A Novel Approach to Automated Stock Trading Using Reinforcement Learning

Ellie Burton, Will Riemer, Caleb Thompson, and Cloie Dale

***Abstract*— Developing robust and automatic stock trading strategies in modern volatile markets remains a significant challenge. We introduce in this paper a hybrid approach that harnesses the strengths of different reinforcement learning techniques to learn and fine-tune strategies in stock holding for an improved outcome. The proposed system is an ensemble of Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC). The unique abilities these algorithms leverage include the stability of PPO; the continuous control skills of TD3, and the entropy-driven exploration mechanism of SAC all provide an excellent fit for the environment under scrutiny. This enables this strategy to depict market dynamics better and to manage the peculiarities of continuous action spaces, such as those of stock trading. We compare this approach to a previous ensemble method that utilizes PPO, DDPG, and A2C. As TD3 and SAC are direct successors of DDPG and A2C, respectively, we expect to see an improved result compared to the pervious ensemble. This expectation is realized through outperforming the previous ensemble and the DIJA in returns during the testing period.**

***Index Terms*— Algorithmic Trading, Continuous Control, Deep Reinforcement Learning, Ensemble Methods, Market Simulation, Portfolio Optimization, Risk Management, Sharpe Ratio, Stock Trading**

I. INTRODUCTION

In current fast-paced financial markets, it has become extremely valuable to design an automatic trading strategy that can be consistent throughout the intricacies and volatility of the stock market. Traditional portfolio optimization requires a "first-then" methodology for estimating risks and returns before calculating optimization strategy. This often ultimately fails with fluidity and uncertainties present in stock trading. This opens the door for a novel strategy using applied deep reinforcement learning directly for guiding trading strategies via market experience.

Evolving DRL techniques in stock trading show promising results, each carrying their respective strengths as well as weaknesses. Deep Q-Networks (DQN) offers a very robust framework for decision-making in environments with discrete actions, while Twin Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor-Critic (SAC) algorithms have great strengths in continuous action spaces which are essential to mimicking realistic trading conditions. The TD3 method solves the well-known overestimation problems in deterministic policy gradient methods, and in SAC, the exploration of market dynamics is encouraged due to the use of entropy regularization.

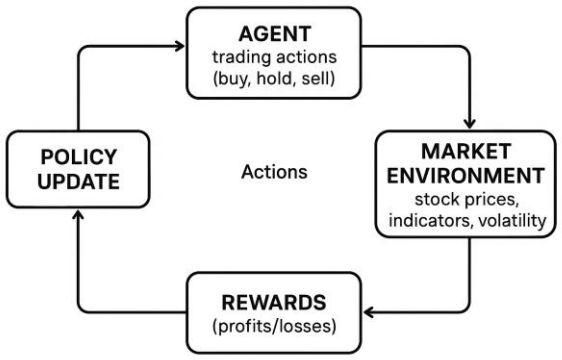

Figure 1. Overview of Reinforcement Learning in Algorithmic Trading

Guided by these complementing features, this paper suggests an ensemble trading strategy that combines PPO, TD3, and SAC. This ensemble will allow our strategy to change with market trends to maximize profits and minimize risk. In this approach, we develop a realistic market simulator where the individual states and actions are defined to incorporate the multiplicity of financial data. Each algorithm is trained independently within this environment for the purpose of maximizing cumulative returns given the consequent factors of risk. This is accomplished by risk metrics in the gain function, such as the Sharpe ratio, to create a trade-off between profit and volatility.

The rest of this paper is organized as follows. Section 2 overviews related works on algorithmic trading. Section 3 details our methodology, including each trading strategy and the proposed ensemble method. Section 4 reviews our exploratory data analysis and preprocessing. Section 5 details all training, validation and experimentation that was done. Section 6 provides insights into the results of our research and section 7 provides final notes on the work and further considerations.

II. RELATED WORKS

This section will include a comprehensive review of the relevant literature on algorithmic trading and the different approaches taken to implement it for portfolio optimization. We will look at works surrounding the core concepts of algorithmic trading, its effects on trading markets, and various machine learning and reinforcement learning approaches to handling the problem of algorithmic trading.

A. Algorithmic Trading

Nuti et al. [1] offers a comprehensive overview of algorithmic trading, covering its applications in corporate trading, including the use of fundamental, technical, and quantitative indicators for pre-trade analysis. It explores the development of algorithms designed to determine entry and exit strategies for trades. Algorithmic trading is particularly effective in high-frequency trading (HFT) within highly liquid markets. For it to be effective, algorithmic trading requires an environment where trades can be executed swiftly and frequently. This approach is most commonly applied in U.S. stock markets, large-cap cryptocurrencies, exchange-traded funds (ETFs), and foreign exchange markets.

The widespread use of algorithmic trading and its impact on the stability of foreign exchange markets is explored by Chaboud et al. [2], where a correlation was identified between algorithmic trading and reduced excess volatility and arbitrage opportunities. However, conflicting findings are presented by Boehmer et al. [3], where they studied 42 equity markets and found that algorithmic trading consistently contributed to increased short-term volatility. While the impact of algorithmic trading on market volatility remains debated, the increased liquidity associated with the high-frequency nature of many algorithmic strategies suggests that it enhances financial flexibility and stimulates market activity.

In recent years, artificial intelligence has been increasingly adopted in corporate settings to enhance analysis beyond traditional technical and fundamental indicators in order to optimize algorithmic trading strategies. Reinforcement learning has gained significant attention in the literature on AI and algorithmic trading due to its intuitive reward-based framework. This allows learning agents to adapt to market volatility, offering promising solutions for improving decision-making in complex and unpredictable environments. The following section provides a review of relevant literature on reinforcement learning applications in algorithmic trading.

B. Q-Learning Applications

Deep Q Networks (DQN) are among the most widely studied reinforcement learning methods in algorithmic trading. Li et al. [4] compare two prominent reinforcement learning approaches, DQN and Asynchronous Advantage Actor-Critic (A3C), within the context of stock market trading. The study trains and tests both models on historical data from five high liquidity stocks, demonstrating how integrating a Long Short-Term Memory (LSTM) deep learning framework on top of each method can enhance performance. Certain practices were put into place to decrease noise, such as removing the open time from the trading period as news and notices typically cause the most volatility at these times and employing stacked denoising autoencoders (SDAE) to obtain a more robust representation of each operation. The improved methods using LSTM, coined DQN-extended and A3C-extended were implemented and tested against a baseline buy and hold strategy as well as their predecessors. Both enhanced methods outperformed their parent methods, while A3C-extended significantly outperformed DQN-extended.

Massahi and Mahootchi [5] introduce an improved variant of DQN, known as Double DQN (DDQN), to implement a strategy for trading in futures markets. DDQN, which uses two Q-functions to evaluate and improve the policy, provides a less biased estimate of Q values. While DQN showed superior results in short-term scenarios, DDQN notably reduced investment risk, illustrating the model's ability to make more informed decisions and generate more reliable trade signals.

Shavandi and Khedmati [6] propose a multi-agent framework for trading foreign exchange currencies, in which each agent is trained and fitted to perform only in a specific timeframe. A DQN trading algorithm for multiple agents was devised in a hierarchical manner, where agents trade collaboratively, and flow of information is top down. Behaviors of agents in higher time frames were transferred to agents of lower time frames. This allows the lowermost timeframe agent to generate optimal trading signals. The results of the framework were tested in a non-episodic environment and measured against the results of each agent acting individually, and it was found that the framework significantly outperformed the individual agents, while agents trained in lower time frames outperformed those in higher timeframes.

Although Deep Q Networks provide proof of concept for the effectiveness of reinforcement learning application in algorithmic trading, they face limitations that make them an unsuitable choice for practical application in real world trading scenarios. DQN is fixed to a discrete action space, making it unsuitable for highly dynamic trading scenarios where multiple stocks need to be bought and sold at variable amounts. DQN must approximate Q values for every state-action pair, which can become inefficient and unreliable in highly dimensional observation spaces that are needed to provide accurate trading models. A policy iteration approach is a more suitable choice for algorithmic trading as they can act in continuous action spaces and generalize about the environment in states that haven’t yet been visited. The flexibility offered by these models makes them an intuitive solution to algorithmic trading problems. The following section discusses some of the policy iteration approaches used in algorithmic trading.

C. Policy Iteration Learning Applications

Twin delayed deep deterministic policy gradient (TD3) is an improved version of the deep deterministic policy gradient method (DDPG), a method that has been well published with regards to the problem of algorithmic trading. Despite this, TD3 has not received much attention within the literature when it comes to application in trading scenarios. TD3 is well suited for continuous action spaces, which can help when implementing a trading strategy that involves holding, buying, and selling varying amounts at any given time step. It also proves to be more stable than DDPG by using double Q learning and delayed policy updates.

Hossain and Rahman [7] propose a TD3 approach trading using historical stock market closing prices. Rewards were calculated using a logarithmic transformation of the net reward of each trade made. An episodic environment was used to train, validate, and test the model, with it being examined in both a stock market and cryptocurrency market environment. The model was tested against standard algorithmic trading algorithms, two supervised learning models, and a DQN model. Models were evaluated based on total return and

Sharpe ratio, where it was found that TD3 showed better results than its machine learning counterparts with respect to the evaluation metrics.

PPO has become popularized as being one of the most efficient reinforcement learning algorithms in continuous action space problems. Park et al. [8] propose a PPO approach for trading a single stock-Apple (AAPL)-given information on the price performance of the S&P 500 as well as relevant interest rate data from the given time period. Three models were implemented, one using only the historical stock data in its observation space, one using the historical stock data and the S&P data, and one that also included the interest rate data. Rewards were calculated using profit when the agent made a trade, and by subtracting opportunity cost if the agent wasn't making trades. This was done to ensure the agent is not withholding from trading to minimize risk; however, results suggest that it may have had a negligible effect. All three models produced a negative average return per trade, but this can partially be attributed to the lack of sophistication of the environment. The third model showed extremely low variance in its action selection, showing promise in the ability of PPO to be used to learn a stable, low volatility trading strategy.

Han and Wang [9] formulate a novel SAC approach in which the actor produces a coarse policy for each individual stock which is acted upon using decision logic. Market turning point predictions and a ranking of each stock's relative value are used to decide which stocks to buy and the quantity of orders. The proposed framework was tested against other SAC frameworks and against the market. Results showed that each while the proposed model produced stronger results than the baseline, both models outperformed the market, exhibiting SAC's ability to pinpoint optimal policy.

D. Ensemble Approach

The work in this paper is heavily influenced by the work from Yang et al. [10], which implements an ensemble approach using PPO, DDPG, and A2C methods to trade 30 stocks from the Dow Jones Index. The ensemble approach outperforms each individual method as well as the Dow Jones average, proving it to be a reliable model for trading in a high dimensional environment. We propose an improved ensemble strategy, using TD3 and SAC instead of DDPG and A2C. TD3 provides a more stable solution than DDPG by using twin critics and policy smoothing, while SAC provides more exploration and better performance in high dimensional environments than A2C. Using the same dataset and splits for training, testing, and validation, we look to test and compare the results of our ensemble strategy against that of [10], with the expectation that our approach is more reliable and thus will generate a more profitable strategy.

III. METHODS

In our work, we implement a number of reinforcement learning models from the *Stable Baselines* library, including Proximal Policy Optimization (PPO), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC). Using the process, environments, and results described by Yang et al. [10], we can compare the aforementioned models and our ensemble approach with the results from the previous study.

A. Problem Description

Mirroring the approach described in Yang et al.[10], our problem environment was modeled in the following way:

State: The state for our environment contains the information below [10]. Please note that the vectors are of size 30 due to the input data tracking the prices of 30 major stocks over a period of just over 10 years.

- The current account balance for the model
- A vector of size 30 representing the closing prices for each of the stocks being tracked for a given day.
- A vector of size 30 representing how many shares of each stock is currently held by the model.
- A vector of size 30 containing the current Moving Average Convergence/Divergence (MACD) indicator values for each of the stocks.
- A vector of size 30 containing the Relative Strength Index (RSI) indicator values for each stock.
- A vector of size 30 containing the Commodity Channel Index (CCI) indicator values for each stock.
- A vector of size 30 containing the Average Directional Index (ADX) indicator values for each stock.

The technical indicators (MACD, RSI, CCI, & ADX) are used for tracking price trends and momentum for stocks. These are discussed in more depth in the data section.

Actions: There are 3 potential actions which the model may perform for each stock: buying, selling, or holding [10]. The action space is continuous as the model is allowed to buy any number of shares, hold its current number of shares, or sell any amount of its current held shares for each of the 30 stocks [10].

Reward: The reward calculation for each step involves simply calculating the difference between the model's current balance and the previous balance. This result is scaled accordingly to help stabilize the learning of the model [10].

Assumptions: Once again following the procedure detailed in Yang et al. [10], we incorporate a number of assumptions and constraints into our model's environment. These include the assumption that our model can immediately buy or sell stock at the given close price, and the assumption that there is a transaction fee of 0.1% of the transaction value [10].

B. Proximal Policy Optimization (PPO)

PPO is a reinforcement learning approach that builds upon existing policy gradient methods by providing simpler implementation and improved sample complexity [11]. PPO is based on trust region policy optimization (TRPO). TRPO uses an objective function with importance sampling to compute policy loss and only subjects itself to a policy update if the difference between the old and new policy (calculated using Kullback-Leibler divergence) is within a user defined constraint called the trust region. Below is the objective function algorithm for TRPO:

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}\hat{A}_t\right] \\ \text{subject to} \quad & \hat{\mathbb{E}}_t[\mathrm{KL}[\pi_{\theta_{\text{old}}}(\cdot \mid s_t), \pi_\theta(\cdot \mid s_t)]] \leq \delta. \end{aligned}$$

Figure 3.1.1 TRPO Algorithm [11]

Using the trust region disallows excessive policy updates that may skew the policy parameters, thus solving the unconstrained optimization problem [11]. PPO significantly improves this implementation by replacing the trust region with a clipping function that reduces the size of policy updates if they are outside of a defined range with respect to the old policy. This way, instead of invalidating extreme policy updates, they are clipped to the bounds of the acceptable range, allowing for more effective updates and a far less computationally expensive implementation. Below is the clipped objective function used in PPO:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Figure 3.1.2 PPO Clipped Objective Function [11]

PPO implores an actor critic network with shared parameters for learning the policy and value functions [11]. A value function error coefficient and an entropy bonus are added to ensure both value function learning and exploration are being optimized for. The resulting final loss equation is depicted below:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right],$$

Figure 3.1.3 PPO Loss Function [11]

PPO will run an iteration of the policy on T timesteps to collect data for updating. The advantage function is calculated for the batch of steps and stochastic gradient descent is used to perform mini batch updates from the batch of T steps [11]. Below is a high-level visualization of the algorithm:

Algorithm 1 PPO, Actor-Critic Style

```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy π_θold in environment for T timesteps
        Compute advantage estimates Â_1, ..., Â_T
    end for
    Optimize surrogate L wrt θ, with K epochs and minibatch size M ≤ NT
    θ_old ← θ
end for
```

Figure 3.1.4 Pseudocode of the PPO Algorithm [11]

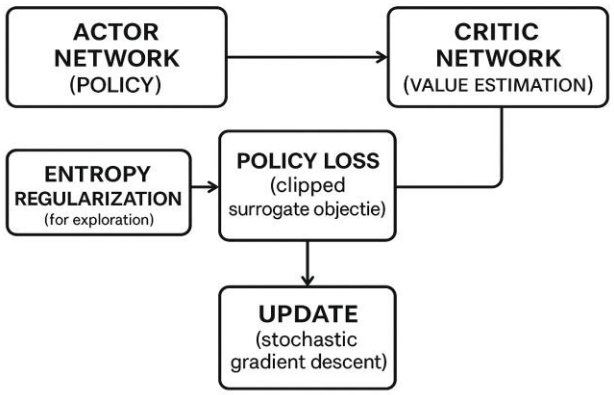

Figure 3.1.5 Structure of the PPO Algorithm

C. Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is a direct successor of DDPG, which is one of the most popular trading strategies because of its superior performance in continuous action spaces [10]. TD3 uses an almost identical strategy to DDPG. However, it utilizes 3 modifications correct for the overestimation bias caused by function approximation error common with the DDPG method, which are highlighted below in Figure 3.2.1 [12].

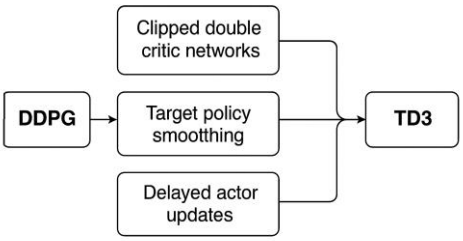

Figure 3.2.1 TD3 Algorithm Enhancements over DDPG

Firstly, TD3 utilizes clipped double critic networks instead of a single critic. The method uses two distinct critic networks and takes the minimum value from the critic estimates at each step to form the target value. Secondly, TD3 implements delayed actor network updates to minimize the accumulation of residual error caused by the critic and actor networks updating simultaneously. Thirdly, TD3 introduces target policy smoothing by applying random noise to the target policy.

TD3 maintains a replay buffer R that holds the transition vector (s, a, s', r, d) where s is the current state, a is the current action, s' is the next state, r is the return for the current state-action pair, and d is a binary value that expresses if s' is the terminal state [12]. At each time step, the learning target update can be given by:

$$y = r + \gamma \min_{i=1,2} \quad Q_{\theta_i'}\left(s', \pi_{\varphi_1}(s')\right) \quad (1)$$

Figure 3.2.2. TD3 Learning Target Update [12]

where π is the actor, Q_i is the two critics, and gamma is the discount factor. Both Q_1 and Q_2 are estimated using π, and π is independently optimized in respect to Q_i because the critic functions are upper bounded by each other [12].

TD3 adds a small amount of random noise to the target update to limit target variance. This is done to regulate variance to ensure that similar actions should have a similar value [12]. The noise is introduced as a random integer between –c and c where c is any constant integer. To do this the target update is modified to:

$$y = r + \gamma Q_{\theta'}\big(s', \pi_{\varphi'}(s') + \varepsilon\big),$$

$$\varepsilon \sim clip(N(0, \sigma), -c, c) \qquad (2)$$

Figure 3.2.3. TD3 Target Update [12]

The actor is updated on a delay to limit the accumulation of residual error from Q and pi being updated simultaneously, as is common in DQN based algorithms. In TD3, the policy is updated every *n* time steps while Q_i is updated at every time step. By delaying the update, the value error remains as small as possible limiting error accumulation [12].

The improved function approximation within TD3 makes it better suited for stock trading than DDPG. Within our ensemble method the implementation of TD3 found in the stable baseline library, which was originally implemented by OpenAI, will be used [13]. The pseudo code for the TD3 algorithm used by the library can be found in figure 3.2.4.

Algorithm 1 Twin Delayed DDPG

Input: initial policy parameters θ, Q-function parameters ϕ_1, ϕ_2, empty replay buffer $\mathcal{D}$
Set target parameters equal to main parameters $\theta_{\text{targ}} \leftarrow \theta$, $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
repeat
  Observe state s and select action $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{Low}, a_{High})$, where $\epsilon \sim \mathcal{N}$
  Execute a in the environment
  Observe next state s', reward r, and done signal d to indicate whether s' is terminal
  Store (s, a, r, s', d) in replay buffer $\mathcal{D}$
  If s' is terminal, reset environment state.
  if it's time to update **then**
    for j in range(however many updates) **do**
      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
      Compute target actions

$$a'(s') = \text{clip}\left(\mu_{\theta_{\text{targ}}}(s') + \text{clip}(\epsilon, -c, c), a_{Low}, a_{High}\right), \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

      Compute targets

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

      Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s, a) - y(r, s', d)\right)^2 \qquad \text{for } i = 1, 2$$

      if j mod `policy_delay` $= 0$ **then**
        Update policy by one step of gradient ascent using

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s))$$

        Update target networks with

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho)\phi_i \qquad \text{for } i = 1, 2$$
$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta$$

      end if
    end for
  end if
until convergence

Figure 3.2.4. Stable Baselines TD3 Pseudocode [13]

D. Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) is at the forefront of the recent off-policy actor-critic algorithms, distinguished by taking a maximum entropy framework into account in the learning world. The main distinction that sets Soft Actor-Critic apart from traditional methods of reinforcement learning, which seeks to merely maximize the expected return, is that this algorithm computes a combined objective with a maximization of entropy in terms of its stochastic policies and their further exploration.

Soft Actor-Critic aims to maximize the cumulative reward and the expected entropy of the policy, thereby balancing exploitation and exploration. The objective function is defined as such that the policy is rewarded for maximal return as well as maintaining a certain level of randomness. This is mathematically expressed below:

$$J(\pi) = \sum_t \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}\big(\pi(\cdot \mid s_t)\big) \right]$$

Figure 3.3.1 SAC Objective Function

Similar to other off-policy approaches, a replay buffer R is created, responsible for maintaining transition tuples (s, a, s′, R, d), where s is the current state, a is the action taken, s′ is the next state, R is the reward, and d is the terminal flag. The critic networks in SAC learn a soft Q-value function. This requires the entropy term to be merged with the standard Bellman backup. An overestimation bias is mitigated from a value-based method because SAC adopts a twin-critic approach, with independent critics that drive toward two separate Q-value estimates made for use within the Bellman error calculation:

$$y(r, s') = r + \gamma \, \mathbb{E}_{a' \sim \pi} \left[\min_{i=1,2} Q^i_{\text{target}}(s', a') - \alpha \log \pi(a' \mid s') \right]$$

Figure 3.3.2 SAC Target Soft Q-Value Function

A further key detail is the automatic entropy coefficient update. This allows the possibility of dynamic compensation for the entropy term relative to a reward; as a result, the degrees of exploration throughout training are automatically calibrated.

Thus, the reparameterization trick is employed by the policy gradients to propagate gradients through the stochastic action sampling process while conditioning efficient training and ensuring stable policy updates through disentanglement of the reward and entropy parts of the objective function.

Overall, the union of the twin critic networks, entropy-augmentation of the objective, and interleaved adaptive learning rate schedule renders SAC an excellent performer for advanced environments involving continuous action spaces such as trading stocks. We will use the implementation of SAC available in the stable-baselines library; a pseudocode for the SAC algorithm in the library is presented in Figure 3.3.3.

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$
  end for
end for

Figure 3.3.3 Stable Baselines SAC Pseudocode

E. Ensemble Approach

Each one of the previously described agents (PPO, TD3, SAC) are well suited for stock trading. However, each agent will perform differently in a given market environment. Since market environments change frequently, it is best that the trading strategy changes to reflect the market. Our ensemble approach will allow the strategy best suited for the current market environment to predict and trade for a set time period before the agents are reassessed. The steps of our approach are as follows:

- Step 1: We use a growing window of *n* months to train the three agents concurrently. We have chosen to retrain the agents every 3 months.
- Step 2: A 3-month validation rolling window follows the training window to verify which agent performs best in the current market. The agent with the largest Sharpe Ratio is considered to be performing best [10]. Sharpe Ratio is calculated as:

$$sharpe\,ratio = (\bar{r}_p - r_f)/\sigma_p$$

Figure 3.4.1. Sharpe Ratio Equation

where r_p is the expected return of the portfolio, r_f is the risk-free rate, and σ_p is the standard deviation of the portfolio.

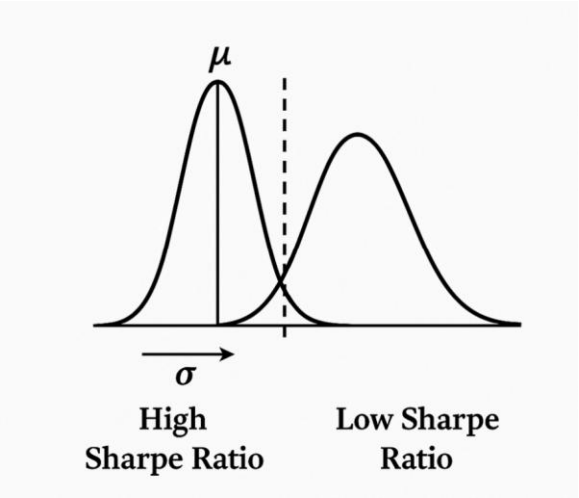

Figure 3.4.2 Sharpe Ratio Visualization

- Step 3: The best agent as determined by step 2 is used to predict and trade for the next 3 months before the model returns to step 1.

This method allows us to prioritize the overall portfolio value relative to the risk profile to maximize returns to a variety of market environments.

IV. DATA

We use 30 stocks of the Dow Jones Industrial Average (DJIA) as our trading universe. Historical daily price data (open, high, low, close) and volume, along with other relevant information, are obtained through the Compustat database via the Wharton Research Data Services (WRDS) [14]. This dataset spans the period from January 1, 2009, to May 8, 2020. By using these 30 large-cap stocks, we aim to study a liquid and widely tracked index that is representative of the U.S. equity market.

A. Exploratory Data Analysis

Before any modeling or trading strategy design, we perform exploratory data analysis (EDA) on the dataset to gain insights into its structure and characteristics and better inform our decision making. Relevant data columns are outlined to the right in Table 1.

We next make certain completeness checks of the dataset, including missing or null entries in the price, volume, or fundamental fields. This dataset is complete and requires no cleansing in this regard. This data does not require outlier detection, as we do not wish to exclude unusual market circumstances from the training. We then explore summary statistics of the data to inform our understanding. Figures 4.1, 4.2, and 4.3 show relevant data analysis.

TABLE I
RELEVANT DATA COLUMNS

Column	*Description*
datadate	Date
tic	Indicates which stock is referenced
adjcp	Closing price adjusted for stock splits and dividends
open	Market open price of a stock for a given day
high	The highest traded price of a stock during that trading day
low	The lowest traded price of a stock during that trading day
volume	The total number of shares traded during that day.
macd	Moving Average Convergence Divergence: a momentum-based technical indicator
rsi	Relative Strength Index: A momentum oscillator that measures the magnitude of recent price changes
cci	Commodity Channel Index: a momentum-based technical indicator
adx	Average Directional Index: quantifier of the strength of a trend
turbulence	Measure of market volatility/risk

Figure 4.1 Adjusted Closing Prices Over Time

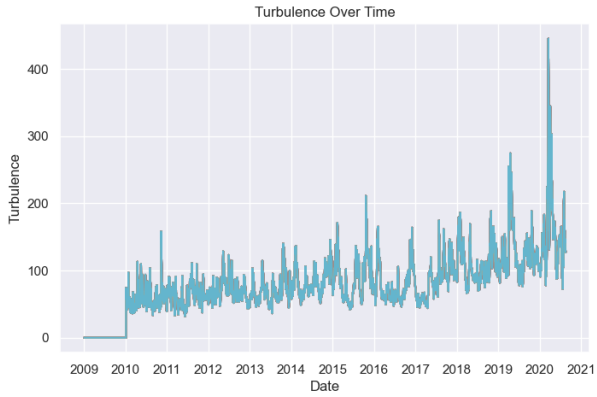

Figure 4.2 Turbulence Over Time

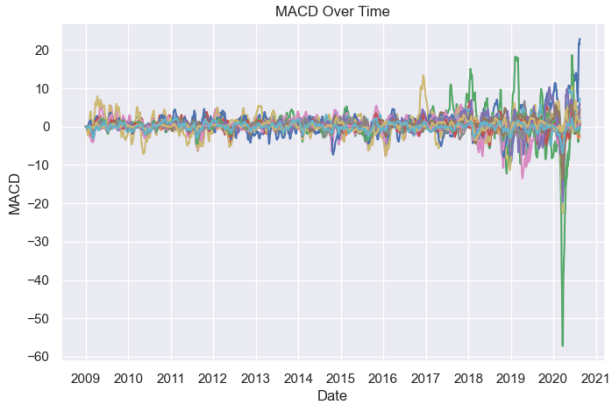

Figure 4.3 MACD Over Time

Closing price analyses illustrate that most DJIA stocks exhibit moderate positive skew over time, which is consistent with typical equity returns. We also see a slight positive trend in turbulence with a large spike in 2020, almost certainly due to the economic effects of the COVID-19 pandemic. MACD shows relatively stable values, with more volatility after 2017. The major outlier (green line) around 2020 is BA, The Boeing Company, and can also likely be explained via travel shutdowns with the COVID-19 pandemic. Overall, we observe expected trends among the data.

B. Preprocessing

After completing the EDA, we preprocessed the data to prepare it for modeling and strategy development. Following industry-standard practices for back-testing and performance evaluation, we split the dataset into in-sample (training/validation) and out-of-sample (testing) periods by date:

1) **Training Subset (In-Sample)**
   (01/01/2009 - 09/30/2015)
   This subset is used to train reinforcement learning agents. The algorithms iteratively update their parameters to learn optimal policies based on historical market conditions.
2) **Validation Subset (In-Sample)**
   (10/01/2015 - 12/31/2015)
   We then evaluate the agents' performance on this subset to prevent overfitting and to fine-tune hyperparameters. We primarily use the performance metrics of Sharpe ratio.
3) **Trading Subset (Out-of-Sample)**
   (01/01/2016 - 05/08/2020)
   The final subset is treated as unseen data to assess how our algorithms perform under market conditions not used during training or validation.

This process is summarized in Figure 4.4 below.

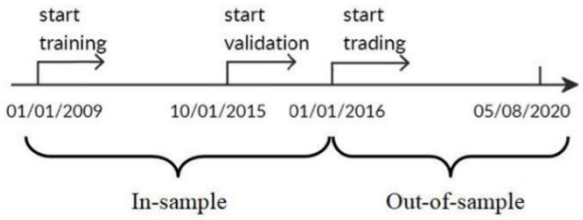

Figure 4.4 Stock Data Splitting

V. EXPERIMENTS

To ensure consistency and efficiency, the reinforcement learning algorithms utilized in our experiments (PPO, TD3, SAC, DDPG, and A2C) were implemented using the Stable Baselines library. Stable Baselines simplifies the training and hyperparameter tuning processes, providing a standardized environment and comprehensive functionalities suitable for conducting comparative studies of these RL algorithms. This also allows for consistency with previous research's ensemble methods. We perform an experiment to test the validity of the proposed method by comparing the performance against previously proposed methods and the Dow Jones Industrial Average (DJIA).

The environment used for all algorithms is based on the one described in the previous ensemble method to ensure a fair comparison between the methods [10]. The observation space for the prescribed environment includes the current account balance, stock price for each company for the given day, number of stocks owned, the four technical indicators previously described (MACD, RSI, CCI, ADX), and market turbulence for the day. The given algorithm is allowed to buy, sell, or hold each stock on a given day. Since the amount for each action is not set, the action space remains continuous. The change in the portfolio value is given as the reward function. Additionally, if the turbulence crosses the set threshold of 90% higher than average, the algorithm is instructed to liquidate all assets until the market turbulence drops below the threshold.

A. Turbulence

While reinforcement learning algorithms can be applied to complex problems such as financial trading, rare and unfamiliar experiences can throw a wrench in their effectiveness. The primary threat in this experiment involves sudden, unexpected world events which may greatly affect the stock market. As the environment deals solely with financial information, and no information regarding global events was taken into consideration, we decided to mirror the approach taken in [10] by using a turbulence threshold for all models. Our dataset contained a turbulence index for every day of trading, calculated using equation 5.1. Please note that r_t is a vector of stock returns for the time period t, μ is a vector of the average returns for the stocks, and Σ is the covariance matrix for the historical returns for the stocks [10]. The index values were already included within our dataset.

$$turbulence_t = (r_t - \mu)\Sigma^{-1}(r_t - \mu)'$$

Figure 5.1 Turbulence Index Equation [10]

Similar to [10], we implemented a turbulence threshold, above which the models would liquidate their entire portfolio at the current market price and hold cash until the turbulence subsided. This helped hedge against sudden unfamiliar changes which could lead to negative returns for the reinforcement learning agents.

B. Training & Validation Process

The training and validation process for the different algorithms mirrors the approach taken by Yang, Liu, Zhong, and Walid, and is as follows [10]. After splitting the data into training (in-sample) and trading (out-of-sample) sets as described in figure 4.4, a repetitive process occurs where each individual algorithm is trained, validated, and tested on subsequent 3 month periods for the duration of the data. First, as discussed previously, the algorithms are pretrained on the data from January 2009 – October 2015. Then, each model is validated on the next 3-month period, from October 2015 to January 2016. In the ensemble methods, the highest performing algorithm, as determined by the Sharpe ratio, is used to trade in the following 3-month period, the first one being January 2016 – April 2016. This process then repeats, with the 3-month window shifting forward. The second iteration involves training all algorithms on the data from October 2015 – January 2016, Validation then occurs from January 2016 – April 2016, and then trading happens from April 2016 – July 2016.

This then repeats for every 3-month period until the final trading window: April 2020 – July 2020. Figure 5.2 below illustrates this process. Please note that for the methods involving a single reinforcement learning algorithm, the same training and validation process occurs to control the experiment, however, no choice is made between algorithms as only one is involved and tested at a time.

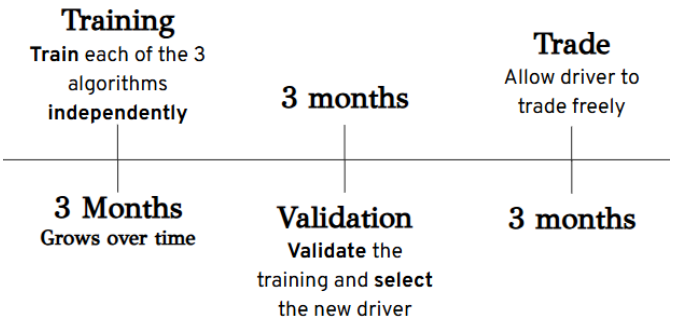

Figure 5.2 Training/Validation/Trading process for ensemble methods.

The success of each algorithm is determined by the return of their portfolio over the trading subset timeframe. These returns are then compared to the returns from the Dow Jones Industrial Average (DJIA) for the same amount of time as well as those from other methods. A strong method will have a

higher return than DJIA and outperform most, if not all, of the reinforcement learning methods tested.

VI. RESULTS

This section outlines the results of our experimental methods. We will explore the effectiveness of individual and ensemble algorithms and compare each with the Dow Jones Industrial Average (DJIA.)

A. Algorithm Performance

Figure 5.1 recaps the execution of individual reinforcement learning algorithms plotted against each of their portfolio gains. Out of all individual algorithms, TD3 shows superiority over the others with relative holding of returns. DDPG exhibited solid performance with a slightly higher level of volatile returns.

It is noteworthy that the flat regions shown on the cumulative return graph represent times when the market is volatile. At such times, the model will not trade; instead, it will simply hold its cash to minimize any potential losses. This conservative strategy strengthens the stability and resilience of our ensemble method in challenging times.

Figure 5.2 shows the portfolio value percent increase of tested algorithms, including our proposed ensemble strategy, as well as the DJIA. Our ensemble outperformed all three single algorithms. The proposed ensemble (PPO/TD3/SAC) also outperformed the ensemble set of the previously proposed ensemble method (PPO/DDPG/A2C.) It can also be seen that while individual algorithms had varying degrees of volatility and returns, the ensemble methods were significantly less prone to high volatility leading to larger returns. Both ensemble methods also benefited from smoother and more robust growth

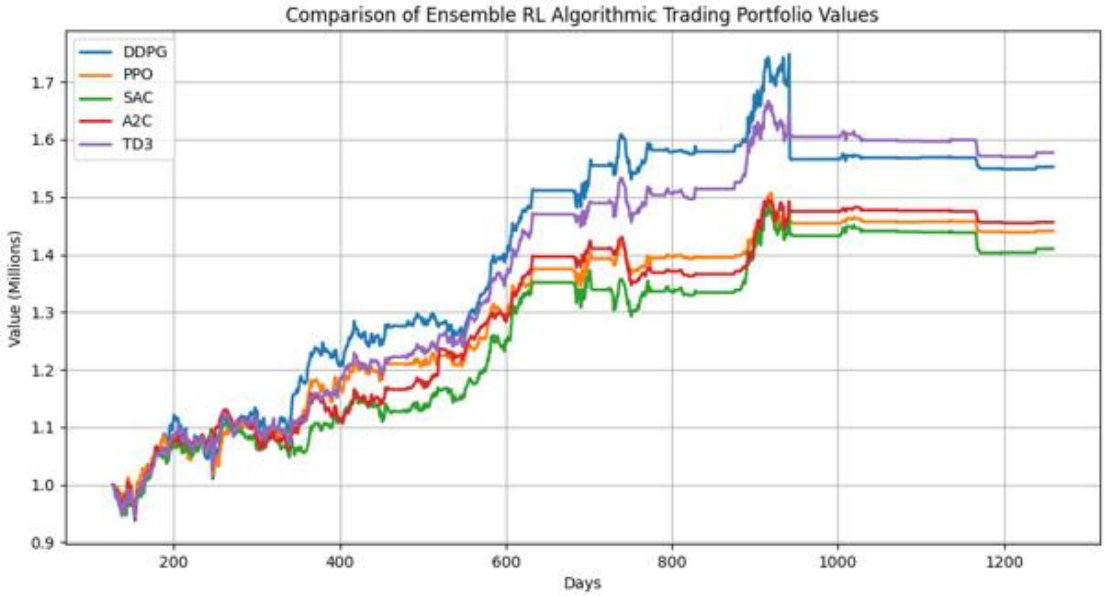

Figure 5.1 Individual Algorithm Portfolio Values

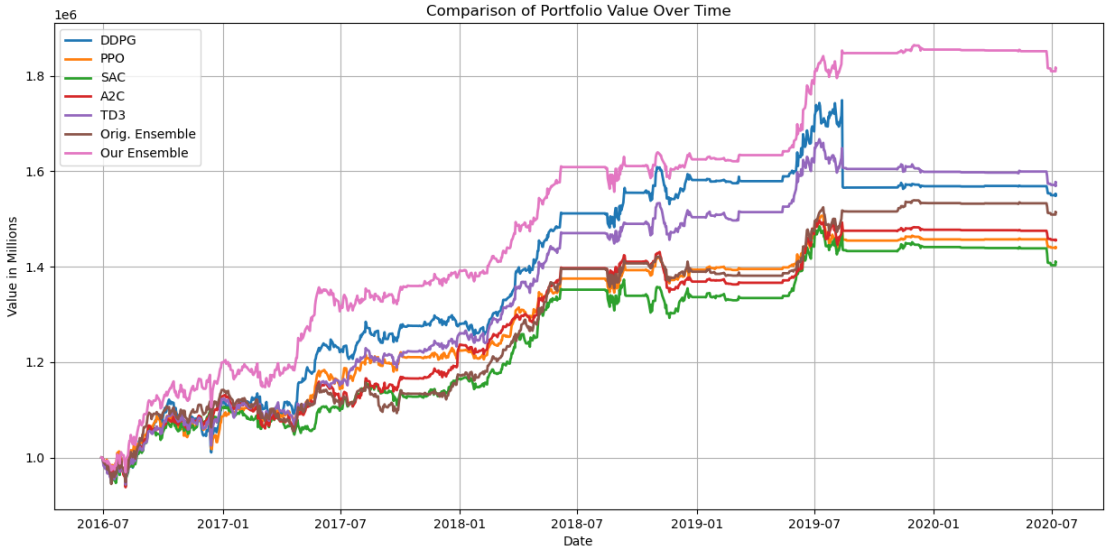

Figure 5.2 Individual and Ensemble Algorithm Portfolio Value

Figure 5.3 below shows the portfolio value percent increase of tested algorithms, including our proposed ensemble strategy, as well as the DJIA. Our ensemble outperformed all three single algorithms. The proposed ensemble (PPO/TD3/SAC) also outperformed the ensemble set of the previously proposed ensemble method (PPO/DDPG/A2C.) It can also be seen that while individual algorithms had varying degrees of volatility and returns, the ensemble methods were significantly less prone to high volatility leading to larger returns. Both ensemble methods also benefited from smoother and more robust growth.

The DJIA was used as a base benchmark to track general market performance. As seen in Figure 5.3, both the ensemble procedures and the non-optimized ensemble methods hold a substantial advantage over the DJIA across the entire evaluation period. Among the various ensemble methods, the superior performance can be claimed by the PPO/TD3/SAC ensemble over both time periods while retaining lower volatility than DJIA. Thus, these market observations prove that ensemble methods are effective in varying market conditions.

The results provide very strong evidence that ensemble reinforcement learning strategies are applicable to strong and adaptive performance in erratic financial markets.

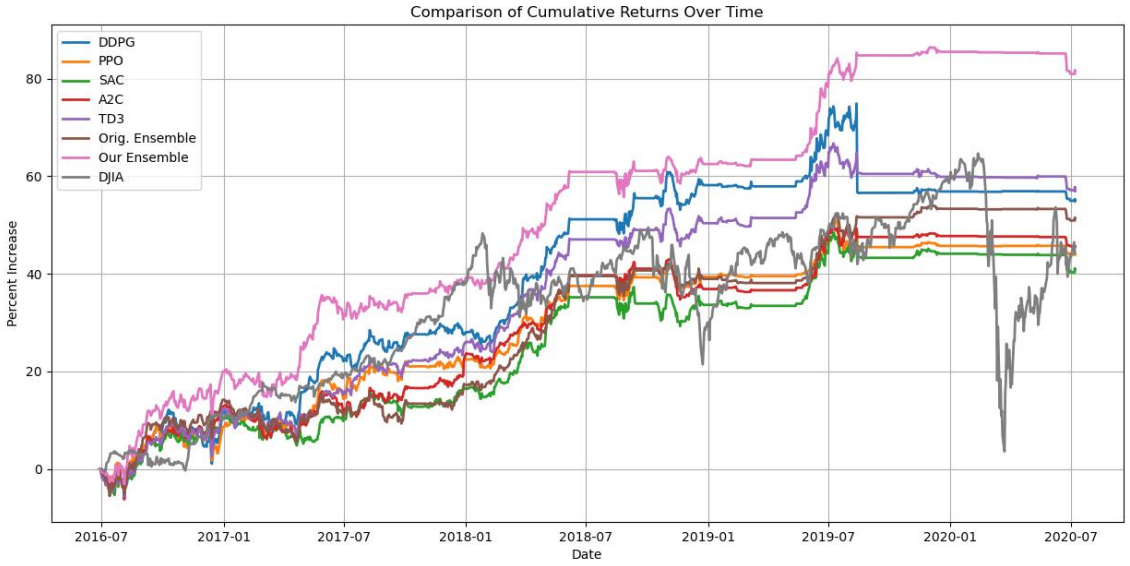

Figure 5.3 Individual and Ensemble Algorithm Percent Returns

B. Sharpe Ratio Analysis

The models best suited for each market are best outlined using each methods Sharpe Ratio over time. Figures 5.4 and 5.5 show the results of the validation Sharpe Ratios for each model. These results highlight TD3's ability to limit volatility in comparison to other policy gradient methods. However, TD3 was less adapted to more stable markets. During time frames with stable markets, algorithms that prioritize maximizing rewards, such as DDPG, are typically the most beneficial.

Our trading time frame included several high volatility time frames that our ensemble, and the algorithms used within it, are best suited for due to the focus on policy stability. This property is significantly highlighted during the COVID-19 pandemic outbreak at the end of 2019. SAC had a substantially higher Sharpe Ratio than other methods due to its bias towards safe trades and dynamic entropy calibrations that tend to limit its portfolio value. The varied strengths of the algorithms used in our ensemble combined to provide a more robust solution to the problem than other methods.

Figure 5.4 Validation Sharpe ratios over time

Start Date	End Date	A2C	DDPG	PPO	SAC	TD3
10/2/2015	1/4/2016	0.0128062	-0.02295	0.0013337	0.12585684	0.0279799
1/4/2016	4/5/2016	0.2136337	-0.000958	0.0068908	0.0176751	0.0225202
4/5/2016	7/5/2016	0.1144705	0.123542	0.0146376	0.1250828	0.0259832
7/5/2016	10/3/2016	0.204443	0.012843	0.0129475	0.17360256	0.06595832
10/3/2016	1/3/2017	0.2726743	0.6067615	0.480999	0.49550023	0.3993659
1/3/2017	3/4/2017	0.1823513	0.5199284	0.4379221	0.2433679	0.293505
3/4/2017	4/4/2017	0.2364093	0.2223541	0.3159694	0.44366628	0.2691652
4/4/2017	7/5/2017	0.2364093	0.2223541	0.3159694	0.44366628	0.2691652
7/5/2017	10/3/2017	0.0807973	0.3134547	0.093306	0.2724204	0.2422378
10/3/2017	1/3/2018	0.1706149	0.428732	0.4844326	0.4040705	0.4223549
1/3/2018	4/5/2018	0.0001807	-0.025286	0.0078488	-0.0602102	-0.0386579
4/5/2018	7/5/2018	0.0020109	0.1792716	-0.0393513	0.130864	0.0683159
7/5/2018	10/3/2018	0.1104961	0.1715393	0.1088915	0.157553	0.2772256
10/3/2018	1/4/2019	0.3254381	-0.355266	-0.4069779	-0.3584809	-0.3410678
1/4/2019	4/5/2019	-0.0572096	0.0859617	0.004349	1.1308445	0.0448776
4/5/2019	7/4/2019	0.3452939	0.2833933	0.3050585	0.2590012	0.2184567
7/4/2019	10/4/2019	0.1400472	-0.021773	-0.044979	-0.1177084	-0.0216487
10/4/2019	1/6/2020	-0.4102	-0.375187	-0.3465762	-0.1047612	-0.3225479
1/6/2020	4/6/2020	-0.3758117	-0.384956	-0.47344	-0.3791046	-0.4044592

Figure 5.5 Validation Sharpe ratios over time

VII. CONCLUSION

This section will conclude the research of this paper, discuss the results presented in the previous section, applicability to real-world investing, and future directions of research building on this work.

A. Discussion of Results

In our work we have investigated using different methods for training reinforcement learning agents. We have seen how leveraging the strengths of different approaches in an ensemble strategy can lead to a significant increase in profitability. Table 2 below shows the precise earnings and gains of each method.

TABLE 2
FINAL RETURNS ON ALL STRATEGIES

Model	*Final Balance*	*Total Gains*
Original Ensemble	$1,607,294.61	60.73%
A2C	$1,456,393.90	45.64%
DDPG	$1,552,380.74	55.24%
PPO	$1,441,059.82	44.11%
SAC	$1,409,988.08	41.00%
TD3	$1,577,075.75	57.71%
New Ensemble	$1,816,351.27	81.64%

Each individual agent performed exceptionally well under the proposed trading strategy using turbulence as a technical indicator for generating trading signals. TD3 and DDPG proved to produce the best results, with A2C, PPO, and SAC close behind. It should be noted that A2C outperformed its replacement in the ensemble strategy, SAC. This is likely due to the off-policy nature of SAC demanding more training than it received in our experimentation. Despite this shortcoming, our ensemble approach including TD3, PPO, and SAC far outperformed that of Yang et al. [10] in cumulative return. These results are consistent with the hypothesis that the introduction of more sophisticated and diverse algorithms would lead to a more robust algorithmic trading strategy.

B. Applicability

In real-world investment cases, the ensemble method has merit. The combination of reinforcement learning algorithms allows for dynamic adjustment based on market attributes, ensuring it will adapt to a given situation and demonstrate resilience in a world of uncertain trading. By stressing risk-adjusted returns via the Sharpe Ratio, the model captures some real-world defense interests relevant to institutions and hedge funds. The conservative strategy during periods of heightened turbulence further illustrates the system's practical applicability, as real-world investors frequently adopt risk-averse behaviors to preserve capital during volatile market phases. The outperformance over typical benchmarks such as the Dow Jones Industrial Average demonstrates its efficiency and potential with algorithmic trading portfolios and advisory services.

In practice, some complexities of the model must also be addressed. There may be a variety of blockages to real time implementation of a highly complex model such as this ensemble approach. Beyond time complexity issues, there may also be market liquidity issues, execution costs, or execution slippage, all of which were not explicitly present in the model. Future iterations of the model could benefit substantially should these critical market pressure points be incorporated, ultimately leading to the model's robustness and yielding far higher diagnostic accuracy in actual execution environments.

C. Future Works

There are many directions for future prospective research. The first suggestion we offer is pushing this model further to some types of financial instruments other than stocks, such as bonds, commodities, or cryptocurrencies. This could provide diversified investment strategy as well as an audit of the versatility of reinforcement learning algorithms in non-traditional investment spaces.

Furthermore, incorporating real-time macroeconomic indicators or sentiment AI from financial news/lending bodies as an enhancement could increase the precision of the model and enhance its agility for reacting to sudden market movements; the modeling of broader market trends would be an opportunity including elements like interest rates, inflationary indices, and the employment situation.

In conclusion, this research demonstrates the significant potential of ensemble reinforcement learning methods for algorithmic trading, offering superior performance compared to traditional approaches. By combining the strengths of diverse agents and incorporating market turbulence signals, our approach addresses key challenges in financial markets while providing a foundation for more sophisticated trading systems. As both reinforcement learning techniques and market environments continue to evolve, we believe this work represents an important step toward more intelligent, adaptive, and profitable automated trading strategies.

REFERENCES

[1] G. Nuti, M. Mirghaemi, P. Treleaven, and C. Yingsaeree, "Algorithmic Trading," *Computer*, vol. 44, no. 11, pp. 61-69, Nov. 2011, doi: 10.1109/MC.2011.31.

[2] A. P. Chaboud, B. Chiquoine, E. Hjalmarsson, and C. Vega, "Rise of the Machines: Algorithmic Trading in the Foreign Exchange Market," *J. Finance*, vol. 69, no. 5, pp. 2045-2084, Oct. 2014, doi: 10.1111/jofi.12186.

[3] E. Boehmer, K. Fong, and J. Wu, "Algorithmic Trading and Market Quality: International Evidence," *J. Financial Quantitative Analysis*, vol. 56, no. 8, pp. 2659-2688, 2021, doi: 10.1017/S0022109020000782.

[4] Y. Li, W. Zheng, and Z. Zheng, "Deep Robust Reinforcement Learning for Practical Algorithmic Trading," *IEEE Access*, vol. 7, pp. 108014-108022, 2019, doi: 10.1109/ACCESS.2019.2932789.

[5] M. Massahi and M. Mahootchi, "A Deep Q-Learning Based Algorithmic Trading System for Commodity Futures Markets," *Expert Syst. Appl.*, vol. 237, p. 121711, Mar. 2024, doi: 10.1016/j.eswa.2023.121711.

[6] A. Shavandi and M. Khedmati, "A multi-agent deep reinforcement learning framework for algorithmic trading in financial markets," *Expert Syst. Appl.*, vol. 208, p. 118124, Nov. 2022, doi: 10.1016/j.eswa.2022.118124.

[7] M. S. Hossain and M. A. Rahman, "A deep reinforcement learning-based approach for portfolio management with risk-sensitive reward function," *Expert Syst. Appl.*, vol. 213, p. 121245, Mar. 2023, doi: 10.1016/j.eswa.2023.121245.

[8] J.-H. Park, J.-H. Kim, and J.-H. Huh, "Deep Reinforcement Learning Robots for Algorithmic Trading: Considering Stock Market Conditions and U.S. Interest Rates," *IEEE Access*, vol. 12, pp. 20705-20725, 2024, doi: 10.1109/ACCESS.2024.3361035.

[9] X. Han and J. Wang, "R2-SAC: A Relaxation-and-Refinement SAC Agent for Stock Portfolio Trading," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Hyderabad, India, 2025, pp. 1-5, doi: 10.1109/ICASSP49660.2025.10888366.

[10] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading," in *Proc.*

1st ACM Int. Conf. AI Finance, Oct. 2020, doi: 10.1145/3383455.3422540.

[11] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv:1707.06347, Jul. 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[12] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," arXiv:1802.09477, 2018. [Online]. Available: https://arxiv.org/pdf/1802.09477

[13] "Twin Delayed DDPG — Spinning Up documentation," OpenAI. [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/td3.html

[14] Wharton Research Data Service, "Standard & Poor's Computat," 2015.

[15] S. Fujimoto, H. Van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," arXiv:1802.09477, 2018. [Online]. Available: https://arxiv.org/pdf/1802.09477

[16] V. Mnih et al., "Playing Atari with Deep Reinforcement Learning," arXiv:1312.5602, Dec. 2013. [Online]. Available: https://arxiv.org/pdf/1312.5602

[17] Y. Li, "Deep Reinforcement Learning," arXiv:1810.06339, 2018. [Online]. Available: https://arxiv.org/pdf/1810.06339

[18] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2020. [Online]. Available: http://www.incompleteideas.net/book/RLbook2020.pdf

[19] E. O. Arwa and K. A. Folly, "Reinforcement Learning Techniques for Optimal Power Control in Grid-Connected Microgrids: A Comprehensive Review," *IEEE Access*, vol. 8, pp. 208992-209007, 2020, doi: 10.1109/access.2020.3038735.