Application of multilayer perceptron to deep reinforcement learning for stock market trading and analysis

Hima Keerthi Sagiraju, Shashi Mogalla
Department of Computer Science and System Engineering, Andhra University College of Engineering, Visakhapatnam, India

Article Info

Article history:

Received Mar 28, 2021
Revised Oct 9, 2021
Accepted Oct 13, 2021

Keywords:

Actor-critic framework
Automated stock trading
Deep Q-networks
Deep reinforcement learning
Markov decision process
Multilayer perceptron

ABSTRACT

Trading strategies to maximize profits by tracking and responding to dynamic stock market variations is a complex task. This paper proposes to use a multilayer perceptron method (a part of artificial neural networks (ANNs)), that can be used to deploy deep reinforcement strategies to learn the process of predicting and analyzing the stock market products with the aim to maximize profit making. We trained a deep reinforcement agent using the four algorithms: proximal policy optimization (PPO), deep Q-learning (DQN), deep deterministic policy gradient (DDPG) method, and advantage actor critic (A2C). The proposed system, comprising these algorithms, is tested using real time stock data of two products: Dow Jones (DJIA-index), and Qualcomm (shares). The performance of the agent linked to the individual algorithms was evaluated, compared and analyzed using Sharpe ratio, Sortino ratio, Skew and Kurtosis, thus leading to the most effective algorithm being chosen. Based on the parameter values, the algorithm that maximizes profit making for the respective financial product was determined. We also extended the same approach to study and ascertain the predictive performance of the algorithms on trading under highly volatile scenario, such as the pandemic coronavirus disease 2019 (COVID-19).

Corresponding Author:

Hima Keerthi Sagiraju
Department of Computer Science and System Engineering, Andhra University
Visakhapatmam, Andhra Pradesh, India
Email: keerthisagiraju86@gmail.com

1. INTRODUCTION

A stock market is referred to be a public market that exists to issue, buy, and sell shares or stocks [1], [2]. Stock market or share market is a dynamic ingredient of an economy of free-market, where, the organizations are authorized to capital in exchange to provide shares to the sponsors in the organization proprietorship [3]. Automated stock trading is also termed as "algorithmic trading". Stock trading can be automated using machine learning (ML) techniques such as reinforcement learning (RL) and deep reinforcement learning (DRL) [4]-[6]. In general, manual stock trading is a continuously evolving process involving feedback from the financial markets and implementing new ideas to optimize the trading strategies. To model this trading environment, the framework of Markov decision process (MDP) [7], on which RL is based, can be used. Recent advancements in applications of DRL to automatic trading, with the strategy to maximize profits, can be a vital tool for investment companies, as it can be used to plan capital allocation and maximize investment performance [8].

In the domain of business, forecasting is considered to be difficult since there exists a lot of factors that are dynamic in behaviour and that produces complexities in the market [9]-[11]. However, forecasting is important for the investors to observe the risks in real time, such that the return on investments (ROI) can be obtained higher. This study proposes DRL model for forecasting the stock price so that the investors can find a way to raise their ROI. But the main challenge is to design the best profitable DRL scheme to a complex and dynamic stock market. This study is a step in that direction and aims to investigate and propose possible solutions for this problem. Day-to-day basic operations of trading are sold, buy, and hold. The algorithms can implement these operations to simulate real time trading sessions using real data. The real time stock trading data can be accessed through online platform (example: Yahoo Finance). Data processing methods can be used to reduce the noise in the stock data [12]. This improves the quality of the stock market prediction. One of the objectives of this paper is to determine the desired profitable sessions that can occur in stock trading activities, with the aim to maximize profits. As a deviation to normal market behavior, unprecedented global events such as pandemic (COVID-19) has shown to have a profound bearish impact on the stock market. Prolonged forced restriction by government led to cease of commercial activity, which in turn effected economy (example: service industry) and hence the reason for the stock markets to react negatively (bearish). It would be interesting research to investigate the validity and performance (example: trade profitability) of the algorithms under such (COVID-19) unanticipated dynamic events.

This study proposes to investigate the above-mentioned research problems with an anticipation that, the results can be used by the financial institutions or individuals to device profitable trading strategies or to forecast profitability under similar future events. In this regard, this research work contributes to develop a deep reinforcement model to predict and forecast the profitability of the stock in the market. In this work, multi-layer perceptron (MLP) is developed as the agent for interacting with the environment. This MLP agent is responsible for updating the policy and produce actions so as to predict the future events with less error rate. Using the concept of multilayer perceptron four algorithms are investigated for their predictive performance in this study: deep Q-learning (DQN), deep deterministic policy gradient (DDPG), proximal policy optimization (PPO) and actor critic (A2C). These algorithms are applied to Yahoo Finance datasets of two stocks, Dow Jones industrial average (DJIA) and Qualcomm.

The remaining sections of the paper are organized as follows: section 2 deals with the description of related work and background research. Section 3 consists of the proposed trading methodology. Section 4 includes the definition of datasets, evaluation metrics, and experimental setup. Section 5 discusses the experimental results. Finally, section 6 provides the conclusion and the future work.

2. BACKGROUND AND RELATED WORK

To develop the model framework, the following features from existing literature, are applied in this study [4], [6], [12]-[15]:

- The significance of RL in automating the stock trading.
- Adaptation of deep Q-learning methodology.
- The application of multilayer perceptron (MLP) and artificial neural networks (ANN).
- DRL and RL based algorithms to automate/predict the stock market trading.
- Data-mining framework (wavelet transformation) that discretize the overall process.
- A neural network-based stock price prediction and trading system using technical analysis.

2.1. Reinforcement learning (RL)

With the aim to automate trading process, RL techniques can be applied in the real time trading [4]. RL is a type of machine learning where an agent learns to behave in an environment by performing actions that maximizes reward. The process of reinforcement learning is as follows: Initially, an agent is considered in the environment. The steps are allocated to the agent to determine the better outcome. The agent then moves accordingly and receives the rewards based on its actions. The entire process is a hit and trial-based method. In a stock trading scenario, the agent could be a bot trader, the actions could be volumes of stock to trade and the environment could be the price movement of the stock [16].

2.2. Supervised learning and unsupervised learning

Early machine learning (ML) techniques such as supervised learning (SL) and unsupervised learning (USL) have been widely investigated to develop predictive models to mimic the behavior of stock markets. In SL, the learning is guided by a dataset containing desired outcomes. The SL process uses the dataset to get trained and then it can start making a prediction or decision when it is given new data. Example: the prediction could be, Is it a profitable transaction or loss? In USL, the model learns through observation and it automatically finds the patterns and relationships by creating clusters in the dataset, which may be useful for strategic decision making. There are many studies that uses the SL for the classifications and predictions problems. The support vector machine (SVM) and convolution neural network (CNN) classification model was developed for iris recognition system [17]. The novel feature extraction method was developed for improving the classification performance of the remotely sensed data using SL [18]. Using back propagation algorithm [19], MLP-ANN [20] and extreme learning machine (ELM) [21], the fault detection and classification method was developed for rotating machinery. The butterfly optimization algorithm utilizing the Levy flight (BOALF) and modified butterfly optimization algorithm (BOARN) was proposed for detecting the pneumonia diseases [22]. The deep learning model using global average polling (GAP) techniques with pre-trained CNN was developed for the earlier detection of diabetic retinopathy [23]. The SVM [24] and data mining [25] is used for predicting the performance of students in the online learning process. The framework for grid search model based on the walk-forward validation methodology [26]. The ML models are developed for predicting the failure of heart utilizing grey wolf optimization (GWO) for the feature selection [27]. The long short-term memory (LSTM) network is used for fault classification of the transmission line [28]. The pattern recognition method between normal and autistic affected children was developed using deep learning algorithm [29]. The malaria parasite classification method was developed using CNN and K-nearest neighbors (KNN) [30]. The CNN also used for the development of license plate recognition system [31]. The power flow analysis in the transmission line to evaluate the loss was performed through ELM [25]. The novel bio-inspired algorithm based on the supervised learning was developed for the earlier detection of breast cancer [32]. The offline handwritten signature authentication system was developed using edge features and deep feedforward neural networks (DFNN) [33]. The improved teaching learning-based optimization (ITLBO) [34] algorithm and MOST constructive algorithm and the improved TLBO (MCO-ITLBO) [35] was developed for the prediction of weight and structures.

Similarly, there are many studies that replicate or predict the stock trading results using RL [4]. In early studies, Q-learning method has been applied to derive the optimal decision of the result. However, the major shortcoming of Q-learning was that, it did not have the ability to estimate value for unseen states [36]. Other learning methods like recurrent reinforcement network (RRN) and LSTM [37], [38] did not provide better improvements either. Studies have shown that the ability of Q-learning can be enhanced, by incorporating DRL techniques, to attain improved predictive results as output of an experiment [39]. In this paper, an effective solution for stock trade prediction is proposed by combining deep learning and RL techniques to attain improved predictive results as output of the experiment (Figure 1).

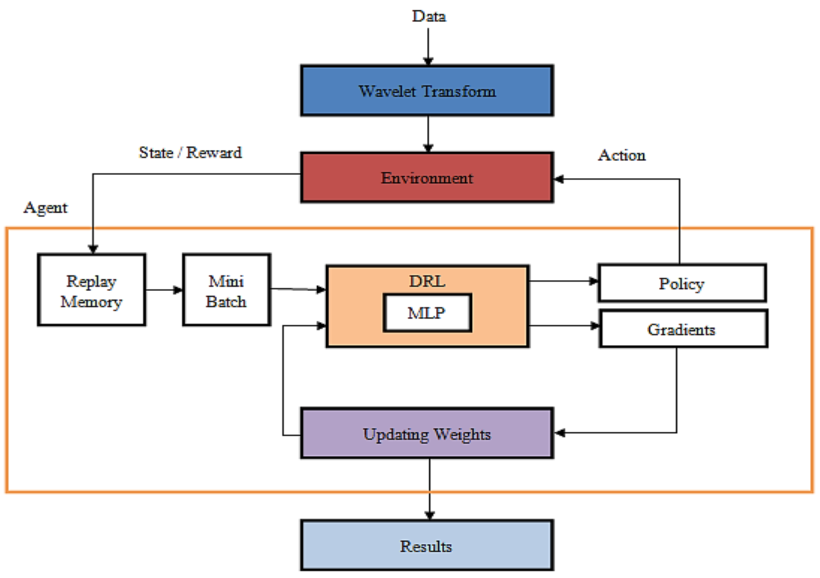

Figure 1. Proposed architecture

2.3. Deep reinforcement learning (DRL)

As there is no automatic way to detect intrusions, implementing a reward function aligned with intrusion detection is exceedingly challenging for intrusion detection systems (IDS) [26]. Typically, network features related with intrusion events are identified manually and saved in datasets of network features. These datasets are used to train supervised machine learning systems for detecting intrusions. As a novel technique, we replaced the environment with a sampling function of recorded training intrusions, substituting the standard DRL paradigm (based on interaction with a live environment).

3. PROPOSED METHOD

The architecture of the proposed model of stock prediction is shown in Figure 1. This research work contributes to develop a deep reinforcement model to predict the stock based on the market analysis. The main components of a reinforcement model are: environment, agent, reward calculation, policy updation, and action. The stock market data is set up as the environment. In this work, the agent developed for interacting with the environment is the multi-layer perceptron (MLP). Before starting the learning process of the MLP, the model parameters such as replay memory and mini batch size are initialized. For every input of data from the environment, reward/state is calculated. Based on this reward, the MLP manipulates the policy to produce an action which will help in improvising the performance. As the learning progresses, the weights get updated and the results are recorded.

3.1. Wavelet transforms

The time domain and frequency domain transforms of a signal are known as wavelet transforms. The basic wavelet is treated as the simulation unit, and useful information is recovered from the signal or noisy information is removed. Signals can be divided into a series of high frequency and low frequency components using the wavelet transform to better preserve the original information. The data is converted using the Wavelet transform, then the coefficients that are more than a complete standard deviation distant (out of all the coefficients) are removed, then the new coefficients are inverse transformed to get the denoised data. Preprocessing of data to remove inconsistencies, redundancies and null values from the input data has been widely studied. This is entirely done to remove or reduce the noise when the data is transformed. There are circumstances where RL model could get arbitrary commotion which is typically present in monetary information and consider it as information which could be followed up on. This can prompt incorrect exchanging signals. Here we h apply wavelet transform to eliminate noise and to avoid losing important data. Hansen *et al.* [12] provides a comprehensive survey of wavelet application in data mining. The authors showed how wavelet transform can be successfully applied to support the creative process of data mining in the phases of data understanding, data preparation, modeling, and evaluation. In this examination, we applied discrete wavelet transform (DWT) to extract features from the non-stationary characteristics of stock time-series data to remove noise.

The DWT can be developed from a scaling function with a restriction that it must be orthogonal to its discrete translations, such as the dilation function,

$$\emptyset(x) = \sum_{i=-\infty}^{\infty} a_i \emptyset(Sx - i) \tag{1}$$

where, S is the scaling factor and a_i is the finite set of coefficients defining the scaling function. The most common family of orthonormal wavelets refer to the family of Daubechies. These wavelets are often denominated by the number of nonzero coefficients a_i, and so the family starts with Daubechies 4 and Daubechies 6 wavelets.

3.2. Stock market environment

Before training a deep reinforcement learning agent, we carefully build the environment to simulate real world trading which allows the agent to perform interaction and learning. To develop a model various parameter, need to be considered like shares, stock price history, and financial results. The environment is built to define action, state and reward.

3.3. Markov decision process (MDP)

The Markowitz model becomes infeasible as the number of investments grows because it is written as a quadratic program (much less efficient than a linear program). The Markov decision process (MDP) assigns a weight to each asset's Markov state, along with its associated expected return and standard deviation, indicating how much of our capital should be invested in that asset. The present weight invested and the economic state of all assets is stored in each state of the MDP. Application of MDP modelling for financial markets has been widely studied [40], [41]. MDP can be described as a sequential decision problem and can be used to determine the probability distribution on states. The output obtained in MDP is completely based on the current state and action of an agent. Main components of MDP are as follows:

- State s = [p, h, b] = a vector that includes stock prizes p belongs to $R + {}^\wedge D$, the stock shares h $(Z + {}^\wedge D)$ and the remaining balance b (R+), where D denotes the number of stocks and Z+ denotes non-negative integers.
- Action a = a vector of actions over D stocks. The allowed actions on each stock include selling, buying, or holding, which results in decreasing, increasing, and no change of the stock shares h, respectively.
- Reward r (s, a, s') = the direct reward of taking action a at state s and arriving the new state s'.

- Policy = the trading strategy at state s, which is the probability distribution of actions at state s.
- Q-value Q (s, a) = the expected reward of taking action 'a' at state 's' following policy.

In the above architecture, at each time step, the state is given by the environment and the agent takes the action according to state and receives the reward and next state. The (s, a, r, s') are saved in the replay memory. For optimization part in each iteration, a batch of (s, a, r, s') is selected for training. The replay memory has specific capacity and after it is filled, a random (s, a, r, s') is substituted with new (s, a, r, s').

3.4. Multilayer perceptron (MLP)

Performance of DRL techniques can be improved by adopting neural networks to approximate the function efficiently, thereby, enhancing its capability to handle both, large state and action spaces [15]. Particularly, multilayer concept (MLP) is chosen as a predictive tool for this study showed promising results in applications of financial forecasting [8], [15], power forecasting [42], attack detection [43]-[45], facial expression recognition [46] and wind speed prediction [47]. Here, MLP is used to describe time arrangement information that attempts to predict the near distant future information dependent on its past information. This is huge in the field of stock market investment, as investors, need to settle on right choices at perfect chance to boost the monetary benefits just as forecast of future development of an organization. MLP is a feedforward neural organization that has three layers; an input layer, hidden layers and output layer. The neural network can be trained with loss function to reduce the risk.

Beginning from starting arbitrary loads, multi-layer perceptron (MLP) limits the loss function by consistently refreshing these loads. After computing the loss, a backward pass propagates it from the output layer to the past layers, furnishing each weight boundary with an update value mean to diminish the loss. The loss is utilized to prepare the neural network. In gradient descent, the gradient $\nabla Loss_W$ of the loss with respect to the and devoted from W, all the more officially, this is communicated as,

$$W^{i+1} = W^{i} - \epsilon \nabla Loss_{w}^{i} \quad (2)$$

where 'i' is the emphasis step, and $\epsilon$is the learning rate with a value bigger than 0. The calculation stops when it arrives at a preset most extreme number of emphases; or when the improvement in the loss is under a specific, small number. Here, the number of resources which describe the large dataset is reduced. It is used to define the data variables with sufficient accuracy. New features can also be created from the existing features of dataset by combining the features. The study [15] provides a comprehensive survey of an artificial neural network-based stock trading system.

3.5. Gradient descent

The model finds a predicted value during the training iterations. The error between the predicted value and the actual value is expressed as (3).

$$Error = Y'(Predicted) - Y(Actual) \quad (3)$$

This error value related to the loss function or cost function of the gradient descent algorithm. During the course of training iteration, the loss is calculated. Consider that the data has N points for which the error has to be minimized. Then the loss function is given as:

$$Loss = \frac{1}{N}\sum_{k=1}^{N}(Y' - Y)^{2} \quad (4)$$

The calculation stops when it arrives at a preset most extreme number if emphases or when the improvement of loss is under a specific modest number. So, loss can be obtained by subtracting the original data from predicted data. Here we acquire a risk**.** Our main aim is to maximize the investment performance and reduce the risk.

3.6. Sharpe ratio

The Sharpe ratio is a risk-adjusted return statistic for a financial portfolio. A portfolio with a higher Sharpe ratio is considered superior to its rivals. Investors typically use the Sharpe Ratio to assess the success of investment management products and professionals. It's computed by dividing the portfolio's excess return compared to the risk-free rate by the portfolio's excess return standard deviation. When compared to similar portfolios with a lower level of diversification, adding diversification should improve the Sharpe ratio. This is true if investors accept the assumption that risk equals volatility, which is logical but may be too restrictive to apply to all investments. The Sharpe ratio of a portfolio determines its risk-adjusted return. A negative Sharpe ratio, on the other hand, means would be better off dealing in a risk-free asset than the one we are now dealing in. Here, we apply Sharpe ratio metric [41]. By the accurate Sharpe ratio value we can choose the best performing specialist among, PPO, A2C, DQN, and DDPG. The working of calculations is clarified in 3.8. We utilize a developing window of n month to hold our four specialists simultaneously. We pick the best performing specialist with most note value sharpe proportion. The Sharpe proportion [19] more prominent than 1.0 is viewed as great. The Sharpe proportion is,

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p} \tag{5}$$

where R_p = return of portfolio, R_f = risk-free rate, and σ_p = standard deviation of portfolio's excess return.

3.7. Deep reinforcement learning (DRL) algorithms

The deep RL algorithms i.e., DQN, DDPG, PPO and A2C are applied individually [13]. The results after applying the algorithms are compared and observed. The algorithms are: Q-learning a model-free reinforcement learning calculation to become familiar with an agent what action to take under what circumstances. "Q" names the capacity that profits the award used to give the support and can be said to represent the "quality" of an activity taken in each state. It does not have the ability to value for unseen states.

3.7.1. Deep Q-networks (DQN)

To deal with the limitation of Q-learning, DQN forms two-dimensional clusters with the help of neural network. Q-learning is a model-free reinforcement learning algorithm for determining the worth of a certain action in a given state. It can handle problems with stochastic transitions and rewards without requiring adaptations and does not require a model of the environment (thus "model-free"). DQN use a Neural Network to assess the Q-learning work. Although Q-learning is a relatively resilient algorithm, its fundamental weakness is its lack of generality. Deep Q-network is one of the algorithms of Reinforcement Learning. Q-learning follows dynamic programming when viewed as renewing numbers in a two-dimensional array (action space * state space). It means that the Q-learning agent has no idea what to do when it encounters states it hasn't encountered previously. To put it another way, a Q-learning agent can't estimate value for states that aren't visible. DQN solves this problem by introducing neural network, which replaces the two-dimensional array. DQN implements a true off-policy update in discrete action space. The experience replay function can solve the correlation and non-static distribution problems. The update formula of Q-learning:

$$Q'(s,a) = Q(s,a) + [r + \gamma max_a \cdot Q(s',a') - Q(s,a)] \tag{6}$$

The loss function of DQN is as follows:

$$L(\theta) = E\left[\left(TargetQ - Q(s,a;\theta)\right)^2\right] \tag{7}$$

$$TargetQ = r_{t+1} + [\gamma max_a \cdot Q(s',a';\theta)] \tag{8}$$

('a' represents action of the agent, 's' represents a state of agent, 'r' is a real value representing the reward of the action, indicates the mean square error loss of the network parameter, and Q′, s′, a′ represents the updated value of Q, s and a.)

3.7.2. Deep deterministic policy gradient (DDPG)

Is a model-free off-policy calculation for learning nonstop activities. It utilizes experience replay and slow-learning target networks from DQN, and it depends on deterministic policy gradient (DPG), which can work over consistent activity spaces. DDPG [41] is off-arrangement on the grounds that our dataset (the replay buffer) was created from a more established strategy. It consolidates thoughts from DPG and DQN. It utilizes experience replay and slow-learning target networks from DQN, and it depends on DPG, which can work over constant activity spaces. It is a calculation which simultaneously learns a Q-function and an approach. It utilizes off-policy information and the Bellman condition to learn Q-function. The actor is an arrangement network that accepts the state as information and outputs the specific activity (constant), rather than a probability distribution over activities. Test code clarification: The prerequisite is to construct the DDPG model and test on financial exchange [41]. The critic network is then updated by minimizing the loss function L(θ^Q) which is the expected difference between outputs of the target critic network Q′ and the critic network Q, i.e,

$$L(\theta^Q) = E_{s_t,a_t,r_t,s_{t+1}} \sim buffer\left[\left(y_i - Q(s_t,a_t;\theta^Q)\right)^2\right] \tag{9}$$

DDPG is effective at handling continuous action space, and so it is appropriate for stock trading.

3.7.3. Proximal policy optimization (PPO*)*

An effective reinforcement learning [48] to solve consistent control activities. Here, this calculation is tried on the choice of continuous control tasks, with both linear and deep approaches, and thought about state-of-the-art plan enhancement strategies. This was presented by the OpenAI group in 2017. It is quite possibly the most mainstream RL strategies usurping the deep Q-learning technique. It guarantees that another update of the arrangement doesn't transform it a lot from the past strategy. It is an on-strategy based technique. First request enhancer like gradient descent technique can be utilized to upgrade the outcome. This is as follows:

$$L^{PG}(\theta) = E_t[\log \pi\theta(a_t|s_t) * A_t] \tag{10}$$

L^{PG} (θ)=policy loss, E_t is expected, log ∏θ (a_t| s_t) is log probability of taking that action at that state, A_t is the advantage if A>0, this action is better than the other action possible at that state.

3.7.4. Advantage actor critic(A2C)

We know the Q vale can be learned by parameterizing the Q function with a deep neural network. The A2C [36] this algorithm includes:

- The "critic" that assesses the value capacity which could be the activity value (Q value) or the state value (V value).
- The "actor" that updates the policy distribution toward the path recommended by the critic (utilizing strategy gradients). Here, both the critic and the actor capacities are defined utilizing deep neural networks built utilizing different layer perceptron. The critic neural organization defines the Q value thus it is referred to as Q actor critic.

This can be formulated as:

$$\nabla_\theta J(\theta) = \mathbb{E}_t[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t] \tag{11}$$

Where J provides the gradient, at refers to the expected reward at time t, 'a' is the action performed by an agent at time instance 't' and 's' is the state where the agent is present at time instance 't'.

3.8. Decision

After the best agent is picked, we use it to predict and trade. The highest the agent Sharpe ratio, the better its return is relative to the amount of investment risk it has taken. Therefore, we pick the trading agent that can maximize the return adjusted to the increasing risk.

4. EXPERIMENTAL SETUP

In this section, we describe all the elements used to explore our proposed model for algorithmic stock trading. This includes the discussion of the financial products considered for dataset. The datasets use evaluation metrics for market trading and the implementation of the proposed stock model.

4.1. Datasets

We use the dataset of Dow Jones Index (DJIA) and Qualcomm. The dataset is extracted from Yahoo finance website and spans for a period of 30 years (1992-01-02 to 2020-11-30). The data is split into train, test and COVID-19. The training dataset is from 1992-01-02 to 2012-01-03 and the testing dataset is from 2012-01-03 to 2020-11-13. The COVID-19 data includes 2020-03-01 to 2020-11-13. In this, we can compare the effectiveness of RL in terms of training and showing result.

4.2. Experimental methodology

We considered the preprocessed data as input. The entire dataset is split into training data and testing data. The training data is used primarily to get the results using the different algorithms included. This process of using training data is often called as learning step. Later, we use the remaining data i.e., the testing data to explore the output. Here, this process of using testing data is referred to as construction step. This also includes the consideration of the data of the past recent pandemic time that ran out of the profits.

4.3. Evaluation Metrics

An evaluation metrics quantifies the performance of a predictive model. It gives us a measure to compare different models of stock trading agents. They are the financial ratios that compare a stock or indices. The metrics considered for this study are [16] presented in Table 1. Each metrics is described by its function and use.

Table 1. Evaluation metrics

Evaluation Metrics	Description
Cumulative Return	Calculated by subtracting the portfolio's final value from its initial value, and then dividing by the initial value.
Annual Return	The annualized return generated during the trading activity.
Annualised Volatility	The modelling of risk associated with trading activity.
Sharpe Ratio	The return of trading activity compared to its riskiness.
Sartiono Ratio	Similar to the Sharpe ratio with negative risk penalized only.
Skewness	It is a degree of distortion from the symmetrical bell curve or the normal distribution. It is the measure of lack of symmetry in data distribution.
Kurtosis	Measure of whether the data is heavy-tailed or light-tailed relative to normal distribution.
Maximum Drawdown	The maximum percentage loss during the trading period.
Daily value at risk	Metric that estimates the risk of an investment over a specific period of time.

4.4. Hyper parameters

- *Max_open_positions* = 5: A transaction that has been established but has not yet been closed out with an opposing trade is referred to as an open position.
- *Max_Account_balance* = 147483647 (any random high number): Maximum Account Balance refers to the amount designated by the Committee as the maximum principal amount of a Security at any one moment.
- *Max_num_shares* = 147483647 (any random high number): A share is the smallest denomination of stock in a firm.
- *Max_share_price* = 5000: the price of one share of stock in a corporation. A share's price is not set in stone, but changes with market conditions. If the company is regarded to be doing well, it will likely rise, whereas if the company isn't fulfilling expectations, it will likely decrease.
- *Initial_account_balance* = 1000000: There is no minimum investment amount required to begin trading in the Indian stock market. Simply put, you must have enough capital to cover the cost of a stock. As a result, you do not require a large sum of money to begin trading in India.
- *Lookback_window_size* = 30 (how many days to consider for every trade i.e., how much vector to pass into multilayer perceptron.

5. EXPERIMENTAL RESULTS

The outcomes of applying the algorithms to the data of two stock products is depicted below. The dataset after splitting into training data set, testing set and COVID-19 set, different algorithms are applied individually to testing and COVID-19 data sets. The results obtained are in Tables 1 and 2. From the Tables 2 and 3, it can be analyzed that on considering the DJIA related table, the Sharpe ratio as 0.51 which is relatively high for PPO algorithm than other algorithms when applied to the same testing data set. The Sortino ratio is 3.45 which is desirable i.e. Sortino ratio > 2 yields better profits. When A2C is used for DJIA, the evaluation metrics i.e. the Sharpe ratio is relatively less which is indicating less profits and so is the Sortino ratio. For Qualcomm stock, the Sharpe ratio is 0.57 and the Sortino ratio is 3.47 when DQN is applied to the data set.

Table 2. Comparison of RL agents using different evaluation metrics for historical data (2012-01-03 to 2020-11-20) Dow Jones

Algorithm	PPO	DQN	DDPG	A2C
Cumulative Return (%)	0.65	0.54	-0.133	0.856
Annual Return (%)	0.073	0.08	-0.015	0.096
Annual Volatility (%)	953.96	1200.32	1228.87	920.302
Sharpe Ratio	0.51	~0.5	-0.36	0.22
Sortino Ratio	3.45	~0.63	-0.36	0.55
Skew	31.62	35.75	-38.77	32.75
Kurtosis	1043.82	1350.85	1615.52	1589.48
Max Drawdown	-110.595	-100	-112.624	-100.267
Daily Value at Risk	-118.274	Nan	-156.569	-115.137

Table 3. Comparison of RL agents using different evaluation metrics for historical data (2012-01-03 to 2020-11-20) Qualcomm

Algorithm	PPO	DQN	DDPG	A2C
Cumulative Return (%)	0.57	0.834	0.004	-1.531
Annual Return (%)	0.064	0.094	0.075	-0.174
Annual Volatility (%)	27520.219	4064.36	2381.188	7632.955
Sharpe Ratio	0.26	0.57	-0.32	0.27
Sortino Ratio	2.54	3.47	-0.33	1.19
Skew	46.4	32.85	-38.41	41.15
Kurtosis	2178.83	1233.31	1679.91	1861.04
Max Drawdown	-156.971	-118.526	-112.416	-153.05
Daily Value at Risk	-3439.217	-502.905	-303.025	-953.462

This yields the best outcome for Qualcomm. When DDPG is applied, the Sharpe ratio is -0.32 and the Sortino ratio is -0.33. This is producing reduced profit-oriented outcome and so it is clear that DDPG is not efficient to maximize profit for Qualcomm stock. The sales output for DJIA and Qualcomm are given in (Figure 2). As shown in Figure 2, most part of the output is covered by DJIA. The algorithms efficiency (Figure 3) can also be determined based on the results and the metric values obtained in Tables 2 and 3.

Now, on comparing the results of the two products i.e. DJIA, and Qualcomm, the graphical representations in Figures 4 and 5 are the outcomes based on the data considered. Including the overall performance, it is shown that DJIA has the best result with more accurate profits generated. Then the Qualcomm stock produced better and intermediate profits with significant loss comparatively.

From Tables 2 and 3, when the data is exposed with four different algorithms, the Sharpe ratio is greater than 1 and is more accurate for the DQN and PPO algorithms, it is moderate for DDPG and is less for actor critic (AC2). The value of metrics may be good for DQN but the algorithm itself is unstable when compared to DDPG. DQN algorithm may not produce consistent result as DDPG does. This is same for the remaining metrics when calculated by applying the algorithms to both the testing data as well as the COVID-19 data. So, the results indicate that the algorithm that is yielding the best outcome is PPO. However, the least profit-oriented algorithm is AC2 algorithm. The DQN and DDPG algorithms provided relatively moderate results. When these two algorithms are considered, DDPG policy gradient method produced better outcome since it combines ideas from DPG and DQN. It utilizes Experience Replay and moderate taking in target networks from DQN, and it depends on DPG, which can work over consistent activity spaces. This cannot be achieved by using DQN algorithm. The efficient order of the algorithms can be as in Figures 2 and 3.

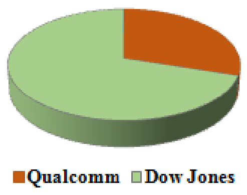

Figure 2. Sales

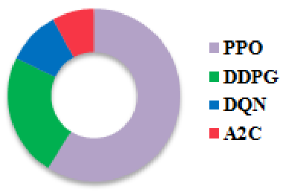

Figure 3. Algorithm efficiency

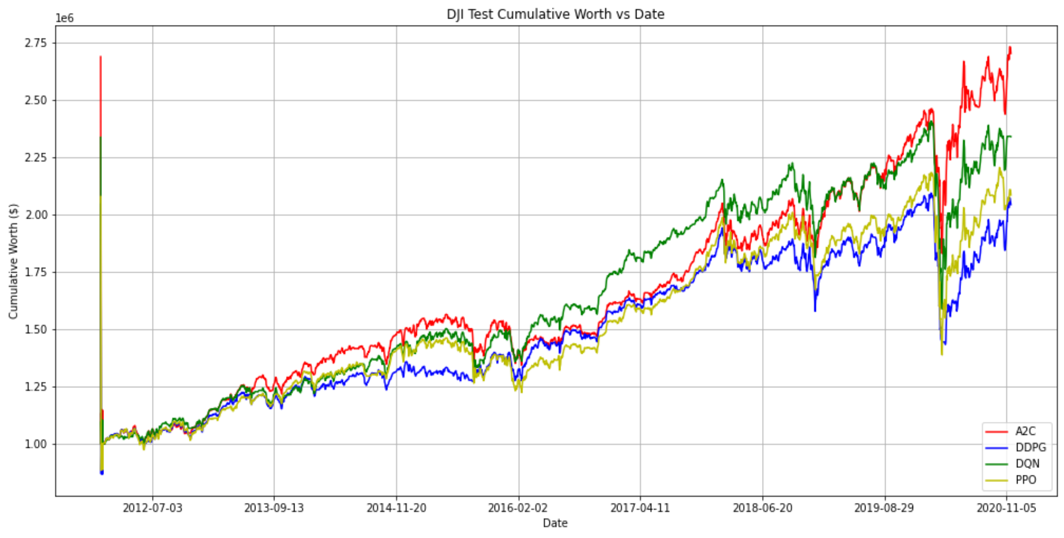

Figure 4. Graphical representation of DJIA cumulative worth based on test data

From the Tables 4 and 5, the results can be analyzed: Considering the DJIA, the Sharpe ratio is 1.84 when PPO is applied and it is 1.92 when DQN is applied. The Sortino ratio is 5.94 when PPO is used and 14.11 when DQN is used. But also considering the other metrics, the cumulative return is 7.389, annual return is 10.197 when DQN is applied. But the skewness being high i.e., 10.42, the consistency is obtained through PPO rather than DQN where PPO's skewness is produced as 6.06 which is the lowest value and is therefore leading to the best outcome.

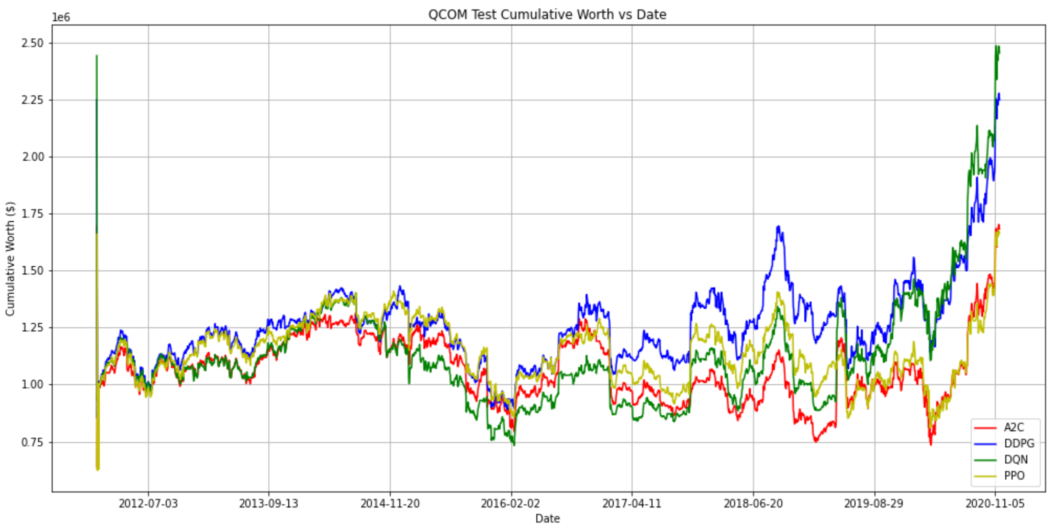

Figure 5. Graphical representation of Qualcomm cumulative worth based on test data

Table 4. Comparison of RL agents using different evaluation metrics for COVID-19 data (2020-03-01 to 2020-11-20) Dow Jones

Algorithm	PPO	DQN	DDPG	A2C
Cumulative Return (%)	4.099	7.389	3.235	3.228
Annual Return (%)	5.625	10.197	4.736	4.423
Annual Volatility (%)	1067.217	6339.088	4500.354	4710.833
Sharpe Ratio	1.84	1.92	~ -1.6	-1.72
Sortino Ratio	5.94	14.11	~ -4	-1.82
Skew	6.06	10.42	8.01	-7.76
Kurtosis	48.48	121.61	65.72	84.73
Max Drawdown	-256.956	-1312.984	-111.639	-399.901
Daily Value at Risk	-126.65	-750.41	Nan	-625.664

Table 5. Comparison of RL agents using different evaluation metrics for COVID-19 data (2020-03-01 to 2020-11-20) Qualcomm

Algorithm	PPO	DQN	DDPG	A2C
Cumulative Return (%)	0.008	-1.458	-0.211	0.01
Annual Return (%)	0.01	-1.98	-0.288	0.005
Annual Volatility (%)	688.143	1556.209	8760.611	878.646
Sharpe Ratio	1.75	1.84	-0.99	2.38
Sortino Ratio	2.79	7.73	-1.04	5.93
Skew	-0.4	9.27	-10.36	2.89
Kurtosis	13.1	102.98	133.74	13.86
Max Drawdown	-128.366	-118.834	-116.528	-127.747
Daily Value at Risk	-81.926	-184.725	-1137.987	-102.386

While evaluating the least performance, A2C algorithm is yielding less output where the Sharpe ratio is -1.72 indicating huge loss in market and the Sortino ratio is -1.82 that is far away from optimal value of Sortino ratio 2. Now for Qualcomm company, the Sharpe ratio is 1.84 for DQN and is 1.75 for PPO. The Sortino ratio is 7.73 for DQN and is 2.79 for PPO. However, considering the skew metrics, it is efficient for PPO since the skew value for PPO (-0.4) is less than that of DQN (9.27). On this basis, PPO algorithm produced better output than other algorithms. A2C and DDPG produced least profits for Qualcomm company out of which A2C produced relatively better result with Sharpe ratio=2.38 and Sortino ratio=5.93.

Based on the output values, DJIA performs better than Qualcomm as it produces intermediate result with significant loss. PPO is the algorithm which is yielding good result with the overall evaluation metrics values and A2C is the algorithm associated with least profits and significant loss when compared to the other algorithms. Also, the other two algorithms i.e., DQN and DDPG are yielding better outputs in some scenarios and huge loss-oriented results when all the evaluation metrics such as skewness and Kurtosis are considered for analysis.

From the results obtained when algorithms are applied on the COVID-19 data sets (Figure 6 and Figure 7), DJIA is associated with more accurate and consistent result which produced the best profits when compared to Qualcomm. The result for all the algorithms showed positive profits for this stock. Qualcomm is the next stock which produced the moderate profits and an intermediate output with significant loss. The values for this stock data during COVID-19 time is not as efficient as the DJIA. There exists a drastic change in the economy and share markets of the two companies. The COVID-19 crisis pulled the stock marketing strategies down. Irrespective of this, the DJIA withstood the crisis and loss more terribly and efficiently when compared to Qualcomm.

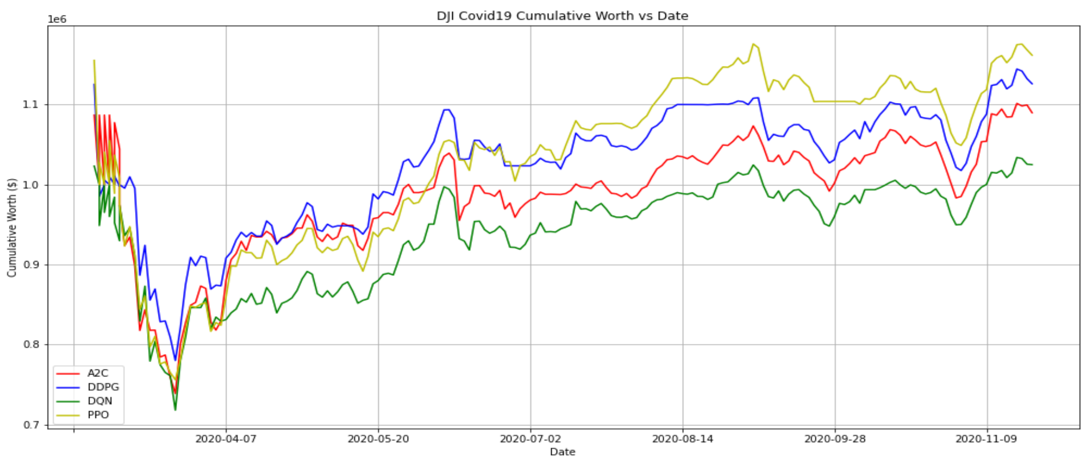

Figure 6. Graphical representation of DJIA simulation (COVID-19 dataset)

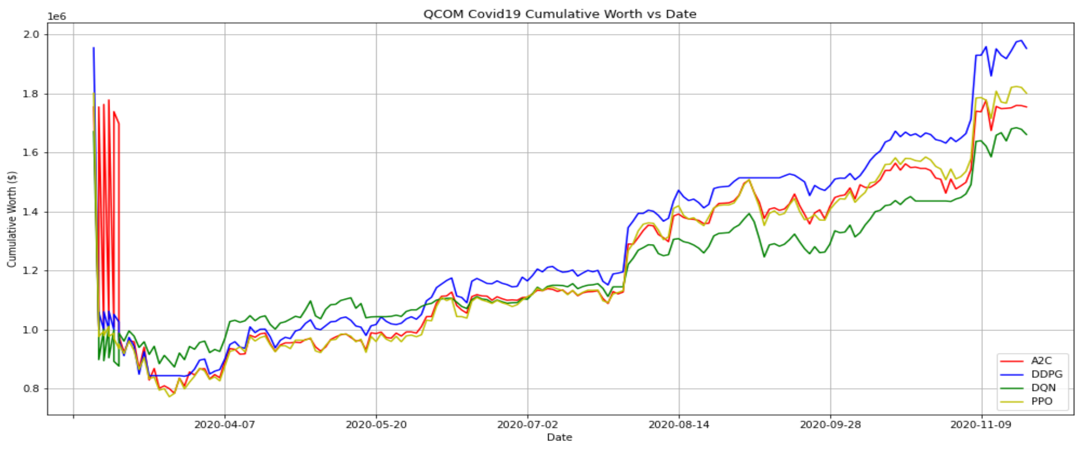

Figure 7. Graphical representation of Qualcomm simulation (COVID-19 dataset)

6. CONCLUSION

In this paper, we investigated the efficiency of four algorithms to simulate stock trading (DJIA and Qualcomm) under normal trading conditions and further extended the study to evaluate their ability to simulate under distressed (bearish) market conditions caused by pandemic (COVID-19). PPO, DQN, DDPG and advantage actor critic (A2C) algorithms have been enhanced with MLP to increase the model ability to handle large state and action spaces. We applied these algorithms to the agent, to explore the best strategy-oriented algorithm, based on the Sharpe ratio, Sortino ratio, Skewness factor and Kurtosis. The agent is trained to maximize profits in shortest time. The results from this study clearly show successful adaptation of enhanced DLR algorithms to automate stock trading, with the objective to maximum profitability. Simulated results show that PPO and DQN outperformed the other two algorithms, both under normal day-to-day trading scenario, as well as, under bearish market conditions (COVID-19). The results further reiterate that, PPO and DQN can be implemented to develop a profitable trading strategy under highly volatile stock market scenarios, with good degree of confidence. For the data collected from 2012 to 2020, it was found that, in the case of DJIA, the PPO algorithm produced a relatively high Sharpe ratio of 0.51 and high Sortino ratio of 3.45. In the case of Qualcomm stock, the DQN model was found to perform better with a Sharpe ratio of 0.57 and Sortino ratio of 3.47. For the data in the COVID-19 pandemic, it was found that, in the case of DJIA, the Sharpe ratio was 1.84 when PPO was applied and it was 1.92 when DQN was applied. The Sortino ratio was 5.94 when PPO was used and 14.11 when DQN was used. The PPO algorithm produces lowest skewness and thus has better consistency leading to best outcome. In the case of Qualcomm, the Sortino ratio was 5.94 when PPO was used and 14.11 when DQN was used. The PPO algorithm found to produce better results with least skewness. Thus, the proposed model of stock prediction can be used by the investors, individuals, or financial institutions to develop profitable trading techniques or to predict profitability under similar events in the future.

REFERENCES

[1] R. J. Barro, “The stock market and investment,” *The review of financial studies*, vol. 3, no. 1, pp 115-131, 1990, doi: 10.1093/rfs/3.1.115.

[2] R. J. Teweles and E. S. Bradley, *The stock market -7th Edition*, New York, USA: John Wiley & Sons, 1998.

[3] S. M. Idrees, M. A. Alam, and P. Agarwal, “A Prediction Approach for Stock Market Volatility Based on Time Series Data,” in *IEEE Access*, vol. 7, pp. 17287-17298, 2019, doi: 10.1109/ACCESS.2019.2895252.

[4] P. G. Necchi, “Reinforcement Learning for Automated Trading,” *Mathematical EngineeringPolitecnico di Milano: Milano, Italy*, 2016.

[5] T. Théate and D. Ernst, “An Application of Deep Reinforcement Learning to Algorithmic Trading,” *Expert Systems with Applications*, vol. 173, 2020, doi: 10.1016/j.eswa.2021.114632.

[6] G. Jeong and H. Y. Kim, “Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning,” *Expert Systems with Applications*, vol. 117, pp. 125–138, 2019, doi: 10.1016/j.eswa.2018.09.036.

[7] A. Gupta and B. Dhingra, “Stock market prediction using Hidden Markov Models,” *2012 Students Conference on Engineering and Systems*, 2012, pp. 1-4, doi: 10.1109/SCES.2012.6199099.

[8] E. Chong, C. Han, and F. C. Park, “Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies,” *Expert Systems with Applications*, vol. 83, pp. 187-205, 2017, doi: 10.1016/j.eswa.2017.04.030.

[9] R. Sutton, D. Mcallester, S. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” *Proceedings of the 12th International Conference on Neural Information Processing Systems*, pp. 1057–1063, 1999.

[10] Z. Zhang, S. Zohren, and S. Roberts, “Deep Reinforcement Learning for Trading,” *Journal of Financial Data Science Spring*, vol. 2, no. 2, pp. 25-40, 2020, doi: 10.3905/jfds.2020.1.030.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[12] K. B. Hansen, “The virtue of simplicity: On machine learning models in algorithmic trading Big Data & Society,” vol. 7, no. 1, *Big Data & Society*, 2020, doi: 10.1177/2053951720926558.

[13] V. Mnih *et al.*, “Asynchronous methods for deep reinforcement learning,” *The 33rd International Conference on Machine Learning,* vol. 48, pp. 1928-1937, 2016.

[14] Z. Xiong, X-Y. Liu, S. Zhong, H. Yang, and A. Walid, “Practical Deep Reinforcement Learning Approach for Stock Trading,” *arXiv preprint arXiv:1811.07522*, 2018.

[15] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, “Predicting stock market index using fusion of machine learning techniques,” *Expert Systems with Applications*, vol. 42, no. 4, pp. 2162-2172, 2015, doi: 10.1016/j.eswa.2014.10.031.

[16] B. Lim, S. Zohren, and S. Roberts, “Enhancing Time Series Momentum Strategies Using Deep Neural Networks,” *The Journal of Financial Data Science,* vol. 1, no. 4, pp. 19-38, 2019, doi: 10.3905/jfds.2019.1.015.

[17] M. Choudhary, V. Tiwari, and U. Venkanna, “Enhancing human iris recognition performance in unconstrained environment using ensemble of convolutional and residual deep neural network models,” *Soft* Computing, vol. 24, pp. 11477–11491, 2020, doi: 10.1007/s00500-019-04610-2.

[18] D. Lu and Q. Weng, “A survey of image classification methods and techniques for improving classification performance,” *International Journal of Remote Sensing*, vol. 28, no. 5, pp. 823-870, 2007.

[19] A. A. Mas’ud, A. Jamal, S. Adewusi, and A. Sundaram, “Rotating blade faults classification of a rotor-disk-blade system using artificial neural network,” *International Journal of Power Electronics and Drive Systems*, vol. 12, no. 3, pp. 1900-1911, 2021, doi: 10.11591/ijpeds.v12.i3.pp1900-1911.

[20] E. Yan, J. Song, C. Liu, J. Luan, and W. Hong, “Comparison of support vector machine, back propagation neural network and extreme learning machine for syndrome element differentiation,” *Artificial Intelligence Rev.*, vol. 53, pp. 2453–2481, 2020, doi: 10.1007/s10462-019-09738-z.

[21] S. Ishak, S-P. Koh, J-D. Tan, S-K. Tiong, and C-P. Chen, “Corona fault detection in switchgear with extreme learning machine,” *Bulletin of Electrical Engineering and Informatics*, vol. 9, no. 2, pp. 558-564, 2020, doi: 10.11591/eei.v9i2.2058.

[22] B. I. Khaleel and M. Y. Ahmed, “Pneumonia detection using butterfly optimization and hybrid butterfly optimization algorithm,” *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2037-2045, 2021, doi: 10.11591/eei.v10i4.2872.

[23] M. M. Assadullah, “Barriers to Artificial Intelligence Adoption in Healthcare Management: A Systematic Review,” *SSRN*, 2019, doi: 10.2139/ssrn.3530598.

[24] E. A. Mahareek, A. S. Desuky, and H. A. El-Zhni, “Simulated annealing for svm parameters optimization in student’s performance prediction,” *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 3, pp. 1211-1219, 2021, doi: 10.11591/eei.v10i3.2855.

[25] L. Imen and L. Djamel “Power flow variation based on extreme learning machine algorithm in power system”, *International Journal of Power Electronics and Drive Systems*, vol. 10, no. 3, pp. 1244-1254, 2019, doi: 10.11591/ijpeds.v10.i3.pp1244-1254.

[26] D. P. Chassin, J. C. Fuller, and N. Djilali, “Grid LAB-D: An Agent-Based Simulation Framework for Smart Grids,” *Journal of Applied Mathematics*, vol. 2014, 2014, doi: 10.1155/2014/492320.

[27] M. T. Le, M. T. Vo, N. T. Pham, and S. V. T. Dao, "Predicting heart failure using a wrapper-based feature selection," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 3, pp. 1530-1539, 2021, doi: 10.11591/ijeecs.v21.i3.pp1530-1539.

[28] A. M. S. Omar, M. K. Osman, M. N. Ibrahim, Z. Hussain, and A. F. Abidin, "Fault classification on transmission line using LSTM network," *Indonesian Journal of Electrical Engineering and Computer Science,* vol. 20, no. 1, pp. 231-238, 2020, doi: 10.11591/ijeecs.v20.i1.pp231-238.

[29] N. A. Ali, A. R. Syafeeza, A. S. Jaafar, and M. K. M. F. Alif, "Autism spectrum disorder classification on electroencephalogram signal using deep learning algorithm," *IAES International Journal of Artificial Intelligence*, vol. 9, no. 1, pp. 91-99, 2020, doi: 10.11591/ijai.v9.i1.pp91-99.

[30] W. Lumchanow and S. Udomsiri, "Image classification of malaria using hybrid algorithms: Convolutional neural network and method to find appropriate K for K-Nearest neighbor," *Indonesian Journal of Electrical Engineering and Computer Science*, vol 16, no. 1, pp. 382-388, 2019, doi: 10.11591/ijeecs.v16.i1.pp382-388.

[31] V. Malathi, N. S. Marimuthu, and S. Baskar, "Intelligent approaches using support vector machine and extreme learning machine for transmission line protection," *Neurocomputing*, vol. 73, no. 10–12, pp. 2160-2167, 2010, doi: 10.1016/j.neucom.2010.02.001.

[32] S. Qiu , F. Fei, and Y. Cui, "Offline Signature Authentication Algorithm Based on the Fuzzy Set," *Mathematical Problems in Engineering*, vol. 2021, 2021, doi: 10.1155/2021/5554341.

[33] T. S. Gunawan and M. Kartiwi, "On the use of edge features and exponential decaying number of nodes in the hidden layers for handwritten signature recognition," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 12, no. 2, pp. 722-728, 2018, doi: 10.11591/ijeecs.v12.i2.pp722-728.

[34] B. Naik, J. Nayak, and H. S. Behera, "A TLBO based gradient descent learning-functional link higher order ANN: An efficient model for learning from non-linear data," *Journal of King Saud University - Computer and Information Sciences,* vol. 30, no. 1, pp. 120-139, 2018, doi: 10.1016/j.jksuci.2016.01.001.

[35] M. Jouyban and M. Khorashadizade, "Using the modified k-mean algorithm with an improved teaching-learning-based optimization algorithm for feedforward neural network training," *International Journal of Electrical and Computer Engineering*, vol. 11, no. 6, pp. 5277-5285, 2021, doi: 10.11591/ijece.v11i6.pp5277-5285.

[36] W. F. Sharpe, "The Sharpe ratio," *Journal of Portfolio Management*, vol. 21, no. 1, pp. 49-58, 1994, doi: 10.3905/jpm.1994.409501.

[37] D. Munandar, A. F. Rozie, and A. Arisal, "A multi domains short message sentiment classification using hybrid neural network architecture," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2181-2191, 2021, doi: 10.11591/eei.v10i4.2790.

[38] I. H. Sarker, "Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions," *SN Comput. Sci.,* vol. 2, no. 420, 2021, doi: 10.1007/s42979-021-00815-1.

[39] P. C. Pendharkar and P. Cusatis, "Trading financial indices with reinforcement learning agents," *Expert Syst. Appl.*, vol. 103, pp. 1-13, 2018, doi: 10.1016/j.eswa.2018.02.032.

[40] O. B. Sezer, A. M. Ozbayoglu, and E. Dogdu, "An Artificial Neural Network-based Stock Trading System Using Technical Analysis and Big Data Framework," *Proceedings of the SouthEast Conference,* 2017, pp. 223–226, doi: 10.1145/3077286.3077294.

[41] A. Ilmanen, *Expected returns: An investor's guide to harvesting market rewards*, New York, US: Wiley Online Library, 2011, doi: 10.1002/9781118467190.

[42] A. Verma and V. Ranga, "Machine Learning Based Intrusion Detection Systems for IoT Applications," *Wireless Pers Commun,* vol. 111, pp. 2287–2310, 2020, 10.1007/s11277-019-06986-8.

[43] T. B. Seong, V. Ponnusamy, N. Z. Jhanjhi, R. Annur, and M. N. Talib, "A comparative analysis on traditional wired datasets and the need for wireless datasets for IoT wireless intrusion detection", Indonesian *Journal of Electrical Engineering and Computer Science*, vol. 22, no. 2, pp. 1165-1176, 2021, doi: 10.11591/ijeecs.v22.i2.pp1165-1176.

[44] D. Dikii, S. Arustamov, and A. Grishentsev, "DoS attacks detection in MQTT networks," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 1, pp. 601-608, 2021, doi: 10.11591/ijeecs.v21.i1.pp601-608.

[45] A. F. Jabbar and I. J. Mohammed, "Bot Detector FW: an optimized botnet detection framework based on five features-distance measures supported by comparisons of four machine learning classifiers using CICIDS2017 dataset," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 1, pp. 377-390, 2021, doi: 10.11591/ijeecs.v21.i1.pp377-390.

[46] M. R. Mahmood, M. B. Abdulrazaq, S. R. M. Zeebaree, A. Kh. Ibrahim, R. R. Zebari, and H. I. Dino, "Classification techniques' performance evaluation for facial expression recognition," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 2, pp. 1176-1184, 2020, doi: 10.11591/ijeecs.v21.i2.pp1176-1184.

[47] Y. Amellas, O. E. bakkali, A. Djebli, and A. Echchelh, "A Short-term wind speed prediction based on MLP and NARX network models," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 1, pp. 150-157, 2019, doi: 10.11591/ijeecs.v18.i1.pp150-157.

[48] T. P. Lillicrap *et al*., "Continuous control with deep reinforcement learning," *International Conference on Learning Representations (ICLR),* 2016.