Independent Research Project

Applying Deep Reinforcement Learning to Systematic Equities Trading Strategy

Project advisor: Professor Ali Kakhbod

Ying Qian

University of California, Berkeley - Haas School of Business

October 2024

Contents

1. Introduction

The financial markets are complex, dynamic environments where accurate prediction and timely decision-making are crucial for profitability. Traditional rule-based trading systems sometimes hard to adapt to dynamic market conditions and incorporate a variety of data sources. Therefore, there has been a trend in utilizing Deep Reinforcement Learning (DRL) to develop trading agents capable of learning optimal policies through interaction with the market environment. DRL offers a promising approach to automate trading by continuously improving decision-making based on market data, while adapting to volatile and unpredictable market environments.

This research project focuses on applying DRL, specifically the Asynchronous Advantage Actor-Critic (A3C) algorithm, in combination with a Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (Bi-LSTM) architecture, to develop a systematic trading strategy. A3C is an algorithm that allows parallel learning across environments, making it well for complex tasks like equities trading, where timely and accurate decisions are very important. The use of CNN-BiLSTM is motivated by its ability to capture both local patterns in financial time series data through convolutional layers and long-term dependencies through the Bi-LSTM layers, which is crucial for accurately forecasting price movements and trends.

To enhance the agent's ability to make informed trading decisions, sentiment scores were incorporated into input data. Sentiment data provides a qualitative perspective on market conditions, enabling the agent to integrate news and public sentiment with quantitative indicators. This combined approach aims to improve the robustness and performance of the trading strategy by leveraging both technical and sentiment-based signals.

The research evaluates the performance of the sentiment-aware A3C-CNN-BiLSTM agent and A3C-CNN-BiLSTM agent against baseline DRL algorithms such as DQN (Deep Q-Networks), A2C (Advantage Actor-Critic), and PPO (Proximal Policy Optimization) across major stock indices like the S&P 500, NASDAQ, and DJIA (Dow Jones). Key metrics, including cumulative return, Sharpe ratio, and maximum drawdown, are used to evaluate the profitability and risk control capabilities of the agents. The objective is to demonstrate how applying advanced DRL algorithms, neural network architectures, and sentiment scores can yield systematic trading strategies that adapt to different market conditions and enhance overall profitability.

2. Literature Review

The application of Deep Reinforcement Learning (DRL) in financial trading has gained attention in recent years due to its ability to learn adaptive and data-driven trading strategies. Traditional quantitative finance models, such as statistical arbitrage and factor-based approaches, rely on predefined rules and assumptions, often finding it difficult to generalize to the highly volatile and complex nature of financial markets. In contrast, DRL offers a flexible and dynamic approach to learning trading policies directly from market data, enabling continuous improvement and adaptability.

Actor-critic methods have emerged as more robust alternatives for DRL in trading. The A3C (Asynchronous Advantage Actor-Critic) algorithm proposed by Mnih et al. (2016) has been particularly impactful. A3C allows multiple agents to interact with their environments asynchronously, enabling faster and more diverse exploration of the state-action space. This parallelism makes it suitable for complex, time-sensitive works like equities trading. The use of an advantage function further stabilizes the training process by reducing the variance in policy updates, improving convergence. Studies such as Deng et al. (2016) and Jiang et al. (2017) have demonstrated the applicability of A3C in equity and cryptocurrency trading, showing improved performance over traditional RL models.

Xu et al. (2021) explore the application of the A3C reinforcement learning algorithm to develop a stock trading strategy. Their study aims to enhance the decision-making process in stock trading by leveraging the A3C's capability of handling continuous decision spaces and real-time updating of trading policies. The authors detail their approach by constructing a trading environment that incorporates realistic market dynamics. By training the A3C agent within this environment, they demonstrate that the algorithm can autonomously learn and execute profitable trading strategies over time. Their empirical results indicate that A3C outperforms some traditional trading strategies, leading to higher profitability and better risk-adjusted returns.

The use of deep neural networks (DNNs) for feature extraction is crucial for modeling financial time series, as they allow the agent to learn complex patterns in the data. CNNs (Convolutional Neural Networks) have been widely adopted for financial prediction tasks due to their ability to capture local dependencies and patterns in time-series data. For instance, Bao et al. (2017) applied CNNs for feature extraction in stock price prediction, showing that CNNs outperform traditional models in extracting key features from market data. However, CNNs are limited in capturing long-term dependencies.

To address this limitation, RNNs (Recurrent Neural Networks), particularly LSTMs (Long Short-Term Memory) networks, have been used for their ability to handle temporal dependencies and sequential data. Bi-LSTM (Bidirectional LSTM) networks further improve this capability by processing input data in both forward and backward directions, making them well-suited for predicting stock price movements. Studies like Fischer & Krauss (2018) and Chen et al. (2018) have shown that LSTM and Bi-LSTM models outperform traditional RNNs in financial time-series prediction, as they better capture both short-term and long-term dependencies in market data.

Combining CNNs and LSTMs has been found to enhance both spatial and temporal feature extraction. Zhang et al. (2019) demonstrated that a CNN-LSTM architecture could effectively model both short-term and long-term dependencies in market data, leading to improved trading performance. Furthermore, adding a sentiment analysis component to DRL models has been a growing area of research, as market sentiment plays a significant role in price movements. Studies like Bollen et al. (2011) and Hu et al. (2018) have shown that public sentiment, derived from news articles, social media, and financial reports, can serve as a powerful predictor of asset price movements.

The literature review provides an overview of the success of DRL algorithms in financial trading. The use of advanced neural network architectures, such as CNNs and LSTMs, has improved feature extraction from financial time series, while sentiment analysis has added a qualitative factor to predictive modeling. This research aims to contribute to this growing field by developing a sentiment-aware A3C-CNN-BiLSTM model that applies the strengths of these advancements to create a robust, adaptive and systematic trading strategy that outperforms traditional baselines in multiple financial markets.

3. Methodology

3.1 Data

To train the DRL model, I use multiple data sources, each providing different insights into market conditions. The data includes 5 macroeconomic indicators, 13 pricing and technical indicators and 1 financial news sentiment data. Data related to the stock index prices is dated from 1st January 2018 to 31st December 2023, and the pricing data is downloaded using python yfinance library.

3.1.1 Macroeconomic Indicators

I obtain macroeconomic data through the Federal Reserve Economic Data (FRED) API, provided by the Federal Reserve Bank of St. Louis. The macroeconomic indicators collected include the Interest Rate, Yield Spread, GDP, Unemployment Rate, and Inflation Rate.

The Interest Rate (the Federal Funds Rate) represents the cost of borrowing and directly influences equity markets. The Yield Spread, calculated as the difference between the 10-year and 2-year treasury constant maturities, acts as a predictor of economic cycles and is particularly useful for assessing market sentiment and the risk of recessions. GDP serves as a measure of the economy's overall activity and health. The Unemployment Rate offers insights into labor market conditions and consumer purchasing power. Finally, the Inflation Rate, represented by the Consumer Price Index (CPI), reflects changes in the purchasing power of money and has significant implications for corporate earnings and stock valuations.

3.1.2 Technical Indicators

The technical indicators are derived from pricing data using the Python TA library. I select price, volume, and 11 technical indicators to provide insights into stock trends, momentum, and potential reversals.

The 5 and 10-day Exponential Moving Averages (EMA) track price movements over time, giving more weight to recent prices for better trend analysis.

The Moving Average Convergence Divergence (MACD) measures the relationship between two EMAs and is used to identify potential buy or sell signals.

The Relative Strength Index (RSI) evaluates the magnitude of recent price changes to determine if a stock is overbought or oversold.

Bollinger Bands (high and low) consist of a moving average and two standard deviations, indicating market volatility and potential price reversals.

On-Balance Volume (OBV) uses volume flow to forecast stock price changes.

The Average Directional Index (ADX) assesses the strength of a trend, while the Momentum (MOM) indicator measures the speed or rate of price changes, comparing the current price with a previous period.

The Money Flow Index (MFI) integrates price and volume data to reveal buying or selling pressure.

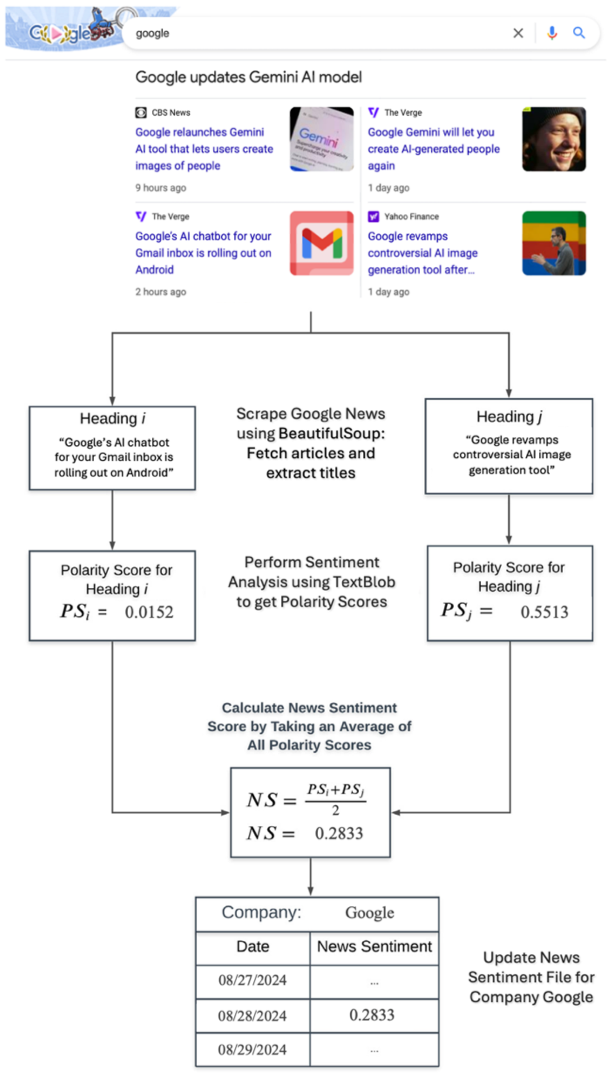
Lastly, the Parabolic SAR (Stop and Reverse) is a trend-following indicator that identifies potential entry and exit points by highlighting the possible direction of an asset's price movement.

3.1.3 News Sentiment

In addition to stock pricing and macroeconomic indicators, I incorporate text-based data to capture market sentiment on a given date. To gather sentiment-related news data, I use Google News, a news aggregator with robust search capabilities that allow filtering by keywords, date ranges, and relevance. The web scraping process is automated using BeautifulSoup, where these parameters are leveraged to effectively query and collect relevant news articles. Information such as the date, title, snippet, source, and link of the articles is retrieved and stored in CSV files.

Since article titles provide a concise summary and tone, they are sufficient for gauging the sentiment of the full article. Sentiment analysis is performed on these extracted titles using TextBlob, which calculates polarity (ranging from -1 for negative sentiment to +1 for positive sentiment) and subjectivity (ranging from 0 for objective to 1 for subjective). The polarity scores are used to compute a news sentiment score for the stock on a given date.

These sentiment scores are then aggregated daily to create a market sentiment indicator. The process of scraping Google News using BeautifulSoup and conducting sentiment analysis with TextBlob is illustrated through an example in figure 1.


google
Google updates Gemini AI model
CBS News
Google relaunches Gemini AI tool that lets users create images of people
9 hours ago
The Verge
Google Gemini will let you create AI-generated people again
1 day ago
The Verge
Google's AI chatbot for your Gmail inbox is rolling out on Android
2 hours ago
Yahoo Finance
Google revamps controversial AI image generation tool after...
1 day ago
Heading *i*
"Google's AI chatbot for your Gmail inbox is rolling out on Android"
Scrape Google News using BeautifulSoup: Fetch articles and extract titles
Heading *j*
"Google revamps controversial AI image generation tool"
Polarity Score for Heading *i*
$PS_i = 0.0152$
Perform Sentiment Analysis using TextBlob to get Polarity Scores
Polarity Score for Heading *j*
$PS_j = 0.5513$
Calculate News Sentiment Score by Taking an Average of All Polarity Scores
$NS = \frac{PS_i + PS_j}{2}$
$NS = 0.2833$
Update News Sentiment File for Company Google

Company:	Google
Date	News Sentiment
08/27/2024	...
08/28/2024	0.2833
08/29/2024	...

Fig. 1. Example with Detailed Steps for Google News.

Once all data are collected, they are combined to create a comprehensive input dataset. During preprocessing, any missing values are filled to ensure data completeness. The dataset is then split into training and testing sets based on the date range, with the first 80% of the data allocated for training and the remaining 20% reserved for testing.

3.2 Environment Setup

The trading environment for the DRL agent is designed to simulate a financial market where the agent interacts with historical market data to make trading decisions.

The **state representation** of the environment consists of a rolling window of historical data that captures key market features, such as adjusted close prices, technical indicators, macroeconomic indicators and news sentiment influencing the market. The size of the window, defined by the `window_size` parameter, determines the length of the historical sequence the agent observes at each step, providing context for its decisions. The state also includes buy and sell states and information on the agent's portfolio, such as the current cash balance, portfolio values, profits and inventory to fully capture the financial states at any point.

The **action space** available to the agent consists of three discrete choices: Buy, Sell, or Hold.

- **Buy** ($a = 1$): The agent buys one unit of the asset at the current price, reducing its cash by the price of the asset, adding it to its inventory.
- **Sell** ($a = 2$): The agent sells one unit of the asset if it holds any from its inventory, increasing its cash by the current price and realizing any profit or loss.
- **Hold** ($a = 0$): The agent takes no action, maintaining its current portfolio position without making any transactions, potentially allowing it to wait for more favorable market conditions.

Thus, the action space can be defined as: $A_t \in \{0, 1, 2\}$.

The **reward function** is designed to align the agent's incentives with profitability. The reward after each action is based on the agent's ability to generate positive returns. For each sell action, the reward R_t is calculated as the percentage return on the sold asset, represented by:

$$R_t = \frac{P_t - P_{buy}}{P_{buy}}$$

where P_t is the selling price at time t, and P_{buy} is the purchase price. If the agent sells at a higher price than the purchase price, the reward is positive, representing a profit; otherwise, it’s negative, reflecting a loss. There’s a small penalty if trying to sell without inventory. For each buy action, there’s no immediate reward, and a small penalty if trying to buy due to insufficient cash.

Reward for holding depends on the market condition. When holding with a position, the agent receives small positive reward for holding during an uptrend, and small penalty for holding during a down trend. When holding without a position and waiting for a better entry point, the agent receives small reward for holding when prices are dropping (avoiding bad entry) and receives small penalty for not taking an action during an uptrend. This reward structure encourages the agent to optimize for risk-adjusted returns, balancing profit potential with the risk of negative outcomes.

3.3 A3C Algorithm

The Asynchronous Advantage Actor-Critic (A3C) algorithm is selected because its parallel training and actor-critic framework enable efficient learning of complex market dynamics and adaptive decision-making in volatile environments. It combines the strengths of both policy-based and value-based approaches, which is used to train the agent to interact with the market environment by learning optimal trading strategies. A3C uses multiple agents that run in parallel, interacting with their respective environments independently. This parallelization helps the algorithm explore the state-action space more efficiently and avoid local optima, making it suitable for dynamic, complex environments. Figure 2 illustrates the architecture of multiple agents running in parallel, interacting with its own independent environment, updating their local actor and critic networks; a shared global policy and value network are then updated asynchronously by the local agents.

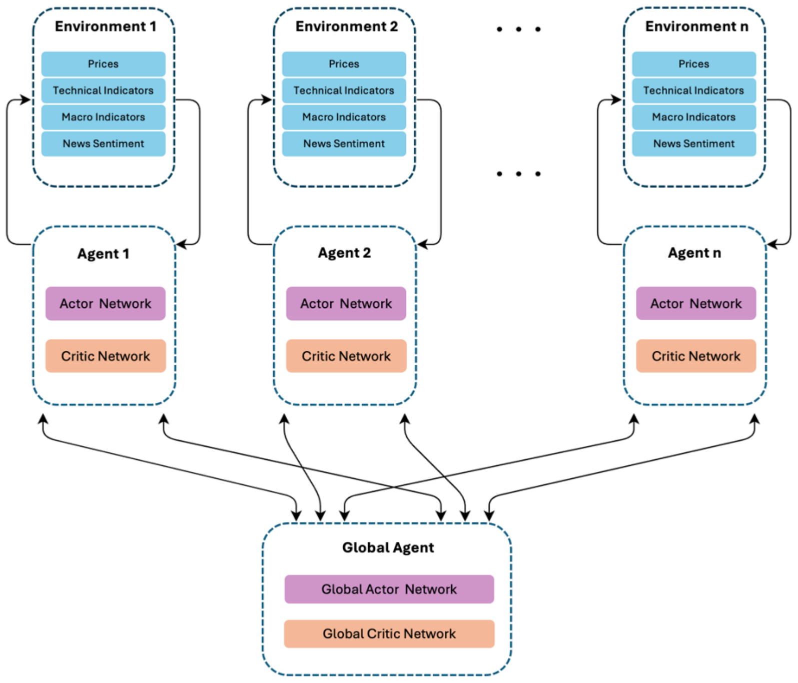


Fig.2. the architecture of A3C model

There're two key components in A3C: the actor and the critic. The actor is responsible for learning a policy $\pi(s, \theta)$, where s is the current state, and θ represents the parameters of the policy (the weights of the actor network). The policy outputs a probability distribution over the action space, allowing the agent to decide which action to take at each time step. Meanwhile, the critic estimates the value function

$V(s, \psi)$, where ψ is the parameter of the value network. The value function assesses how good it is for the agent to be in a particular state s by estimating the expected cumulative future rewards from that state. The key idea of A3C is to update both the actor and critic networks using advantage. The advantage $A(s, a)$ measures how well a particular action performed compared to the expected value of the state. It is calculated as

$$A(s, a) = r_t + \gamma V(s_{t+1}, \psi) - V(s_t, \psi)$$

where r_t is the reward at time step t, γ is the discount factor, $V(s_t, \psi)$ is the value of the state s at time step t. This advantage function is used to update both the actor and critic. The actor is updated by maximizing the likelihood of actions that lead to higher advantages, and the critic is updated to reduce the difference between the estimated value and the actual reward plus the next state's value.

The actor's policy is updated using the policy gradient method:

$$\nabla_\theta log\pi(s_t, a_t, \theta)A(s_t, a_t)$$

the gradient of the log-probability of the chosen action a_t is scaled by the advantage, encouraging actions that lead to higher returns. The critic's parameters are updated by minimizing the mean-squared error between the value estimate and the expected reward:

$$L(\psi) = \left(r_t + \gamma V(s_{t+1}, \psi) - V(s_t, \psi)\right)^2$$

Entropy regularization is added to the loss function to encourage the agent to explore a wider range of actions, rather than getting stuck in suboptimal or overly deterministic policies. A more exploratory policy is crucial, particularly during the early stages of training, as it allows the agent to explore a broader range of strategies. This is especially important in dynamic and uncertain financial markets, where exploring different actions can lead to discovering more profitable strategies over time.

The entropy of the policy $\pi(s_t, \ \theta)$for a given state is defined as:

$$H\left(\pi(s_t, \theta)\right) = -\sum_a \pi(a|s, \theta) log\pi(a|s, \theta)$$

Incorporating entropy regularization into the A3C algorithm modifies the policy gradient and update it to include this entropy term. The total loss for the actor now becomes:

$$L_{actor} = -\left(\nabla_\theta log\pi(s_t, a_t, \theta)A(s_t, a_t)\right) - \beta H\left(\pi(s_t, \theta)\right)$$

The first term in this formula is the standard policy gradient, where the agent is encouraged to take actions that lead to higher advantages, and the second term represents the entropy regularization, with entropy coefficient β being a hyperparameter that controls the weight of the entropy penalty. A higher value promotes more exploration by encouraging higher entropy, while a lower entropy coefficient encourages more exploitation of known strategies. For the critic, the loss function remains the same.

Each agent then accumulates policy gradients over time by interacting with its environment. Once the agent has accumulated enough gradients, it sends them to the global agent. The global agent then aggregates the gradients from all agents and updates the global policy parameters accordingly.

3.4. Actor and Critic Network Architecture

The actor network consists of input layer, three Convolutional layers, BiLSTM layer, connection layer and output layer. Figure 3 displays the architecture of Actor and Critic Network.

The **input layer** to both the Actor and Critic networks is the state vector, which consists of pricing and technical indicators, macroeconomic variables and sentiment scores. The state representation covers a window of past n-days, encapsulating temporal dependencies. It can be represented as a 2D tensor $X \in \mathbb{R}^{T \times F}$, where T is the window size (number of time steps) and F is the number of features at each time step.

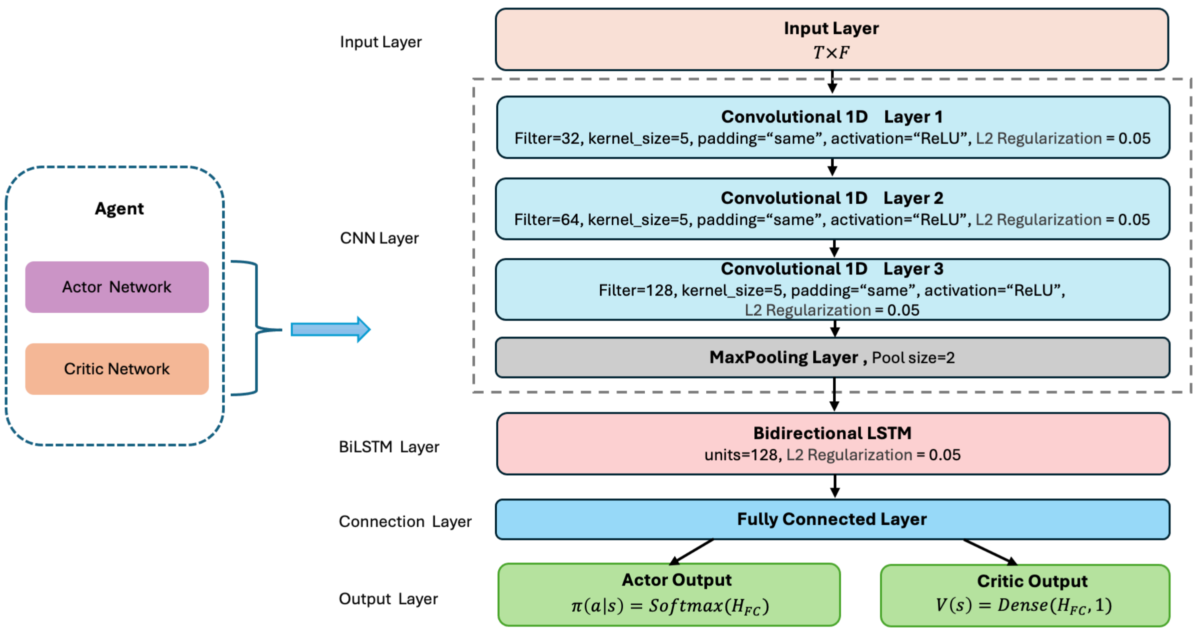

Agent
Actor Network
Critic Network
Input Layer
CNN Layer
BiLSTM Layer
Connection Layer
Output Layer
Input Layer
$T \times F$
Convolutional 1D Layer 1
Filter=32, kernel_size=5, padding="same", activation="ReLU", L2 Regularization = 0.05
Convolutional 1D Layer 2
Filter=64, kernel_size=5, padding="same", activation="ReLU", L2 Regularization = 0.05
Convolutional 1D Layer 3
Filter=128, kernel_size=5, padding="same", activation="ReLU", L2 Regularization = 0.05
MaxPooling Layer , Pool size=2
Bidirectional LSTM
units=128, L2 Regularization = 0.05
Fully Connected Layer
Actor Output
$\pi(a|s) = Softmax(H_{FC})$
Critic Output
$V(s) = Dense(H_{FC}, 1)$

Fig.3. Actor and Critic Network Architecture

Both the Actor and Critic networks apply a series of 1D **Convolutional Neural Network** (CNN) layers to extract local features from the time-series data. CNN excels at extracting spatial and temporal patterns from high-dimensional inputs, thus enhancing the agent's ability to make decisions and identify trends or patterns based on complex data.

The first Conv1D layer uses 32 filters with a kernel size of 5 and applies the ReLU activation function. This layer identifies local patterns in the input data. The second and third Conv1D layer use 64 and 128 filters with the same kernel size and activation function further refining the patterns detected by the first layer. Each convolutional layer is followed by Batch Normalization and Dropout for regularization.

A **MaxPooling** layer with pool size of 2 is applied following the Conv1D layers. This layer reduces the dimensionality of the data by down-sampling, reducing the number of parameters and computations in subsequent layers, and retaining the most important features from each region of the feature map, which helps to make the model more robust to variations and prevent overfitting.

Next, a **Bidirectional LSTM** (Bi-LSTM) layer as an advanced type of recurrent neural network (RNN) layer is applied to capture temporal dependencies in the time-series data, with 128 units and L2 regularization, and followed by Layer Normalization and Dropout. It is designed to acquire dependencies in sequential data both forwards and backwards and meanwhile extends the capacity of Long Short-Term Memory (LSTM). There are two LSTM layers in Bi-LSTM's hidden layer. One processes the input sequence from the beginning to the end (forward direction), while the other processes it from the end to the beginning (backward direction) to better model the temporal structure of the input data.

After the Bi-LSTM, the output is passed through a **Fully Connected (Dense) Layer**, which transforms the features into a lower-dimensional space suitable for the final decision-making step. Dropout is applied after this layer to further regularize the network.

The final **output layer** for the Actor Network uses the Softmax activation function to output a probability distribution over the possible actions (Buy, Sell, Hold). The Softmax function ensures that the sum of probabilities is 1, making it suitable for action selection.

The Critic Network mirrors the Actor Network in terms of the input, convolutional, and pooling layers and Bi-LSTM, except the output, which is a single scalar value representing the state value. It is a prediction of the expected cumulative future rewards from the current state, used by the actor to estimate the advantage function.

4. Implementation

4.1. Training Process

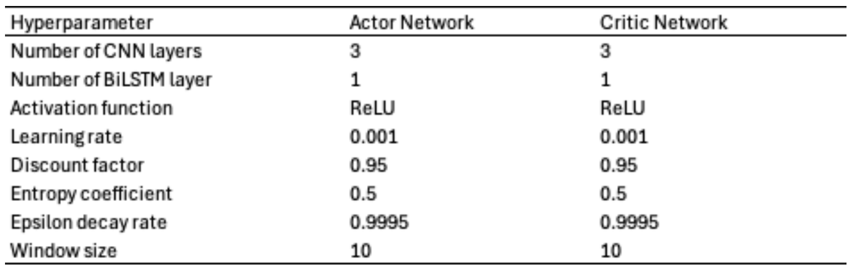
The training process for the A3C agent involves iterative learning over multiple episodes to develop a robust trading strategy that balances profitability and risk. The A3C agent, designed with both actor and critic networks, is trained to learn an optimal policy for buying, selling, or holding assets based on the environment's state, which consists of historical market data and sentiment information.

To evaluate the impact of sentiment analysis on trading performance, two variants of the A3C agents were trained: one incorporating sentiment data into its state representation (sentiment-aware A3C-CNN-BiLSTM) and one that only relied on traditional data (A3C-CNN-BiLSTM). This comparative approach allows for the assessment of how sentiment influences decision-making and profitability in trading, especially when paired with advanced DRL techniques.

The first agent, referred as A3C-CNN-BiLSTM, was trained on a dataset consisting solely of market features such as historical prices, volume, technical indicators and macro indicators. The state representation used a window size of 10, capturing the last 10 days of market data to inform each action. The CNN-BiLSTM architecture enabled the agent to learn both short-term patterns through convolutional layers and long-term dependencies via Bi-LSTM layers. The goal was to evaluate how well this agent could predict and respond to market movements based on purely quantitative data. The agent was trained over 30 episodes, with both the actor and critic networks using the Adam optimizer with a learning rate of 0.001.

The second agent, the sentiment-aware A3C-CNN-BiLSTM, was trained using an enriched dataset that incorporated both traditional market features and sentiment scores derived from news. This additional sentiment information aimed to provide a more holistic understanding of the market, which can influence price movements. The training setup for this agent was like the first agent, it used a window size of 10, 30 episodes, and the Adam optimizer for both actor and critic networks. The state representation for this agent was augmented with sentiment scores, allowing it to learn not only from market data but also from the influence of public sentiment on price dynamics.

The actor network is trained using policy gradients, where the objective is to maximize the expected cumulative reward over time. The actor is updated by adjusting the probabilities of taking actions based on their advantages. Actions that lead to positive advantages are made more probable, while those leading to negative outcomes are suppressed. The critic network is trained using the advantage function, which helps the critic learn to better evaluate the agent's actions, leading to more accurate value estimates. Table 1 lists the paraparameters used during the training.



Hyperparameter	Actor Network	Critic Network
Number of CNN layers	3	3
Number of BiLSTM layer	1	1
Activation function	ReLU	ReLU
Learning rate	0.001	0.001
Discount factor	0.95	0.95
Entropy coefficient	0.5	0.5
Epsilon decay rate	0.9995	0.9995
Window size	10	10

Table 1. The hyperparameters

An epsilon-greedy strategy is adopted to encourage exploration of different trading actions. Initially, the agent explores various actions by selecting them randomly with a high probability (epsilon = 1). As the agents learn and gain more experience, epsilon decay at a rate of 0.9995 after each step, reducing the probability of random actions and shifting the agent towards exploitation of learned policies. During each time step, the actor network outputs a probability distribution over the actions, and the agents select an action either by exploring or by exploiting the action with the highest probability as epsilon decreases.

To prevent the agents from becoming overly deterministic and to maintain exploration during training, entropy regularization is applied. An entropy coefficient (0.5) is included in the loss function of the actor, promoting diverse action selection and preventing premature convergence to suboptimal policies. This regularization ensures that the agent does not settle on a single action too early, allowing for broader exploration of trading opportunities. In addition, to prevent overfitting, early stopping is applied to both actor and critic models by monitoring the loss functions during training. If the loss does not improve for 10 epochs, training is stopped, and the model weights are restored to their best-performing state.

During training, 4 local agents interact with separate instances of the trading environment to explore different market scenarios and gather diverse experiences. Each local agent periodically synchronizes its gradients with a global agent, which aggregates these updates to improve the shared actor and critic networks, thus enabling faster convergence and more stable learning. After each episode, the global agent saves the updated actor and critic models, capturing the learned policies and value functions for future trading decisions. Once training is completed, the global actor and critic models are saved.

4.2 Testing Process

The model's performance is evaluated on unseen data to test its generalizability. Environment is instantiated for testing, and a separate test_env is created using the test data. This ensures that the agent's performance is evaluated on new data that it hasn't seen during training. The agent is reset to avoid any carryover of state from training to ensures that models from training do not interfere with testing.

The global actor and critic models saved from training are loaded to initialize the A3C agent for testing. This ensures that the agent uses the learned policies and value functions from the best-performing training episodes. During testing, the epsilon-greedy strategy is no longer applied. Instead, the agent exploits its learned policy by selecting the action with the highest probability generated by the actor network. This ensures that the agent consistently takes the most confident action according to its training, leading to more stable and consistent trading behavior.

Then, the agent's testing performance is assessed based on key performance metrics, such as cumulative return, Sharpe ratio, and maximum drawdown, which help measure profitability, risk-adjusted returns, and risk exposure, respectively.

4.3 Evaluation Metrics

The evaluation of the A3C agent involves measuring key performance metrics to assess the effectiveness and risk-adjusted returns of the trading strategy. These metrics provide insights into both profitability and risk management over the trading period. Below is the description of each metric used for evaluation.

Cumulative return (CR) measures the total return of the trading strategy over a specified period. It represents the growth of an initial investment, indicating the overall profitability of the trading strategy.

$$CR = \frac{V_T - V_0}{V_0}$$

where V_0 is the initial portfolio value, V_T is the final portfolio value.

Annualized return (AR) measures the average return per year over the trading period, which helps standardize performance for comparison across different time horizons. It is calculated by scaling the cumulative return to a yearly basis.

$$AR = (1 + CR)^{\frac{252}{N}} - 1$$

where N is the number of trading days in the evaluation period.

Annualized standard deviation, referred to as volatility, measures the amount of variation in the portfolio’s returns. It is a risk measure indicating the consistency of returns. Higher volatility suggests more risk, as returns fluctuate more widely.

$$Annualized\ Std = \sigma_{daily} \times \sqrt{252}$$

where σ_{daily} is the standard deviation of the daily returns.

Sharpe Ratio (SR) is a risk-adjusted performance metric that compares the excess return of the trading strategy to its volatility. It measures the return earned per unit of risk, providing a standardized measure to compare different strategies.

$$SR = \frac{AR - R_f}{Annualized\ Std}$$

where R_f is the risk-free rate of return, typically the return of government bonds or cash.

Sortino Ratio is similar to the Sharpe Ratio but focuses only on downside risk instead of overall volatility. It provides a more accurate view of the risk by penalizing only the downside deviations rather than both upward and downward movements.

$$Sortino\ Ratio = \frac{AR - R_f}{Downside\ Std}$$

where Downside Std is the standard deviation of the negative returns.

Maximum Drawdown (MDD) is a measure of the largest peak-to-trough decline in portfolio value over the trading period. It assesses the worst possible loss the strategy experiences, helping to evaluate the strategy's vulnerability to extreme downturns.

$$MDD = \frac{\max_{t}(V_{peak} - V_{trough})}{V_{peak}}$$

where V_{peak} is the maximum value before a decline and V_{trough} is the minimum value following the peak.

Maximum Drawdown Length measures the duration of the longest drawdown, which is the period between the peak and the subsequent recovery to the previous peak. It gives insights into the time it took for the strategy to recover from the worst loss. This metric helps in understanding the resilience and recovery capabilities of the strategy. A shorter drawdown length indicates that the portfolio recovers quickly from losses, while a longer length suggests prolonged periods of underperformance.

4.4 Baseline DRL Algorithms

To evaluate the performance of A3C agent, the proposed A3C-CNN-BiLSTM and sentiment-aware A3C-CNN-BiLSTM agents are compared with a few baseline algorithm: DQN (Deep Q-Networks), A2C (Advantage Actor-Critic), and PPO (Proximal Policy Optimization).

DQN (Deep Q-Networks) is one of the foundational algorithms in DRL that employs a value-based approach, where a neural network is used to approximate the Q-value function, which estimates the expected reward of taking a certain action in each state. DQN uses experience replay to stabilize training and a target network to update Q-values, which helps mitigate the issue of instability in training.

A2C (Advantage Actor-Critic) is an enhancement over DQN that uses both a policy (actor) and value function (critic) to learn the optimal strategy. The advantage function is used to stabilize training by reducing variance in policy updates. A2C is synchronous, multiple parallel agents interact with their environments simultaneously, and their experiences are aggregated to update the shared model, which improves learning efficiency.

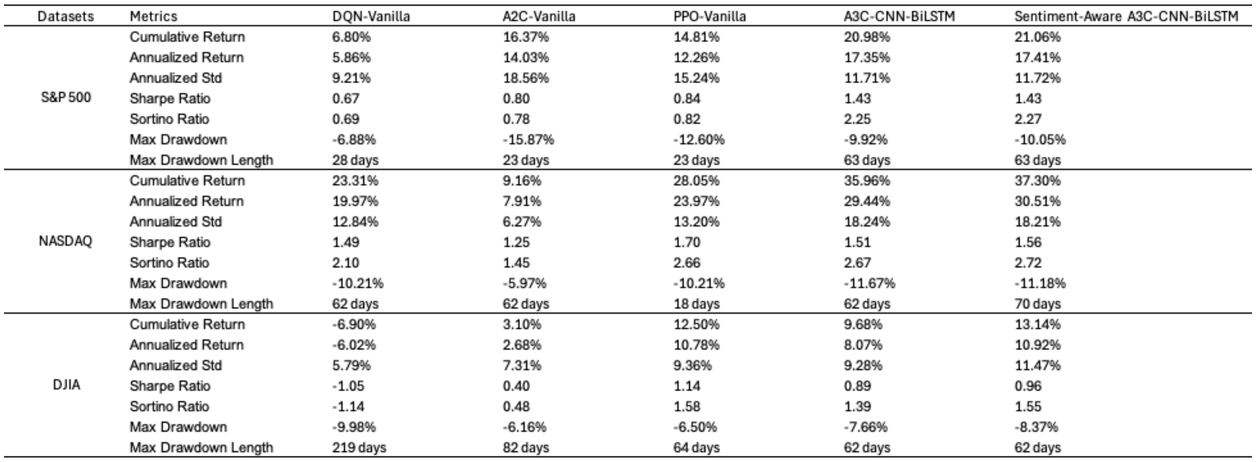
PPO (Proximal Policy Optimization) is a modern DRL algorithm that builds on A2C and other actor-critic approaches by using a clipped objective function to ensure that policy updates are not too large, making training more stable and efficient. PPO leverages a surrogate objective function to penalize deviations from the current policy, preventing drastic changes and allowing smoother updates.

Each of these algorithms has its strengths and is used as a baseline to understand how proposed A3C agents perform relative to different learning methodologies, ranging from value-based (DQN) to actor-critic (A2C and PPO).

5. Results and Analysis

5.1. Testing results and comparison with baselines

Table 2 displays the testing performance comparison between the proposed agents with baseline DRL agents. The testing results of the A3C-CNN-BiLSTM and sentiment-aware A3C-CNN-BiLSTM agent demonstrate strong performance across different financial indices: the S&P 500, NASDAQ, and DJIA.



Datasets	Metrics	DQN-Vanilla	A2C-Vanilla	PPO-Vanilla	A3C-CNN-BiLSTM	Sentiment-Aware A3C-CNN-BiLSTM
S&P 500	Cumulative Return	6.80%	16.37%	14.81%	20.98%	21.06%
	Annualized Return	5.86%	14.03%	12.26%	17.35%	17.41%
	Annualized Std	9.21%	18.56%	15.24%	11.71%	11.72%
	Sharpe Ratio	0.67	0.80	0.84	1.43	1.43
	Sortino Ratio	0.69	0.78	0.82	2.25	2.27
	Max Drawdown	-6.88%	-15.87%	-12.60%	-9.92%	-10.05%
	Max Drawdown Length	28 days	23 days	23 days	63 days	63 days
NASDAQ	Cumulative Return	23.31%	9.16%	28.05%	35.96%	37.30%
	Annualized Return	19.97%	7.91%	23.97%	29.44%	30.51%
	Annualized Std	12.84%	6.27%	13.20%	18.24%	18.21%
	Sharpe Ratio	1.49	1.25	1.70	1.51	1.56
	Sortino Ratio	2.10	1.45	2.66	2.67	2.72
	Max Drawdown	-10.21%	-5.97%	-10.21%	-11.67%	-11.18%
	Max Drawdown Length	62 days	62 days	18 days	62 days	70 days
DJIA	Cumulative Return	-6.90%	3.10%	12.50%	9.68%	13.14%
	Annualized Return	-6.02%	2.68%	10.78%	8.07%	10.92%
	Annualized Std	5.79%	7.31%	9.36%	9.28%	11.47%
	Sharpe Ratio	-1.05	0.40	1.14	0.89	0.96
	Sortino Ratio	-1.14	0.48	1.58	1.39	1.55
	Max Drawdown	-9.98%	-6.16%	-6.50%	-7.66%	-8.37%
	Max Drawdown Length	219 days	82 days	64 days	62 days	62 days

Table 2. The performance of various trading agents on three indices.

In the S&P 500, the sentiment-aware A3C-CNN-BiLSTM agent achieves a cumulative return of 21.06% and an annualized return of 17.41%, surpassing the standard A3C-CNN-BiLSTM and outperforming all other baseline DRL agents, showcasing the benefit of incorporating sentiment data into the model. The Sharpe Ratio of 1.43 and Sortino Ratio of 2.27 show good risk-adjusted returns compared to the baselines like DQN-Vanilla and A2C-Vanilla, which exhibit considerably lower Sharpe ratios. The max drawdown of -10.05% for the sentiment-aware agent indicates moderate risk, while the drawdown period is longer than the other baseline agents.

The outperformance is even more pronounced in the NASDAQ, where the sentiment-aware A3C-CNN-BiLSTM achieves the highest cumulative return of 37.30% and an annualized return of 30.51%, emphasizing the value of sentiment data in improving trading decisions. Additionally, it has a Sortino Ratio of 2.72, demonstrating strong downside protection and it's the highest among all agents. A Sharpe Ratio of 1.56 indicates good risk-adjusted returns, although PPO-Vanilla gets the highest of 1.70. The sentiment-aware agent has a max drawdown of -11.18%, which is a slight improvement over the A3C-

CNN-BiLSTM agent -11.67%. However, both A3C agents experience a longer drawdown period (62 and 70 days) compared to the quicker recovery of PPO-Vanilla (18 days).

For the DJIA, the A3C-CNN-BiLSTM agent performs moderately well with a cumulative return of 9.68% and an annualized return of 8.07%, outperforming A2C-Vanilla and DQN-Vanilla baselines. The agent maintains a good balance between return and risk, as indicated by a Sharpe Ratio of 0.89 and Sortino Ratio of 1.39. The sentiment-aware A3C agent again improves on these metrics, with a cumulative return of 13.14% and a higher Sharpe Ratio of 0.96, indicating that sentiment-aware enhancements contribute positively across various datasets. Both A3C agents show shorter drawdown periods compared to the other baselines.

Figure 4-6 display the cumulative returns and provide a visual comparison of the performance of different DRL algorithms, including DQN-Vanilla, A2C-Vanilla, PPO-Vanilla, and the proposed A3C agents across the S&P 500, NASDAQ, and DJIA indices.

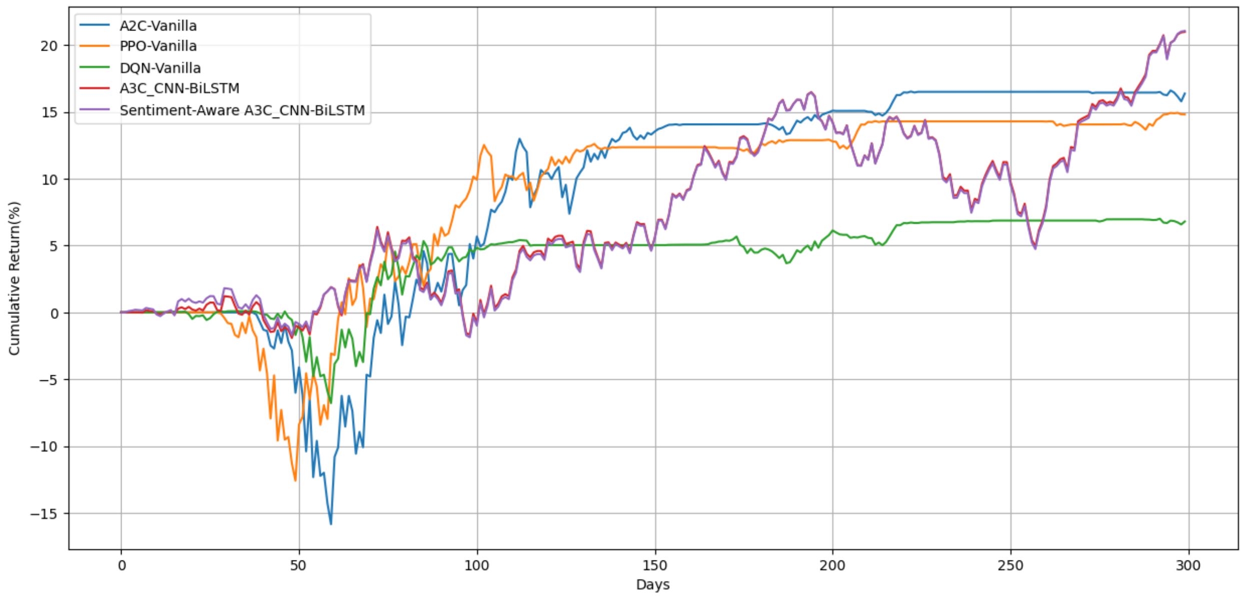


Fig. 4. Cumulative return (%) of proposed agents and baselines in S&P 500.

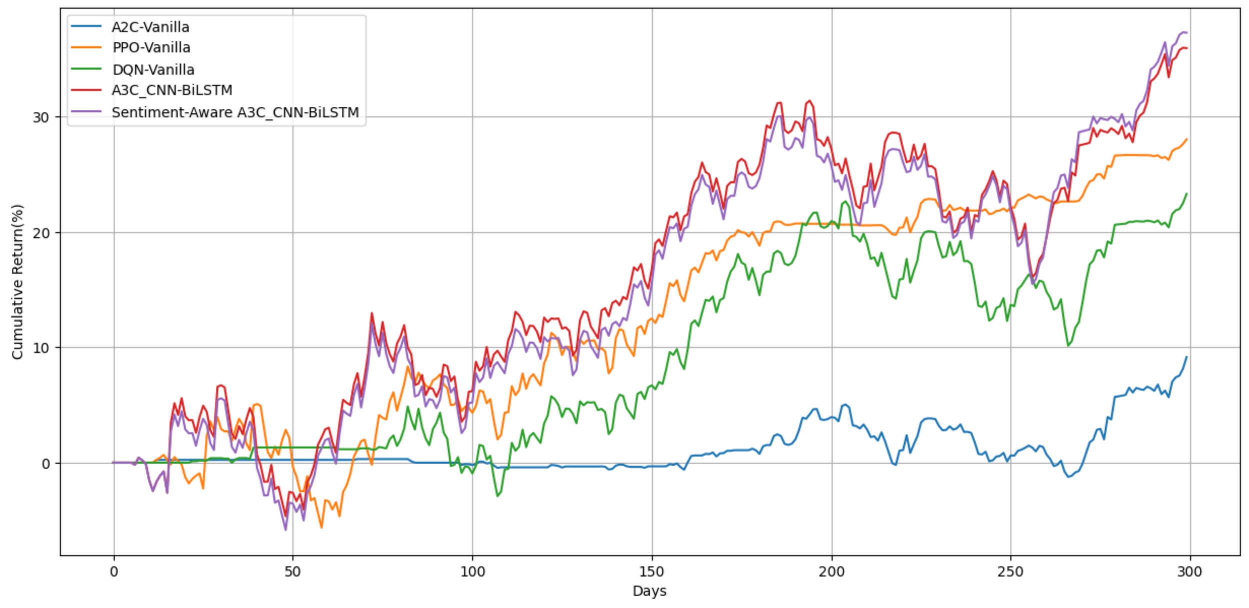


Fig. 5. Cumulative return (%) of proposed agents and baselines in NASDAQ.

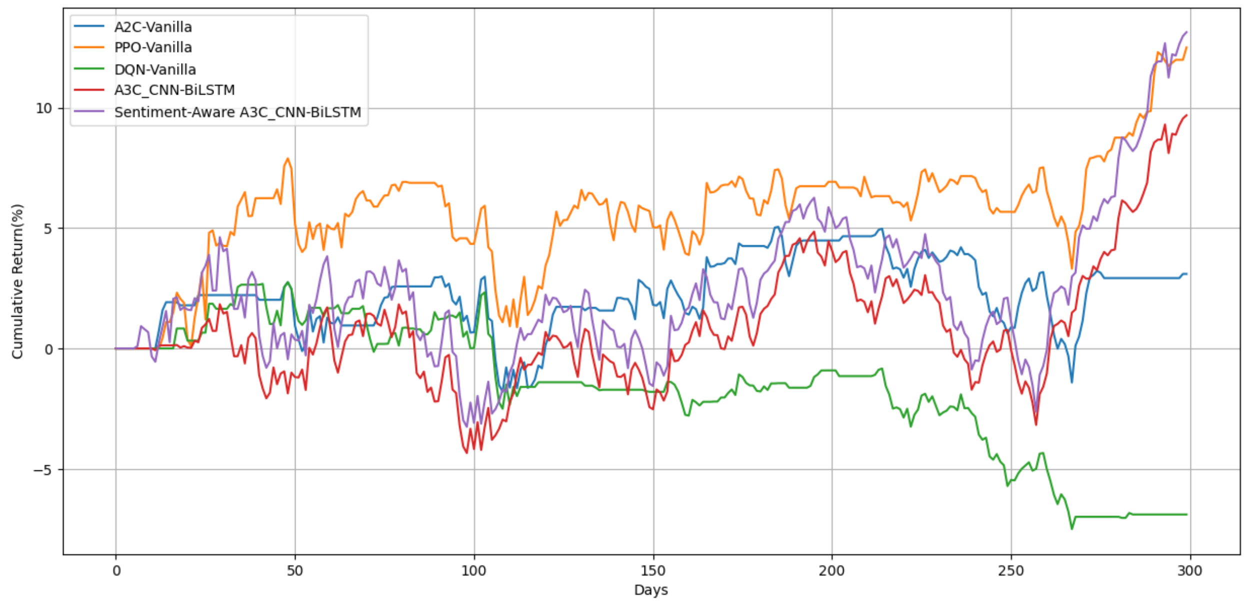


Fig. 6. Cumulative return (%) of proposed agents and baselines in DJIA.

In all three indices, the sentiment-aware A3C-CNN-BiLSTM agent achieves the highest cumulative returns, indicating that incorporating sentiment data enhances trading performance, demonstrating strong growth and stability across different market conditions. PPO-Vanilla shows moderate performance, while DQN-Vanilla and A2C-Vanilla lag behind, showing lower returns and higher volatility.

Overall, the results indicate that sentiment-aware A3C-CNN-BiLSTM outperforms traditional DQN-Vanilla, A2C-Vanilla, and PPO-Vanilla baselines. Additionally, leveraging sentiment data makes better trading decisions and manages risks effectively, leading to an overall more robust and profitable trading strategy compared to baselines.

5.2. Analysis of Strategy

The backtesting results of the sentiment-aware A3C-CNN-BiLSTM agent across the S&P 500, NASDAQ, and DJIA indices show that the agent effectively identifies profitable trading opportunities, leading to notable gains.

Figure 7 shows the trading activities during backtesting in S&P 500. Sentiment-aware A3C-CNN-BiLSTM agent gains over $7000, and it effectively identifies the buying and selling opportunities throughout the backtesting period. The buying and selling signals are well-distributed across market uptrends and corrections, indicating that the model capitalizes on short- to medium-term price movements. The model maintains profitability by executing trades during clear upward trends and quickly exiting positions during downward movements, demonstrating effective use of sentiment data for market timing. The trading activities align well with market uptrends, with timely buying at lower points before price increases and selling near local peaks. There is consistent alignment with market movement, and the model avoids prolonged exposure to downturns, indicating an effective strategy for capturing upward momentum and minimizing losses.

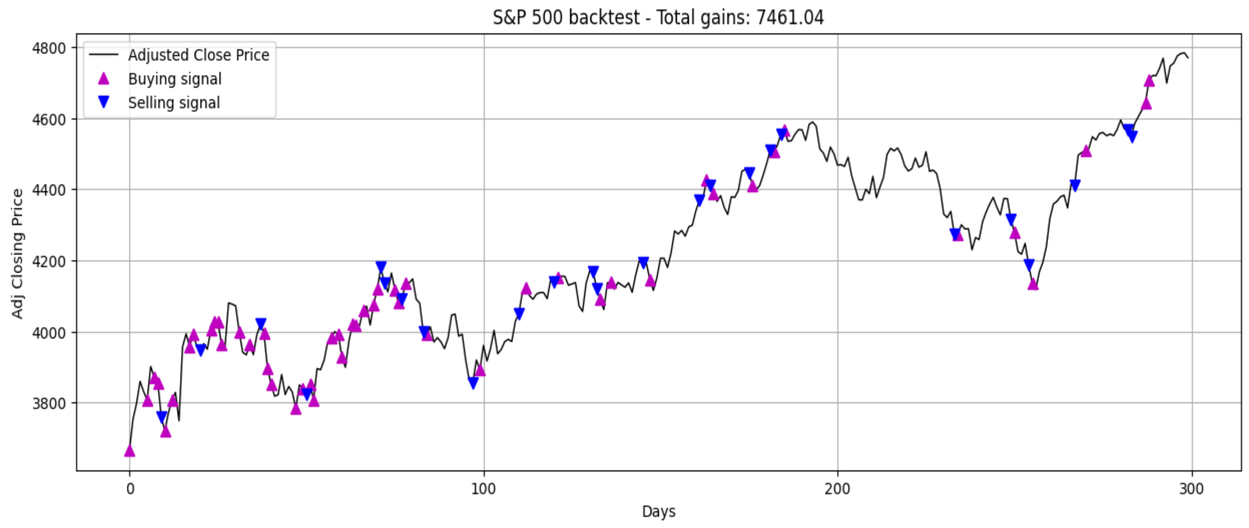


Fig. 7. Trading activities during backtesting in S&P 500.

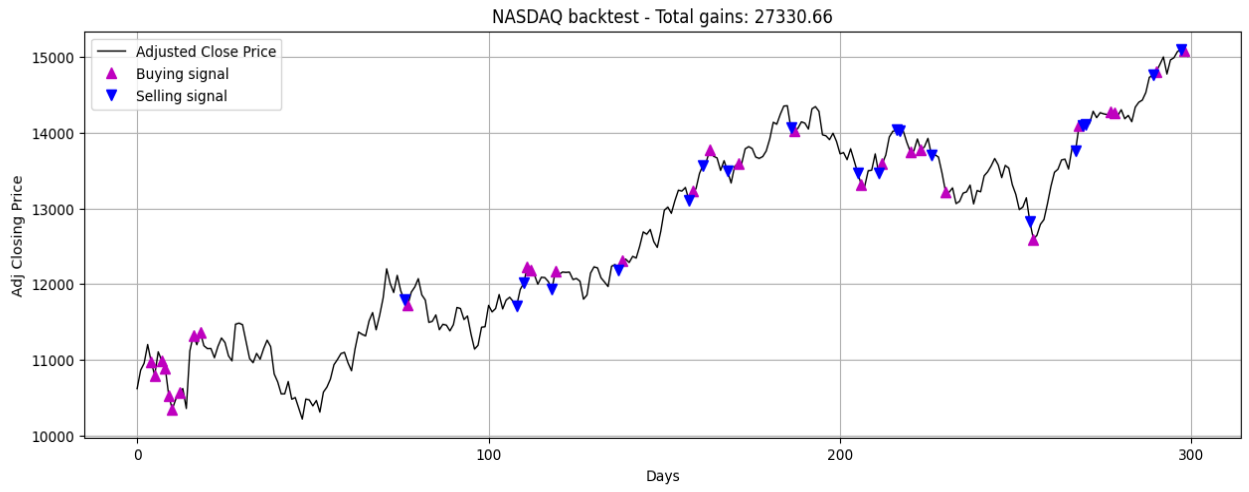


Fig. 8. Trading activities during backtesting in NASDAQ.

The agent's performance in the NASDAQ index is particularly strong as illustrated in figure 8. The signals indicate that the sentiment-aware A3C-CNN-BiLSTM agent successfully captures a significant portion of the market's upward trend, buying early in the upswings and selling before market corrections. The agent effectively rides the uptrends while protecting against losses during dips. The sentiment data likely contributes to these timely decisions, helping the agent distinguish between temporary pullbacks and genuine trend reversals. The effective timing of trades and limited exposure to downside risks result in substantial cumulative returns.

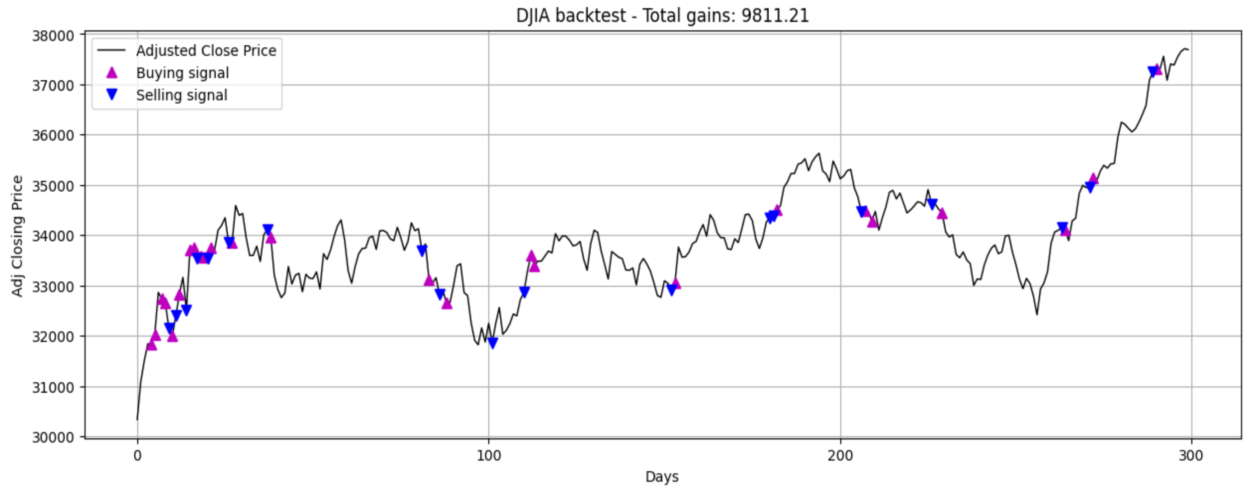


Fig. 9. Trading activities during backtesting in DJIA.

In the DJIA index, figure 9 presents that the sentiment-aware A3C-CNN-BiLSTM agent achieves a substantial gain, with signals spread throughout both periods of volatility and sustained trends. The agent's trading decisions reflect a balanced approach, taking advantage of market rebounds and holding through periods of steady growth. The use of sentiment data may help the agent mitigate risk during market drawdowns.

Across all indices, the sentiment-aware A3C-CNN-BiLSTM agent demonstrates effective decision-making by aligning its buy and sell actions with market trends. The use of sentiment data enables the agent to optimize its entries and exits, enhancing returns and minimizing drawdowns. The profits across different market conditions (bullish trends in NASDAQ, balanced trends in S&P 500, and mixed movements in DJIA) indicates the robustness of the agent. It shows that integrating sentiment scores into deep reinforcement learning can enhance trading strategies by providing a deeper understanding of market dynamics.

6. Improvements and Future Work

6.1 Improvements

Current reward functions could be enhanced by incorporating better risk-return trade-offs. For example, instead of just using returns, risk-adjusted metrics like Sharpe ratio or Sortino ratio could be utilized as reward function. Additionally, utilizing a reward function that factors in transaction costs, slippage, and liquidity constraints could help the agent simulate real-world trading and optimize for net returns more realistically.

Another improvement is feature extraction, which can be achieved by incorporating more advanced techniques for analyzing time-series data, such as attention mechanisms or transformers. These methods have demonstrated superior performance in capturing long-term dependencies in sequential data.

From the testing results, sentiment-aware A3C-CNN-BiLSTM agent has relatively longer drawdown period. To better control risk, risk management measures can be incorporating into the current framework. A risk-aware DRL agents could be developed on top the current one, which actively manage and mitigate risk. This could involve implementing stop-loss strategies, dynamic position sizing, or volatility-adjusted portfolios to ensure the agent controls downside risk more effectively.

Furthermore, since trading decisions are both discrete and continuous, a hybrid discrete-continuous action spaces approach could be developed. The approach could combine Twin Delayed Deep Deterministic

Policy Gradient (TD3) with A3C. It involves using A3C for discrete policy decisions i.e. buy, hold or sell and TD3 for continuous decisions within those policies i.e. quantity to trade.

6.2 Future Work

An extension and future work of the current framework is to work on multi-agent reinforcement learning, where multiple agents with potentially different strategies interact in the same environment. Agents could be specialized, for example, one agent is for long-term risk-adjust return, one agent is for short-term return and another one is for volatility. They can cooperate with each other and share information to improve the overall performance.

I could also experiment with other DRL algorithms such as Soft Actor-Critic (SAC) and Twin Delayed Deep Deterministic Policy Gradient (TD3) and explore ensemble agents that combine these with the current sentiment-aware A3C-CNN-BiLSTM agent. SAC uses a maximum entropy framework to encourage exploration, which could provide better performance in volatile markets. TD3 is an enhancement of the DDPG (Deep Deterministic Policy Gradient), designed to address the instability and overestimation biases present in standard actor-critic methods in continuous action spaces. By leveraging the strengths of SAC's exploration capabilities, TD3's stability in continuous action learning, and the sentiment-aware enhancements of A3C, an ensemble approach could potentially yield a more adaptive and higher-performing trading strategy.

Another future work I'm interested is to develop a market regime detection method to identify different phases i.e. bull, bear, volatile markets, and dynamically adjust the trading agent's algorithm. That could potentially improve the performance in diverse market conditions and reduce risk exposure. For instance, using TD3 for volatile markets, that can quickly adjust position sizes, and switch to A3C or PPO during stable periods for more exploratory and consistent policies.

By considering these potential improvements and future work, the sentiment-aware A3C-CNN-BiLSTM agent will be further enhanced to deliver more robust, adaptive, and profitable trading strategies across market environments.

7. Conclusion

This research project explored the application of Deep Reinforcement Learning (DRL) for systematic equities trading strategy with a particular focus on the A3C (Asynchronous Advantage Actor-Critic) algorithm enhanced by a CNN-BiLSTM architecture. The A3C algorithm was chosen due to its ability to effectively balance policy learning (actor) and value estimation (critic), providing a solid foundation for decision-making in dynamic and complex trading environments. The CNN-BiLSTM architecture was designed to improve feature extraction by combining the spatial representation capabilities of Convolutional Neural Networks with the temporal sequence modeling strengths of Bidirectional Long Short-Term Memory networks. This combination enabled the agent to capture both local and long-term dependencies in market data, allowing for more informed trading decisions.

The inclusion of sentiment scores further improved the agent's performance, resulting in the sentiment-aware A3C-CNN-BiLSTM agent outperforming traditional DRL baselines such as DQN-Vanilla, A2C-Vanilla, and PPO-Vanilla across major stock indices S&P 500, NASDAQ, and DJIA. The sentiment-aware enhancements allowed the agent to make more informed trading decisions by integrating both quantitative market indicators and qualitative sentiment data, which led to higher cumulative returns and better risk-adjusted performance.

The research demonstrates that combining DRL algorithms A3C with architectures like CNN-BiLSTM and additional features such as sentiment data can substantially enhance trading performance. The results highlight the potential of deep reinforcement learning in developing robust and profitable trading strategies, paving the way for further exploration of advanced algorithms, hybrid approaches, and market-specific adaptations. Future work can build on these findings by incorporating multi-agent systems, alternative reward functions, and more complex ensemble agents to enhance the adaptability and profitability of trading strategies in diverse market conditions.

References

Bollen, J., Mao, H., & Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science,* 2(1), 1-8.

Chen, J., Shen, C., & Yi, J. (2018). Stock prediction using deep learning and sentiment analysis. *Proceedings of the 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE),* 90-94.

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2016). Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE Transactions on Neural Networks and Learning Systems,* 28(3), 653-664.

Fischer, T., & Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2), 654-669.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 1861-1870.

Hu, Z., Liu, B., Bian, J., Liu, T. Y., & Liu, B. (2018). Listening to Chaotic Whispers: A Deep Reinforcement Learning Network for Stock Trading. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2820-2828.

Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059.*

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature,* 518(7540), 529-533.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., ... & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 1928-1937.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347.*

Xu, Y., & Cohen, S. (2018). Stock Movement Prediction from Tweets and Historical Prices. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL),* 1970-1979.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long short-term memory. *PloS one,* 12(7), e0180944.

Koratamaddi, P., Wadhwani, K., Gupta, M., & Sanjeevi, S. G. (2024). Market sentiment-aware deep reinforcement learning approach for stock portfolio allocation. *Engineering Science and Technology, an International Journal*, 24, 1-13.

Huang, Y., Wan, X., Zhang, L., & Lu, X. (2024). A novel deep reinforcement learning framework with BiLSTM-Attention networks for algorithmic trading. *Expert Systems With Applications, 240,* 122581.

Liu, X.-Y., Xiong, Z., Zhong, S., Yang, H. (B.), & Walid, A. (2024). Practical deep reinforcement learning approach for stock trading. *Mathematics of Systems Research Department, Nokia-Bell Labs.*

Cheng, L.-C., Huang, Y.-H., Hsieh, M.-H., & Wu, M.-E. (2021). A novel trading strategy framework based on reinforcement deep learning for financial market predictions. *Mathematics,* 9(23), 3094.

Xu, Y., Feng, S., Yang, J., Wang, X., & Yang, C. (2021). Stock Trading Strategy Based on A3C Reinforcement Learning Algorithm*.* *Mathematical Problems in Engineering*, 2021, 1-11.