Received: 16 July 2019 | Revised: 21 November 2019 | Accepted: 29 November 2019

DOI: 10.1111/exsy.12514

ORIGINAL ARTICLE

Expert Systems WILEY

Trend following deep Q-Learning strategy for stock trading

Jagdish Chakole | **Manish Kurhekar**

Department of Computer Science and Engineering, Visvesvaraya National Institute of Technology, Nagpur, Maharashtra, India

Correspondence
Jagdish Chakole, Department of Computer Science and Engineering, Visvesvaraya National Institute of Technology, Nagpur, Maharashtra, India.
Email: jagdish06@students.vnit.ac.in

Abstract

Computers and algorithms are widely used to help in stock market decision making. A few questions with regards to the profitability of algorithms for stock trading are can computers be trained to beat the markets? Can an algorithm take decisions for optimal profits? And so forth. In this research work, our objective is to answer some of these questions. We propose an algorithm using deep Q-Reinforcement Learning techniques to make trading decisions. Trading in stock markets involves potential risk because the price is affected by various uncertain events ranging from political influences to economic constraints. Models that trade using predictions may not always be profitable mainly due to the influence of various unknown factors in predicting the future stock price. Trend Following is a trading idea in which, trading decisions, like buying and selling, are taken purely according to the observed market trend. A stock trend can be up, down, or sideways. Trend Following does not predict the stock price but follows the reversals in the trend direction. A trend reversal can be used to trigger a buy or a sell of a certain stock. In this research paper, we describe a deep Q-Reinforcement Learning agent able to learn the Trend Following trading by getting rewarded for its trading decisions. Our results are based on experiments performed on the actual stock market data of the American and the Indian stock markets. The results indicate that the proposed model outperforms forecasting-based methods in terms of profitability. We also limit risk by confirming trading actions with the trend before actual trading.

KEYWORDS

deep learning, reinforcement learning, trend following

1 | INTRODUCTION

Trading actions or decisions that can be performed on a stock at any time are buy, sell, or hold (do nothing). The main objective of a trader is to get optimal profit by taking the right trading decision at the right time. Predicting a future price or trend of a stock and making trading decisions accordingly to get an optimal benefit is a challenging research problem. The stock market is volatile and noisy as the company stock price depends on many dynamic and uncertain factors such as the macroeconomics of the company, countries' geopolitical tensions, other competitor companies, and user sentiments about companies' products. Many research papers are available on the prediction of future stock prices or trends using various techniques and methods using machine-learning algorithms. Almost all of them claim that it is challenging to predict stock prices or trends with perfect accuracy. Trend Following (TF) is a trading strategy where the trading decision on a stock is made purely with the help of the current stock price trend based on the observed market data. A stock trend can be up, down, or sideways. Once the current trend of the stock is identified, TF dictates trading decisions using a predefined strategy. Here, the predefined plan is a rule-based mechanism that decides when to buy or sell a particular stock. All these decisions are based on the current stock trend rather than the future price. A trader takes a long position, that is, first buys the stock and later sells the stock, when she believes that the price trend will go upwards. She opens a short trade, that is, first sells the stock and then buys the stock when she believes that the price will have a downward trend.

Algorithmic Trading (AT) is a relatively new way of stock trading that allows algorithms (computers) to trade stocks. AT is defined as the use of a preprogrammed model that takes a trading decision and executes trading orders automatically without human intervention (Hendershott, Jones, and Menkveld (2011)). AT also increases the liquidity in the stock market (Hendershott et al. (2011)). In case of AT, a trader can back-test the trading strategy on historical data to check the profitability of her trading strategy. AT strategies are grouped into two categories: the strategies based on Machine Learning (ML), and those based on knowledge-based methods (Wang et al., 2016). Knowledge-based AT strategies are based on economic research, trader experience, and quantitative analysis. ML-based AT approaches first learn the trading strategy from historical trading data and then use the learned trading strategy to perform actual trading. So ML-based trading methods find trading strategies that are profitable at the current instance of time.

The authors (Shi, Da Xu, and Liu (1999)) proposed to improve the accuracy of three individual forecasting methods (autoregressive integrated moving average, Brown, and Trend) by using artificial neural networks. They experimented on IBM stocks; and a combined model using artificial neural networks outperformed the three individual forecasting methods. In (Jiang, Xu, Wang, and Wang (2009)), the authors find factors that influence the financial performance of companies listed in China using Genetic Algorithms. In (Polimenis and Neokosmidis (2014)), the authors perform the Granger causality test for the U.S. returns relative to the returns in the Asia-Pacific region to know the dependence of Asia-Pacific returns on the U.S. returns. It is claimed that AT is not capable of providing profit at every trading decision because of the highly complicated nature of the stock market (Yadav (2015)). It is also challenging to predict future trends with the help of AT (Hu et al. (2015)). It is found to be difficult to outperform the stock market when there exists a persistent uptrend or downtrend. To overcome these limitations, many researchers have used Reinforcement Learning (RL) to trade in the stock market (Jeong and Kim (2019)). RL methods are a subset of ML methods. In RL, an agent performs an action from the action space (i.e., the set of operations) on the environment (environment is in a specific state at any point of time, and that is referred to as current state), and in return, the agent gets a reward and changes the state of the environment. The main objective of the RL agent is to maximize cumulative rewards by taking a set of sequential actions on the environment till the desired state is reached (Sutton and Barto (1998)).

In RL, online learning is possible; an agent learns from its experiences. The agent adapts its policy (selection of action for the given state) according to the changing conditions of the environment. This adaptation is based on the reward for action (Wang et al., 2016). The RL method is suitable for AT because of the online learning and reward-based approach. Many researchers have been using RL for trading in the financial markets. Gao and Chan (2000) proposed a trading and portfolio management system that uses a Q-Learning technique and Sharpe ratio maximization algorithm. Sharpe ratio is used to compare the profit on stock investment with respect to the risk associated with the investment. The higher the value of the Sharpe ratio, lower is the risk. In (Moody, Wu, Liao, and Saffell (1998)), the authors proposed a trading system to optimize stock portfolios using the Recurrent RL (RRL) algorithm. The comparison for the performance of Q-Learning and RRL to optimize the portfolios is provided in (Du, Zhai, and Lv (2016)), and the authors concluded that RRL delivers superior results. The authors (Lee, Park, Jangmin, Lee, and Hong (2007)) proposed a model that incorporates multiple Q-Learning agents to provide adequate decision support for the daily stock trading problem. Moody and Saffell (2001) introduced a trading system based on direct RL to optimize stock trading. They used RRL to solve stochastic control problem of investment decisions. Gorse (2011) compares results of RRL and Genetic Programming obtained by applying these methods on daily, weekly, and monthly data of the stock. They observed that, RRL was better than the Buy-and-Hold approach for daily and weekly data, and Genetic Programming was best for monthly data. There are many other profitable traditional trading strategies such as pair trading and TF. Most of the approaches mentioned above only rely on a RL agent for trading strategy. They do not take advantage of the previously known profitable trading strategies. In our proposed model, the RL agent takes advantage of the TF trading strategy. As shown in Table 2, it outperforms other approaches.

Deep leaning also known as deep neural networks is a subset of machine-learning techniques. Deep learning uses artificial neural networks. It is called deep because its architecture uses a many layered neural network. Deep learning successfully influenced many areas of research such as speech recognition, computer vision, and natural language processing (LeCun, Bengio, and Hinton (2015)). Deep learning is capable of capturing high-level features from basic signals. Recently, Zhai et al. (2016) combined RL with deep learning to provide advantages of both, they called it deep RL. Deep Q-Learning is one of the deep RL methods that combines Q-Learning in RL with a deep neural network. It has been trained to learn Atari 2,600 games and the Go game (Mnih et al. (2015)). Deep learning has been successfully used to capture high-level features from basic signals (Lu (2019); Duan, Liu, Yu, Li, and Yeh (2019)). In (Alguliyev, Aliguliyev, and Abdullayeva (2019)) it is used to preserve privacy for big personal data analysis using a sparse denoising autoencoder-based deep neural network.

Advancement in internet technology has revolutionized financial services. Financial Technology is the result of the exploration of technology in financial services (Assarzadeh and Aberoumand (2018)). Stock exchanges are benefitted by the technological enhancement, and as a result, all major stock exchanges are operating online. This also has allowed large amounts of stock pricing data to be available for processing by computers. Our algorithm processes this stock pricing data for providing trading signals.

In this paper, our main objective is to propose a novel AT Strategy that combines the power of both approaches: TF trading strategy and deep Q-Learning. Our goal is to train the deep Q-Learning model to decide when to buy, sell, or hold a stock based on the current price and the historical price. The experimental results of the proposed trading model on the Indian stock market (National Stock Exchange of India and Bombay Stock

TABLE 1 Experimental Data of DJIA, NASDAQ, NIFTY and SENSEX index stocks

Stock name	Time	Total period	Training period	Testing period
DJIA	Start	January 2, 2001	January 2, 2001	January 3, 2006
	End	December 31, 2018	December 30, 2005	December 31, 2018
NASDAQ	Start	January 2, 2001	January 2, 2001	January 3, 2006
	End	December 31, 2018	December 30, 2005	December 31, 2018
NIFTY	Start	September 17, 2007	September 17, 2007	July 28, 2011
	End	December 31, 2018	July 27, 2011	December 31, 2018
SENSEX	Start	September 17, 2007	September 17, 2007	July 28, 2011
	End	December 31, 2018	July 27, 2011	December 31, 2018

Abbreviations: DJIA, Dow Jones Industrial Average; NASDAQ, National Association of Securities Dealers; NIFTY, National Stock Exchange Fifty; SENSEX, Sensitive Index.

TABLE 2 Comparison of trading performance on the test set for three trading strategies

Stock name	Performance evaluation metrics	Buy-and-Hold	Decision Tree	Proposed model
DJIA	Accumulated Return (%)	112.60	53.20	138.27
	Maximum Drawdown (%)	53.77	52.80	49.80
	Average daily return (%)	0.029	0.018	0.033
	Average annual return (%)	8.66	4.09	10.63
	Skewness (%)	0.069	0.334	0.114
	Kurtosis (%)	11.33	11.98	12.08
	Sharpe ratio (%)	1.51	1.00	1.68
	Standard Deviation (%)	1.12	1.06	1.10
NASDAQ	Accumulated Return (%)	193.46	85.85	399.17
	Maximum Drawdown (%)	55.62	39.68	23.62
	Average daily return (%)	0.041	0.018	0.054
	Average annual return (%)	14.88	6.60	30.70
	Skewness (%)	−0.102	0.334	−0.156
	Kurtosis (%)	7.46	11.98	6.902
	Sharpe ratio (%)	1.24	1.00	2.93
	Standard Deviation (%)	1.31	1.06	1.07
NIFTY	Accumulated Return (%)	97.89	86.57	136.44
	Maximum Drawdown (%)	22.52	22.44	15.88
	Average daily return (%)	0.042	0.038	0.051
	Average annual return (%)	13.62	12.05	18.99
	Skewness (%)	−0.161	−0.333	0.032
	Kurtosis (%)	1.96	2.45	2.25
	Sharpe ratio (%)	1.87	1.84	2.50
	Standard Deviation (%)	0.95	0.88	0.87
SENSEX	Accumulated Return (%)	98.12	98.92	152.43
	Maximum Drawdown (%)	22.67	25.49	22.58
	Average daily return (%)	0.042	0.041	0.055
	Average annual return (%)	18.72	18.87	29.07
	Skewness (%)	−0.160	−0.145	−0.145
	Kurtosis (%)	1.99	2.25	2.09
	Sharpe ratio (%)	1.90	1.99	2.53
	Standard Deviation (%)	0.94	0.89	0.93

Abbreviations: DJIA, Dow Jones Industrial Average; NASDAQ, National Association of Securities Dealers; NIFTY, National Stock Exchange Fifty; SENSEX, Sensitive Index.

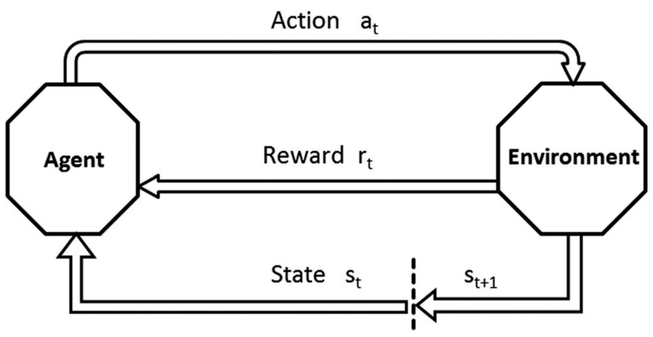

FIGURE 1 Typical reinforcement learning system

Exchange) and the US stock market [Dow Jones Industrial Average (DJIA) and National Association of Securities Dealers Automated Quotations (NASDAQ)] indicate that it is fairly optimal in terms of profitability. The remaining parts of this paper are organized as follows: Section 2 introduces in detail the deep Q-Learning approach and the TF Strategy. Section 3 presents the implementation details of our proposed trading system. Section 4 contains the experimental results, which include the performance of the proposed model, and Section 5 concludes this paper and discusses future directions.

2 | METHODOLOGY

This section gives a brief introduction about RL and its techniques Q-Learning and deep Q-Learning. This section also discusses the TF approach to stock trading.

2.1 | Reinforcement Learning

In a stochastic process, the output of any state depends on the corresponding probability. A Markov process is a stochastic process in which the next state depends only on the current state. RL is an example of the Markov process. A RL system consists of a learning Agent and its Environment, as shown in Figure 1. In RL, the state s represents the environment by values of the selected features at that instant of time. At each time step t, the environment is in a certain state s_t. The RL agent interacts with the environment by taking action a_t, and in response, the state of the environment changes to s_{t+1}. It is well known that the exact prediction of the future price of the stock from past prices is not possible, but the future price of the stock still depends on the current price. Thus, as RL and the price of stock both follow the Markov process, it is intuitive to use RL in trading stocks. The RL agent takes action a_t on the environment at time t from a set of actions A by observing current state s_t of environment. The choice of action is based on the control policy π of the agent. Control policy π decides which action should be invoked for a certain state. In response, the agent receives reward r_t, and the environment enters the next state s_{t+1}. Based on the rewards received, the agent updates its control policy π, that is, with experience, it improves its control policy π. The agent performs a sequence of actions until it reaches the terminal or the final state. The agent's objective is to maximize its cumulative rewards across the episode. An episode consists of a sequence of state changes from the start state to the final state, and this is the typical life cycle of most of the RL algorithms.

Q-Learning is a RL algorithm. In this algorithm, to provide the best action for the state, the agent maintains a state-action pair value table known as q-table. Construction of q-table for continuous state space is generally not feasible. This problem is addressed by deep Q-Learning, a variant of Q-Learning. In deep Q-Learning, the q-table is replaced by nonlinear function approximator such as a deep neural network.

2.2 | Q-learning

The RL agent's objective is to get maximal accumulated reward R_t given by

$$R_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \gamma^3 \cdot r_{t+3} + \cdots \tag{1}$$

$$R_t = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k}. \tag{2}$$

Discount factor $\gamma \in [0, 1]$ is used to discount rewards in the future, and to avoid infinite return in cyclic Markov processes. r_t is the reward at time t that can be positive or negative. The reward function can be appropriately defined to achieve the goal of the agent. The total rewards in Q-Learning are formulated as a state-action value function $Q(s, a)$ that indicates the expected accumulated reward while performing an action a on the current state s of the environment and then following the policy π. This is formulated as follows:

$$Q_\pi(s,a) = \mathbb{E}\ [(R_t | s_t = s, a_t = a, \pi)]. \tag{3}$$

The objective of the Q-Learning algorithm is achieved by finding optimal control policy π^* that gives maximum expected accumulated rewards $Q^*(s, a)$.

$$Q^*(s,a) = \max_\pi\ Q_\pi(s,a) \tag{4}$$

The above Q-function is governed by the Bellman equation:

$$Q^*(s,a) = r_t + \gamma \cdot \max_{a'}\ Q_\pi(s', a'), \tag{5}$$

where r_t indicates immediate reward and new state $s^{'}$ is obtained by taking action a on state s. In Q-Learning, the Q-value $Q(s, a)$ is updated by performing the iterative update for each state-action pair (s, a) using the following equation, where α is the learning rate.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha\left[r_t + \gamma \cdot \max_a\ Q(s_{t+1},a) - Q(s_t,a_t)\right] \tag{6}$$

Optimal Q-function $Q^*(s, a)$ is obtained as a result of convergence of this iterative update.

2.3 | Deep Q-Learning (Q-Learning with deep neural networks)

It is computationally not feasible to build optimal Q-table when state space and action space are large (Mnih et al. (2015)). This computational problem for large state-action space is alleviated by using function approximator for state-action trade off selection such as a neural network, $Q(s, a) = Q(s, a, \theta)$ as shown in Figure 2 where θ is the neural network for function approximation. Neural network (θ) will provide the q-values of all action for the state. For state-action pair (s_t, a_t), $Q(s_t, a_t;\theta)$ is the predicted q-value, and $\left[r_t + \gamma \cdot \max_a Q(s_{t+1}, a;\theta)\right]$ is the target q-value for the Neural network. So, state-action value updation for deep Q-Learning will be done by the following equation.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha\left[r_t + \gamma \cdot \max_a\ Q(s_{t+1},a;\theta) - Q(s_t,a_t;\theta)\right] \tag{7}$$

As the same neural network (θ) is used for target value calculation and prediction, it will create learning instability. Learning instability created by combining Q-Learning algorithms with a deep neural network is alleviated by deep Q-Networks (DQNs) by two novel approaches. First is the experience replay mechanism used by DQNs, in which the agent at each time-step t stores transitions tuple $e_t = (s_t, a_t, r_t, s_{t+1})$ consisting of state

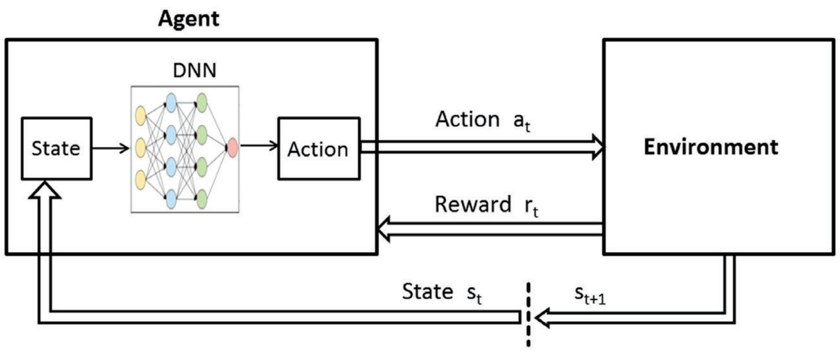

FIGURE 2 Deep Q-learning. DNN, deep neural network

s_t at time t, action a_t on state s_t, reward received r_t, and the next state s_{t+1} into a buffer memory $D = e_1, e_2, ..., e_t$; and then samples from memory are taken uniformly for training.

Second is the use of two different DQN $Q(s, a;\theta)$ and $Q(s, a;\theta^{'})$, one for prediction and the other for the target network, where θ and $\theta^{'}$ are present and immediate previous parameters, respectively. The target network is same as the neural network used for prediction, but its parameters are frozen, and after every fix steps, its parameters are copied from parameters of neural prediction network. DQN is updated iteratively using the Bellman equation, and the mean-square-error method is used to minimize error while updating the model. Present DQN parameter θ is updated with respect to previous DQN parameter $\theta^{'}$, by reducing loss as shown below.

$$L_i(\theta_i) = \mathbb{E}\left[\left(r + \gamma \cdot \max_{a'} Q(s', a'; \theta_{i'}) - Q(s, a; \theta_i)\right)^2\right] \tag{8}$$

We get a gradient as given below by differentiating above loss function with respect to current DQN parameters:

$$\triangle_{\theta_i} L_i(\theta_i) = \left(r + \gamma \cdot \max_{a'} Q(s', a'; \theta_{i'}) - Q(s, a; \theta_i)\right) \triangle_{\theta_i} Q(s, a; \theta). \tag{9}$$

DQN uses a greedy approach to find optimal Q-value, $a = \max_a Q(s, a; \theta)$, so, it selects action a that maximizes the Q-value. The exploration and exploitation trade off for action selection is based on ϵ-greedy strategy. It selects random action with probability ϵ and selects action as per $Q(s, a;\theta)$ with probability $1 - \epsilon$. Initially, the value of ϵ is maximum, and it decreases with the number of iterations.

2.4 | Trend Following

TF is a trading strategy where trading decisions like buying and selling are taken purely according to the observed stock price trends. The TF approach is to go with a trend. TF does not forecast the stock price. Instead, TF identifies the current trend and momentum of the stock and the market. Once TF identifies the trend, trades are performed as per a predefined strategy. The TF approach is to continuously observe the trend; if the trend swings, then swap the trading positions. A trend can be computed using technical analysis indicators like the Relative Strength Index (RSI; Fong, Si, and Tai (2012)), Commodity Channel Index (CCI), and so forth. These technical analysis indicators are discussed in Section 2.5. The TF method continuously monitors the current trend or momentum. If the trend or momentum is upwards, then it is estimated that stock price will increase, so take a long position, and close the previous short position if any. Also, it will hold any prior long positions. On the other hand, if the trend or momentum is downwards, then it is estimated that stock price will decrease, so take a short position, and close the previous long position if any. Also, it will hold any prior short positions.

2.5 | Technical analysis of stocks

Traditionally, in trading or investing in the stock market, two tools are used to make trading decisions: technical analysis and fundamental analysis. The fundamental analysis is useful for long term investing to know about the company's financial condition, management, revenue, and expected growth. The technical analysis is useful for short-term traders having holding duration ranging from 1 minute, 1 hr, 1 day or 1 week, or at most a few weeks. The technical analysis provides indicators that indicate the current price trend, momentum, and so forth. In our proposed models, we have used three momentum indicators which are the RSI, CCI, and Williams Percent Range (%R).

2.5.1 | Relative Strength Index (RSI)

RSI is a momentum indicator. It provides an overbought or oversold signal (Kumar & Thenmozhi, 2006). This indicator's value ranges from 0 to 100. If the value is below 30, it signals oversold, and if the value is above 70, it signals overbought. RSI is calculated from 14 previous trading sessions by using the formula given in Equation (10).

$$RSI = 100 - \frac{100}{1 + \frac{average\_gain}{average\_loss}} \tag{10}$$

2.5.2 | Commodity Channel Index (CCI)

CCI is an oscillating momentum indicator. It also provides an overbought or oversold signal (Kumar & Thenmozhi, 2006). It provides an overbought signal when the value is above +100 and an oversold signal when the value is below −100. CCI is calculated using the formula given in Equation (11).

$$CCI_t = \left[\frac{1}{0.015}\right] * \frac{A_t - Simple\_Moving\_Average(A_t)}{Mean\_Absolute\_Deviation(A_t)} \tag{11}$$

$$Where, A_t = \frac{Low\_price_t + High\_price_t + Close\_price_t}{3}$$

2.5.3 | Williams Percent Range (%R)

This is also a momentum indicator that provides an overbought and oversold signal. Its value ranges from 0 to −100. It signals overbought when the value is above −20 whereas it signals oversold when the value is below −80. Generally, %R calculations are based on the previous 14 sessions (i.e., $n = 14$) using a formula given in Equation (12).

$$\%R_t = \frac{Highest\ High(n\,days) - Close_t}{Highest\ High(n\,days) - Lowest\ Low(n\ days)} \tag{12}$$

3 | PROPOSED TRADING SYSTEM

In this section, we propose an AT system using deep Q-Learning and TF approach. Our system operates on a single stock for simplicity. Also, only one trading action is allowed on each trading day t. The trading decision a_t that can be taken on each trading day can be hold (0), long (1), or short (−1); and it gives reward r_t. The goal of the trading agent is to learn state-action value function $Q(s, a)$ using a deep Q-function that maximizes accumulated profits in the long term. While calculating rewards, action taken in favour of the trend is rewarded more. Trading charges were included in this study. The trading or brokerage is considered to be 0.1% of the trading amount on both sides, buy or sell.

As shown in Figure 3, the deep learning module uses a state-action value $Q(s, a)$ function approximator. It provides the best action for the given state as per the control policy. As the stock market is volatile, our trading decision is based on both, the deep learning module and the current trend of the market provided by the technical analysis module. If the deep learning module signals a buy action, then the trend is observed, and if the trend is also in favour of buy, then a buy action is performed. The agent is positively rewarded if its trading action is according to the trend. Thus, the agent is implicitly trained to follow the trend. We use the RSI, CCI, and %R indicators to know the trend that is calculated from

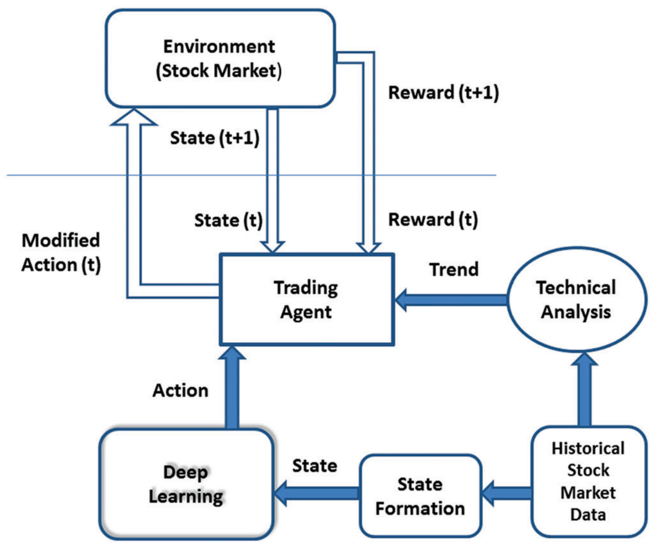

FIGURE 3 Proposed trading system

the current and the historical stock data. Suppose the current price of a stock is 11,500 and the RL agent gives a buy signal, then the trend is checked; and if the trend is also in favour of a buy, then that stock is bought. If after some days the price of the stock increases to 11,600, and if RL agent gives a sell signal and trend is also in favour of a sell, then that stock is sold. The RL agent gets 100 = (11,600 – 11,500) as positive reward. Instead, if the price of the stock decreases to 11,400 in the same scenario, the RL agent gets 200 = (11,600 – 11,400) as a negative reward.

One cannot guarantee an algorithm's performance on stock returns. This is because the stock market reacts to many different factors including economic and political conditions. Online learning of the trading agent is required to quickly adapt to changing market conditions. In the proposed model, online learning is very convenient. Initially, we train the deep Learning model with historical data. On every trading day, a new sample is added to the training buffer (s_t, a_t, r_t, s_{t+1}). Thus, the model improves with every trading day. Two separate DQNs are used to avoid learning instability. The target Q-value is calculated using an old DQN $Q(s, a;\theta')$, and predicted Q-value is calculated using a current DQN $Q(s, a;\theta)$. The state of the environment consists of the previous 50-days' stock closing price as state $s = (C_t, C_{t-1}, C_{t-2}, .., C_{t-49})$, where C_t is closing price of stock at day t. Our experimental results indicate that a state size of 50 days gives optimal results.

4 | EXPERIMENTAL RESULTS

In this section, experimental details such as experimental data, experimental scheme, performance evaluation metrics, and results are discussed.

4.1 | Data

To verify our proposed model, we conducted various experiments on the American and the Indian stock market data. Two index stocks DJIA and NASDAQ are from the USA. Two index stocks National Stock Exchange Fifty (NIFTY) and Sensitive Index (SENSEX) are from India. The DJIA is the index stock, and it is a representative of 30 companies trading on the two American stock exchanges, New York Stock Exchange and the NASDAQ. The NASDAQ index is the index of the NASDAQ stock exchange. The NIFTY is the index of National Stock Exchange, and it is representative of 50 different stocks. The SENSEX is the index of Bombay Stock Exchange, and it is a representation of the top 30 companies on Bombay Stock Exchange.

Stock market data has been accessed from Yahoo! Finance. Missing values of trading days were borrowed from the values of the previous days. Data then is divided into training and testing, as shown in Table 1. The American stock market index DJIA and NASDAQ data set are as shown in Figure 4. The Indian stock market index NIFTY and SENSEX data set are as shown in Figure 5. The code of the proposed model in the Python programming language and the experimental data is available at the URL: https://sites.google.com/view/jagdishbchakole/.

4.2 | Experimental scheme

To perform buy or sell operation in the stock market, the trader has to pay transaction cost which includes brokerage, stamp duty, stock exchange charges, and so forth. The transaction charge is considered to be 0.10% of the total trading amount on both sides, buy or sell. The risk management tool (stop loss) is not included in this study for a fair comparison of the proposed model with the Buy-and-Hold trading strategy.

4.3 | Performance evaluation metrics

The performance of the proposed model is compared with two trading methods, the Buy-and-Hold trading strategy and trading strategy using the Decision Tree method using the performance evaluation metrics as explained below.

4.3.1 | Accumulated Return

Accumulated Return is the change in the value of initial investment compared to the current value of the investment. The value of the Accumulated Return should be positive and as large as possible. The percentage Accumulated Return is calculated using the formula given in Equation (13).

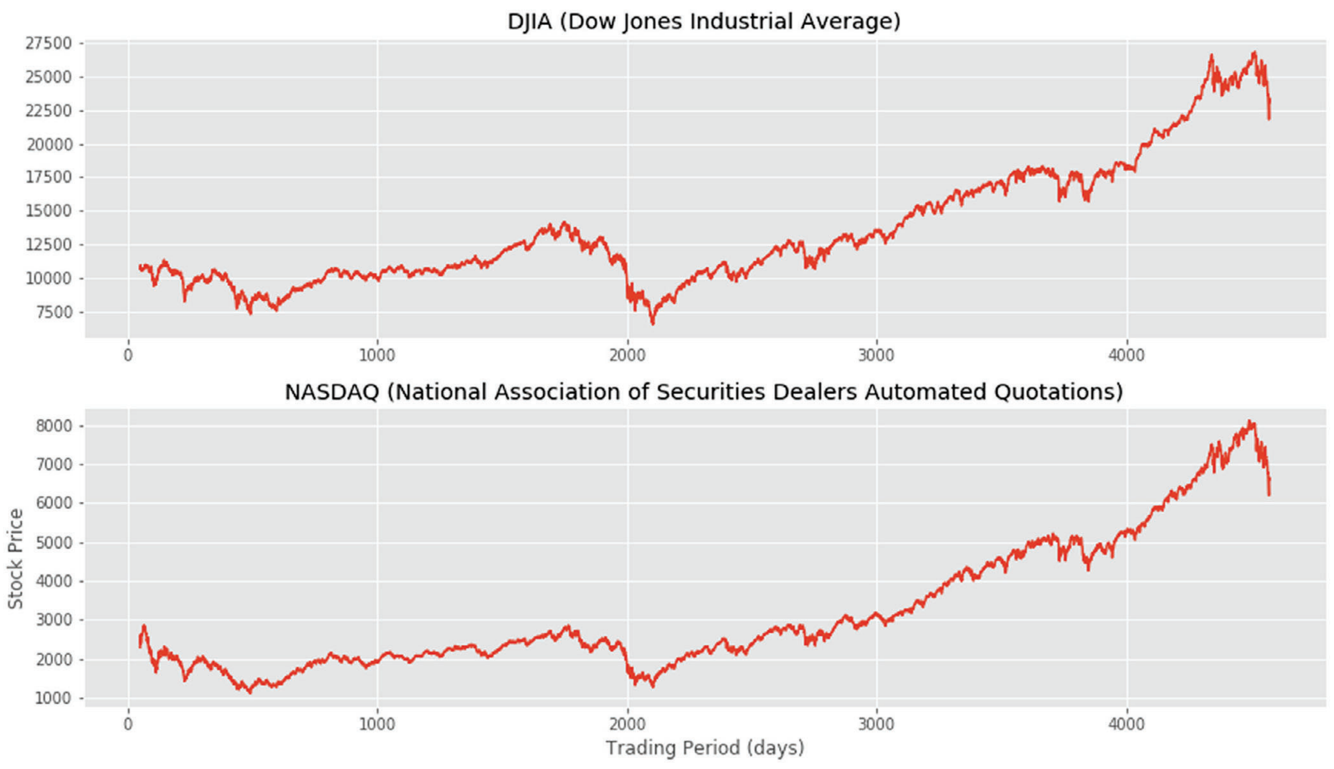

FIGURE 4 Daily close price series for DJIA and NASDAQ. First 1,305 data are used as training data set and remaining as testing data set

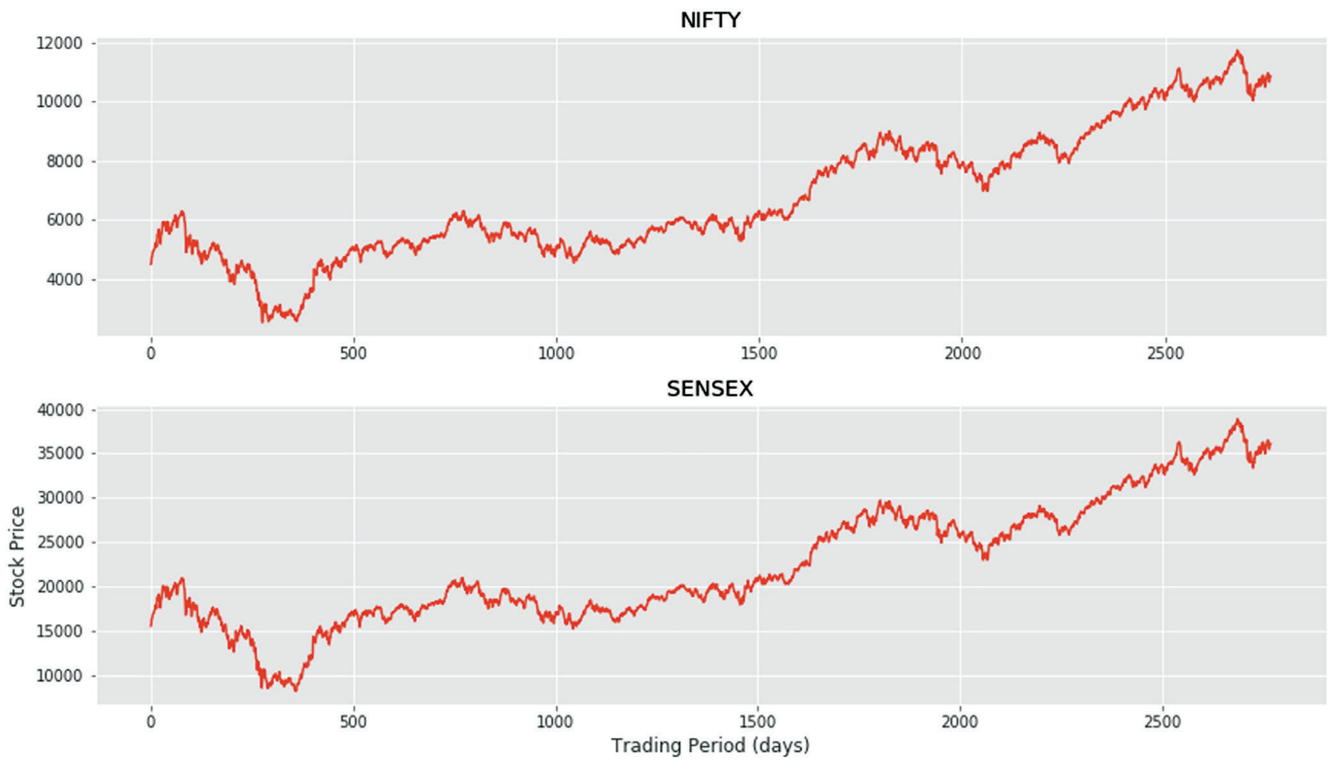

FIGURE 5 Daily close price series for NIFTY and SENSEX. First 950 data are used as training data set and remaining as testing data set. NIFTY, National Stock Exchange Fifty; SENSEX, Sensitive Index

$$\%Accumulated\ Return = \frac{Current\ value\ of\ Initial\ investment - Initial\ investment}{Initial\ investment} * 100 \quad (13)$$

4.3.2 | Maximum Drawdown

A drawdown is a tool used to measure risk in the investment and is calculated using Equation (14).

$$\%Drawdown_t = \frac{P_{max} - P_t}{P_{max}} * 100, \quad (14)$$

where P_t is the value of the portfolio at time t and P_{max} is the historically high value of the portfolio. Maximum Drawdown is the maximum value of drawdown up to time *t* and is calculated using the Equation (15).

$$MDD_t = \max_{0 \le q \le t} (\%Drawdown_q) \quad (15)$$

Ideally, the value of the Maximum Drawdown should be as small as possible. It is used to compare the risk associated with two or more trading strategies.

4.3.3 | Average daily return

Average daily return of the investment or stock is an average of the daily return for the given period. The daily return of the investment is calculated using the formula mentioned in Equation (16).

$$\%Daily\ Return = \frac{Value_t - Value_{t-1}}{Value_{t-1}} * 100 \quad (16)$$

4.3.4 | Average annual return

The average annual return is the average of yearly return for some number of years. We calculate it as total return divided by the number of years. The optimal value of this metric is positive and as large as possible.

4.3.5 | Sharpe ratio

A Sharpe ratio is used to understand the return on portfolio compared to its risk and is calculated by using Equation (17). The Sharpe ratio is used to compare risk associated with different trading strategies, and the strategy with the highest Sharpe ratio has the least risk as compared to the other trading strategies.

$$Sharpe\ ratio = \frac{R_p - R_{Risk\ free}}{\sigma_p}, \quad (17)$$

where R_p is expected return on portfolio $R_{Risk\ free}$ is risk free rate σ_p is the Standard Deviation of the portfolio's excess return.

4.3.6 | Standard Deviation

Standard Deviation is used by investors to measure volatility associated with the stock return. The larger value of the Standard Deviation indicates more volatile stock.

4.3.7 | Skewness and Kurtosis

The Skewness measures the symmetry or asymmetry in the distribution. The perfect symmetric distribution has zero Skewness. The data is skewed if Skewness is below −1 or above 1. Kurtosis is used to measure the degree to which the value of variable varies below or above the mean. The value of Kurtosis above 3 indicates a large variation about the mean.

4.4 | Experimental results

We performed experiments using the proposed model on two index stocks of the Indian stock market and two index stocks of the American stock market. We compared the performance of the proposed model with the traditional Buy-and-Hold strategy and the Decision Tree with a

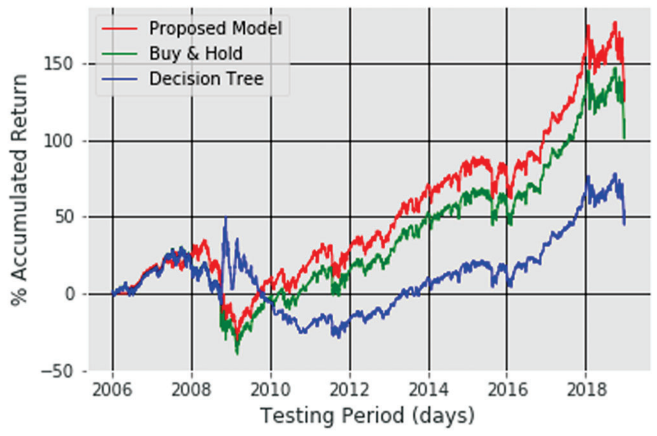

FIGURE 6 Performance of the different trading strategies on Dow Jones Industrial Average index stock on the test set in terms of percentage Accumulated Return

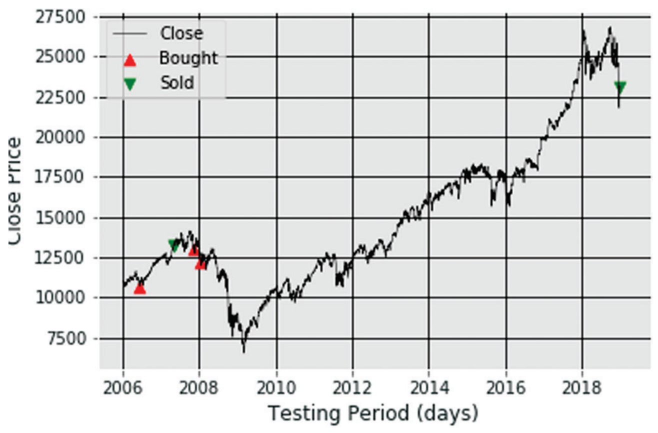

FIGURE 7 Trading decisions taken by the proposed trading system on Dow Jones Industrial Average index stock on the test set

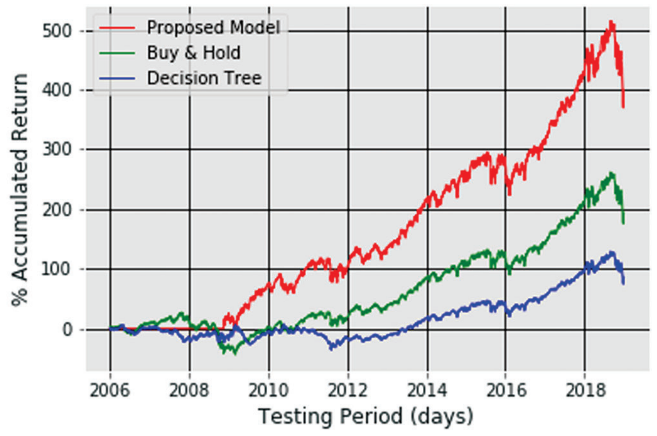

FIGURE 8 Performance of the different trading strategies on National Association of Securities Dealers Automated Quotations index stock on the test set in terms of percentage Accumulated Return

supervised Learning method. For the fair comparison, all these three methods were tested on the same stocks for the same period. The transaction cost is included in this study, and risk control measure, such as stop loss, is not included for a fair comparison with the Buy-and-Hold strategy.

Figures 6, 8, 10, and 12, show the performance of these methods on index stocks DJIA, NASDAQ, NIFTY, and SENSEX, respectively, in terms of Accumulated Return. Figures 7, 9, 11, and 13 depict the different trading positions held by the proposed model at different times on index stocks DJIA, NASDAQ, NIFTY, and SENSEX, respectively.

The proposed model outperformed both the Decision Tree strategy as well as the Buy-and-Hold strategy in terms of the Accumulated Return on the test data set. Table 2 shows a detailed comparison of these models, using parameters for comparing the performance over Accumulated

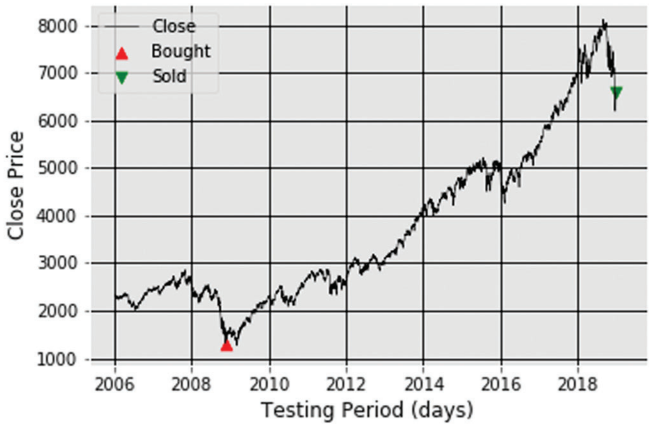

FIGURE 9 Trading decisions taken by the proposed trading system on National Association of Securities Dealers Automated Quotations index stock on the test set

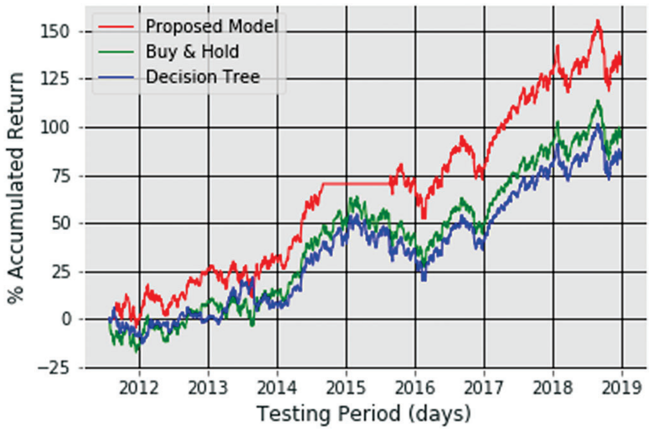

FIGURE 10 Performance of the different trading strategies on National Stock Exchange Fifty index stock on the test set in terms of percentage Accumulated Return

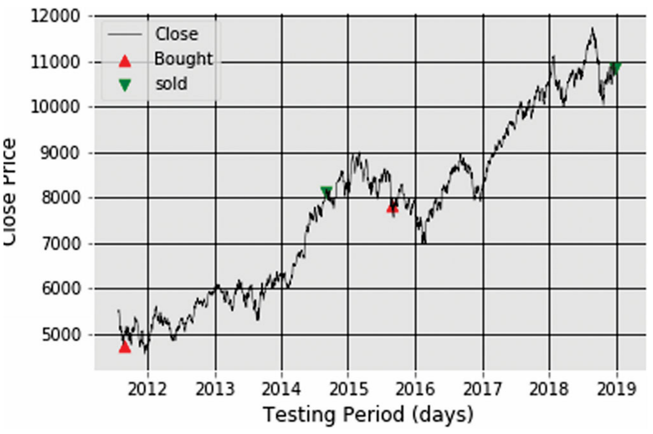

FIGURE 11 Trading decisions taken by the proposed trading system on National Stock Exchange Fifty index stock on the test set

Return, Maximum Drawdown, Average daily return, average annual return, Skewness, Kurtosis, Sharpe ratio, and Standard Deviation. Accumulated Return is the profit or loss with respect to the initial invested amount. Maximum Drawdown is the measure of historical risk on investment. The Maximum Drawdown is the difference between peak and trough value of the investment for a specific period. The proposed trading system performed better than the Decision Tree method as well as the Buy-and-Hold strategy in terms of various performance evaluation metrics as shown in Table 2. The comparison of the proposed model, the Buy-and-Hold method, and the trading strategy using the Decision Tree in terms of the percentage Accumulated Return is depicted in Figure 14.

The results shown in Table 2 conclude that the Accumulated Return of the proposed model is 22.79 and 159.90% more than the Buy-and-Hold and the Decision Tree strategy, respectively, on the test set of the stock DJIA. Similarly, it is 106.33 and 364.96%; 39.38 and 57.60%; 55.35

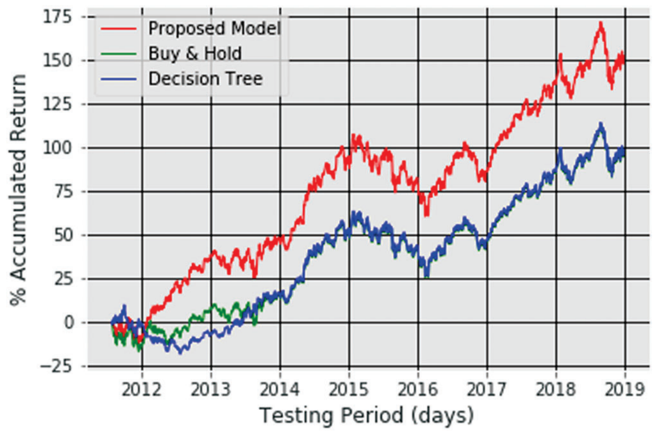

FIGURE 12 Performance of the different trading strategies on Sensitive Index index stock on the test set in terms of percentage Accumulated Return

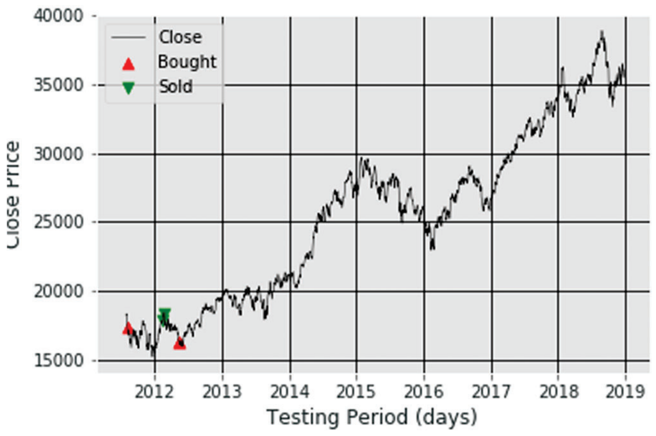

FIGURE 13 Trading decisions taken by the proposed trading system on Sensitive Index index stock on the test set

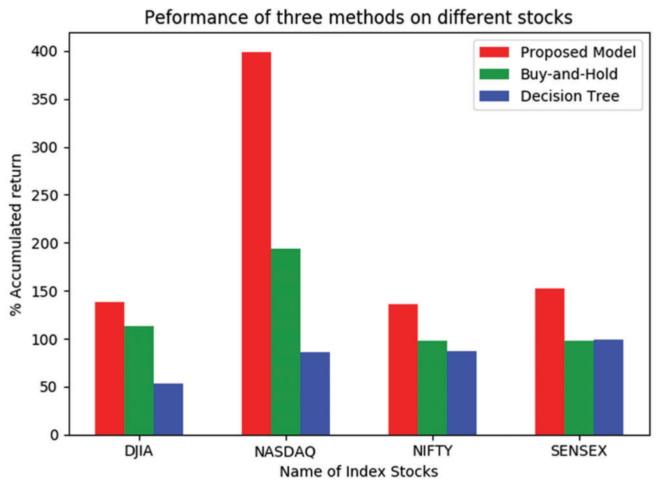

FIGURE 14 Performance of three methods on four index stocks on the test set. DJIA, Dow Jones Industrial Average; NASDAQ, National Association of Securities Dealers Automated Quotations; NIFTY, National Stock Exchange Fifty; SENSEX, Sensitive Index

TABLE 3 *T* values of the Student's *t* test between Proposed Model and Buy-and-Hold; Proposed Model and Decision Tree on four index stocks

Stock name	Proposed model and Buy-and-Hold	Proposed model and Decision Tree
DJIA	12.8	42.3
NASDAQ	34.4	55.9
NIFTY	18.6	24.1
SENSEX	27.4	27.3

Abbreviations: DJIA, Dow Jones Industrial Average; NASDAQ, National Association of Securities Dealers; NIFTY, National Stock Exchange Fifty; SENSEX, Sensitive Index.

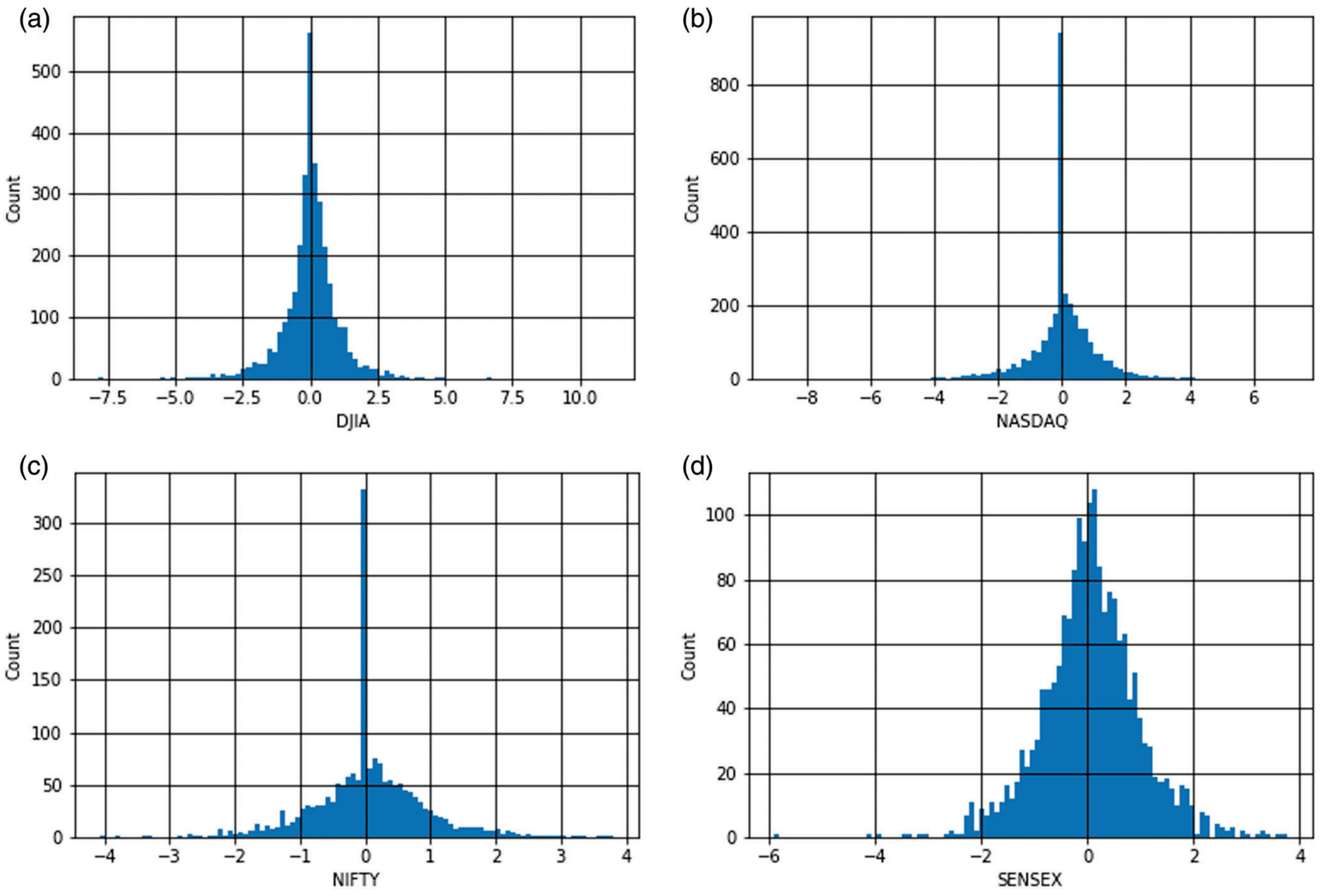

FIGURE 15 Histogram of daily percentage change (percentage daily return) of the proposed model on four index stocks test data set. Histogram of the (a) Dow Jones Industrial Average (DJIA), (b) National Association of Securities Dealers (NASDAQ), (c) National Stock Exchange Fifty (NIFTY), and (d) Sensitive Index (SENSEX) index stocks

and 54.09% more on the test set of the index stocks NASDAQ, NIFTY, and SENSEX respectively. This indicates that the proposed model outperformed the remaining two methods on all test sets in terms of Accumulated Return.

The Maximum Drawdown should be minimum and Table 2 indicates that the Maximum Drawdown of the proposed model is considerably less as compared to the Maximum Drawdown of the remaining two methods on all test sets except on the index stock SENSEX.

The percentage average daily return and percentage average annual return of the proposed model outperforms the rest. The Skewness of the proposed model on all test sets is fairly acceptable. The Kurtosis of the proposed model is more on the American stocks compared to the Indian stocks that indicated that the American stocks can get more variation about mean return as compare to the Indian stocks. The Sharpe ratio of the proposed model is better than the rest two methods on all test data sets except the NIFTY index stock where the Sharpe ratio of the Decision Tree is better. On all test sets, the Standard Deviation of the proposed model is less than the Standard Deviation of the Buy-and-Hold methods that indicates it is less volatile.

The Accumulated Return of the proposed model are comfortably better than the Buy-and-Hold and Decision Tree strategy as reported in Table 2. To check whether the result in terms of the Accumulated Return of the proposed model is statistically different from the Buy-and-Hold and the Decision Tree strategy, we performed the Student's *t* test between the proposed model and the Buy-and-Hold strategy and the proposed model and Decision Tree strategy for Accumulated Return on all four index stocks on test data set. The results of the Student's *t* test are reported in Table 3, and all the *t* values are greater than the 5% level of significance, concluding that the results are statistically different.

The histogram of daily percentage return by the proposed model on the test data set of all four index stocks is shown in Figure 15. Also, to check the normality of daily percentage return by the proposed model statistically, we performed the Shapiro–Wilk test. The result of the test indicates that daily percentage return by the proposed model on the test data set of all four index stocks are not normally distributed.

5 | CONCLUSIONS

In this paper, we have proposed training a RL agent to obey the TF approach. The RL agent receives rewards based on the performance of its trading decisions. The RL agent can only take a trading decision if trend information provided by the TF approach is in favour of the trading decision. As the stock market is volatile and unpredictable, risk control becomes a significant factor. The TF approach controls the risk as it aligns with a trend once it is identified. The RL agent reinforces with experience by observing rewards. We provided rewards in a way that are consistent with the TF approach. We compared our proposed model with Decision Tree and Buy-and-Hold approaches, and it outperforms both these models in terms of percentage Accumulated Return, Maximum Drawdown, Average return, and the Sharpe ratio. Comparatively, it is less volatile; and the Skewness and Kurtosis on the proposed model are fairly acceptable. The performance of the proposed model is best for the NASDAQ. We used deep Q-Learning method to train the RL agent. More RL methods can be explored to improve performance in the future. Similarly, along with RSI, CCI, and %R, some more Trend Following tools can be used in the future. The experimental results show that the performance of the proposed model is best on the NASDAQ index stock of the U.S. stock market, which is mostly comprised of technology companies. Thus, in the future, it can be investigated whether this model is better suited for technology stocks. Our proposed approach gives above-average returns if the market has a particular trend, but if the market is sidewise, then its performance is average. This approach can further be improved by including more information in the state of the environment. The code and experimental data of the model are publicly available at the URL mentioned in Section 4.1.

ORCID

Jagdish Chakole https://orcid.org/0000-0003-0242-7297

REFERENCES

Alguliyev, R. M., Aliguliyev, R. M., & Abdullayeva, F. J. (2019). Privacy-preserving deep learning algorithm for big personal data analysis. *Journal of Industrial Information Integration*, *15*, 1–14.

Assarzadeh, A. H., & Aberoumand, S. (2018). Fintech in Western Asia: Case of Iran. *Journal of Industrial Integration and Management*, *3*(03), 1850,006.

Du, X., Zhai, J., & Lv, K. (2016). Algorithm trading using q-learning and recurrent reinforcement learning. *Positions*, *1*, 1.

Duan, N., Liu, L.-Z., Yu, X.-J., Li, Q., & Yeh, S.-C. (2019). Classification of multichannel surface-electromyography signals based on convolutional neural networks. *Journal of Industrial Information Integration*, *15*, 201–206.

Fong, S., Si, Y.-W., & Tai, J. (2012). Trend following algorithms in automated derivatives market trading. *Expert Systems with Applications*, *39*(13), 11,378–11,390.

Gao, X., & Chan, L. (2000). An algorithm for trading and portfolio management using q-learning and Sharpe ratio maximization. *Proceedings of the International Conference on Neural Information Processing*, 832–837.

Gorse, D. (2011). Application of stochastic recurrent reinforcement learning to index trading. *ESANN 2011 proceedings, 19th European symposium on artificial neural networks, computational intelligence and machine learning (2010)*, 123–128.

Hendershott, T., Jones, C. M., & Menkveld, A. J. (2011). Does algorithmic trading improve liquidity? *The Journal of Finance*, *66*(1), 1–33.

Hu, Y., Liu, K., Zhang, X., Su, L., Ngai, E., & Liu, M. (2015). Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing*, *36*, 534–551.

Jeong, G., & Kim, H. Y. (2019). Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, *117*, 125–138.

Jiang, Y., Xu, L., Wang, H., & Wang, H. (2009). Influencing factors for predicting financial performance based on genetic algorithms. *Systems Research and Behavioral Science: The Official Journal of the International Federation for Systems Research*, *26*(6), 661–673.

Kumar, M., and M. Thenmozhi (2006), Forecasting stock index movement: A comparison of support vector machines and random forest. Paper presented at: Indian Institute of Capital Markets 9th Capital Markets Conference Paper.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.

Lee, J. W., Park, J., Jangmin, O., Lee, J., & Hong, E. (2007). A multiagent approach to *q*-learning for daily stock trading. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *37*(6), 864–877.

Lu, Y. (2019). Artificial intelligence: A survey on evolution, models, applications and future trends. *Journal of Management Analytics*, *6*(1), 1–29.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529.

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, *12*(4), 875–889.

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, *17*(5–6), 441–470.

Polimenis, V., & Neokosmidis, I. (2014). The global financial crisis and its transmission to asia pacific. *Journal of Management Analytics*, *1*(4), 266–284.

Shi, S. M., Da Xu, L., & Liu, B. (1999). Improving the accuracy of nonlinear combined forecasting using neural networks. *Expert Systems with Applications*, *16*(1), 49–54.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction mit press*. Cambridge: MA.

Wang, Y., D. Wang, S. Zhang, Y. Feng, S. Li, and Q. Zhou (2016), Deep q-Trading.

Yadav, Y. (2015). How algorithmic trading undermines efficiency in capital markets. *Vand. L. Rev.*, *68*, 1607.

Zhai, J., Q. Liu, Z. Zhang, S. Zhong, H. Zhu, P. Zhang, and C. Sun (2016). Deep q-learning with prioritized sampling, in *International Conference on Neural Information Processing*, (LNCS, volume *9947*), pp. 13–22, Springer.

AUTHOR BIOGRAPHIES

Jagdish Chakole received a Bachelor of Engineering degree in Computer Engineering from Umrer College of Engineering, Umrer, India in 2008 and a Master of Technology degree in Computer Science and Engineering from Ramdeobaba College of Engineering and Management, Nagpur, India in 2013. Since January 2018, he is a PhD scholar in the Department of Computer Science and Engineering, VNIT, Nagpur, India. His current research interests include Machine Learning, Algorithmic Trading, and Quantitative Finance.

Manish Kurhekar is Associate Professor in the Department of Computer Science and Engineering, VNIT, Nagpur, India. He received a Bachelor of Technology degree from VNIT, Nagpur, India, in 1997, a Master of Technology degree from the Indian Institute of Technology, Bombay, Mumbai, India, in 1999, and the PhD degree from VNIT, Nagpur, India, in 2015 all in Computer Science and Engineering. Before VNIT Nagpur, he has worked for 10 years in various organizations, including, Texas Instruments, IBM India Research Labs, and Persistent Systems Ltd. His areas of interests are Theoretical Computer Science, Multiagent Systems, Machine Learning, Algorithmic Trading, Quantitative Finance, and Systems Biology.

How to cite this article: Chakole J, Kurhekar M. Trend following deep Q-Learning strategy for stock trading. *Expert Systems*. 2020;37:e12514. https://doi.org/10.1111/exsy.12514