soichiro Takahashi, Generalized exponential moving average (EMA) model with particle filtering and anomaly detection, Expert Syst. Appl. 73 (2017) 187–200.
[30] Mualla Gonca Yunusoglu, Hasan Selim, A fuzzy rule based expert system for stock evaluation and portfolio construction: An application to Istanbul Stock Exchange, Expert Syst. Appl. 40 (3) (2013) 908–920.
[31] Yaohao Peng, Pedro Henrique Melo Albuquerque, Jader Martins Camboim de Sá, Ana Julia Akaishi Padula, Mariana Rosa Montenegro, The best of two worlds: forecasting high frequency volatility for cryptocurrencies and traditional currencies with support vector regression, Expert Syst. Appl. 97 (2018) 177–192.
[32] Xavier Glorot, Antoine Bordes, Yoshua Bengio, Deep sparse rectifier neural networks, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.
[33] Geoffrey E. Hinton, Ruslan R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.
[34] Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: Proc. icml, vol. 30, 2013, p. 3.
[35] Diederik P. Kingma, Jimmy Ba, Adam: a method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[36] J. Welles Wilder, New Concepts in Technical Trading Systems, Trend Research, 1978.
[37] Yoshua Bengio, Aaron Courville, Pascal Vincent, Representation learning: A review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798–1828.
[38] Johan Hastad, Almost optimal lower bounds for small depth circuits, in: Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing, ACM, 1986, pp. 6–20.
[39] Guido F. Montufar, Razvan Pascanu, Kyunghyun Cho, Yoshua Bengio, On the number of linear regions of deep neural networks, in: Advances in Neural Information Processing Systems, 2014, pp. 2924–2932.
[40] Mian Mian Lau, King Hann Lim, Investigation of activation functions in deep belief network, in: Control and Robotics Engineering (ICCRE), 2017 2nd International Conference on, IEEE, 2017, pp. 201–206.
[41] Florian Glaser, Kai Zimmermann, Martin Haferkorn, Moritz Weber, Michael Siering, Bitcoin-asset or currency? revealing users' hidden intentions, 2014.
[42] David Easley, Maureen O'Hara, Soumya Basu, From mining to markets: The evolution of bitcoin transaction fees, 2017.
[43] Masafumi Nakano, Akihiko Takahashi, Soichiro Takahashi, Creating investment scheme with state space modeling, Expert Syst. Appl. 81 (2017) 53–66.