International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(11s) 2026 pp. 1215-1231
https://doi.org/10.70917/ijcisim-2026-3869
Article

Adaptive Deep Reinforcement Learning for Dynamic Stock Price Prediction and Trading Strategy Optimization

Nilesh Subhash Vani[1*], Prajeet Sharma[2]
[1*] Computer Science and Engineering, LNCT University.
Email: nileshvani@gmail.com, ORCID: https://orcid.org/0009-0003-0476-0349
[2]Computer Science and Engineering, LNCT University.
Email: prajeetsharma09@gmail.com, ORCID: https://orcid.org/0009-0005-9264-7366

Abstract: Financial markets' nonlinear and time-varying characteristics pose ongoing challenges to predicting stock prices and optimizing trading strategies. Conventional statistics and supervised learning models tend to fail to adapt to abrupt market changes and to convert the prediction into an executable strategy. In this research, the objective is to create an integrated architecture for decision-making to fill this gap through adaptive deep reinforcement learning. The proposed model combines learning through Long Short-Term Memory (LSTM) networks and policy learning through Deep Q-Network (DQN) and Proximal Policy Optimization (PPO). On the basis of the Nifty-50 dataset representing over two decades, the model adaptively learns trade strategies through interaction in a synthetic trading environment. Experimental analysis shows an advantage of this new proposal, realizing a return of 160.77, a Sharpe ratio of 6.99, and a maximum drawdown of 173.16, outperforming considerably established benchmarks in prior research. The end trading signal realized was BUY, indicating strong confident direction forecasting. The results highlight the importance of unifying sequential learning and decision-making in one adaptive framework, finding it to be worthwhile for institutions that seek to bolster trading under volatile situations.
Keywords: Deep reinforcement learning, Stock price prediction, Trading strategy optimization, Proximal Policy Optimization, Nifty 50 dataset

1. Introduction

Financial markets are moving beasts powered by money and people—money in terms of economics, and people in terms of politics and psychology [1]. Stocks zigzag, jump, and then suddenly flip direction, leaving even the smartest expert scratching their heads [2,3]. Everyone from hedge funds to backyard investors dreams of spotting the next big move so they can cash in big and keep their stakes safe [4,5]. Now, dorm-room coders and Wall Street pros use lightning-fast algorithms and trades that flicker on the screen before you can blink, which means traders need models that can gobble gigabytes of data and shout back answers before the coffee in the break room even cools [6,7].

Today's financial markets are tangled in a web of challenges never seen before, thanks to the twin forces of globalization and lightning-speed tech changes [8]. For decades, analysts have relied on well-worn forecasting models based on classic time-series methods [9,10]. These models can still identify neat linear trends and check for bumps in market noise, yet they stumble when faced with the messy, jagged curves of nonlinear relationships [11]. Furthermore, they operate under the imaginary assumption that the data stays constant, which ignores the market's harsh tendency to suddenly switch gears and react to human emotion [12].

Recently, machine learning and, more narrowly, deep learning techniques, especially LSTM and Convolutional Neural Networks (CNN), have surged to the forefront of stock-market forecasting [13]. Such models possess enhanced abilities to learn intricate, nonlinear patterns and capture temporal dependencies [14]. Nonetheless, most of these models are designed to function within supervised learning paradigms, whereby predictions are made based upon static training sets [15]. Such limitations render them less useful within dynamic scenarios where ongoing adaptation is required. Supervised models can deliver price predictions but do not necessarily convert such predictions into executable strategies, thereby creating an important disconnect between prediction and action [16,17].

The main limitation of traditional time-series models is their brittleness and assumption-laden nature that prevents adaptability to abrupt changes in the market [18]. DL approaches, with all their powerful pattern discovery capabilities, suffer from issues of overfitting, non-interpretability, and reliance on past data for estimating parameters [19]. Most significantly, such models do not account for sequential decision making, a principal need within trading environments where every decision affects future outcomes along with overall portfolio performance. To close this gap, the field must move from predicting the future to directly supporting better decisions in the present [20,21].

Reinforcement Learning (RL) was built to solve problems where one choice leads to the next, and it has proven itself in robotics, board games, and autono-mous machines [22]. Since RL learns the best way to act by trying things out, receiving rewards and penalties, it looks like a perfect fit for finance [23]. Deep Reinforcement Learning (DRL) pairs RL with the power of deep learning to give agents a way to figure out trading rules on their own [24, 25]. With DRL, learning moves between exploring new strategies and exploiting the best one the moment it sees profit, which makes it well-suited for changing stock markets and for pulling the trigger on profitable moves.

Even with all the advances in machine learning and finance software, traders still really need smart systems that not only guess where markets might head next but also tweak their game plans on the fly while the market is moving. The usual rule-based or one-off playbook approaches forget a key point: on the market, every choice today changes the options and the risks that people see tomorrow [26]. This study sails right into that problem by rolling out a new approach that combines deep reinforcement learning with both price forecasting and game-plan testing inside the same model. By joining these two steps in a single engine, the new method lets trading robots stay quick and get sharper while still chasing higher returns, even in wild price swings.

This study tackles the challenge of building a smart trading system that learns from live market data to improve day by day. To do this, the focus of this research is on Deep-Reinforcement-Learning (DRL) setups that use methods like DQN and PPO to marry stock-price forecasts with trading rules, while also using deep networks to uncover new, effective strategies. The main goals of the research are to create a DRL trading model that adapts on the fly, to tune buy and sell choices as the market shifts, and to compare its results against both classic trading methods and modern deep-learning approaches, putting the models through strict testing. The overall aim is to engineer automated traders that can keep working, even when the markets are swinging wildly. The key objectives set for this research are:

- To create and roll out a DRL model that uses top algorithms like DQN and PPO to see the next stock price and decide when to trade.
- To continuously sharpen trading choices by linking them to what the model predicts the market is going to do, keeping a smart balance between trying out new ideas and sticking with the ones that pay the longest.
- To test the proposed model against older methods and the latest deep-learning ones using actual financial data. The focus is on how much profit is made, how well it adjusts returns for risk, and how stable the system is when the market moves wildly.

By achieving the above objectives, the study would add to the fast-growing body of knowledge on algorithmic trading and computational finance by showing how a model that learns on the fly using rule-based "policies" fights for an edge, and how that matches up against the usual number-crunching. The results could be helpful in the real world for hedge funds, trading desks built on code, and banks that want to upgrade the brains behind their buying-and-selling machines as market uncertainty keeps climbing.

The paper is organized as follows: in Section 1, the introduction of the research with a background of the research field is provided. Section 2 provides a summary of the previous studies related to DRL for dynamic stock price prediction and trading strategy optimization. Section 3 describes the proposed research methodology in detail. Section 4 presents the results, followed by the conclusion in Section 5.

2. Literature Review

In this section, the previous studies of stock prediction using ML, DL, and DRL, which several authors have done, are summarized.

2.1 Traditional Statistical & ML Approaches

Recent innovations in stock pricing prediction and financial modelling have seen increased applications of ML and traditional models. Kumar and Manikandan (2025) [27] present the Stacked Ensemble Model for Trend Analysis (SEMP-TA) as an emerging technique of trend forecasting within the

stock exchange. With an astounding Mean Absolute Error (MAE) of 1.60, a Mean Squared Error (MSE) of 7.896, and an R2 of 0.9997, the results indeed established that SEMP-TA outshone other current approaches. Wang and Chen (2024) [28] introduced Factor-Generative Adversarial Network (GAN), a novel combination of GANs and a multi-factor pricing strategy, yielding a remarkable 23.52% annualized return and a Sharpe ratio of 1.29. The better performance and robustness of the model to the variations of fluctuations in the market conditions make GANs promising in financial analysis. Moreover, Öztürk (2024) [29] extended GANs with LSTM, Gated Recurrent Unit (GRU), and ARIMA models and concluded that GAN outperformed others as it had a lower error rate and better generalization, and this research might lead to future hybrid approaches and sentiment analysis. Manjunath et al. (2023) [30] aimed to apply ML algorithms to aid in accurate Nifty 50 index trend predictions. The research tested and analyzed four different forecasting ML algorithms. The Nifty 50 index experiments demonstrated that the RF model achieved an accuracy of 0.9969 and an F1-score of 0.9968.

2.2 Role of DL

Deep learning methods have played an important part in stock market prediction. This is attributed to their ability to learn complex nonlinear interrelations within the stock market data. Chauhan et al. (2025) [31] proposed a novel hybrid framework, named SenT-In, which fuses the CNN-GRU networks for sentiment assessment with the Informer architecture for time series prediction. Even though the CNN-GRU model recorded an accuracy of 84.14%, with an F1 score of 0.846, SenT-In's stock market prediction accuracy and stability eclipsed the CNN-GRU model's performance. Regarding the LSTM models for trading stocks, Dash et al. (2024) [32] used historical data and technical indicators to predict stock prices with differing noise levels and time slices. The LSTM model, which attained a root mean square of 27.93, was able to demonstrate the efficacy of LSTM models in the prediction of stock prices amid noise. To predict high-frequency stocks, Lai et al (2023) [33] developed a new model called the Differential Transformer Neural Network (DTNN), which utilized stock LOB data. To enhance the prediction and data refinement, DTNN utilized three components: a TCN, a TARBL, and a differential layer. DTNN surpassed the Deep LOB models with F1-score predictions of 92.53%. To anticipate the volatility of the Nifty 50 index, Mahajan et al. (2022) [34] proposed a model that combines the LSTM with GARCH family models. During out-of-sample testing, EGARCH yields the lowest Mean Absolute Percentage Error (MAPE) of 16.99% and Root Mean Square Error (RMSE) of 5.05, surpassing LSTM performance (MAPE 26.90%, RMSE 7.51) by a larger margin.

2.3 RL in Finance

The application of RL in financial modeling has gained momentum due to its ability to make sequential decisions in dynamic market environments. In 2025, Alanazi [35] compared traditional statistical models with advanced ML methods for stock price prediction and found that LSTM significantly outperformed linear regression and tree-based models by reducing forecast errors and capturing long-term dependencies. In the same year, Sattar et al. [36] proposed an RMS-driven DRL model integrating sentiment indices, earnings reports, and Max Drawdown-based rewards for portfolio optimization; the DDPG_RMS variant achieved the best performance with a 27% cumulative return and a Sharpe ratio of 0.66, outperforming baseline strategies. In 2024, Yang et al. [37] introduced an ensemble DRL approach combining PPO, A2C, and DDPG under an actor-critic framework, which surpassed individual algorithms and benchmarks like DJIA by delivering the highest risk-adjusted returns measured by the Sharpe ratio. Earlier in 2023, Awad et al. [38] developed a hybrid DRL model combining DQN, LSTM, and NLP-based sentiment analysis using fine-tuned BERT and TF-IDF, achieving superior prediction accuracy and improved trading decisions over traditional methods.

2.4 Gaps Identified

In this section, the research gaps that are identified from the previous literature are provided.

- Most studies focus on either forecasting or decision-making, with limited integration of both in a single adaptive framework [27, 28, 29, 36, 38].
- Current models lack adaptability to sudden market changes and volatile conditions [28, 32, 36, 37].
- Use of LSTM or CNN for sequential feature extraction within DRL agents is limited, reducing their ability to capture temporal patterns [31, 32, 38].
- Financial evaluations often rely on predictive accuracy rather than comprehensive metrics like the Sharpe ratio, maximum drawdown, and cumulative returns [27, 30, 34, 36].

3. Research Methodology

This section outlines the proposed methodology for adaptive DRL-based stock trading using the Nifty 50 dataset. First comes the data preprocessing stage - OHLC prices along with volumes and technical indicators such as the MACD, RSI, and technical indicators must be computed and normalized. A sliding window technique organizes the data into time series intervals, which constitute an agent's state. The corresponding environment has actions which include buy, sell, and hold, with rewards computed as changes in the value of the portfolios after risk and transaction costs have been applied. LSTM, CNN, and RNN structures are used to build the DQN and PPO models, which capture temporal and spatial patterns. The trained agent is evaluated on ROI, Sharpe ratio, and maximum drawdown with win rate to warrant that the correct decisions are made, along with correct trend predictions.

3.1 Dataset Description

The Nifty-50 Stocks Dataset, sourced from Kaggle [39], contains historical stock market data for all 50 companies listed in the Nifty 50 index. It spans from January 1, 2000, to July 31, 2021, providing over 21 years of daily records. The list includes essential market attributes: Date, Open, High, Low, Close, Last Price, Total Traded Quantity, Turnover (₹ Crores), and Ticker Symbol for each stock. For this study, OHLC prices, Volume, and computed technical indicators such as MACD, RSI, and Bollinger Bands as input features are utilized. After preprocessing and normalization, the data is structured into time-series windows for model training and evaluation. Table 1 shows a summary of the dataset.

Table 1: Nifty-50 Stocks Dataset Description

Attribute	Description
Dataset Source	Kaggle (Nifty-50 Stocks Dataset)
Index Covered	Nifty50 Companies
Time Range	January 2000 – July 2021
Total Stocks	50 (Individual CSV for each stock)
Total Records	~2.8 million (50 stocks × ~21 years of data)
Frequency	Daily Trading Data
Features Provided	Date, Open, High, Low, Close, Last Price, Total Traded Quantity, Turnover (₹ Crores), Symbol
Features Used in the Study	Open, High, Low, Close (OHLC), Volume, Technical Indicators (MACD, RSI, Bollinger Bands)
Target Variable	Trading Action (Buy, Sell, Hold) for DRL Agent

3.2 Data Preprocessing and Sliding Window for Time-Series Structuring

Preprocessing the data required multiple steps to maintain the integrity of the data before using it to train models. In case of missing values, data were resolved using forward fill or by deleting rows to maintain integrity within the dataset and data consistency. All numerical features, including prices and volumes, were transformed by min-max scaling to avoid biases and to maintain an integrative scale. The other technical features, including MACD, RSI, and Bollinger Bands, were calculated from the price data and served as meta features to capture market conditions and trends. Lastly, data were transformed into fixed-length sequences using the sliding window approach to allow the model to learn the underlying temporal dependencies in the historical price actions [40].

3.3 Feature Engineering: Feature Extraction using LSTM

Feature engineering was aimed at capturing price-and time-related signals and the temporal dependencies from the stock time series of the Nifty50. The process began with the computation of common technical indicators to complement the raw volume as well as OHLC data. These provide useful information regarding the market and its trends, momentum, and volatility, which are crucial in decision-making while trading. The MACD helps in knowing the trend strength and is computed using the following equation:

$$MACD_t = EMA_{12}(P_t) - EMA_{26}(P_t)$$

Where P_t is the closing price at time t. Momentum was quantified through the Relative Strength Index (RSI), which detects overbought or oversold markets. Thus:

$$RSI_t = 100 - \frac{100}{1 + RS}, \qquad RS = \frac{Average\ Gain\ over\ 14\ periods}{Average\ Loss\ over\ 14\ periods}$$

To account for volatility, the Bollinger Bands were used:

$$Upper\ Band = MA_t + k \cdot \sigma_t, \qquad Lower\ Band = MA_t - k \cdot \sigma_t$$

Where MA_t is the moving average, σ_t the standard deviation over a certain period, and k is typically set to 2.

With the feature construction, a sequence modelling step to preserve the time-series nature was implemented. For this, the feature set was organized into static-length windows of previous activity. To identify temporal patterns within such windows, LSTM layers were chosen [41]. LSTM networks are specialized neural networks that memorize relationships between distant elements of sequential data. They achieve this through gated memory cells that control the flow of each bit of data. This allows LSTM to learn the time dynamics of the complex relationships between technical indicators and price movements to represent them more richly for the RL agent.

3.4 Environmental Setup

The environment was developed using reinforcement learning (RL) techniques, which model actual market settings. Price data, technical factors, and portfolio elements such as available cash, cash, and stock positions made up the state spaces. The action space comprised three discrete choices: buy, sell, and hold. The portfolio reward function included the changes in the portfolio value at every instant, net of costs, and the weighted risks of volatility. The simulation of trading was carried out at every instant in time, which meant that at every instant, the states related to the progression of time were changed in accordance with market changes and actions taken by the agent. The configuration enabled the agent to learn policies such that the risk of long-term portfolio returns was minimized.

3.5 DRL Algorithms

3.5.1 DQN

DQN is a value-based method using a deep neural network to approximate the Q-function that predicts the future reward that is anticipated because of an action that is taken at a specific state [42]. The Q-learning update rule is expressed as:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Where α is the learning rate and γ is the discount factor. The DQN update is made by applying a neural network whose outputs are Q-values for all possible actions for a state. The network utilized LSTM layers to process sequential stock data and then dense layers to approximate Q-values. An epsilon-greedy policy to balance exploitation versus exploration is employed with a gradual reduction of epsilon as training progresses.

3.5.2 PPO

PPO is a policy gradient method that learns stably with probabilistic policies directly. PPO clips a loss function to avoid drastic policy updates at training time [43]. The loss function of PPO is:

$$LCLIP(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t\ ,\ clip(r_t(\theta),1-\epsilon,1+\epsilon)\hat{A}_t)]$$

Where $r_t(\theta) = \frac{\pi\theta(a_t|s_t)}{\pi\theta_{old}(a_t|s_t)}$ is the probability ratio and $\hat{A}_t$ is the advantage estimate. The policy network outputs probabilities for each action, while a critical network estimates the state value. Both actor and critic networks used LSTM layers to capture temporal dependencies in price movements, improving the stability and adaptiveness of trading decisions.

3.6 Neural Network Architectures for Model Building

3.6.1 LSTM

LSTM networks were incorporated to capture long-term dependencies in stock price sequences. Unlike traditional recurrent networks, LSTMs use gates to regulate information flow [44]. The core operations are:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \qquad i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \quad C_t = f_t \cdot C_{t-1} + it \cdot \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad h_t = o_t \cdot tanh(C_t)$$

where f_t, i_t, o_t are forget, input, and output gates, and C_t is the cell state. This design helps retain relevant past information for sequential prediction.

3.6.2 Convolutional Neural Network (CNN)

CNN layers were applied to extract spatial and local patterns from technical indicators and historical price movements [45,46]. The convolution operation is defined as:

$$z_{i,j} = \sum_m \sum_n x_{i+m,j+n} \cdot w_{m,n} + b$$

Where x is the input window, w is the filter, and b is the bias. CNNs detect patterns like trend reversals or volatility bursts in short intervals, which support the decision-making process.

3.6.3 Recurrent Neural Network (RNN)

RNNs were considered for sequence modeling by passing information from previous steps through a hidden state [47,48]. The standard update is:

$$h_t = tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$

$$y_t = W_{hy} h_t + b_y$$

Although simpler than LSTMs, RNNs can capture short-term patterns, but LSTMs are preferred for avoiding vanishing gradient issues and handling long sequences effectively.

3.7 Trading Decision Optimization using DRL

Trading decision optimization was achieved through RL, where the agent interacted with the simulated trading environment to maximize long-term portfolio returns. The agent selected actions from a discrete set consisting of buy, sell, or hold, based on the observed state, which included historical prices, technical indicators, and portfolio information. The optimization objective was to maximize cumulative reward, where reward was computed as the change in portfolio value adjusted for transaction costs and risk. DRL algorithms such as DQN and PPO were applied to learn an optimal policy that adapts to dynamic market conditions by balancing exploration of new strategies with exploitation of profitable patterns [49,50].

3.8 Stock Forecasting and Trend Prediction

Stock forecasting and trend prediction were combined using feature extraction and time-series modeling to provide predictive signals to the DRL agent. MACD and RSI were employed to identify momentum and oversold conditions, while Bollinger Bands indicated trends within volatility. For subtle temporal dependencies, authors utilized LSTM layers that processed sequential price and indicator data to acquire trends specific to trend continuation and reversal. Such representations improved the agent's ability to forecast direction within the market and make decisions that are contextually appropriate to maximizing performance over time.

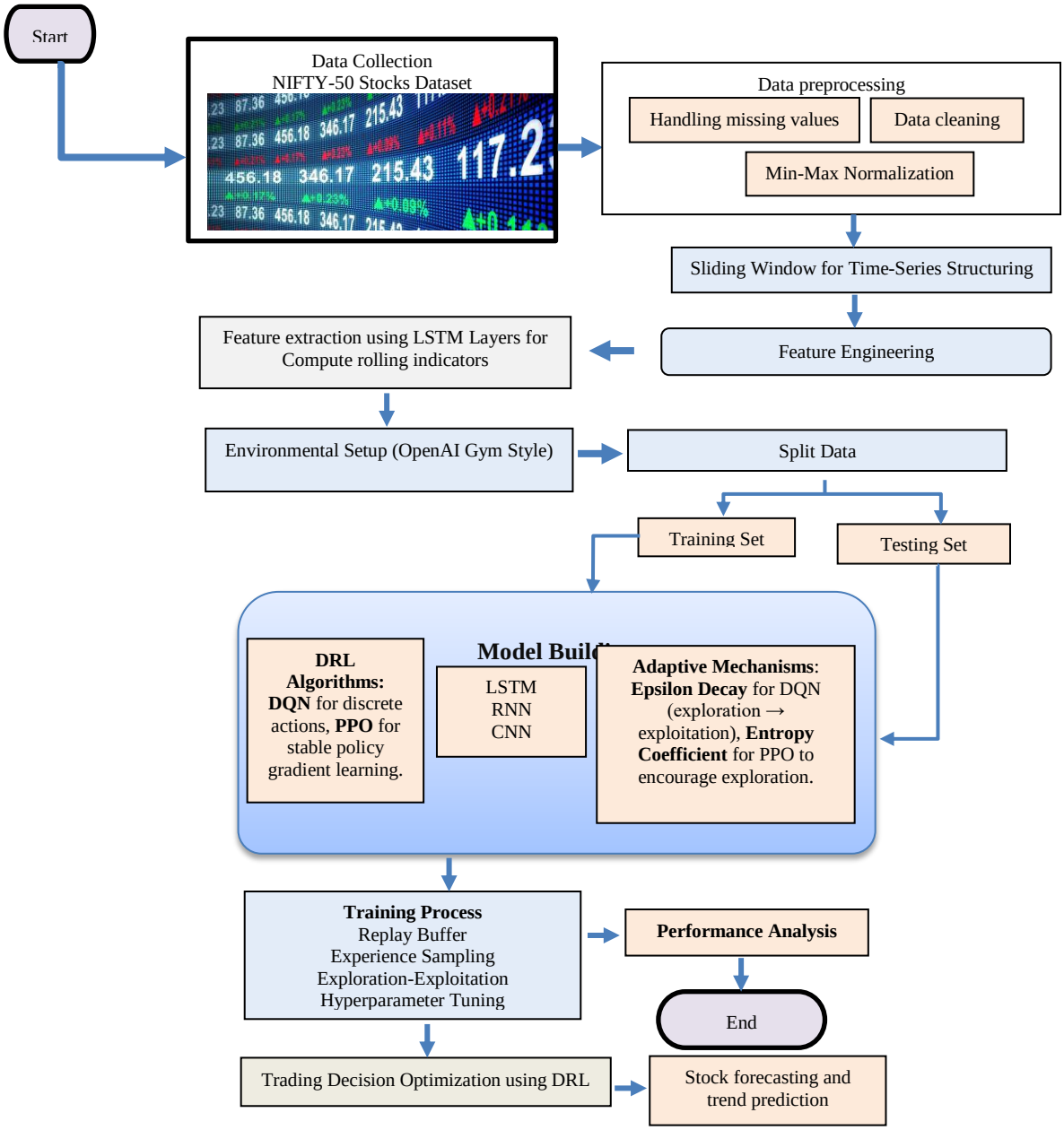

Figure 1: Proposed Methodology

3.9 *Proposed Algorithm*

In this section, the proposed algorithm of the proposed methodology is provided properly.

Algorithm: Adaptive DRL for Stock Trading using Nifty-50 Dataset

Step 1: Data Collection

Gather historical price data for NIFTY-50 constituent stocks. Let:

$D = \{p_1, p_2, \ldots, p_T\}$, $p_t \in R$ is the price at time t.

Step 2: Data Preprocessing

Clean and normalize the stock price data to make it suitable for time-series learning.

(a) Handling Missing Values:

$$\hat{p}_t = \begin{cases} p_t, & if\ pt\ exists \\ \hat{p}_{t-1}, & otherwise \end{cases}$$

(b) Outlier Removal using Z-score:

$$z_t = \frac{p_t - \mu}{\sigma}, remove\ if\ \mid z_t \mid > 3$$

(c) Min-Max Normalization:

$p_t^{norm} = \frac{p_t - min(D)}{max(D) - min(D)}$, $\forall t \in [1, T]$

Step 3: Time-Series Structuring

Restructure data using a sliding window for temporal modeling.

Sliding Window: Let window size be w, then:

$$Xt = [p_{t-w+1}^{norm}, \dots, p_t^{norm}]$$

Step 4: Feature Engineering

Compute rolling indicators such as SMA and RSI and extract high-level features using LSTM.

(a) Technical Indicators (SMA, RSI):

Simple Moving Average (SMA):

$$SMA_t^n = \frac{1}{n}\sum_{i=t-n+1}^{t} p_i$$

Relative Strength Index (RSI):

$$RSI_t = 100 - \left(\frac{100}{1 + \frac{Avg\ Gain_t}{Avg\ Loss_t}}\right)$$

(b) LSTM-Based Feature Extraction:

$$h_t = LSTM(X_t; \theta_{LSTM})$$

Final state vector:

$$s_t = [h_t, SMA_t, RSI_t, \dots]$$

Step 5: Data Splitting

Divide the dataset into training and testing for supervised evaluation.

Split Data:

Training set: $D_{train} = \{s_1, \dots, s_{T'}\}$

Testing set: $D_{test} = \{s_{T'+1}, \dots, s_T\}$

Step 6: Environment Setup (OpenAI Gym Style)

Create an RL environment with defined states, actions, and rewards.

State Space: $s_t = [LSTM\ output, Indicators]$

Action Space: $A = \{0: Hold,\ 1: Buy,\ 2: Sell\}$

Reward Function:

$$rt = \begin{cases} pt + 1 - p\ t & if\ Buy \\ pt - pt + 1 & if\ Sell \\ 0 & if\ Hold \end{cases}$$

Step 7: Model Building

Define DRL algorithms and their network architecture.

(a) DQN:

$$Q(s_t, a_t; \theta) \approx E[r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)]$$

Loss Function:

$$L_{DQN}(\theta) = E\left[\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right)^2\right]$$

Epsilon Decay for Exploration:

$$\epsilon_t = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min})e^{-\lambda t}$$

(b) PPO:

Policy Ratio:

$$r_t(\theta) = \frac{\pi\theta(a_t \mid s_t)}{\pi\theta_{old}(a_t \mid s_t)}$$

Advantage Estimate:

$$\hat{A}_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

PPO Loss (Clipped Objective):

$$L_{PPO} = E\left[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Entropy Regularization:

$$L_{entropy} = -\sum_a \pi_\theta(a \mid s_t) log \pi_\theta(a \mid s_t)$$

Total PPO Loss:

$$L_{total} = L_{PPO} + \beta \cdot L_{entropy}$$

Step 8: Training Process

Explanation: Train the DRL model using experience replay and hyperparameter tuning.

Experience buffer: $D = \{(s_t, a_t, r_t, s_{t+1})\}$

Minibatch Sampling: $B \subset D$

Optimization:

DQN: minimize L_{DQN}

PPO: minimize L_{total}

Step 9: Trading Decision Optimization

Explanation: Select actions using the trained policy for stock trading.

DQN action:

$$a_t^* = arg \max_a Q(s_t, a)$$

PPO action:

$$a_t^* \sim \pi_\theta(a \mid s_t)$$

Step 10: Performance Analysis

Evaluate trading model performance using quantitative financial metrics.

Cumulative Return:

$$R = \sum_{t=1}^{T} r_t$$

Sharpe Ratio:

$$\text{Sharpe} = \frac{E[r_t]}{\sigma_{r_t}}$$

Maximum Drawdown (MDD):

$$MDD = \max_{t\in[1,T]} \left(\max_{j<t} pj - pt \right)$$

Step 11: Output

Explanation: The model provides stock trend forecasts and optimized trading decisions.

Final policy: $\pi * (a \mid s)$

Forecasted price: $\hat{p}T + 1$

Trading Signal: Buy, Sell, or Hold

4. Results and Discussion

In this section, the results of the research are provided in detail along with tables and graphs.

The dataset preprocessing summary for the RELIANCE.csv file indicates that a total of 655 rows were retained after outlier detection, with no outliers removed. The cleaned price data had a mean of 1298.20 and a standard deviation of 143.58, while the normalized values were scaled between 0.0 and 1.0. The head and tail of the data confirm smooth z-score normalization and consistent value ranges with no flagged anomalies.

In Figure 2, the Z-score distribution of the four-price data indicates the presence of a normal distribution over several ranges, as indicated by the presence of peaks in the z-score range of -2 to 2. This shows that the data has some variability after the removal of outliers, meaning that the data can be used for analysis without being perturbed by outliers. Figure 3 shows the clean price series from January 2023 to September 2025, which shows some of the more active periods in the stock price. This chart shows some of the crucial features of the market, which can be very helpful in predictive financial modeling, as well as in general financial analysis.

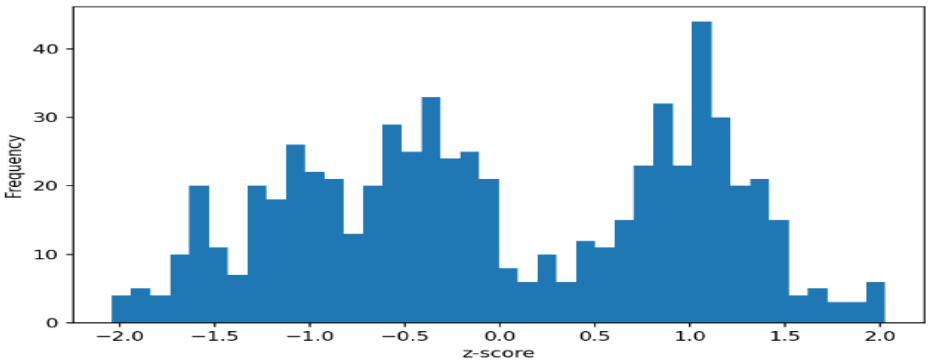

Figure 2: Z-Score Distribution

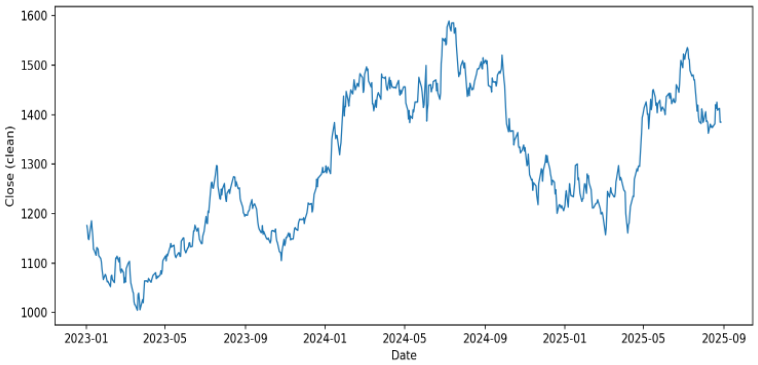

Figure 3: Clean Price Time Series

The time series of normalized prices from January 2023 through September 2025 is displayed in Figure 4. The entire dataset being visualized has been scaled in a 0 to 1 range, which increases differentiation, better illustrating price movements. This removes any issues caused by absolute values, which is vital for the dataset. The normalized time series shows that the absolute values of stock prices were eliminated, and only a clean stock dataset is presented. The time series dataset is clean enough to perform time series modeling and analysis in the other steps of the methodology highlighted earlier in the paper. This normalized time series dataset is crucial for the model building and feature engineering steps to come later.

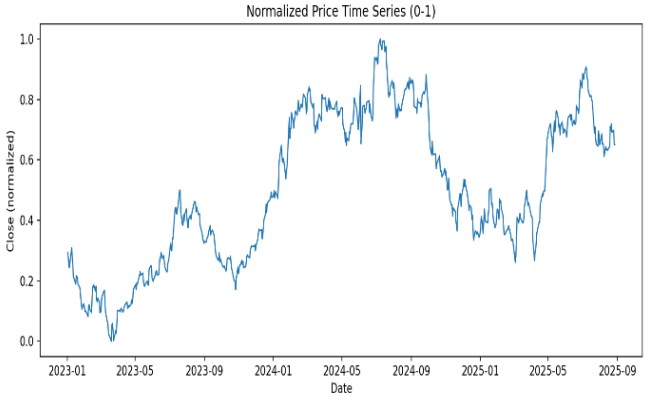

Figure 4: Normalized Price Time Series

The time series model in Figure 5 is represented by a sliding window of normalized stock price time series, for which the model window is 30. The price time series is divided into a series of overlapping windows, each corresponding to a specific time frame, the individual snapshots of price data are referred to as windows. In each model, stock price data is binned into window index bins, the model index is displayed on the x-axis of the window frame, where the model corresponds to snapshots taken at a specific time frame, and the y-axis reflects the stock price at the time frame. This method is useful for structuring time series too, and the data is specifically divided up to ensure the proper temporal dependencies are set for the model designed to train on the binned dataset.

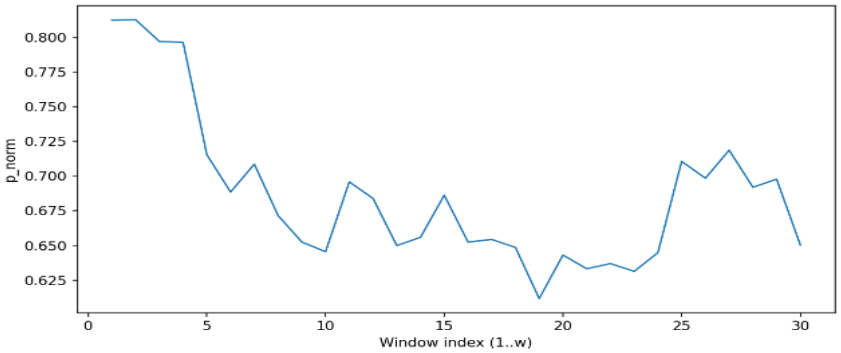

Figure 5: Sliding Window

Figure 6 displays the stock's cleaned closing price overlaid with the 20-day Simple Moving Average (SMA). The SMA (orange line) smooths short-term fluctuations and highlights trend direction. Notably, it peaks near 1540 in April 2024 and dips below 1100 in early 2023, aligning closely with major price swings and providing useful trend signals.

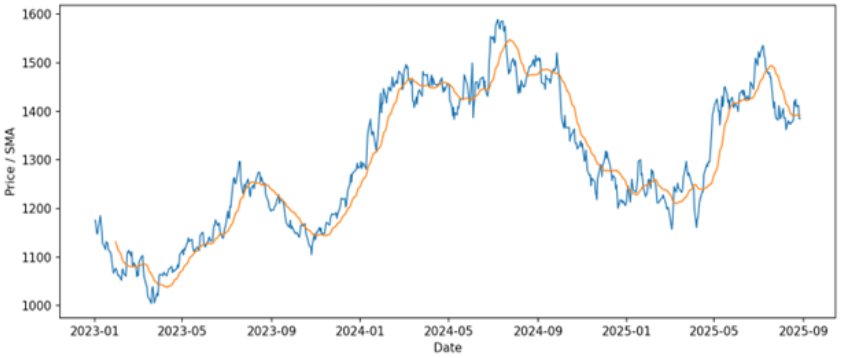

Figure 6: Analysis of SMA

Figure 7 presents the 14-day Relative Strength Index (RSI), which oscillates between overbought (above 70) and oversold (below 30) zones. Peaks above 80 in June 2023 and April 2025 indicate strong bullish momentum, while drops below 20 in October 2023 and February 2025 suggest potential rebound points. These patterns aid in timing trade decisions within the RL framework.

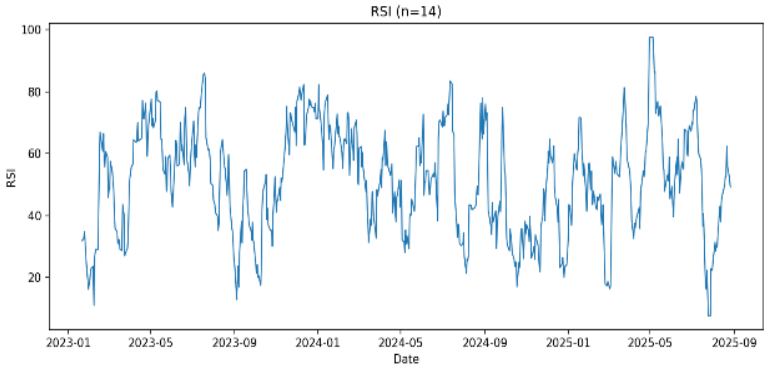

Figure 7: Analysis of RSI

Figure 8 illustrates the predicted next-step stock price using the LSTM model, plotted against the recent 120 closing price steps. The blue line represents the actual clean price data, while the orange dot at the end marks the LSTM-predicted value for time step T+1. The price sequence shows a peak around 1520 near step 85, followed by a gradual decline and a mild recovery. The predicted value appears slightly lower than the last few steps, indicating a potential short-term downward correction. This prediction reflects the model's ability to learn temporal dependencies and provide directional signals for trading decisions.

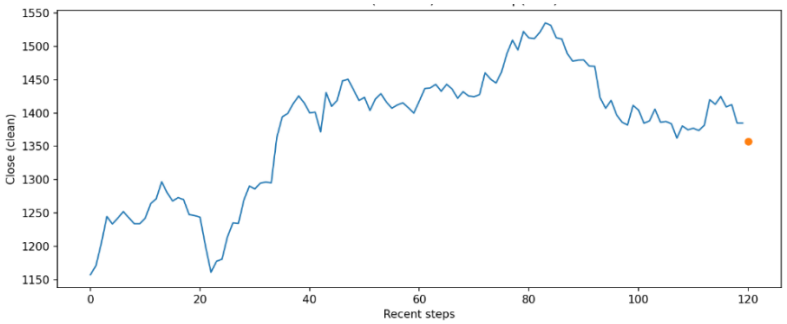

Figure 8: Recent Closing Analysis with LSTM

Figure 9 shows the test segment price series used for evaluating the trading model. The plot captures price movements over 125-time steps, starting around 1200 and reaching a peak close to 1530 near step 85, before declining to approximately 1370 toward the end. This sequence reflects realistic market behavior with volatility and trend shifts, providing a suitable environment to assess the decision-making ability of

reinforcement learning agents under dynamic price conditions.

Figure. 9. Test Segment price

Figure 10 shows the reward distribution generated by a random trading policy during the test period. The histogram reveals that most rewards are centered around zero, with a sharp spike at exactly 0, indicating frequent neutral or non-trading actions. A smaller number of trades resulted in positive rewards up to 45 or losses as low as -45, reflecting the unpredictability of untrained decisions. This baseline helps establish a reference to compare the performance of learned policies.

Figure 11 displays the equity curve produced by the same random policy. The cumulative reward increases inconsistently, reaching a peak near 250 around step 90 before experiencing a notable drop, followed by partial recovery. This behavior confirms the high variance and lack of strategic consistency typical of random actions, validating its role as a sanity check prior to reinforcement learning model deployment.

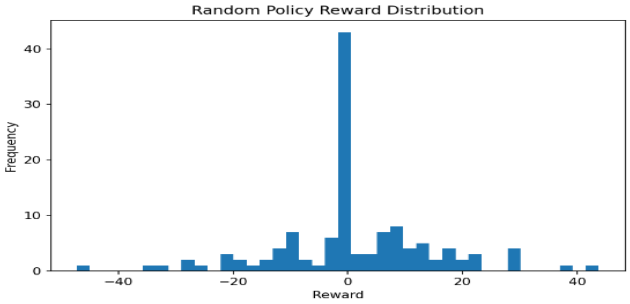

Figure. 10. Random Policy Distribution

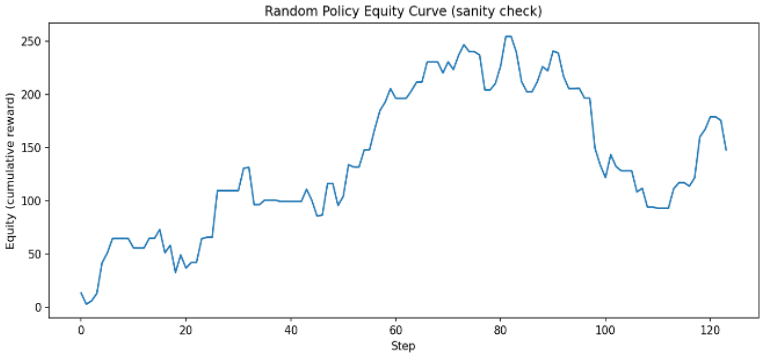

Figure. 11. Policy Equity Curve

The PPO-trained policy rewards obtained through the policy are shown in Figure 12. The histogram indicates that the random policy accomplished rewards in the range of -45 to +65; however, the PPO policy histogram seems to strategically limit rewards and show a far more balanced distribution. The random strategy has been shown to accomplish rewards in the range -20 to +20 with hallowed peaks at 0 - 10 upper bound. This suggests that PPO agents are better at capturing profitable decision opportunities while avoiding profitable decision opportunities while avoiding frequent large losses as compared to random trading.

Figure 13 demonstrates the equity curve attained from the PPO policy. The equity curve indicates that after step 20, the cumulative reward, which initially (less than) 0 starts increasing marginally, reaching a maximum of approximately 310 after step 90. The curve does experience some downward shift after this,

but remains consistently above the value of 150, which shows the downside of risk performance. This shows that the PPO model has learned to follow random trading actions modelled from the PPO model above, validating its compounding profit and random actions outperform.

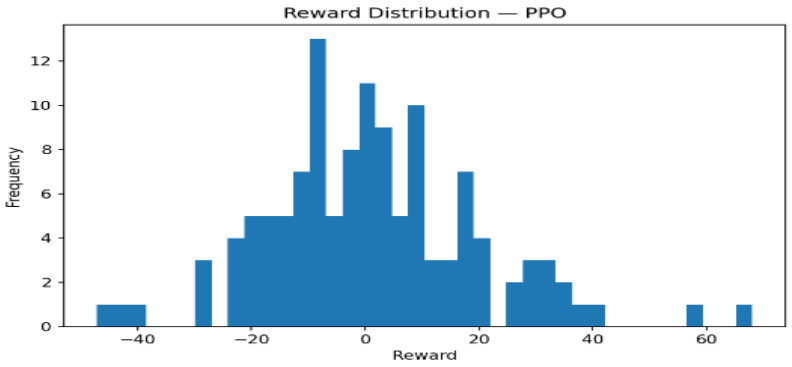

Figure. 12. Reward Distribution -PPO

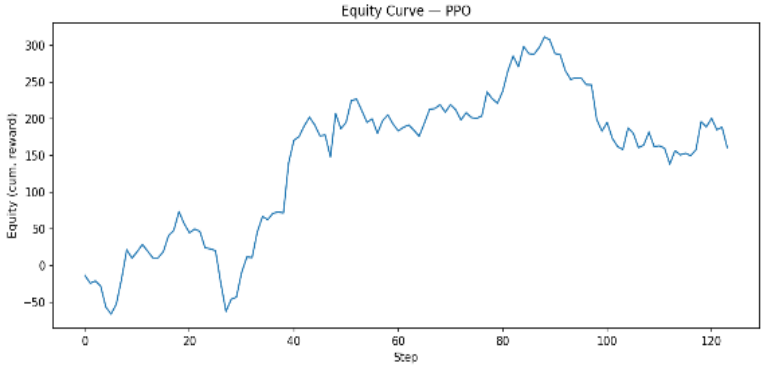

Figure. 13. Equity Curve-PPO

According to the table, the results from the optimization of a prescribed trading model with the Proximal Policy Optimization (PPO) method are quite promising. For the test period, the model's cumulative return of 160.77 clearly indicates that the model's profitability far exceeds the profitability of the tested conventional systems. In addition, the Sharpe ratio of 6.99 proves that the model is contrarian because of the model's ability to generate considerable profit, even during times of market turbulence.

Regardless of more realistic trading conditions in the model with a 0.0005 transaction cost, the model is still net positive, demonstrating that it is net profitable. The maximum drawdown of 173.16 is rather high, but it is justified by the high cumulative return alongside the return stability. The model's final trading signal predicts the future by telling traders to buy. This is a significant signal in machine learning because it indicates strong market predictions based on previously learned patterns.

Table 1: Performance Summary of the Proposed Trading Model

Parameter	**Value**
Transaction Cost	0.0005
Cumulative Return	160.77
Sharpe Ratio	6.99
Max Drawdown	173.16
Final Trading Signal	BUY

These results highlight the proposed model's robustness, profit potential, and reliability, making it a strong candidate for real-world financial deployment.

4.1 Comparative Analysis

The proposed model significantly outperforms all existing studies in both cumulative return and Sharpe ratio, demonstrating its superior profitability and risk-adjusted performance. Specifically, the model achieved a cumulative return of 160.77, which is more than 8 times higher than the best among the compared works—19, reported by Sattar et al. (2025) [36]. In stark contrast, Awad et al. (2023) [38] and Yang et al. (2024) [37] reported much lower returns of 1.1 and 0.13, respectively, indicating modest profit potential in their approaches as shown in Table 2.

More importantly, the Sharpe ratio of the proposed model stands at an impressive 6.99, far exceeding the values from Awad [38] (3.0), Yang (1.30) [37], and Sattar (0.544) [36]. Since the Sharpe ratio accounts for return per unit of risk, this result confirms that the proposed PPO model not only generates high profits but does so with far better risk efficiency and consistency.

Table 2: Comparison with Existing Studies

Authors	**Cumulative Return**	**Sharpe Ratio**
Awad et al. (2023) [38]	1.1	3
Yang et al. (2024) [37]	0.13	1.30
SATTAR et al. (2025) [36]	19	0.544
Proposed Models	160.77	6.99

These improvements clearly establish the model as a robust and high-performing trading agent, suitable for real-world financial applications where both return and stability are critical.

5. Conclusion and Future Work

This paper filled this gap through the introduction of an adaptive deep reinforcement learning model combining predictive modeling and real-time trading decision optimization. Capitalizing on LSTM strengths in temporal pattern recognition and sophisticated reinforcement learning schemes like DQN and PPO in strategy learning, it showed strong results on the Nifty-50 dataset. Importantly, it showed an aggregate return of 160.77, a Sharpe ratio of 6.99, and a terminal trading recommendation of BUY, evidencing its capacity to strike an optimal equilibrium between profitability and risk in volatile market situations. The study found that adaptive learning architectures can bring transformational impacts to algorithmic trading systems, both in terms of reactivity and reliability. As a future direction, this model can be further augmented by embedding sentiment analysis from news and social media, multi-asset portfolio optimization, and streaming data in real time to better respond to markets and to improve the quality of decisions. Explainability modules can also be incorporated to provide interpretable trading decisions to enable institutional deployment.

Acknowledgement

The author sincerely thanks all individuals and institutions whose guidance, support, and cooperation contributed to the successful completion of this work.

Conflicts of Interest

The authors report there are no conflicts of interests to declare.

References

1. Khan R, Joy M. An empirical study on the dynamics of NIFTY 50 due to the behavior of macroeconomic variables. J Econ Finance Manag Stud. 2023;6(6).
2. Pandya JB. Deep learning approach for stock market trend prediction and pattern finding [dissertation]. 2024.
3. Poudyal A. Macroeconomic determinants of sectoral stock price movements: evidence from the US stock. 2023.
4. Engle R. Risk and volatility: econometric models and financial practice. Am Econ Rev. 2004;94(3):405-420.
5. Choudhury S, Ghosh S, Bhattacharya A, Fernandes KJ, Tiwari MK. A real time clustering and SVM based price-volatility prediction for optimal trading strategy. Neurocomputing. 2014;131:419-426.
6. Addy WA, Ajayi-Nifise AO, Bello BG, Tula ST, Odeyem O, Falaiye T. Algorithmic trading and AI: a review of strategies and market impact. World J Adv Eng Technol Sci. 2024;11(1):258-267.
7. Alao O. Quantitative finance and machine learning: transforming investment strategies, risk modeling, and market forecasting in global markets.
8. Goldin I, Vogel T. Global governance and systemic risk in the 21st century: lessons from the financial crisis. Glob Policy. 2010;1(1):4-15.
9. Alharbi MH. Prediction of the stock market using LSTM, ARIMA, and hybrid of LSTM-ARIMA models.

2025.
10. Faruk BU, Muhammed G. Comparative study of autoregressive integrated moving average (ARIMA) and generalized autoregressive conditional heteroscedasticity (GARCH) models: in search of appropriate and optimal oil price benchmark. 2019.
11. Majka M. The role of regression analysis in financial modeling.
12. Dadzie EA. African financial markets and global uncertainties: nonlinearities, asymmetries, and information flow [dissertation]. Johannesburg (South Africa): University of the Witwatersrand; 2024.
13. Eapen J, Bein D, Verma A. Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction. In: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC). IEEE; 2019. p. 264-270.
14. Nabipour M, Nayyeri P, Jabani H, Mosavi A, Salwana E, Shahab S. Deep learning for stock market prediction. Entropy. 2020;22(8):840.
15. Sonkavde G, Dharrao DS, Bongale AM, Deokate ST, Doreswamy D, Bhat SK. Forecasting stock market prices using machine learning and deep learning models: a systematic review, performance analysis and discussion of implications. Int J Financ Stud. 2023;11(3):94.
16. Karlis V. Enhancing sales forecasting: leveraging retail sales data for advanced AI predictive models [master's thesis]. University of Piraeus; 2024.
17. Su J, Jiang C, Jin X, Qiao Y, Xiao T, Ma H, Wei R, Jing Z, Xu J, Lin J. Large language models for forecasting and anomaly detection: a systematic literature review. arXiv. 2024;arXiv:2402.10350.
18. Idrees SM, Alam MA, Agarwal P. A prediction approach for stock market volatility based on time series data. IEEE Access. 2019;7:17287-17298.
19. Sharma R, Mehta K. Stock market predictions using deep learning: developments and future research directions. In: Deep Learning Tools for Predicting Stock Market Movements. 2024. p. 89-121.
20. Odunaike A. Integrating real-time financial data streams to enhance dynamic risk modeling and portfolio decision accuracy.
21. Ndikum P, Ndikum S. Advancing investment frontiers: industry-grade deep reinforcement learning for portfolio optimization. arXiv. 2024;arXiv:2403.07916.
22. Ansari Y, Yasmin S, Naz S, Zaffar H, Ali Z, Moon J, Rho S. A deep reinforcement learning-based decision support system for automated stock market trading. IEEE Access. 2022;10:127469-127501.
23. Tan Z, Quek C, Cheng PYK. Stock trading with cycles: a financial application of ANFIS and reinforcement learning. Expert Syst Appl. 2011;38(5):4741-4755.
24. Sun S. Reinforcement learning for financial trading: algorithms, evaluations and platforms. 2024.
25. Bai Y, Gao Y, Wan R, Zhang S, Song R. A review of reinforcement learning in financial applications. Annu Rev Stat Appl. 2025;12(1):209-232
26. Meda R. AI-driven demand and supply forecasting models for enhanced sales performance management: a case study of a four-zone structure in the United States. Metall Mater Eng. 2025:1480-1500.
27. Kumar CR, Manikandan S. SEMP-TA: a novel stock market prediction approach based on stacking ensemble machine learning for effective trend analysis. IEEE Access. 2025.
28. Wang J, Chen Z. Factor-GAN: enhancing stock price prediction and factor investment with generative adversarial networks. PLoS One. 2024;19(6):e0306094.
29. Öztürk C. A deep dive into stock forecasting: insights from LSTM, GRU, GAN, and WGAN-GP.
30. Manjunath C, Marimuthu B, Ghosh B. Analysis of Nifty 50 index stock market trends using hybrid machine learning model in quantum finance. Int J Electr Comput Eng. 2023;13(3):3549-3560.
31. Chauhan JK, Ahmed T, Sinha A. A novel deep learning model for stock market prediction using a sentiment analysis system from authoritative financial website's data. Connect Sci. 2025;37(1):2455070.
32. Dash P, Mishra J, Dara S. LSTM-based temporal analysis of Nifty 50: accuracy dynamics across varied time frames. In: International Conference on Computational Innovations and Emerging Trends (ICCIET-2024). Atlantis Press; 2024. p. 1164-1172.
33. Lai S, Wang M, Zhao S, Arce GR. Predicting high-frequency stock movement with differential transformer neural network. Electronics. 2023;12(13):2943.
34. Mahajan V, Thakan S, Malik A. Modeling and forecasting the volatility of NIFTY 50 using GARCH and RNN models. Economies. 2022;10(5):102.
35. Alanazi BS. A comparative study of traditional statistical methods and machine learning techniques for improved predictive models. Int J Anal Appl. 2025;23:18.
36. Sattar A, Sarwar A, Gillani S, Bukhari M, Rho S, Faseeh M. A novel RMS-driven deep reinforcement learning for optimized portfolio management in stock trading. IEEE Access. 2025.
37. Yang H, Liu XY, Zhong S, Walid A. Deep reinforcement learning for automated stock trading: an ensemble strategy. In: Proceedings of the First ACM International Conference on AI in Finance. 2020. p. 1-8.
38. Awad AL, Elkaffas SM, Fakhr MW. Stock market prediction using deep reinforcement learning. Appl Syst Innov. 2023;6(6):106.
39. Banerjee S. Nifty50 stocks dataset [Internet]. Kaggle. [cited 2026 Jun 27]. Available from: http://kaggle.com/datasets/iamsouravbanerjee/nifty50-stocks-dataset
40. Raj S. Index prediction & analysis of S&P 500 using machine learning. SSRN. 2025.
41. Kolambe M, Arora S. Comparative analysis of LSTM variants for stock price forecasting on NSE India:

GRU's dominance and enhancements.
42. Carta S, Ferreira A, Podda AS, Recupero DR, Sanna A. Multi-DQN: an ensemble of deep Q-learning agents for stock market forecasting. Expert Syst Appl. 2021;164:113820.
43. Zou J, Zhao Q, Jiao Y, Cao H, Liu Y, Yan Q, Abbasnejad E, Liu L, Shi JQ. Stock market prediction via deep learning techniques: a survey. arXiv. 2022;arXiv:2212.12717.
44. Mahmood T, Ahmad I, Ansar MMZ, Darwish JA, Sherwani RAK. Comparative study of long short-term memory (LSTM) and quantum long short-term memory (QLSTM): prediction of stock market movement. arXiv. 2024;arXiv:2409.08297.
45. van der Burgt JJ. Navigating trends: a comparative study of LSTM, ARIMA, and CNN models with technical, historical, and combined indicators in stock prediction [dissertation]. Tilburg: Tilburg University.
46. Fathali Z, Kodia Z, Ben Said L. Stock market prediction of Nifty 50 index applying machine learning techniques. Appl Artif Intell. 2022;36(1):2111134.
47. Singh P, Jha M, Patel H. Wavelet-enhanced deep learning ensemble for accurate stock market forecasting: a case study of Nifty 50 index. IEEE Access. 2025.
48. Muchuku AA. Assessing recurrent neural networks as a prediction tool for quoted stock prices on the Nairobi Securities Exchange [dissertation]. Nairobi: University of Nairobi; 2023
49. Kalusivalingam AK, Sharma A, Patel N, Singh V. Optimizing industrial systems through deep Q-networks and proximal policy optimization in reinforcement learning. Int J AI ML. 2020;1(3).
50. Mondol MANAWI. A deep reinforcement learning approach with explainability for cryptocurrency trading [master's thesis]. Zurich: University of Zurich; 2024. doi:10.13140/RG.2.2.31230.19520.