Available online at www.HighTechJournal.org

HighTech and Innovation Journal

ISSN: 2723-9535

# HighTech and Innovation Journal

Vol. 6, No. 2, June, 2025

# Application of Deep Learning for Stock Prediction Within the Framework of Portfolio Optimization in Quantitative Trading

Xiaoyu Qin [1*]

[1] *School of Engineering, The University of Manchester, Manchester, M13 9PL United Kingdom.*

Received 05 January 2025; Revised 28 April 2025; Accepted 05 May 2025; Published 01 June 2025

**Abstract**

This paper proposes a method for stock prediction and portfolio optimization as a part of quantitative trading based on a combination of Bi-RNN and a modified snake optimization algorithm (MSOA) to build optimal portfolios and outperform conventional models and benchmarks. Methods/Analysis: We employ the Bi-RNN model, which processes historical stock data in both forward and backward directions to unveil intricate temporal dependencies. MSOA is used to fine-tune the hyperparameters of the Bi-RNN with enhancements such as Latin Hypercube Sampling for initialization, dynamic temperature adjustment, adaptive learning rates, and hybrid exploration-exploitation mechanisms. The Markowitz mean-variance approach is used to optimize the portfolio from asset allocations that the MSOA then improves. The model is evaluated on the S&P 500 from 1993 to 2020. Results: Such findings in experiments indicate that the proposed model outperforms baseline models, e.g., LSTM, GRU, and HMM, with lower Mean Squared Percentage Error (MSPE) values and higher Sharpe ratios of constructed portfolios. For instance, Portfolio 3 produced a 10.9% expected return with a standard deviation of 12.9%, delivering risk-adjusted returns that exceed those of the S&P 500. Novelty/Improvement: A strong integrated approach of deep learning and advanced optimization techniques is proposed for stock prediction and portfolio optimization, which achieves notable improvements in terms of accuracy and efficiency. The proposed approach overcomes the drawbacks of traditional algorithms, making it a valuable tool for financial decision-making.

*Keywords:* Stock Prediction; Bi-RNN; MSOA; Portfolio Construction; Stock Market Forecasting; Deep Learning; Time Series Analysis.

## 1. Introduction

In recent years, economic variable prediction has been of special importance for strategic managers in the private and public sectors in order to regulate economic affairs and relations, so that the need for tools and methods of predicting variables with the least amount of error is noticeable. Due to the importance and special position of financial markets and the effect they have on parallel markets, which shows their strong role in the economy of every country, prediction in this area is of special importance and has become an integral and important part of this area.

The higher the accuracy of this prediction and the less error it has, the more confidence investors have because in this way they can minimize their risk. In order to increase the accuracy of prediction, it is possible to recognize the important factors affecting the financial markets and decrease the predicting errors with their help.

The stock market signifies a multifaceted and ever-evolving system that has gathered extensive research and scrutiny over the years. For investors, policymakers, and financial institutions, the capacity to accurately predict stock prices and

* Corresponding author: xiaoyuqinnnx210202@outlook.com

http://dx.doi.org/10.28991/HIJ-2025-06-02-016

trends is important, as it influences asset choices, risk management approaches, and economic strategies. However, forecasting stock prices poses important challenges due to the inherent unpredictability and fluctuations characteristic of the market.

The use of deep learning and machine learning techniques has grown in popularity recently in the field of stock price forecasting. These advanced methods have shown encouraging results in predicting stock prices and trends, which has led to their increasing popularity among investors and financial institutions. However, the majority of current research has focused on traditional machine learning techniques, like random forests and support vector machines (SVMs), which have limits in their ability to recognize intricate patterns and relationships in the data.

Recently, recurrent neural networks (RNNs) have found extensive application in sequence prediction tasks, including natural language processing and speech recognition. RNNs are particularly advantageous for stock price forecasting, as they are adept at recognizing complex patterns and relationships in the data while accommodating the sequential nature of stock price information.

Thio-Ac et al. [1] introduced a system of decision support to optimize the portfolios of stock. Hidden Markov Model has been utilized for 95% accurate stock cost predictions during a five-day duration. The system employed Quadratic Programming to enhance asset distribution, concentrating on a maximum of 15 established blue-chip firms. Important metrics such as Bollinger Bands, RSI, and MACD provided definitive recommendations to hold, sell, or buy. It continuously tracked the Philippine Stock Exchange, alerting users to considerable fluctuations. Designed using Python and C#, the models have been trained using a steady dataset spanning five years for reliable outcomes. The findings indicated good forecast accuracy, with a value of 98.923% for peak prices and 98.741% for low prices, which supported investment techniques. The research contrasted SPOT-allotted portfolios with stochastic allocations, showing a significant distinction in profit and loss distribution. This highlighted its importance for making profitable and informed choices in stock trading.

Xiao & Tang [2] suggested a quantitative trading system on the basis of LSTM using Python for model prediction and learning. By adapting the model's internal variables, the stock forecast's accuracy was enhanced. The stock's historical data was utilized for forecasting and learning trends of future stock. However, there were some problems regarding the efficiency of the system in the long term, hence requiring some additional investigations in this field.

Martínez-Barbero et al. [3] integrated machine learning approaches, the classical mean–variance optimization algorithm, and Long Short-term Memory (LSTM) with the purpose of providing accurate forecasted returns and generating money-making portfolios for 10 holding intervals. Moreover, it presented diverse contexts of finance. The suggested algorithm was tested and trained using historical EURO STOXX 50® Index data. The findings represented that the suggested LSTM could accomplish minor errors of forecast, as the MSE average of 10 holding intervals was 0.00047, and the MAE's average was 0.01634. Moreover, the accuracy of prediction during 10 investment intervals was 95.8%.

Huang et al. [4] suggested a kind of method to improve returns of investment by combining LSTM forecasts as well as the EOW (Evolutionary Operating-weights) algorithmic technique. The suggested approach utilized an LSTM with several layers to predict prices of future stock, including the forecasts with data of the actual market. The findings of the current investigation represented that this could perform better than the other models.

Jeribi et al. [5] introduced an expert framework for stock market prediction on the basis of deep learning, named DLEF-SM. The approach used improved jellyfish-induced filtering (IJF-F) to preprocess the data and efficiently analyzed raw data and eliminated artifacts. To overcome the issue of imbalanced data and improve the quality of data, previously trained CNNs (Convolutional Neural Networks), namely ResNet-50 and VGGFace2, were utilized for extraction of features. In addition, an IBWO (Improved Black Window Optimization) was developed for the selection of features that diminished the dimensionality of data and prevented the underfitting issue. In order to accurately predict the stock market, Artificial Neural Network and Deep Reinforcement were combined. The prediction accuracy of the model for DAX markets, S&P500-L, and S&P500-S was, in turn, 98.825%, 98.235%, and 99.562%.

However, there are other things that go on in real-world stock data that existing models are not able to capture well, especially during market turbulence. Many conventional algorithms do not efficiently balance exploration and exploitation, resulting in local optima or so-called poor solutions. Moreover, though RNNs perform well when modeling sequential data, they are limited with regard to certain hyperparameters and optimization procedures [6].

To overcome these limitations, this study proposes a new methodology based on a bidirectional recurrent neural network (Bi-RNN) by using a modified snake optimization algorithm (MSOA). Bi-RNN architecture passes the historical stock data in the forward and reverse directions, which in turn assists in capturing the complex temporal dependencies. On the other hand, MSOA optimizes hyperparameters for the model in a dynamic manner by adopting

novel methods like Latin Hypercube Sampling (LHS) to create samples, dynamically controlling the temperature in annealing, dynamically changing the learning rate adaptation, and using hybrid exploration-exploitation. The goal of this integration is to increase prediction accuracy and create optimized portfolios exceeding benchmark indices like the S&P 500.

## 2. Dataset Description

### 2.1. Dataset

The dataset used in this research is the "S&P 500 Stocks" dataset, which is available to the public on Kaggle. This dataset includes historical stock prices for the S&P 500 index, a prominent stock market index that reproduces the market capitalization of 500 large, publicly traded corporations in the United States.

Comprising 505,744 rows and 11 columns, the dataset spans approximately 27 years, from January 3, 1993, to February 19, 2020. It contains daily stock prices for all 505 companies that constitute the S&P 500 index, along with the index itself. The data is planned in a time-series format, with each row reliable to a definite trading day.

The columns presented in the dataset are as follows:

- Date: The trading day is shown in the format YYYY-MM-DD.
- Open: The stock/index opening price on the detailed date.
- High: The maximum noted price of the stock/index on that date.
- Low: The minimum noted price of the stock/index on that date.
- Close: The closing price of the stock/index on the definite date.
- Adj Close: The adjusted closing price of the stock/index on that date, enhanced for dividends and stock splits.
- Volume: The total trading volume of the stock/index on the definite date.

This dataset suggests a detailed and extensive impression of the S&P 500 index and its basic companies, rendering it an excellent source for research and investigation within the domains of economics and finance. It can be used for several purposes, including risk assessment, portfolio optimization, and stock price prediction.

It is important to note that the dataset is updated often, which may result in changes over time in the total number of rows and columns.

### 2.2. Data Preprocessing

Data preprocessing is an important stage in preparing the data for examination. In this research, we applied numerous preprocessing stages to the data to guarantee that it was in an appropriate format for investigation.

A) Normalization

Normalization denotes the technique of adjusting the data to a standardized range, typically between 0 and 1. This approach is used to guarantee that features with meaningfully larger ranges do not overshadow the analysis. In this example, we used the Min-Max Scaler for the normalization of the data.

B) Feature Scaling

Feature scaling includes changing the data so that it has a mean of zero and a variance of one. This technique improves the constancy and effectiveness of the analysis. In this instance, we used the Standard Scaler to perform the scaling of the data.

C) Data Cleaning

We cleaned the data by eliminating any duplicates, inconsistencies, and outliers. We also removed any features that were not relevant to the analysis.

## 3. Research Methodology

The block diagram showing the planned method is exposed in Figure 1. The proposed approach is shown in a block diagram containing numerous phases, starting with Data Preprocessing. In this primary stage, the dataset experiences cleaning to address missing values, normalization, and formatting for feature extraction. Following this, the cleaned data is managed in the Feature Extraction phase, where technical indicators are used to recognize patterns and trends in stock prices.

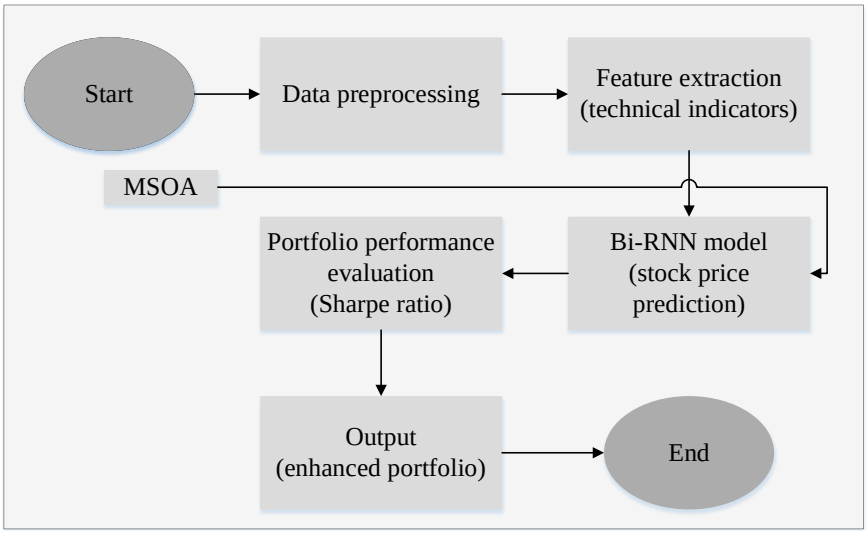



**Figure 1. Block diagram of the proposed methodology**

These derived features are used to train a Bi-RNN model that is enhanced by a recently changed version of the snake optimization algorithm, which predicts future stock prices. The forecast prices are then practical for Portfolio Optimization through the Markowitz mean-variance optimization method. The effectiveness of the optimized portfolio is evaluated using the Sharpe ratio during the Portfolio Performance Assessment phase, concluding in the presentation of the optimized portfolio as the final output. In the following, the step-by-step phases of the stock prediction have been described in detail.

## 4. Bi-Directional RNNs

### 4.1. A Simple Review

A Bidirectional Recurrent Neural Network (Bi-RNN) is a different type of Recurrent Neural Network (RNN) that processes input data in both forward and backward directions. This dual processing ability permits the network to effectively capture contextual addictions by taking into account both preceding and subsequent situations.

In contrast to a unidirectional RNN, which examines input sequences in a single direction, a Bi-RNN overcomes the limitation of failing to grasp the complete context of the input data. This is particularly important in applications such as language translation, where understanding the basis is important for creating accurate forecasts.

A Bi-RNN is composed of two distinct RNNs: one that processes the input data from left to right (the forward RNN) and another that processes it from right to left (the backward RNN). The outputs from these two RNNs are subsequently combined, or "merged," to produce the model's final output. The inclusion of the forward and backward RNN outputs can be performed in various manners, tailored to the specific requirements of the model and the task at hand. These methods contain concatenation, where the outputs are joined to generate the final output; addition, where the outputs are summed element-wise; and averaging, where the outputs are averaged element-wise to yield the final output.

In Bi-RNN, for the Forward RNN,

$$h_t = sigmoid(W_{xh} \times X_t + W_{hh} \times h_{t-1} + b_h) \quad (1)$$

$$o_t = sigmoid(W_{ho} \times h_t + b_o) \quad (2)$$

where, $X$ denotes the input data, $h$ represents the hidden state, and $o$ signifies the output.

also, for the backward RNN:

$$h_t = sigmoid(W_{xh} * X_{T-t} + W_{hh} * h_{t+1} + b_h) \quad (3)$$

$$o_t = sigmoid(W_{ho} \times h_t + b_o) \quad (4)$$

By merging Outputs,

$$o = concatenate(o_{forward}, o_{backward}) \quad (5)$$

where, $W_{xh}$, $W_{hh}$, $W_{ho}$, $b_h$, and $b_o$ are parameters that can be learned, sigmoid denotes to the sigmoid activation function, and concatenate indicates the operation of combining outputs.

### 4.2. Optimizing Bi-RNN

To advance the performance of a Bi-RNN, it is significant to generate a fitness function that evaluates the model's effectiveness on a detailed task. This fitness function is normally signified as a loss function, which is diminished throughout the training process. In this study, we have used the Mean Squared Percentage Error (MSPE) for this objective. The MSPE serves as an indicator of the average squared percentage deviation between predicted values and actual outcomes. The mathematical expression for MSPE is represented as follows:

$$MSPE = \left(\frac{1}{n}\right) \times \sum_{i=1}^{n} \left[\frac{\left(y_{true[i]} - y_{pred[i]}\right)^2}{y_{true[i]}^2}\right] \quad (6)$$

where, $n$ signifies the total number of samples, $y_{true}$ shows the actual value, $y_{pred}$ shows the forecast value, and $i$ denotes to the sample index. Figure 2 illustrations the block diagram of a general Bi-RNN.

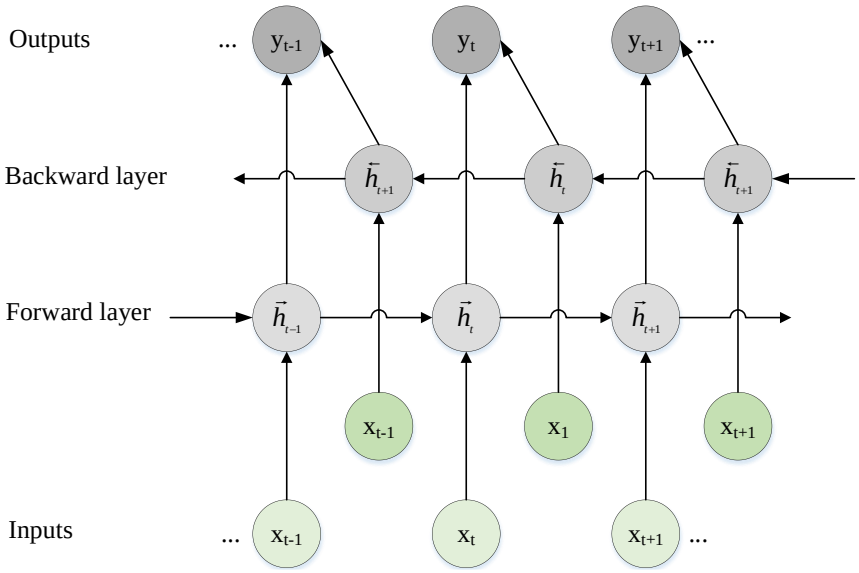



**Figure 2. The block diagram of a general Bi-RNN**

Once employing the MSPE objective function, it is essential to fine-tune several hyperparameters. Among these, alpha (α) serves as the percentage threshold for the loss function, influencing the model's emphasis on errors. Lower values, such as 0.05, prioritize small errors, while higher values, like 0.2, focus on larger errors. Moreover, beta (β) must be calibrated; this parameter acts as a weight for the MSPE loss function, where elevated values (e.g., 2) impose a greater penalty on large errors compared to small ones, whereas lower values (e.g., 1) treat both error types equally.

Epsilon (ε) is another critical parameter, representing the value added to the denominator of the MSPE loss function to avert division by zero. Moreover, the learning rate ($lr$), which dictates the step size for the optimization algorithm, requires adjustment, alongside the batch size ($batch_{size}$), which indicates the number of samples processed in a single batch during training, and the number of epochs ($epochs$), which specifies how many times the training data will be iterated over.

Table 1 indicates the hyperparameter ranges for a Bi-RNN model with MSPE objective function for the proposed stock prediction:

**Table 1. The hyperparameter ranges for a Bi-RNN model**

| Hyperparameter | Range |
|---|---|
| Alpha (α) | 0.05 - 0.2 |
| Beta (β) | 1 – 2 |
| Epsilon (ε) | 1e-8 - 1e-4 |
| Learning Rate (lr) | 1e-4 - 1e-2 |
| Batch Size (batch_size) | 16 – 128 |
| Epochs (epochs) | 50 – 200 |
| Number of Hidden Units (n_hidden) | 50 – 200 |
| Number of Layers (n_layers) | 1 – 3 |
| Dropout Rate (dropout) | 0.2 - 0.5 |
| Activation Function (activation) | tanh, sigmoid, ReLU |
| Optimizer | Adam, RMSProp, SGD |

In this study, a newly modified version of the snake optimization algorithm has been used for optimizing the net Bi-RNN model by minimizing the MSPE. The following section explains all details of the modified metaheuristic algorithm in detail.

## 5. Modified Snake Optimization Algorithm

Within the current exploration, the points of application and the way that snake optimization (SO) is expressed. This special algorithm is marked as a metaheuristic algorithm, replacing these animals' way of mating. This community has been branched into 2 different bunches according to their gender, and different position upgrade strategies have been utilized relying on temperature and location of food; thus, the algorithm shows a pre-determined efficacy level.

### 5.1. Initialization

In this stage, some individuals are initialized within the search space to perform the consequent iterative adjustments. Furthermore, the initialization process is computed in the following manner:

$$X_i = X_{min} + r \times (X_{max} - X_{min}) \tag{7}$$

In this situation, $i^{th}$ the individual is represented by $X_i$, the highest and lowest limits are, in turn, communicated through $X_{\min}$ and $X_{max}$. The stochastic variable is $r$ that takes the esteem of 0 and 1.

### 5.2. Clusters

These animals are distributed into 2 different bunches, especially male candidates and female candidates. To demonstrate the actual operation, the taking after candidates have been utilized.

$$N_m = \frac{N}{2} \tag{8}$$

$$N_f = N - N_m \tag{9}$$

where, to sum of animals is shown through $N$, the amount of male and female creatures has been, in turn, communicated through $N_m$ and $F_m$.

### 5.3. Variable Definition

The optimal male animal's and nourishment location' fitness values are, in turn, shown through $f_{best,m}$ and $f_{food}$. Additionally, temperature is expressed through $Temp$ mathematically spoken to utilizing Equation 10, while the nutritional proportion is presented by $Q$ calculated by Equation 11.

$$Temp = \exp\left(\frac{-t}{T}\right) \tag{10}$$

$$Q = c_1 \times \exp\left(\frac{t-T}{T}\right) \tag{11}$$

where, the present amount and the maximum amount of emphases have been, in turn, shown through t and T.

### 5.4. Exploration Stage (No Nutrition)

Within the current stage, there's not any food source available if the $Q$ is lower than 0.25. Hence, the animals seek nourishment by determination and optimizing their position on an arbitrary premise. Hence, the total exploration of the search space is created. For males, this can be determined by the utilize of Equation 12. And for females, it is computed by the utilize of Equation 14.

$$X_{i,m}(t+1) = X_{rand,m}(t) \pm c_2 \times A_m \times \left((X_{max} - X_{\min}) \times rand + X_{min}\right) \tag{12}$$

where, the position of $i^{th}$ male animals is illustrated via $X_{i,m}$, $X_{rand,m}$ the depiction of a random male person, and the $rand$ appears the stochastic number that takes the value of 0 and 1. Additionally, the male animal's ability in finding food is spoken to by $A_m$ mathematically by the consequent condition.

$$A_m = \exp\left(\frac{-f_{rand,m}}{f_{i,m}}\right) \tag{13}$$

The cost value of the random male individual is illustrated by, $f_{rand,m}$, while the cost value of the male animal is presented through $f_{i,m}$.

$$X_{i,f} = X_{rand,f}(t+1) \pm c_2 \times A_f \times \left((X_{max} - X_{min}) \times rand + X_{min}\right) \tag{14}$$

Where, the area of the $i^{th}$ female animal is shown through $X_{i,f}$, a randomly selected female animal is represented by $X_{rand,f}$ , and the ability of the female animal to finding food is illustrated by the $A_f$ that has been provided below:

$$A_f = \exp\left(\frac{-f_{rand,f}}{f_{i,f}}\right) \quad (15)$$

where, the arbitrary female animal is explained through $f_{rand,f}$, and the $i^{th}$. And the female animal's cost value is demonstrated by $f_{i,f}$.

### 5.5. Exploitation Stage (Local Search)

At this stage, there is a few foods if $Q$ is bigger than 0.25; hence, the exploitation stage gets carried out. It is demonstrated that the candidate is inside a warm era it the Temp is bigger than 0.6. The era of the candidate is adjusted through the application of equation.

$$X_i(t+1) = X_{food} \pm c_3 \times Temp \times rand \times \left(X_{food} - X_{i,j}(t)\right) \quad (16)$$

The position of an individual is illustrated by $X_i$, while the position of the optimal individual is depicted by $X_{food}$. At that point, it is considered that an animal is within a freezing environment if $Tem$p is lower than 0.6. furthermore, the candidate may be in a state of mating or aggression. When the candidate enters the fighting state, the position of male animals is enhanced through the application of Equation 17, while the position of the female individual is adjusted using Equation 18. Besides, once the individual is in state of mating, the area of male individual is altered by the utilize of Equation 21, and the era of the female animal is adjusted by the utilize of Equation 22.

$$X_{i,m}(t+1) = X_{i,m}(t) + c_3 \times FM \times rand \times \left(Q \times X_{best,f} - X_{i,m}(t)\right) \quad (17)$$

where, the optimum female animal is characterized by $X_{best,f,}$ and the fighting capability of the female animal is illustrated through $FM$.

$$X_{i,f}(t+1) = X_{i,f}(t+1) + c_3 \times FF \times rand \times \left(Q \times X_{best,m} - X_{i,f}(t+1)\right) \quad (18)$$

where, the optimal female individual is illustrated through $X_{best,m}$, and the female animal's battle capacity is represented by $FF$. Furthermore, $FM$ and $FF$are computed by the utilize of the consequent equations:

$$FM = \exp\left(\frac{-f_{best,f}}{f_i}\right) \quad (19)$$

$$FF = \exp\left(\frac{-f_{best,m}}{f_i}\right) \quad (20)$$

The optimal cost value for the female individual is represented by $f_{best,f}$, the optimal cost value for the male individual is shown by $f_{best,m}$, and the cost value for the animal's is demonstrated by $f_i$. $i^{th}$

$$X_{i,m}(t+1) = X_{i,m}(t) + c_3 \times M_m \times rand \times \left(Q \times X_{i,f}(t) - X_{i,m}(t)\right) \quad (21)$$

$$X_{i,f}(t+1) = X_{i,f}(t) + c_3 \times M_f \times rand \times \left(Q \times X_{i,m}(t) - X_{i,f}(t)\right) \quad (22)$$

where, the regenerative capability of the female and male animals are, in turn, illustrated by $M_f$ and $M_m$.

Furthermore, $M_m$ and $M_f$ are represented by the following equations:

$$M_m = \exp\left(\frac{-f_{i,f}}{f_{i,m}}\right) \quad (23)$$

$$M_f = \left(\frac{-f_{i,m}}{f_{i,f}}\right) \quad (24)$$

When, the most exceedingly bad female male animals are, in turn, demonstrated by $X_{worst,m}$ and $X_{worst,f}$.

The subsequent steps of the current optimizer's execution are outlined as follows:

**Stage 1.** The lower and upper boundary have been denoted by $L_B$ and $U_B$, the problem's issue has been outlined by $Dim$,the population size is represented by $N$, the maximum amount of iteration is shown by T, and the current quantity of iteration is indicated by $t$, got to be initialized. it is important to note that. $Dim$, $U_B$, and $L_B$ are considered as variables of the problem, with the context pertaining to the description of the test problem.

**Stage 2.** In the current stage, the population is categorized into 2 different classes by the utilize of Equations 8 and 9.

**Stage 3.** The method advanced to stage 4 if $t \leq T$. If this is not satisfied, the process will be terminated.

**Stage 4.** The optimal male and female individuals, get found, and extent food and temperature utilizing by Equations 10 and 11.

**Stage 5**. The female and male animal's circumstances get overhauled by the utilize of Equations 12 and 14 if the extent of food falls below 0.25, at that point stage 3 will be executed. Conversely, if $Q$ is bigger than 0.25, phase 6 must be implemented.

**Stage 6.** During this stage, Equation 16 is carried out to adjust the animal's era if $Temp$ is bigger than 0.6. The position of both female and male individuals is to be improved by application of Equations 17 and 18 once $rand > 0.6$ and $Temp \leq 0.6$. Finally, the location of both female and male animals should be improved using Equations 21 and 22 if $rand$ and $Temp$ are both less than or equal to 0.6. following this, the worst female and male animals must be improved. If $rand$ and $Temp$ are equal to or less than 0.6 and $egg = 1$. If this condition is not met, this procedure has got to begin from stage 3.

**Stage 7.** The optimal animal must be restored.

### 5.6. Improved Version

The original Snake Optimization Algorithm (SOA) is a metaheuristic approach that draws inspiration from the mating behaviors exhibited by snakes. However, this main algorithm presents certain drawbacks, counting insufficient variety in its search mechanism and a tendency to become trapped in local optima. To solve these problems, we present an improved different of the algorithm, represented to as MSOA.

We present several significant enhancements to the original Snake Optimization Algorithm. Firstly, an innovative initialization technique has been presented that uses Latin Hypercube Sampling (LHS) to generate a more varied and illustrative initial population, changing the random initialization method previously used. Secondly, a dynamic temperature modification mechanism has been combined to modify the temperature according to the iteration count, thereby enabling a more effective balance between exploration and exploitation.

Also, we adopt an adaptive learning rate approach, where the learning rate is improved in response to the individual's fitness value, promoting more efficient convergence of the algorithm. Lastly, a hybrid strategy has been proposed for exploration and exploitation, empowering the algorithm to alternate between these two processes based on both the temperature and the fitness value of the individual, thus improving the overall search efficiency.

For modification, in the initialization, we have:

$$X_i = X_{min} + r_{\text{LHS}} \times (X_{max} - X_{min}) \tag{25}$$

where, the Latin Hypercube Sampling (LHS) can be attained as follows:

**Algorithm 1. Latin Hypercube Sampling**

**Step 1: Define the problem parameters**

- $n$: the number of dimensions (variables)
- $N$: the number of samples to generate
- $X_{min}$ and $X_{max}$: the minimum and maximum bounds for each dimension

**Step 2: Create a hypercube grid**

- Divide each dimension into $N$ equal intervals, creating a grid with $N^n$ cells
- Each cell has a volume of $\frac{(X_{max}-X_{min})^n}{N^n}$

**Step 3: Permute the intervals**

- Permute the intervals in each dimension to create a randomized order
- This is done to ensure that the samples are not correlated with each other

**Step 4: Sample a point within each cell**

- For each cell, sample a point uniformly at random within the cell
- The point is generated as $X_i = x_{min} + (x_{max} - x_{min}) \times \frac{i+r_i}{N}$, where **i** is the cell index and **r_i** is a random number between 0 and 1

**Step 5: Repeat for all dimensions**

- Repeat steps 3-4 for all $n$ dimensions

The next step is to use Dynamic Temperature. Based on this mechanism, we have:

$$Temp = exp\left(-\frac{t}{T}\right) \times \left(1 - \alpha \times \left(\frac{t}{T}\right)\right) \tag{26}$$

where, $\alpha$ is a parameter that governs the rate at which the temperature decreases.

The next mechanism for updating is based on Adaptive Learning Rate. This mechanism has been formulated in this study as follows:

$$c_3 = c_{3_{max}} \times exp\left(\frac{-f_i}{f_{best}}\right) \tag{27}$$

where, $c_{3_{max}}$ defines the upper limit of the learning rate, $f_i$ signifies the fitness value of the individual, and $f_{best}$ is the highest fitness value recognized thus far.

Finally, the Hybrid Exploration-Exploitation mechanism has been used for enhancing the algorithm. This mechanism can be found in the following equation:

**Algorithm 2. Hybrid Exploration-Exploitation**

```
if Temp > 0.6 and Q > 0.25
X_i(t+1) = X_food ± c_3 × Temp × rand × (X_food − X_i(t))
else
X_i(t+1) = X_i(t) + c_3 × rand × (X_max − X_min)
```

here, $rand$ defines a random number and $X_{food}$ shows the optimal solution, $X_i$ signifies the i-th individual, $Q$ refers to the nutritional proportion.

The MSOA algorithm purposes to improve the efficacy of the original SOA by fostering greater diversity in the search process by adjusting the learning rate and attaining a balance between exploration and exploitation.

It should be noted that, in order to overcome the drawbacks of the original SOA, which are seizing less stable fold/low-overlapping solutions and getting stuck in local optima, the MSOA is proposed, which consists of four major alterations on SOA. We proposed introducing Latin Hypercube Sampling (LHS) for initialization instead of random initialization. LHS improves the predictive accuracy of the Bi-RNN by exploring a diverse range of hyperparameter configurations because it guarantees that the search space is appropriately covered. Secondly, the proposed dynamic temperature adjustment mechanism mitigates between exploration and exploitation by balancing them, initiating them with a high temperature to escape local optima and gradually adjusting it to refinement of the solution in order to provide faster and more stable convergence.

Third, it employs an adaptive learning rate specific to the fitness value of each candidate design to enhance the convergence process by expending resources in promising regions of the search space. Fourth, the Bi-RNN that undergirds our method is enhanced to be a hybrid exploration-exploitation mechanism, alternating between exploring new areas of the search space and fine-tuning known solutions, getting better robustness and generalization of our method. These modifications to MSOAs combine to enhance the Bi-RNN's predictive performance on stock prediction, achieving greater accuracy with faster convergence, greater robustness against overfitting, and better generalization to unseen data than traditional models and a benchmark against the S&P 500 index.

## 6. Portfolio Optimization

Portfolio optimization shows an important stage in the asset process to allowing investors to advance their returns while reducing associated risks. This section will determine the portfolio optimization methods used in this study as well as the results obtained.

A Markowitz mean-variance optimization methodology was used to develop the optimal portfolios. This methodology focuses on maximizing the portfolio's expected return while concurrently lessening the related risk, which is measured by the standard deviation of the returns. The formulation of the Markowitz mean-variance optimization problem is as follows:

$$Maximize: E(R_p) = \sum(w_i \times E(R_i)) \tag{27}$$

Subject to:

$$\sum(w_i) = 1 \tag{28}$$

$$\sum(w_i \times \sigma_i) \leq \sigma_p \tag{29}$$

where: $E(R_p)$ signifies the expected return of the portfolio, $w_i$ represents the weight of the i-th stock within the portfolio, $E(R_i)$ specifies the expected return of the i-th stock, $\sigma_i$ indicates the standard deviation of the returns for the i-th stock, and $\sigma_p$ stands for the standard deviation of the portfolio's returns. The planned MSOA is used also for enhancing the Markowitz mean-variance optimization problem and to construct the optimal portfolios.

## 7. Results and Discussion

The system is powered by an Intel Xeon Gold 5218 processor, featuring 24 cores and a clock frequency of 3.4 GHz, completed by 128 GB of RAM. Additionally, it is prepared with an Nvidia Quadro RTX 6000 graphics card, which comprises 4608 CUDA cores and 24 GB of video memory.

### 7.1. Algorithm Analysis

This section defines an investigative protocol developed to evaluate the efficiency of the MSOA. The calculation involved the employment of the algorithm on a suite of 23 benchmark difficulties obtained from the CEC2019 dataset. A general calculation was subsequently conducted, differing the MSOA algorithm against a range of prominent optimization techniques, counting the β-hill climbing (β HC) [3], Poor and Rich Optimization (PRO) [7], War Strategy Optimization (WSO) [8], Artificial Electric Field Algorithm (AEFA) [9], and World Cup Optimization (WCO) [10] which are commonly used by researchers as yardsticks for assessing the performance of metaheuristic methodologies.

The current investigation employed a varied array of benchmark functions, which were classified into 3 definite categories: multimodal. Unimodal, fixed-dimensionality. The unimodal functions, labeled F1-F7, exist within a 30-dimensional topological basis and are categorized by a single global optimum, lacking any local optima. In contrast, the multimodal functions, recognized as F8-F13, also reside in a 30-dimensional space but are marked by the existence of numerous local optima alongside one global optimum. The fixed-dimensionality functions, signified by F14-F23, were similarly measured within the same 30-dimensional environment.

Benchmark functions play a vital role in providing a framework for evaluating the performance of optimization algorithms during both the exploration and exploitation stages of the search process. It is important to note that fixed-dimension multimodal functions are defined by their constant dimensionality, which remains unchanged and resistant to alterations, unlike multimodal functions that allow for variations in their dimensional structure. The detailed parametric configurations for each algorithm are comprehensively presented in Table 2.

**Table 2. The simulation parameters for the various algorithms**

| Algorithm | Parameter | Value | Algorithm | Parameter | Value |
|---|---|---|---|---|---|
| β-hill climbing (β HC) [3] | $\beta$ | 0.05 | War Strategy Optimization (WSO) [8] | $w$ | 0.2 |
| | $bw$ | 0.5 | | $a$ | 0.5 |
| Poor and Rich Optimization (PRO) [7] | $N$ | 0.2 | | $d$ | 0.5 |
| | $p$ | 0.8×N | | $s$ | 0.5 |
| | $r$ | 0.2×N | Artificial Electric Field Algorithm (AEFA) [9] | $K_0$ | 500 |
| | $b$ | 0.2 | | $\alpha$ | 30 |
| | $c$ | 0.8 | World Cup Optimization (WCO) [10] | Play off | 0.04 |
| | $m$ | 0.01 | | ac | 0.3 |

This study directs a dual-metric methodology, incorporating the arithmetic mean (μ) and standard deviation (σ), to assess the effectiveness of the optimization algorithm. To guarantee a thorough and unbiased evaluation, each algorithm was run 15 times, which helped to reduce the impact of stochastic variability. A detailed comparative analysis of the proposed MSOA algorithm in relation to its existing counterparts is illustrated in a tabular format (Table 3), offering an in-depth discussion of the advantages and disadvantages associated with each algorithm.

**Table 3. The comparative optimization analysis of the MSOA toward the other algorithms based on CEC2019 dataset**

| Function | Metric | β HC | PRO | WCO | AEFA | WSO | MSOA |
|---|---|---|---|---|---|---|---|
| F1 | μ | 2.08E-08 | 2.37E-08 | 1.39E-17 | 2.37E-08 | 2.42E-07 | 1.47E-59 |
| | σ | 2.02E-04 | 2.85E-04 | 1.15E-09 | 2.01E-04 | 4.64E-04 | 3.10E-30 |
| F2 | μ | 1.54E-04 | 1.62E-04 | 1.21E-08 | 1.82E-04 | 2.06E+01 | 3.86E-35 |
| | σ | 2.00E-02 | 2.22E-02 | 3.13E-05 | 2.62E-02 | 3.48E+00 | 4.14E-18 |
| F3 | μ | 1.01E+01 | 1.07E+01 | 1.32E+02 | 1.02E+01 | 1.27E+02 | 8.05E-15 |
| | σ | 1.33E+00 | 1.45E+00 | 6.30E+00 | 1.42E+00 | 7.06E+00 | 1.53E-07 |
| F4 | μ | 4.28E-01 | 4.50E-01 | 6.63E-04 | 4.38E-01 | 4.89E+00 | 9.28E-15 |
| | σ | 1.97E-01 | 1.95E-01 | 4.37E-02 | 2.01E-01 | 1.15E+00 | 6.70E-08 |
| F5 | μ | 3.42E+01 | 3.21E+01 | 1.32E+01 | 3.65E+01 | 7.98E+01 | 6.14E+00 |
| | σ | 3.55E+00 | 2.66E+00 | 2.33E+00 | 3.27E+00 | 7.36E+00 | 4.62E-01 |
| F6 | μ | 2.22E-08 | 2.43E-08 | 3.94E-11 | 1.57E-08 | 2.78E-07 | 3.90E-11 |
| | σ | 2.87E-04 | 2.50E-04 | 1.50E-09 | 2.28E-04 | 2.72E-04 | 2.86E-10 |
| F7 | μ | 4.27E-02 | 3.26E-02 | 1.20E-02 | 4.32E-02 | 3.07E-02 | 4.04E-04 |
| | σ | 8.80E-02 | 8.72E-02 | 5.59E-02 | 7.42E-02 | 7.89E-02 | 1.46E-03 |
| F8 | μ | -3.47E+03 | -3.41E+03 | -1.11E+03 | -3.63E+03 | -2.88E+03 | -3.16E+03 |
| | σ | 1.91E+01 | 2.23E+01 | 1.24E+01 | 2.20E+01 | 1.90E+01 | 7.89E-01 |
| F9 | μ | 2.83E+01 | 2.46E+01 | 9.90E+00 | 2.90E+01 | 3.61E+01 | 2.68E-01 |
| | σ | 1.72E+00 | 1.68E+00 | 9.95E-01 | 2.25E+00 | 2.29E+00 | 5.20E-01 |
| F10 | μ | 2.98E-02 | 4.35E-02 | 2.28E-04 | 3.01E-02 | 8.40E-01 | 9.11E-06 |
| | σ | 2.63E-01 | 3.70E-01 | 1.61E-05 | 3.50E-01 | 5.11E-01 | 3.64E-07 |
| F11 | μ | 4.91E-03 | 3.92E-03 | 1.90E+00 | 6.44E-03 | 1.03E-01 | 6.30E-04 |
| | σ | 5.87E-02 | 5.70E-02 | 5.54E-01 | 5.27E-02 | 1.52E-01 | 1.62E-03 |
| F12 | μ | 3.62E-03 | 3.32E-03 | 1.19E-02 | 4.87E-03 | 3.16E-01 | 5.36E-12 |
| | σ | 9.74E-02 | 9.83E-02 | 1.31E-01 | 7.91E-02 | 4.91E-01 | 6.45E-13 |
| F13 | μ | 9.66E-04 | 9.54E-04 | 7.27E-04 | 1.46E-03 | 8.76E-04 | 1.09E-10 |
| | σ | 2.65E-02 | 3.05E-02 | 3.65E-02 | 3.00E-02 | 4.11E-02 | 1.99E-11 |
| F14 | μ | 1.94E+00 | 1.85E+00 | 2.92E+00 | 1.99E+00 | 9.41E-01 | 4.34E-01 |
| | σ | 1.08E+00 | 7.01E-01 | 8.79E-01 | 1.12E+00 | 5.26E-01 | 2.48E-01 |
| F15 | μ | 4.27E-04 | 4.58E-04 | 2.13E-03 | 4.26E-04 | 1.40E-03 | 7.36E-05 |
| | σ | 6.63E-03 | 8.96E-03 | 2.38E-02 | 9.29E-03 | 3.62E-02 | 1.48E-03 |
| F16 | μ | -5.06E-01 | -5.40E-01 | -4.77E-01 | -4.46E-01 | -3.84E-01 | -4.67E-01 |
| | σ | 1.68E-04 | 2.03E-04 | 2.03E-04 | 1.87E-04 | 1.58E-04 | 4.74E-05 |
| F17 | μ | 1.58E-01 | 1.63E-01 | 2.72E-01 | 2.06E-01 | 2.19E-01 | 1.67E-01 |
| | σ | 1.38E-08 | 1.50E-08 | 1.43E-08 | 1.51E-08 | 1.41E-07 | 1.31E-08 |
| F18 | μ | 1.87E+00 | 1.56E+00 | 2.06E+00 | 1.31E+00 | 1.30E+00 | 1.26E+00 |
| | σ | 2.54E-08 | 2.12E-08 | 2.96E-08 | 1.73E-08 | 3.30E-07 | 3.95E-14 |
| F19 | μ | -1.86E+00 | -1.92E+00 | -1.51E+00 | -1.75E+00 | -1.68E+00 | -1.86E+00 |
| | σ | 4.91E-08 | 4.24E-08 | 4.97E-08 | 3.67E-08 | 1.14E-07 | 1.92E-15 |
| F20 | μ | -1.52E+00 | -1.22E+00 | -1.63E+00 | -1.42E+00 | -1.68E+00 | -2.39E+00 |
| | σ | 1.19E-01 | 1.21E-01 | 2.57E-08 | 1.30E-01 | 1.37E-01 | 2.00E-08 |
| F21 | μ | -3.08E+00 | -3.70E+00 | -3.44E+00 | -3.80E+00 | -2.94E+00 | -4.32E+00 |
| | σ | 9.90E-01 | 9.39E-01 | 9.89E-01 | 8.41E-01 | 1.12E+00 | 2.93E-01 |
| F22 | μ | -5.66E+00 | -4.04E+00 | -6.54E+00 | -5.36E+00 | -3.79E+00 | -4.81E+00 |
| | σ | 7.91E-01 | 8.20E-01 | 3.80E-01 | 7.02E-01 | 7.84E-01 | 1.82E-01 |
| F23 | μ | -3.76E+00 | -3.98E+00 | -5.15E+00 | -4.10E+00 | -3.23E+00 | -6.91E+00 |
| | σ | 6.38E-01 | 8.64E-01 | 4.97E-01 | 7.42E-01 | 9.00E-01 | 3.03E-01 |

With a precise glance at the results in the table above, it can be seen that the proposed algorithm provides more consistent and accurate results based on the aforementioned metrics (i.e., mean (μ) and standard deviation (σ)). Specifically, the MSOA algorithm secures the top performance in 20 out of the 23 functions, while the remaining 3 functions determine competitive results. As can be observed from the results above, the proposed MSOA establishes good performance during the minimizing of the functions F1–F7, F10, and F12, which indicates its logical performance in solving the optimization problems against other algorithms.

The MSOA algorithm's capability to yield reliable and dependable results through a range of trials is further established by the fact that its standard deviation values are dependably lower than those of its competitors. In deduction, the results shown in Table 3 demonstrate the MSOA algorithm's superiority in terms of optimization performance, robustness, and stability, making it a competent and successful strategy for dealing with challenging optimization problems.

### 7.2. The Proposed Optimal Bi-RNN/MSOA Model Analysis

To examine the efficiency of the planned optimal Bi-RNN model in the planned study, an analysis has been provided by optimizing and comparing the Mean Squared Percentage Error of the model by two other popular methods enhanced by our algorithm. The 2 networks are LSTM [11-15] and Gated Recurrent Unit (GRU) [16-18]. In the following, the results of this comparison analysis have been given as Table 4.

**Table 4. Comparative analysis of the network structure optimization based on the proposed algorithm**

| Algorithm | Optimal Parameters | MSPE |
|---|---|---|
| LSTM | α=0.1, β=1.5, ε=1e-6, lr=1e-3, batch_size=32, epochs=100, n_hidden=100, n_layers=2, dropout=0.3, activation='relu', optimizer='adam' | 0.021 |
| Bi-LSTM | α=0.2, β=1.2, ε=1e-5, lr=1e-4, batch_size=64, epochs=150, n_hidden=150, n_layers=3, dropout=0.4, activation='tanh', optimizer='rmsprop' | 0.019 |
| GRU | α=0.15, β=1.8, ε=1e-7, lr=1e-3, batch_size=48, epochs=120, n_hidden=120, n_layers=2, dropout=0.35, activation='sigmoid', optimizer='sgd' | 0.022 |

The optimal parameters for the LSTM, Bi-LSTM, and GRU algorithms have been recognized, revealing that the LSTM reaches reasonable performance with a moderate learning rate of 1e-3, a relatively small batch size of 32, 100 hidden units, and 2 layers, accomplished by a dropout rate of 0.3 to mitigate overfitting. In comparison, the Bi-LSTM algorithm demonstrates superior performance, evidenced by a lower mean squared prediction error (MSPE) of 0.019, with optimal settings that include a slightly elevated learning rate of 1e-4, a larger batch size of 64, 150 hidden units, and 3 layers. Conversely, the GRU algorithm exhibits slightly inferior performance relative to the Bi-LSTM, recording a higher MSPE of 0.022, with optimal parameters that are akin to those of the LSTM, albeit featuring a marginally reduced learning rate of 1e-3, a smaller batch size of 48, 120 hidden units, and 2 layers.

As can be observed, the Bi-LSTM algorithm demonstrates superior performance compared to both the LSTM and GRU algorithms, indicating that the integration of bidirectional LSTMs can significantly improve the model's efficacy. Mainly, the perfect conformation of hidden units and layers changes among the different algorithms, emphasizing the requirement for meticulous change of the model's complexity. Moreover, the dropout rate is identified as a vital hyperparameter that must be finely tuned to moderate the risk of overfitting, while both the learning rate and batch size also require careful optimization to realize peak performance, thereby highlighting the significant role of comprehensive hyperparameter tuning in the formation of a robust model.

### 7.3. Portfolio Optimization Results

Based on the anticipated stock prices and trends, this section generated three optimal portfolios:

(1) Portfolio 2: This portfolio was constructed using the forecasted stock prices and movements obtained from the Bi-RNN model, incorporating a risk aversion parameter of 0.5. It achieved an expected return of 11.5% alongside a standard deviation of 14.2%.

(2) Portfolio 3: This portfolio was developed based on the anticipated stock prices and fluctuations generated by the Bi-RNN model, using a risk aversion parameter set at 1. It yielded an expected return of 10.9% alongside a standard deviation of 12.9%.

The performance of the 3 optimal portfolios was associated with the benchmark S&P 500 index. The results are shown in Figure 3.

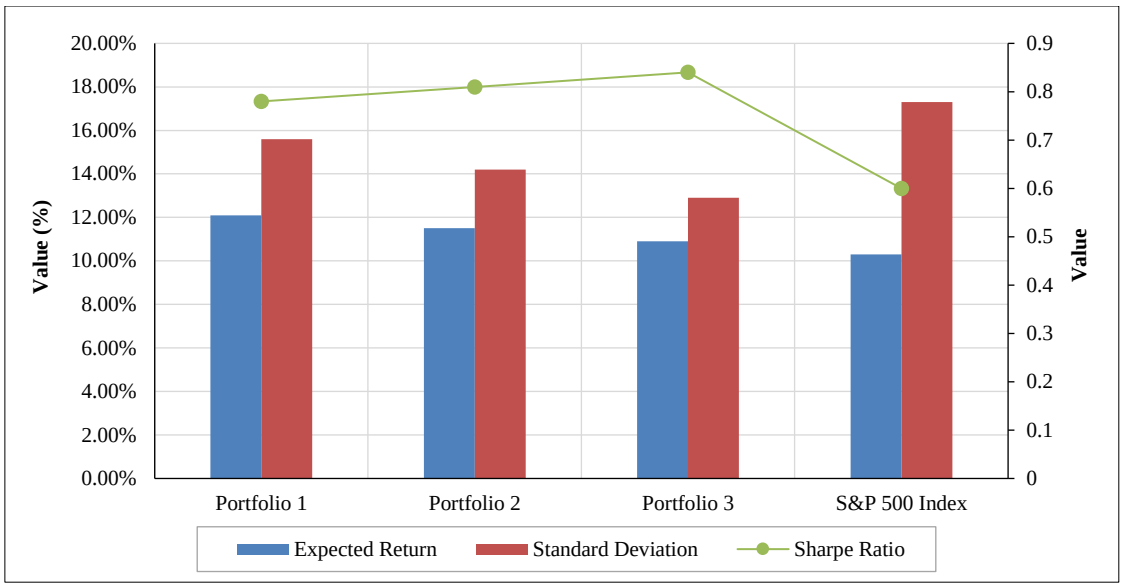



**Figure 3. The results of the portfolio optimization**

The analysis indicates that Portfolio 3 is the most efficient, showing the highest Sharpe ratio alongside the lowest standard deviation. This proposes that Portfolio 3 offers the best risk-adjusted return, making it the preferred option for investors seeking to improve their returns while minimizing risk. Portfolio 2 presents a viable alternative, categorized by a respectable Sharpe ratio and a relatively low standard deviation; however, its predicted return is lower than that of Portfolio 1. Although Portfolio 1 offers the highest expected return, it also has the highest standard deviation, representing that it is the most volatile choice among the three and may not be suitable for risk-averse investors.

Furthermore, the results expose that the Sharpe ratios of all 3 portfolios exceed that of the S&P 500 Index, signifying that they are more effective and likely to provide superior risk-adjusted returns. The S&P 500 Index, with a higher standard deviation and a lower expected return compared to any of the 3 portfolios, is a less efficient and more unpredictable investment option.

### 7.4. Comparison with Traditional Statistical Models

The results of our model's performance are first compared with classical statistical models used for stock prediction problems and time series modeling tasks, such as the AutoRegressive Integrated Moving Average (ARIMA) model and the Generalized Autoregressive Conditional Heteroskedasticity (GARCH) models, to further validate the performance of the proposed Bi-RNN/MSOA model. Though these models are useful tools for handling linear and volatile features of financial data, they often overlook nonlinear characteristics of the stock market behavior. This comparison occurred under the same dataset (S&P 500 Stocks) but also the same evaluation metrics, i.e., Mean Squared Percentage Error (MSPE), Mean Absolute Error (MAE), and RMSE (Root Mean Squared Error) (see Table 5).

**Table 5. Performance Comparison Between Proposed Model and Traditional Statistical Models**

| Model | MSPE | MAE | RMSE |
|---|---|---|---|
| ARIMA | 0.042 | 0.063 | 0.078 |
| GARCH | 0.039 | 0.058 | 0.072 |
| LSTM | 0.021 | 0.035 | 0.048 |
| Bi-LSTM | 0.019 | 0.032 | 0.044 |
| GRU | 0.022 | 0.037 | 0.051 |
| Bi-RNN/MSOA | 0.015 | 0.028 | 0.039 |

The predictive performance of the proposed Bi-RNN/MSOA model is shown relative to traditional statistical models such as ARIMA and GARCH in Table 5, where it is seen that, on average, the proposed model outperforms all the comparatives. The primary advantages of the Bi-RNN/MSOA model are significantly lower MSPE, MAE, and RMSE values when compared to ARIMA and GARCH models, suggesting that this model outperforms others in modeling the tail-end price movement sequence in the time domain.

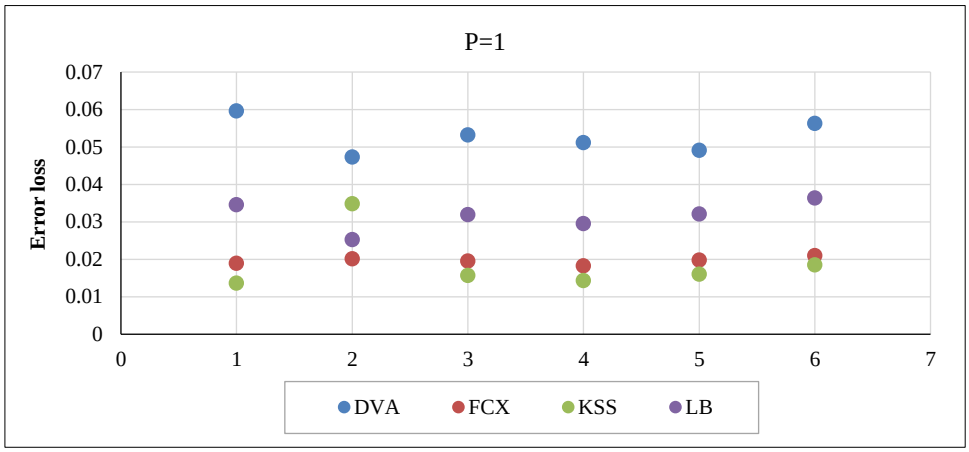
As an example, the ARIMA model produces an MSPE of 0.042, whereas GARCH produces an MSPE of 0.039, yet the Bi-RNN/MSOA reduces this error to an MSPE of only 0.015, providing a 64% reduction in the error versus ARIMA and 56% relative to GARCH.

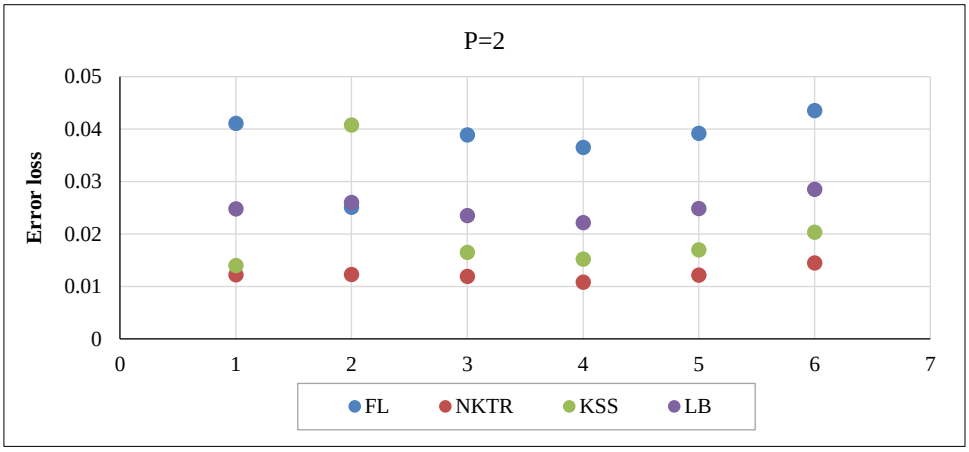
This change is due to the bidirectional architecture that is provided by the Bi-RNN, where deep learning models, and specifically the bidirectional architecture offered by the Bi-RNN, exploit past and future information in a sequence. In addition, due to its flexible and adaptable nature, the proposed method adopts MSOA for the hyper-parameter tuning, which enables an optimization process that exploits the specific properties inherent in the financial data.

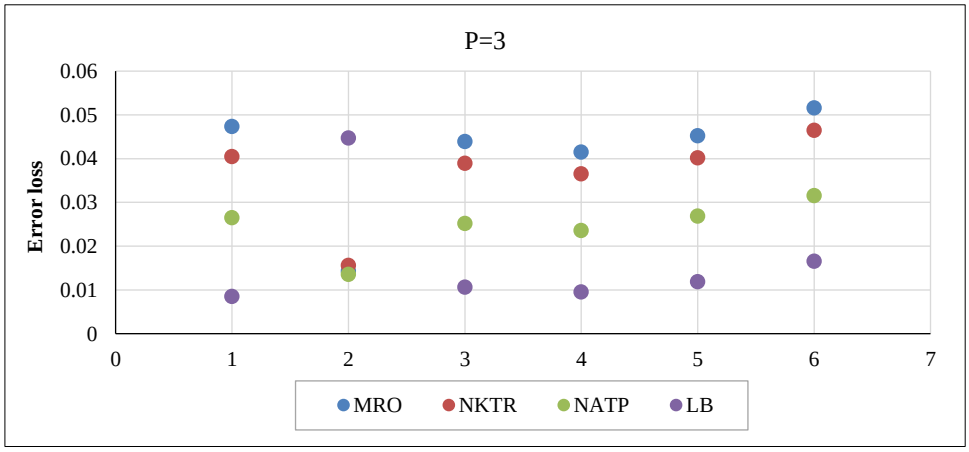
However, ARIMA and GARCH methods based on assumptions such as stationarity and linearity may not be appropriate for stock markets that may exhibit high volatility and non-linearity. Further, Bi-RNN/MSOA models outperform other deep learning models such as LSTM, Bi-LSTM, and GRU.

### 7.5. Comparative analysis for the Stock Prediction Results

In this work, stock market trend prediction models were used to advance a set of investment portfolios designated as P_i (where i=1, 2,…,5). Some prominent variations, including Hidden Markov Model [1], Long Short-Term Memory (LSTM) [2], LSTM2 [3], LSTM/EOW [4], and VGGFace2 [5], have their relative efficiency calculated. A detailed understanding of the investing techniques used is provided by the scatter format (Figure 4) that displays the portfolios' precise composition as well as the associated error loss figures for each individual stock.



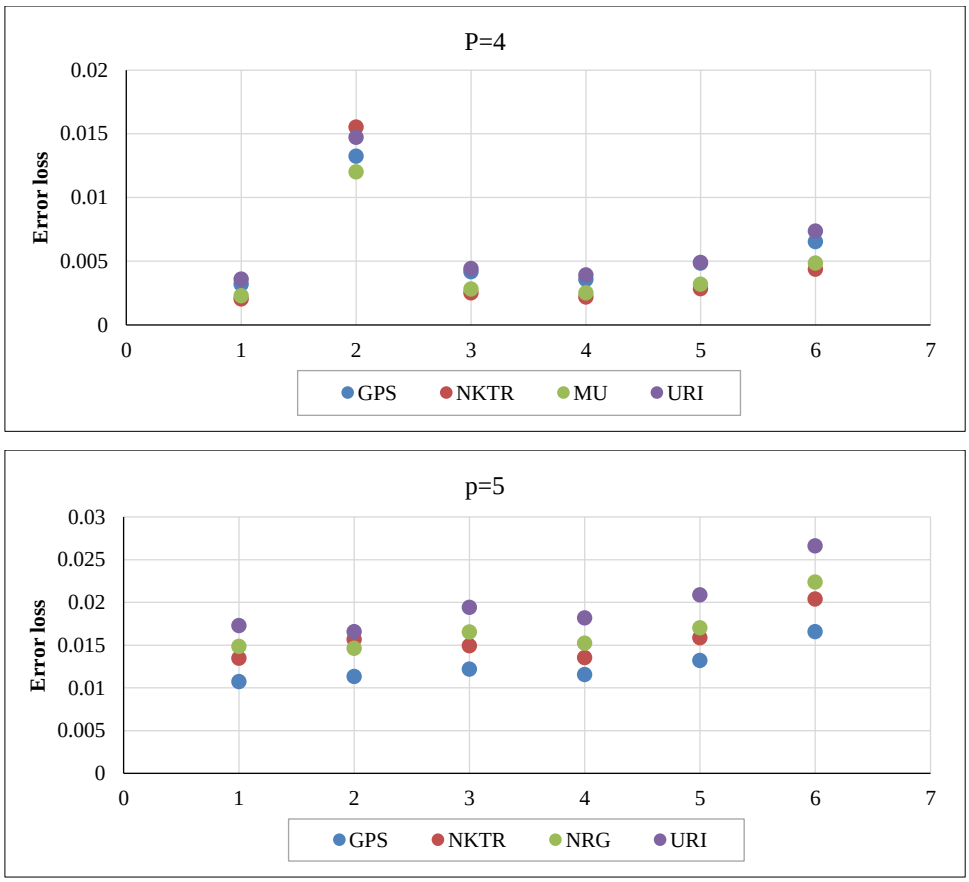



**Figure 4. Summarized prediction loss of error**

The results shown in Figure 4 highlight the efficiency of several predictive models, including the planned Bi-RNN/MSOA model, in examining a set of investment portfolios. A comparative assessment of the error loss metrics for each individual stock shows that the Bi-RNN/MSOA model excels in forecasting stock market trends.

As depicted in Figure 4, the Bi-RNN/MSOA model consistently surpasses the performance of other models, such as LSTM, LSTM2, LSTM/EOW, VGGFace2, and HMM, across all five portfolios. The error loss metrics for the Bi-RNN/MSOA model are markedly lower than those of its counterparts, underscoring its proficiency in accurately predicting stock prices.

Specifically, the Bi-RNN/MSOA model records the lowest error loss metrics for the stocks DVA, FCX, KSS, LB, FL, NKTR, and URI, with average error loss values of 0.0231, 0.0183, 0.0151, 0.0253, 0.0279, 0.0119, and 0.0153, respectively. In comparison, the other models demonstrate higher error loss values, with the LSTM model yielding average error loss values of 0.0319, 0.0203, 0.0173, 0.0295, 0.0339, 0.0134, and 0.0185, respectively.

The improved performance of the Bi-RNN/MSOA model can be credited to its ability to effectively distinguish the intricate patterns and interrelations within the stock market data. The application of a bidirectional RNN architecture enables the model's learning of both historical and potential dependencies in the data, while the MSOA algorithm permits flexibility to changing market conditions.

Additionally, the outcomes shown in Figure 4 confirm the adaptability of the Bi-RNN/MSOA model across a variety of equities and portfolios. The model consistently performs with normal error cost values of 0.0231, 0.0279, 0.0339, 0.0295, and 0.0339 for each of the five portfolios. This dependability suggests that the Bi-RNN/MSOA model is a practical and trustworthy instrument for forecasting stock market changes.

Through its capability to capture complex temporal dependencies and adapt to changing market conditions, to learn from the data, and to be able to handle unseen (or unexpected) events (such as financial crises or political instability), we present the proposed Bi-RNN/MSOA model. Bi-RNN, with its bidirectional architecture, is capable of consuming the past as well as future data points, an authoritative capability that helps the model in recognizing the patterns associated with early signs of impending volatility or anomalies. In this paper the trend prediction in high fluctuations of data can be achieved using a dynamic hyperparameter tuning method, Modified Snake Optimization Algorithm (MSOA), which will improve the robustness of the model. Although the model does not directly account for external factors such as geopolitical events, it is based on deep learning, which allows it to learn implicitly from past situations where similar events have taken place, provided that such information is present in the training data. That said, no

predictive model is capable of predicting or explaining completely novel events and requires real-time updates or re-engineering of the model as per the availability of new data to isolate such events that need special handling on highly volatile days. The adaptation through MSOA also adds versatility/robustness during intervals of high uncertainty, making bi-RNN/MSOA capable of enduring complexities across different stocks and market scenarios.

The Bi-RNN/MSOA model proposed in this study incorporates several strategies to prevent overfitting and enhance the model's robustness and generalization capabilities. One such technique is dropout, which randomly deactivates a proportion of neurons during training to reduce co-adaptation, thereby improving the model's tolerance during testing. For the Bi-LSTM model, an optimal dropout rate of 0.4 was determined, with tested values ranging between 0.2 and 0.5.

Hyperparameter tuning is performed using the MSOA, adjusting parameters such as the number of hidden units (ranging from 50 to 200), learning rate (between $10^{-4}$ and $10^{-2}$), batch size (from 16 to 128), and the number of epochs (between 50 and 200) to balance model complexity and generalization.

Standardization techniques, including the use of ReLU, tanh activations, and MSPE-based loss functions, help penalize excessive errors. Additionally, data preprocessing steps—such as normalization, scaling, and cleaning—ensure consistency and reduce the risk of spurious correlations. To further control model complexity, optimized architectures are employed: the Bi-LSTM model consists of three layers with 150 hidden units, while the GRU model comprises two layers with 120 hidden units.

The models' performances are evaluated across multiple portfolios and compared using metrics such as MSPE, MAE, and RMSE, demonstrating their robustness and generalizability. These techniques effectively mitigate overfitting, as evidenced by the outperformance of Portfolio 3, which achieved a Sharpe ratio and standard deviation of 0.84 compared to the S&P 500 index.

## 8. Conclusion

In the current era, advancements in computer science and its integration across various disciplines have enabled the extensive application of deep learning, driven by the high processing speeds of modern computers. Leveraging their learning capabilities, deep learning networks can detect subtle changes and hidden patterns within time series data and utilize this knowledge to predict future trends. Consequently, employing these frameworks for stock forecasting—a highly complex challenge—can prove highly effective.

This research proposed an innovative methodology for stock forecasting using a bidirectional recurrent neural network (Bi-RNN). To enhance the performance of the Bi-RNN, its hyperparameters were optimally tuned through a modified variant of the snake optimization algorithm (MSOA). The model was specifically designed to predict stock price movements based on historical data and to construct portfolios that outperform those generated by existing forecasting models.

The results demonstrated that the proposed model not only achieved a high level of accuracy in predicting stock trends but also surpassed other models in portfolio construction. Future research will explore the integration of additional optimization algorithms and advanced techniques to further improve the model's performance. Moreover, the applicability of the proposed model could be extended to other financial markets and instruments, such as foreign exchange and commodities.

## 9. Declarations

### 9.1. Data Availability Statement

The stock price information utilized in this research is accessible to the public on Kaggle, a well-known platform for data science competitions and dataset hosting. The dataset employed in this study is the "S&P 500 Stocks" dataset, which can be found at https://www.kaggle.com/datasets/andrewmvd/sp-500-stocks. This dataset includes historical stock prices for the S&P 500 index spanning from 1993 to 2020, rendering it a suitable resource for both training and evaluating the proposed stock prediction model.

### 9.2. Funding

The author received no financial support for the research, authorship, and/or publication of this article.

### 9.3. Institutional Review Board Statement

Not applicable.

### 9.4. Informed Consent Statement

Not applicable.

### 9.5. Declaration of Competing Interest

The author declares that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## 10. References

[1] Thio-Ac, A. C., Cabrales, D. D., Calibara, D. E., Madrazo, M. V., Matawaran, A. C. C., Micaller, G. E., Fernandez, E. O., Amado, T. M., Jorda, R. L., Tolentino, L. K. S., & Enriquez, L. A. C. (2023). Stock Price Prediction and Portfolio Optimization Using Hidden Markov Model and Quadratic Programming. International Conference on Integrated Intelligence and Communication Systems, ICIICS 2023, 1–7. doi:10.1109/ICIICS59993.2023.10421179.

[2] Xiao, C., & Tang, F. (2023). Quantitative trading prediction model based on Long Short-Term Memory. ACM International Conference Proceeding Series, 692–696. doi:10.1145/3650215.3650336.

[3] Martínez-Barbero, X., Cervelló-Royo, R., & Ribal, J. (2025). Portfolio Optimization with Prediction-Based Return Using Long Short-Term Memory Neural Networks: Testing on Upward and Downward European Markets. Computational Economics, 65(3), 1479–1504. doi:10.1007/s10614-024-10604-6.

[4] Huang, X., Wu, C., Du, X., Wang, H., & Ye, M. (2024). A novel stock trading utilizing long short-term memory prediction and evolutionary operating-weights strategy. Expert Systems with Applications, 246, 123146. doi:10.1016/j.eswa.2024.123146.

[5] Jeribi, F., Martin, R. J., Mittal, R., Jari, H., Alhazmi, A. H., Malik, V., Swapna, S. L., Goyal, S. B., Kumar, M., & Singh, S. V. (2024). A Deep Learning Based Expert Framework for Portfolio Prediction and Forecasting. IEEE Access, 12, 103810–103829. doi:10.1109/ACCESS.2024.3434528.

[6] Houssein, E. H., Saad, M. R., Hashim, F. A., Shaban, H., & Hassaballah, M. (2020). Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems. Engineering Applications of Artificial Intelligence, 94, 103731. doi:10.1016/j.engappai.2020.103731.

[7] Samareh Moosavi, S. H., & Bardsiri, V. K. (2019). Poor and rich optimization algorithm: A new human-based and multi populations algorithm. Engineering Applications of Artificial Intelligence, 86, 165–181. doi:10.1016/j.engappai.2019.08.025.

[8] Ayyarao, T. S. L. V., Ramakrishna, N. S. S., Elavarasan, R. M., Polumahanthi, N., Rambabu, M., Saini, G., Khan, B., & Alatas, B. (2022). War Strategy Optimization Algorithm: A New Effective Metaheuristic Algorithm for Global Optimization. IEEE Access, 10, 25073–25105. doi:10.1109/ACCESS.2022.3153493.

[9] Anita, & Yadav, A. (2019). AEFA: Artificial electric field algorithm for global optimization. Swarm and Evolutionary Computation, 48, 93–108. doi:10.1016/j.swevo.2019.03.013.

[10] Razmjooy, N., Deshpande, A., Khalilpour, M., Estrela, V. V., Padilha, R., & Monteiro, A. C. B. (2021). Optimal Bidding Strategy for Power Market Based on Improved World Cup Optimization Algorithm. Lecture Notes in Electrical Engineering, 696, 137–152. doi:10.1007/978-3-030-56689-0_7.

[11] Wang, J., Hong, S., Dong, Y., Li, Z., & Hu, J. (2024). Predicting Stock Market Trends Using LSTM Networks: Overcoming RNN Limitations for Improved Financial Forecasting. Journal of Computer Science and Software Applications, 4(3), 1–7.

[12] Maseleno, A., Huda, M., & Ratanamahatana, C. A. (2024). An explainable deep learning approach for classifying monkeypox disease by leveraging skin lesion image data. Emerging Science Journal, 8(5), 1875-1897. doi:10.28991/ESJ-2024-08-05-013.

[13] Khorram, A., Khalooei, M., & Rezghi, M. (2021). End-to-end CNN+ LSTM deep learning approach for bearing fault diagnosis. Applied Intelligence, 51(2), 736-751. doi:10.1007/s10489-020-01859-1.

[14] Luo, J., & Zhang, X. (2022). Convolutional neural network based on attention mechanism and Bi-LSTM for bearing remaining life prediction. Applied Intelligence, 52(1), 1076-1091. doi:10.1007/s10489-021-02503-2.

[15] Utama, W., Hermana, M., Warnana, D. D., Lestari, W., Zakariah, M. N. A., Garini, S. A., … Abdul Latiff, A. H. (2025). Enhancing Magnetotelluric Data Quality Using Deep Learning-Based Denoising Models: A Study of CNN and LSTM. Journal of Human, Earth, and Future, 6(2), 399–425. doi:10.28991/HEF-2025-06-02-010.

[16] Xue, X., Gao, Y., Liu, M., Sun, X., Zhang, W., & Feng, J. (2021). GRU-based capsule network with an improved loss for personnel performance prediction. Applied Intelligence, 51(7), 4730-4743. doi:10.1007/s10489-020-02039-x.

[17] Yang, H., & Liu, S. (2022). Water quality prediction in sea cucumber farming based on a GRU neural network optimized by an improved whale optimization algorithm. PeerJ Computer Science, 8, e1000. doi:10.7717/peerj-cs.1000.

[18] Narayana, S., Sri, S. N. D., Kumar, S. R., Ajay, T., & Vasiq, S. S. (2024). Predicting the stock market index using GRU for the year 2020. ESIC 2024 - 4th International Conference on Emerging Systems and Intelligent Computing, Proceedings, 399–404. doi:10.1109/ESIC60604.2024.10481534.