Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy

Pham Thi Ngoc Bich - 23520148
Nguyen Thi Minh Phu - 23521186
Phan Thuy Phuong - 23521248
Trinh Tran Tran - 2351624

Lecturer: Luong Ngoc Hoang

University of Information Technology
June 10, 2025

Table of Content

Table of Content

Introduction

Automated trading strategies play an important role in optimizing capital allocation in volatile markets.

We choose an **ensemble deep reinforcement learning (DRL)** framework that **combines PPO and A2C**, modeled as a Markov Decision Process to integrate real-world constraints and maximize Sharpe ratio based returns. This approach leverages the synergy of both algorithms to **enhance market adaptability**.

Table of Content

We model stock trading as a Markov Decision Process (MDP) to learn an optimal trading strategy in a dynamic market.

MDP Tuple

An MDP is defined by:

$$(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$$

Problem Setup

MDP model for stock trading

- **State** $s_t = [p_t, h_t, b_t]$:
  - p_t: stock prices ($\mathbb{R}_+^D$)
  - h_t: shares held ($\mathbb{Z}_+^D$)
  - b_t: cash balance ($\mathbb{R}_+$)
- **Action** a_t: vector of buy/sell/hold decisions for each stock
- **Transition** $\mathcal{P}(s_{t+1}|s_t, a_t)$: evolves based on market dynamics
- **Reward**:

$$r_t = (b_{t+1} + p_{t+1}^\top h_{t+1}) - (b_t + p_t^\top h_t) - c_t$$

- **Discount factor** $\gamma \in (0, 1)$

The following assumption and constraints reflect concerns for practice:

1. **Market Liquidity:** Assumes orders are executed at the closing price without impacting the market, simulating a realistic trading environment.
2. **Nonnegative Balance:** The portfolio's cash balance must remain non-negative. The balance after a trade is constrained by:

$$b_{(t+1)} = b_t + (p_t^S)^\top k_t^S - (p_t^B)^\top k_t^B \geq 0$$

   Where p_t^S and k_t^S are the prices and shares *sold*, p_t^B and k_t^B are the prices and shares *bought*.

Problem Setup

Incorporating Stock Constraints

3. **Transaction Costs:** Each trade incurs a cost of 0.1% of the trade value:

$$c_t = p^\top k_t \times 0.1\%$$

4. **Risk-Aversion for Market Crash:** A turbulence index measures extreme market conditions:

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

Where y_t is the current stock returns, μ is the average historical returns, Σ^{-1} is the covariance matrix. If the index exceeds a threshold, the agent sells all shares and halts buying until the market stabilizes.

The reward function is designed to reflect the change in portfolio value after each trading action, while also accounting for transaction costs and market risk. The general formula is:

$$r(s_t, a_t, s_{t+1}) = \underbrace{\left[b_{t+1} + \mathbf{p}_{t+1}^T \mathbf{h}_{t+1}\right]}_{\text{Portfolio value at } t+1} - \underbrace{\left[b_t + \mathbf{p}_t^T \mathbf{h}_t\right]}_{\text{Portfolio value at } t} - \underbrace{c_t}_{\text{Transaction cost}}$$

- **Reward decomposition:** $r = r_H - r_S + r_B - c_t$

$$r_H = (\mathbf{p}_{t+1}^H - \mathbf{p}_t^H)^T \mathbf{h}_t^H \tag{1}$$

$$r_S = (\mathbf{p}_{t+1}^S - \mathbf{p}_t^S)^T \mathbf{h}_t^S \tag{2}$$

$$r_B = (\mathbf{p}_{t+1}^B - \mathbf{p}_t^B)^T \mathbf{h}_t^B \tag{3}$$

- When $turbulence_t > threshold$, the system **sells all stocks** to **minimize losses**:

$$(2) => r_s = (\mathbf{p}_{t+1} - \mathbf{p}_t)^T \mathbf{k}_t$$

Where:

- $\mathbf{k}_t$: number of shares held before selling.
- $\mathbf{p}_t$: the price of each stock at the time t.

Problem Setup

Returning Maximization as Trading Goal

In real-world, optimizing for short-term profits is not sufficient. The agent needs to consider future rewards when making present decisions.

Bellman equation:

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r(s_t, a_t, s_{t+1}) + \gamma \mathbb{E}_{a_{t+1} \sim \pi(s_{t+1})}\left[Q_\pi(s_{t+1}, a_{t+1})\right]\right]$$

- **Goal**: Maximize the accumulated total reward by:
  - **Buying/holding** stocks expected to increase in price.
  - **Selling** stocks expected to decrease in price.
  - **Automatically adjusting** to market volatility using the Turbulence Index.

Table of Content

Stock Market Environment

We build a realistic trading environment for deep reinforcement learning agents to interact with historical stock data.

The state vector s_t is 175-dimensional, composed of:

- b_t: Current cash balance
- $p_t \in \mathbb{R}_+^{29}$: Adjusted close prices for 29 stocks
- $h_t \in \mathbb{Z}_+^{29}$: Number of shares held
- $M_t \in \mathbb{R}^{29}$: MACD (Moving Average Convergence Divergence)
- $R_t \in \mathbb{R}_+^{29}$: RSI (Relative Strength Index)
- $C_t \in \mathbb{R}_+^{29}$: CCI (Commodity Channel Index)
- $X_t \in \mathbb{R}^{29}$: ADX (Average Directional Index)

Total: $1 + 29 + 29 + 29 + 29 + 29 + 29 = 175$ dimensions.

Stock Market Environment

Action Space

- For each stock, the action is from $\{-k, \ldots, -1, 0, 1, \ldots, k\}$:
  - $-k$: sell k shares
  - 0: hold
  - $+k$: buy k shares
- The action space is normalized to $[-1, 1]$ to suit Gaussian policy distributions.
- For 29 stocks, the total action space is $(2k + 1)^{29}$ (but continuous control is used instead).

Note: k is limited by available balance and max shares per trade.

Table of Content

Methodology

Policy Gradient

- **Goal:** Learn a parameterized policy $\pi_\theta(a|s)$ that maximizes the expected return:

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

- **Key Idea:** Instead of learning a value function, directly optimize the policy parameters.
- **Policy Gradient Theorem:**

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot R_t\right]$$

Actor-Critic is a hybrid algorithm combining policy-based and value-based methods.

- **Actor:** Policy $\pi(a|s)$, selects actions based on the current state.
- **Critic:** Estimates value function $V(s)$ or action-value function $Q(s, a)$, evaluating the actions taken by the actor.

Step 1: Initialization

- Initialize hyperparameters:
  - Learning rates $\alpha_{\text{actor}}, \alpha_{\text{critic}}$
  - Discount factor γ
- Initialize networks:
  - Actor $\pi_\theta(a|s)$
  - Critic $V_\omega(s)$

Step 2: Compute Temporal Difference (TD) Error

- After Actor selects an action and receives reward:
  - TD Target:
  $$y_t = r_t + \gamma V_\omega(s_{t+1})$$
  - TD Error:
  $$\delta_t = y_t - V_\omega(s_t)$$
- TD Error measures the difference between current estimated value and expected future return.

Step 3: Update Critic Network

- Loss function:

$$\mathcal{L}_{critic} = \delta_t^2 = (r_t + \gamma V(s_{t+1}) - V(s_t))^2$$

- Meaning:
  - r_t: actual received reward.
  - $V(s_{t+1})$: predicted value of next state.
  - $V(s_t)$: predicted value of current state.
- Update parameters ω via gradient descent.

Step 4: Update Actor Network

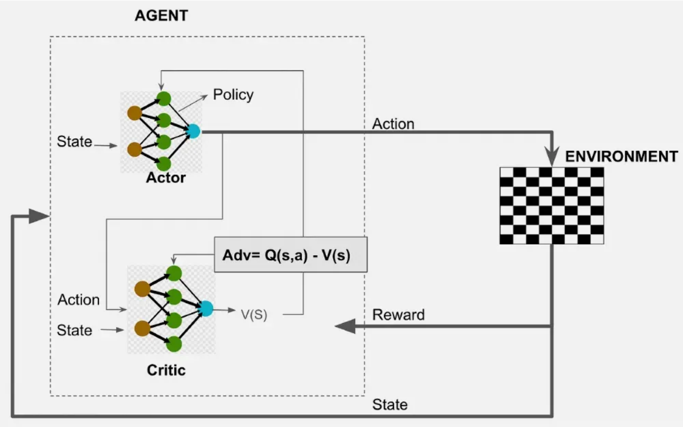
- Use policy gradient:

$$\nabla_\theta J(\theta) \approx \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot \delta_t\right]$$

- Interpretation:
  - $\delta_t > 0$: action was better than expected $\rightarrow$ increase probability.
  - $\delta_t < 0$: action was worse $\rightarrow$ decrease probability.
- Use gradient ascent to update θ.

Step 5: Repeat

- Repeat steps 2 $\rightarrow$ 4 until convergence or maximum episodes reached.



A2C, short for **Advantage Actor-Critic**, is an enhanced version of Actor-Critic. It speeds up and stabilizes training by using multiple agents to collect data simultaneously, and updates the policy using the *advantage function* for more effective learning.

Key difference: Uses Advantage instead of direct TD-error.

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$$

or

$$A(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) - V(s_t)$$

The objective function for A2C:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot A(s_t, a_t)\right]$$

Parallelization:
A2C uses n agents interacting with n parallel environments. Each agent shares a global Actor and Critic model and is responsible for:

- Collecting trajectory data over T steps using the shared policy.
- Sending collected transitions (s_t, a_t, r_t, s_{t+1}) to the central learner.

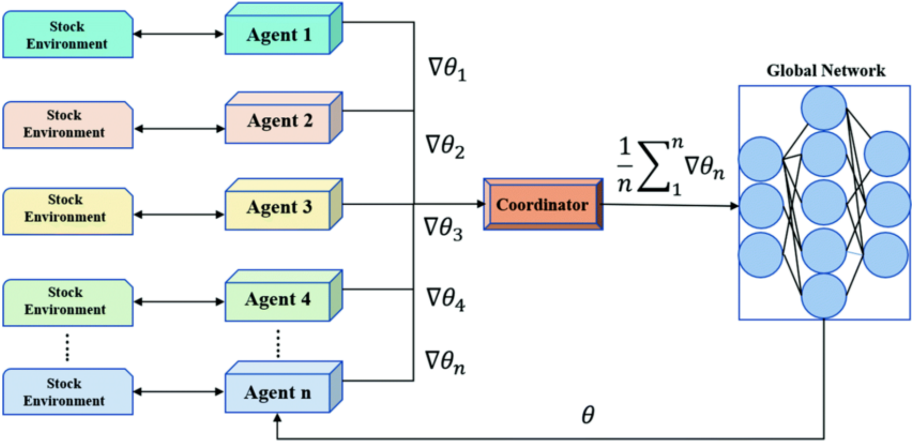
Synchronous update:

After every T steps per agent:

- All agents send their collected data to the central learner.
- The central learner uses the data to compute Advantage and loss.
- It then calculates gradients and updates the global Actor and Critic parameters accordingly.
- The updated Actor and Critic models are broadcast back to all agents.

Methodology

A2C Diagram



Why A2C is suitable for stock trading:

- Enables parallel data collection from the market to accelerate learning.
- Advantage estimation improves action evaluation in noisy environments.

Proximal Policy Optimization (PPO) is an improved policy-gradient algorithm used in our ensemble method to control policy updates. PPO ensures that the new policy does not deviate excessively from the old one by introducing a clipping mechanism to the objective function.

The ratio between the new and old policies is given by:

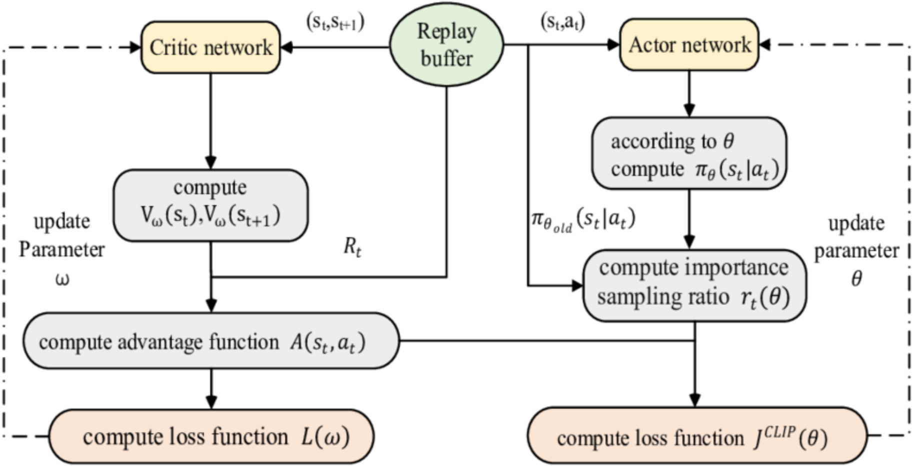
$$\rho_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

Clipped Surrogate Objective:

$$J^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min\left(\rho_t(\theta)A(s_t, a_t),\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)A(s_t, a_t)\right)\right]$$

where

- $A(s_t, a_t)$ is the estimated advantage function.
- When $\rho_t(\theta)$ lies within $[1-\epsilon, 1+\epsilon]$, the objective behaves like the standard policy gradient.
- If $\rho_t(\theta)$ is outside this range, it is clipped to $1-\epsilon$ or $1+\epsilon$ to limit updates and stabilize training.



Critic network
(s_t,s_{t+1})
Replay buffer
(s_t,a_t)
Actor network
according to θ compute $\pi_\theta(s_t|a_t)$
compute $V_\omega(s_t),V_\omega(s_{t+1})$
update Parameter ω
R_t
$\pi_{\theta_{old}}(s_t|a_t)$
update parameter θ
compute importance sampling ratio $r_t(\theta)$
compute advantage function $A(s_t,a_t)$
compute loss function $L(\omega)$
compute loss function $J^{CLIP}(\theta)$

Why PPO is suitable for stock trading:

- PPO ensures stable performance even in highly volatile market conditions.
- The clipping mechanism limits sudden policy changes, reducing the risk of erratic buy/sell actions.
- PPO is easy to implement and trains efficiently, making it well-suited for real-time trading systems.

Ensemble Strategy

To build a robust trading system, we use an ensemble approach that dynamically selects the best-performing agent (PPO or A2C) based on the Sharpe ratio.

Sharpe Ratio:

$$\text{Sharpe ratio} = \frac{\hat{r}_p - r_f}{\sigma_p}$$

where:

- $\hat{r}_p$ is the expected portfolio return
- r_f is the risk-free rate
- σ_p is the standard deviation of returns

Ensemble Procedure:

- **Step 1: Training** — Every 3 months, retrain PPO and A2C using a growing window of historical data.
- **Step 2: Validation** — Use the next 3-month window to evaluate both agents based on their Sharpe ratio. The agent with the highest Sharpe ratio is selected.
- **Step 3: Trading** — The selected agent is deployed to trade for the next 3 months.

Retraining Loop: After each trading phase, repeat from Step 1 with updated market data. This **pretraining** mechanism enables the model to adapt to evolving market conditions.

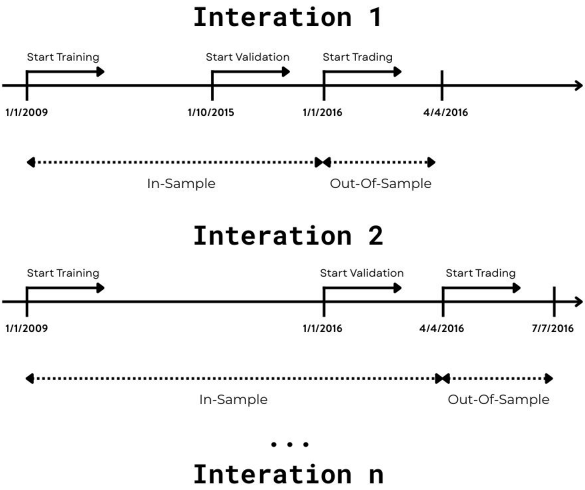

Interation 1
Start Training
Start Validation
Start Trading
1/1/2009
1/10/2015
1/1/2016
4/4/2016
In-Sample
Out-Of-Sample
Interation 2
Start Training
Start Validation
Start Trading
1/1/2009
1/1/2016
4/4/2016
7/7/2016
In-Sample
Out-Of-Sample
...
Interation n

Table of Content

U.S. Market: Data Collection & Preprocessing

- **Data sources:** Yahoo Finance (YahooDownloader)
- **Universe:** Dow 30
- **Feature engineering:**
  - MACD
  - RSI (period = 30)
  - CCI (period = 30)
  - DX (period = 30)
  - Turbulence index
- **Data preprocessing:**

Price	date	close	high	low	open	volume	tic	day
0	2009-01-02	2.727417	2.736134	2.559415	2.581054	746015200	AAPL	4
1	2009-01-02	14.929296	15.076041	14.211022	14.342319	10955700	AXP	4
2	2009-01-02	33.941093	34.173619	32.088396	32.103398	7010200	BA	4
3	2009-01-02	30.344679	30.389960	28.921564	29.050939	7117200	CAT	4
4	2009-01-02	11.166078	11.192413	10.698630	10.803971	40980600	CSCO	4

	date	close	high	low	open	volume	tic	day	macd	rsi_30	cci_30	dx_30	turbulence
0	2009-01-02	2.727417	2.736134	2.559415	2.581054	746015200	AAPL	4	0.0	100.0	66.666667	100.0	0.0
1	2009-01-02	14.929296	15.076040	14.211021	14.342318	10955700	AXP	4	0.0	100.0	66.666667	100.0	0.0
2	2009-01-02	33.941093	34.173619	32.088396	32.103398	7010200	BA	4	0.0	100.0	66.666667	100.0	0.0
3	2009-01-02	30.344683	30.389963	28.921568	29.050942	7117200	CAT	4	0.0	100.0	66.666667	100.0	0.0
4	2009-01-02	11.166075	11.192410	10.698628	10.803968	40980600	CSCO	4	0.0	100.0	66.666667	100.0	0.0

- **Framework:** OpenAI Gym
  - State space: features, cash balance, number of shares
  - Action space: $\{$buy, sell, hold$\}$
  - Reward: change in portfolio value
- Transaction costs & market friction modeled
- Clear train–test split of historical data

- **Time windows:**
  - Train: 2009-01-01 – 2015-09-30
  - Test: 2015-09-30 – 2020-05-08
- **Algorithms:** PPO, A2C, Ensemble
- **Ensemble strategy:**
  - Bull market: weight PPO more heavily
  - Bear market: weight A2C more heavily
  - Sideways market: PPO : A2C = 50 : 50

- **Metrics:**
  - Cumulative Return
  - Annual Volatility
  - Max Drawdown
  - Sharpe Ratio
- **Benchmarks:**
  - DJIA (30 blue-chip stocks)
  - Min-Variance portfolio

- **Data sources:** vnstock package
- **Universe:** VN-Index
- **Feature engineering:** MACD, RSI(30), CCI(30), DX(30), Turbulence
- **Data preprocessing:**

	time	open	high	low	close	volume	tic
0	2014-01-02	2.29	2.29	2.28	2.29	146982	ACB
1	2014-01-03	2.29	2.31	2.28	2.31	98656	ACB
2	2014-01-06	2.28	2.31	2.28	2.29	190009	ACB
3	2014-01-07	2.31	2.35	2.29	2.32	152905	ACB
4	2014-01-08	2.32	2.32	2.29	2.31	58852	ACB

	date	close	high	low	open	volume	tic	macd	rsi_30	cci_30	dx_30	turbulence
0	2014-01-02	29.05	29.36	28.98	29.05	87000	BVH	0.0	100.0	-66.666667	100.0	0.0
1	2014-01-02	9.01	9.07	8.96	9.07	156420	CTG	0.0	100.0	-66.666667	100.0	0.0
2	2014-01-02	6.45	6.50	6.42	6.48	207470	FPT	0.0	100.0	-66.666667	100.0	0.0
3	2014-01-02	31.69	31.69	31.21	31.69	229950	GAS	0.0	100.0	-66.666667	100.0	0.0
4	2014-01-02	2.71	2.74	2.70	2.74	276250	HPG	0.0	100.0	-66.666667	100.0	0.0

- Reuse OpenAI Gym framework
  - Adjust trading hours, transaction costs, and local regulations
- Ensure fair train–test split

- **Time windows:**
  - Train: 2014-01-01 – 2020-09-30
  - Test: 2020-09-30 – 2025-06-01
- **Algorithms:** PPO, A2C, Ensemble

- **Metrics:** Cumulative Return, Sharpe Ratio
- **Benchmarks:** VN-Index (30 highest), Minimum-variance strategy

Table of Content

US Model Performance Evaluation

Sharpe Ratios Overtime

Iter	Val Start	Val End	Model Used	A2C Sharpe	PPO Sharpe
0	2015-10-01	2015-12-31	A2C	0.025	-0.061
1	2015-12-31	2016-04-04	A2C	0.053	-0.002
2	2016-04-04	2016-07-01	PPO	-0.006	-0.001
3	2016-07-01	2016-09-30	PPO	-0.107	-0.098
4	2016-09-30	2016-12-30	A2C	0.557	0.333
5	2016-12-30	2017-04-03	A2C	0.305	0.084
6	2017-04-03	2017-07-03	PPO	-0.125	0.027
7	2017-07-03	2017-10-02	A2C	0.340	0.073
8	2017-10-02	2018-01-02	A2C	0.359	0.316
9	2018-01-02	2018-04-04	A2C	-0.056	-0.210
10	2018-04-04	2018-07-03	A2C	0.191	0.012
11	2018-07-03	2018-10-02	PPO	0.337	0.394
12	2018-10-02	2019-01-03	PPO	-0.205	-0.205
13	2019-01-03	2019-04-04	A2C	0.225	0.160
14	2019-04-04	2019-07-05	A2C	0.001	-0.095
15	2019-07-05	2019-10-03	A2C	-0.244	-0.407
16	2019-10-03	2020-01-03	A2C	0.821	0.261

US Model Performance Evaluation

Comparison with Benchmark Results

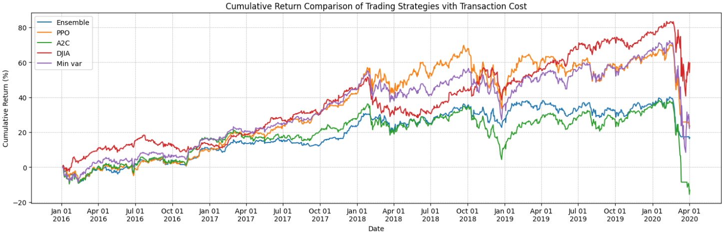


Figure: Performance comparison of the proposed model against benchmark models.

US Model Performance Evaluation

Performance during the stock market crash in the first quarter of 2020.

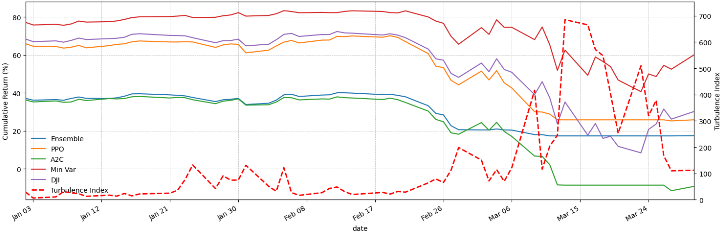


Figure: Performance during the stock market crash in the first quarter of 2020.

VN Model Performance Evaluation

Comparison with Benchmark Results

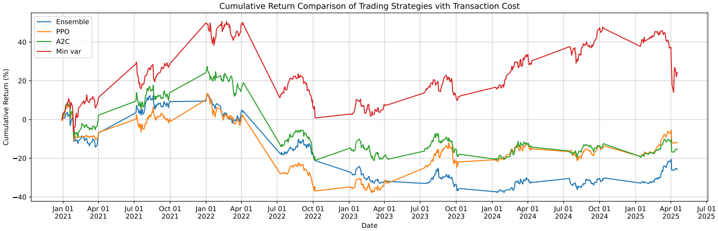


Figure: Performance comparison of the proposed model against benchmark models.

Table of Content

Conclusion

The ensemble strategy incurs **significant training time and computational cost**, yet its performance improvement over A2C and PPO is **marginal**. The gains are small and inconsistent across different periods, while the agents are highly sensitive to hyperparameter tuning. Therefore, the **trade-off is not compelling for practical applications**.