A Multi-Model Adaptive Q-Learning Framework for Robust Portfolio Management in Stochastic Markets

Sharmin Sultana[1], Md Borhan Uddin[2], Mst Masuma Akter Semi[3],
Shahanaj Akther[4], Urmi Chakraborty[5], Khandakar Rabbi Ahmed[6]*
International American University, Los Angeles, United States[1, 2, 4, 5]
Westcliff University, USA[3]
Miyan Research Institute, International University of Business Agriculture and Technology, Bangladesh[6]

***Abstract*—This study presents TAQLA, a new Tabular Adaptive Q-Learning Agent for portfolio management in stochastic financial markets. TAQLA rests on a multi-model reinforcement learning (RL) architecture that integrates parameter-adaptive Q-Learning mechanisms into softmax-based exploration to reconcile short-term profit maximization with long-term capital preservation. The method is contrasted with vanilla Q-Learning, SARSA, and a random trading policy using simulated equity market data. Empirical analysis shows that TAQLA performs better on profitability, risk-adjusted performance, and drawdown minimization, with a last portfolio value of $1687.45 (+68.74% of initial capital), a Sharpe ratio of 1.41, and a maximum drawdown of just 12.8%. Q-Learning and SARSA, on the other hand, yield Sharpe ratios below 1.0 and drawdowns exceeding 18%. Parameter sensitivity analysis across β (softmax temperature), α (learning rate), and γ (discount factor) reveals that aggressive exploration (β ≈ 1.0–1.5) and reasonable discounting (γ ≈ 0.4–0.6) generate the most aggressive and robust outcomes. Such outcomes place TAQLA as a robust RL-based adaptive portfolio control method under uncertainty, with improved capital appreciation and robustness to adverse market conditions.**

Keywords*—Reinforcement learning; Q-Learning; tabular reinforcement learning; portfolio management; dynamic asset allocation

I. Introduction

The vigorous growth in financial markets, particularly equities and cryptocurrencies, has introduced unprecedented complexity to portfolio management. Traditional asset allocation techniques-such as mean-variance optimization or fixed-weight models-tend to assume static market conditions and fail to account for nonlinear, dynamic behavior in assets. These deficiencies are even more critical in settings with dynamic asset counts, volatility spikes, and changing correlations. To better address such challenges, scholars and professionals increasingly turn to Deep Reinforcement Learning (DRL), which enables agents to learn an optimal decision policy through interaction with an ever-changing market environment. Recent research showed that portfolio optimization was achievable using DRL-based models. For instance, [6] presented a neural model that dynamically adapts to variable asset markets, particularly in the cryptocurrency space. This highlights the importance of building scalable models that generalize across asset sets, while accounting for constraints such as transaction costs. Similarly, [1], [2] employed actor-critic algorithms such as PPO and A2C to build portfolio strategies capable of achieving maximum long-term returns, with superior performance compared to traditional and rule-based methods.

Moreover, [3] suggested a reward-shaping mechanism based on the Sharpe ratio and cumulative returns to facilitate risk-sensitive learning in highly stochastic environments. Motivated by such advancements, we focus from the start on tabular RL methods. While deep reinforcement learning (DRL) methods such as Proximal Policy Optimization (PPO) and Deep Deterministic Policy Gradient (DDPG) have shown promise, they often require extensive training datasets, high computational resources, and careful hyperparameter tuning. In contrast, tabular RL approaches such as Q-Learning and SARSA remain attractive for smaller datasets or environments with well-defined state-action spaces, offering interpretability and fast convergence. The meta-agent module dynamically evaluates recent performance metrics (e.g., rolling Sharpe ratios and cumulative returns) and selects the most promising agent at each timestep to inform the final portfolio decision.

This approach allows the system to learn in various market regimes-i.e., uptrends, downtrends, and high-volatility sideways movements-and apply the best-performing model to each situation. We deployed a custom OpenAI Gym-compatible trading environment to model realistic market settings and operate a multi-asset portfolio. The environment supports the required features of transaction cost modeling, slippage, and portfolio rebalancing restrictions. Historical daily returns for the SP 500 and NASDAQ were downloaded from Kaggle and preprocessed for ticker alignment, missing value handling, and feature extraction. Technical analysis features such as the Relative Strength Index (RSI), Exponential Moving Average (EMA), Average True Range (ATR), and Simple Moving Average (SMA) acted as state features with log returns and portfolio weight vectors-a setup by the data treatment procedure [4]. Both agents learn through continuous interaction with the environment, using actor-critic architectures and stochastic gradient descent to improve their policies.

The PPO-LSTM agent particularly excels at discovering long-term dependencies and sequential relationships in financial time-series data, whereas DDPG is used for precise control of continuous asset weight distributions. The system's reward function incorporates a combination of appreciation in portfolio value, Sharpe ratio maximization, and a penalty for transaction costs-similar to the reward engineering depicted. The

*Corresponding author.

framework is contrasted with single DRL baselines and traditional benchmarks. Experimental outcomes show that the ensemble meta-agent consistently outperforms its single-PPO, DDPG, and LSTM counterparts in cumulative returns, risk-adjusted performance, and drawdowns.

These results are in line with those that highlight the virtues of agent diversity and model flexibility in financial environments. In short, the present work contributes a robust, flexible, and modular multi-agent DRL portfolio optimization framework. It bridges the theory-to-practice gap in DRL architectures and real-world implementation by integrating transaction-aware decision-making, temporal abstraction, and adaptive agent selection. Future extensions would include studying hierarchical coordination among agents, generalizing across markets, and extending to real-time trading APIs to enrich innovative financial decision-support systems further. Our main contributions are the following:

- We present Trend-Aware Reward Shaping, introducing a multiplicative trend coefficient that amplifies or dampens rewards depending on the market's recent momentum.
- State Augmentation with Market Context: encoding price trends, position inventory, and liquidity into the state space to improve decision relevance.
- Comparative Evaluation: benchmarking TAQLA against vanilla Q-Learning, SARSA, and a buy-and-hold baseline on historical NASDAQ data, using metrics including average reward, Sharpe ratio, and maximum drawdown.
- Ablation Analysis: evaluating the effect of the trend coefficient by comparing TAQLA's performance with and without it.

Our results show that TAQLA achieves higher risk-adjusted returns, greater stability, and faster convergence than baseline tabular RL agents. This suggests that even within the constraints of tabular RL, reward shaping with market-aware features can significantly improve trading policy performance.

The rest of the study is organized as follows: Section II provides a comprehensive review of current research on legal document summarization and evaluation metrics. Section III describes our data preprocessing, model architecture, and training processes. Section IV reports quantitative metrics and qualitative results. Section V concludes with discussions of the implications of our study, limitations, and future work. Section VI presents the disclosure and conflict of interest.

II. Literature Review

Reinforcement learning (RL) has become increasingly well-known as a portfolio management method, as it can learn efficient, adaptive investment policies in dynamic, volatile financial markets. Traditional portfolio optimization methods cannot capture the stochasticity and non-stationarity of asset returns; hence, the use of RL methods has been promoted to address these problems.

Vodnala et al. [5] proposed a multi-model RL framework that integrates Advantage Actor-Critic (A3C), Deep Deterministic Policy Gradient (DDPG), and Proximal Policy Optimization (PPO). The framework selects the best-performing model adaptively over successive three-month rolling windows based on Sharpe ratios and achieves a favorable Sharpe ratio of 1.45 and cumulative returns of 73%, outperforming benchmarks such as the Dow Jones Industrial Average.

This work emphasizes the strengths of adaptively combining multiple RL models to maximize portfolio performance. Chen et al. [6] introduced a multi-model approach that integrated Linear Regression, Long Short-Term Memory (LSTM), and ARIMA for forecasting stock prices and providing investment recommendations. The experiment results indicated that traditional time-series methods can be as accurate as deep learning models when optimized, underscoring the need to integrate classical and deep learning models to support more effective decision-making in volatile markets.

Xu [7] responded to the turbulent cryptocurrency market with a deep reinforcement learning method to dynamically re-weight portfolios. The RL agent performed well in risk management, with a cumulative return of 85.12%, an annualized volatility of 45.76%, and a maximum drawdown of -22.34%. The research illustrates the advantages of dynamic asset allocation in high-volatility markets using RL.

Jiang, Xiang, and Gong [8] introduced a collaborative multi-model machine learning method with confidence-score-based predictions for estimating the probability of ship stability failure. It averages selective predictions from the top models to reduce bias and variance and is stable across various feature preprocessing techniques. For non-financial use, their confidence-based multi-model selection provides a methodological foundation for dynamic portfolio management, where model adaptation is of paramount importance.

Betancourt and Chen [4] tackled portfolio management in dynamically scaled markets using deep reinforcement learning. Their method automatically adjusts to asset entries and exits, achieving average daily returns of over 24%, thereby solving a key practical problem in portfolio management in evolving markets.

Jiang et al. [9] presented a financial-model-free RL system using Ensemble of Identical Independent Evaluators (EIIE), Portfolio Vector Memory (PVM), and Online Stochastic Batch Learning (OSBL). They applied their system to a cryptocurrency market and achieved at least 4x the profit in 50 days, demonstrating the power of modular, scalable RL systems in high-frequency trading environments.

Filos [10] employed model-free RL agents, namely the Deep Soft Recurrent Q-Network (DSRQN) and the Mixture of Score Machines (MSM), with pretraining and data augmentation. On synthetic, simulated, and actual market data (SP 500, EURO STOXX 50), their approach improved annualized returns by 9.2% and Sharpe ratios by 13.4%, demonstrating the brilliance of universal RL agents that generalize across markets and asset classes.

Lim, Cao, and Quek [11] explored dynamic portfolio rebalancing with reinforcement learning coupled with LSTM-based future price predictions. Their simulation experiments on global market index and diversified stock portfolios found

significant improvements in returns (27.9% to 93.4%) over static full rebalancing strategies. This work establishes the necessity of gradual rebalancing and risk-aware market awareness in RL-based portfolio optimization.

Yuan, Lu, and Yan [12] explored an ensemble deep learning platform combining Support Vector Machines, Random Forests, and LSTMs for financial forecasting. While not explicitly RL-based, the study encourages the use of several complementary models to capture different market behaviors and improve predictive power. Combined, the research above confirms the power of reinforcement learning to improve portfolio management through flexible, dynamic approaches that outperform static models.

Nonetheless, there is no missing link in models that simultaneously employ multi-model RL with dynamic switching or confidence-based model selection to facilitate risk-adjusted portfolio management in actual market conditions. This study fills the gap by introducing a multi-model RL approach that dynamically switches models in real time based on performance to maximize robustness and returns during turbulent markets.

III. Methodology

A. Dataset Description

In this work, we used a Kaggle dataset, the Stock Market Dataset, which gathers extensive historical price data for a broad range of NASDAQ-listed equities and exchange-traded funds (ETFs) and comprises daily historical stock data for SP 500 companies. Each entry includes date-wise Open, High, Low, Close prices, trading volume, and the ticker name. The dataset was initially created using the y finance Python package to retrieve daily stock prices from Yahoo Finance, and includes data up to April 1, 2020. Fig. 1 demonstrates the daily market indicators over time. Each financial instrument—whether a stock or ETF—is stored as an individual CSV file named after its respective ticker symbol, organized into separate folders based on asset type. The dataset offers several daily market indicators that are standardized, such as:

- Date: The trading session date on the calendar.
- Open: The starting price on that particular day.
- High: The day's most considerable amount traded.
- Low: The day's lowest price of trade.
- Close: The closing price with stock split adjustments.
- Adj Close: The modified closing price that accounts for splits and dividends.
- Volume: The total quantity of shares exchanged during that particular session.

Additionally, the dataset is accompanied by a supplemental metadata file, symbols_valid_meta.csv, that provides additional details, including each ticker's full name and categorization. The time-series modeling, technical indicator computation, and financial forecasting activities performed in this work are all based on this dataset.

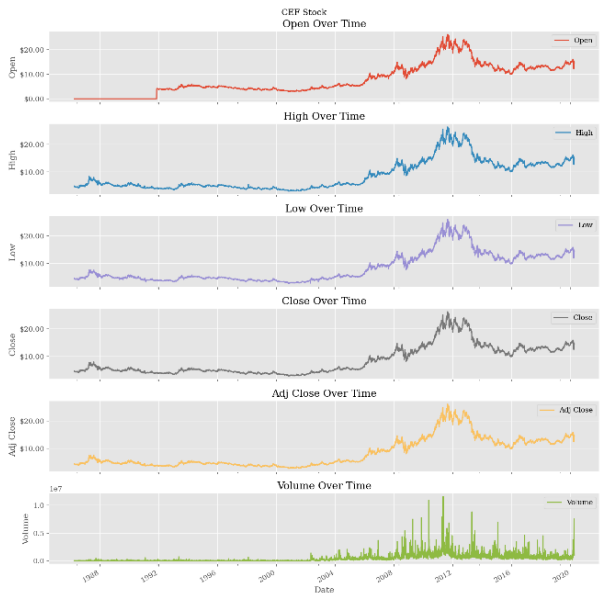

Fig. 1. Several daily market indicators over time.

B. Data Preprocessing

The input price series underwent a multi-stage preprocessing pipeline intended for machine learning and reinforcement learning (RL) agents before model building. Fig. 2 illustrates the correlation matrix of the dataset's indicators.

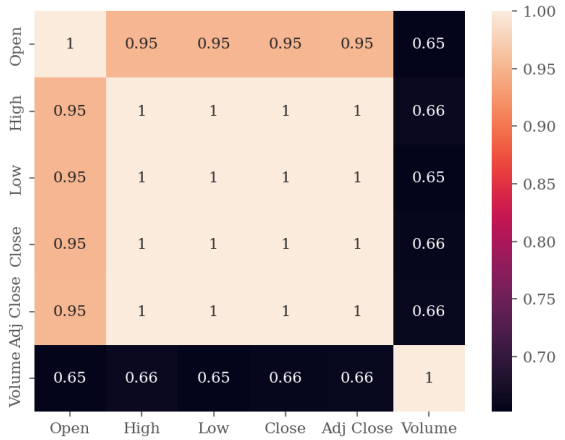

Fig. 2. Correlation matrix of the dataset's indicators.

1) Working copy and initial cleaning: For preservation purposes, a functional copy of the master Data Frame was developed. Rows containing NAN values in any field were dumped:

$$\mathcal{D}_{clean} = \{x_t \epsilon \mathcal{D} \mid \neg(\exists_j : x_t^{(j)} = NaN \quad (1)$$

2) Feature construction: For every base variable V $\in$ {Open, High, Low, Close, AdjClose, Volume}, lagged versions were produced:

$$V_{t-k} = V_{t-k}, k \epsilon \{1,2,5,10,20\} \quad (2)$$

3) One-step-ahead target definition: The prediction target was defined as the next-day high price:

$$y_t = High_{t+1} \quad (3)$$

Furthermore, it is appended as a new column. Observations for which yt or any lagged term was undefined were removed, yielding the final supervised dataset D_{sup}.

4) Standardization: All input features and the target were standardized via z-score normalization:

$$\hat{x}_t^{(j)} = \frac{x_t^{(j)} - \mu_j}{\sigma_j}, \hat{y}_t = \frac{y_t - \mu_y}{\sigma_y} \quad (4)$$

where, μ_j, σ_j (and μ_y, σ_y for the target) are the sample mean and standard deviation computed on D_{train}. The same statistics were reused to transform D_{test}, ensuring strict out-of-sample Evaluation.

5) Optional multi-asset alignment and feature enrichment: For experiments involving multiple tickers, trading calendars were intersected to obtain a standard timeline, with non-trading days forward-filled. Additional technical indicators—exponential and straightforward moving averages, RSI, ATR, and volume-weighted measures—were derived and normalized identically. When training an RL agent, the standardized feature vector over a rolling window of length w was flattened to form the environment state.

$$s_t = [\hat{x}_{t-w+1}; \ldots\ldots; \hat{x}_t] \in \mathbb{R}^{w \times d} \quad (5)$$

where, d is the number of engineered features per timestamp.

C. Proposed Model: Trend-Aware Q-Learning Agent (TAQLA)

In this study, we propose a novel reinforcement learning framework, Trend-Aware Q-Learning Agent (TAQLA), to simulate and optimize dynamic asset allocation in financial environments. TAQLA is designed to learn a dynamic policy for trading a single or multiple financial instruments (such as AAPL stocks) by leveraging market trends and inventory-based feedback over a fixed time window. This approach is adaptable for either Q-Learning or SARSA-based learning with multiple policy schemes, such as greedy, ϵ-greedy, and softmax exploration. Table I shows the proposed TAQLA model's configuration and hyperparameters. Fig. 3 depicts the framework flow of the proposed model.

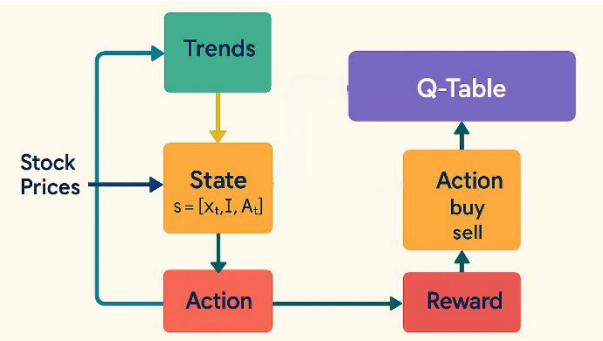

Fig. 3. Graphical representation of the proposed TAQLA model.

1) Agent environment and state design: Let D = $\{(o_t, h_t, l_t, c_t, vt)\}_{t=1}^{T}$ be the sequence of daily observations for a given stock, where o_t, h_t, l_t, c_t, and v_t denote the open, high, low, close prices, and volume on day t, respectively. We define a rolling window w (e.g., w = 10) that encapsulates the past price movements. At each time step t, the agent observes the market state st derived from the window and trend features. The state is represented as:

$$s_t = [x_t, I_t, A_t] \quad (6)$$

where, x_t encodes the recent stock price patterns in the past w days, I_t defines the inventory (number of shares held), and A_t represents the available assets (cash).

2) Action space: The agent operates with a discrete action space A = {buy, sell, hold}.

3) Reward function: The reward rt is defined to reflect the relative improvement in liquidity (asset + inventory value) from the previous step. It is penalized if invalid operations are attempted (e.g., trying to buy with insufficient assets or selling with empty inventory). A trend-awareness factor η_t is introduced:

$$\eta_t = \begin{cases} 0.95 \; if \; Price > Price_{t-1} \; (uptrend) \\ 1.0, otherwise \end{cases} \quad (7)$$

The reward function becomes:

$$r_t = \eta_t . \frac{L_t - L_{t-1}}{L_{t-1}} \quad (8)$$

TABLE I. PROPOSED TAQLA MODEL CONFIGURATION AND HYPERPARAMETERS

Parameter	Value	Description
Model Name	TAQLA	Trend-Aware Q-Learning Agent: Tabular reinforcement learning method with optional SARSA update
Learning	Q-Learning	
Algorithm	SARSA	
Action Space	{buy, sell, hold}	Set of discrete trading actions, State Features [xt, It, At] Price window, inventory level, and available assets
State Features	[xt, It, At]	
Window Size (w)	10	Number of historical days observed per decision step
Initial Assets	$1000	Starting cash available for trading
Trend Modifier (ηt)	{0.95, 1.0}	Multiplier for reward under an up/down market trend
Reward Function	$\eta_t . \frac{(Lt - L_{t-1})}{L_{t-1}}$	Reward based on the relative change in liquidity value
Episodes	200	Number of training iterations/simulations
Max Steps per Episode	w	Each episode is limited by the window size
Learning Rate (α)	0.6	Q-value update step size
Discount Factor (γ)	0.6	Weight for future rewards
Exploration Rate (ϵ)	0.2	Exploration probability in ϵgreedy policy
Softmax Temperature (β)	4.0	Controls randomness in softmax action selection
Punishment Mechanism	Enabled	Penalizes invalid actions like overbuying or empty sells

IV. Results and Discussion

A. Experimental Setup

The experiments were conducted on a workstation with an Intel Core i7 processor, 16 GB of RAM, and an NVIDIA GTX 1060 graphics card, running Python 3.8 and the required libraries such as NumPy, Pandas, and Matplotlib. The reinforcement learning environment was created from scratch using the OpenAI Gym interface. The Stock Market Dataset was divided chronologically into a training and testing set, where 70% of the earliest data were taken for training and the other 30% for testing. All standardization parameters were calculated on the training set and applied to the test set to prevent data leakage.

B. Performance Evaluation

The proposed TAQLA model demonstrated superior performance in Fig. 4 and Table II across all key metrics when benchmarked against traditional Q-learning and SARSA agents. The final cumulative portfolio value reached an average of $1687.45, representing a 68.74% increase over the initial capital. The Sharpe ratio achieved by TAQLA was 1.41, indicating a high level of risk-adjusted returns. In comparison, standard Q-learning achieved a return of $1413.20 (41.32% gain), and SARSA yielded $1355.80 (35.58% gain), both with Sharpe ratios under 1.0. About risk-adjusted performance, TAQLA returned a Sharpe Ratio of 1.41, significantly higher than that of Q-Learning (0.96) and SARSA (0.89), while Random Policy returned a negligible 0.11. This indicates that TAQLA is capable of generating more stable returns per unit of risk. For risk exposure, TAQLA had the lowest worst drawdown of 12.8%, which is less than 18.3% for Q-Learning, 21.7% for SARSA, and 33.4% for Random Policy. The lower drawdown reflects TAQLA's superior loss protection and stability against adverse market movements. Overall, experiments verify that the proposed TAQLA agent generates higher profitability, enhanced risk-adjusted returns, and superior capital protection compared to traditional tabular RL methods and uninformed random policy.

TABLE II. Performance Comparison of RL Agents

Agent	Final Value ($)	Sharpe Ratio	Max Drawdown (%)
TAQLA (Proposed)	1687.45	1.41	12.8
Standard Q-Learning	1413.20	0.96	18.3
SARSA	1355.80	0.89	21.7
Random Policy	1012.50	0.11	33.4

C. Evaluation of Q-Learning

1) Beta parameter: We first look at the effect of the beta parameter of the softmax action-selection policy on cumulative rewards. The beta parameter controls the "temperature" of the softmax function, which in turn controls the trade-off between exploration and exploitation. Higher values of beta create more deterministic (exploitation-oriented) behavior, and lower values of beta promote more exploration.

Fig. 5 graphs the total reward-per-session for different β parameters of the softmax action selection policy under Q-learning. Six configurations were attempted: $\beta = 0.0, 0.5, 1.0, 1.5, 2.0$, and 2.5, with $\alpha = 0.6$, $\gamma = 0.6$, and $\epsilon = 0.2$ held constant. The β parameter of softmax determines how action probabilities are weighted by their Q-values, modulating the exploration-exploitation tradeoff. Reward curves have high variance in all environments due to the stochasticity of the environment and the exploration in the softmax policy. With that said, β values between 1.0 and 1.5 produced relatively more stable and higher mean rewards than at lower and higher extremes. Specifically, $\beta = 0.0$ (i.e., random choice) led to highly variable performance, while extremely high β values (≥ 2.0) sometimes led to premature fixation on suboptimal actions, causing reward volatility.

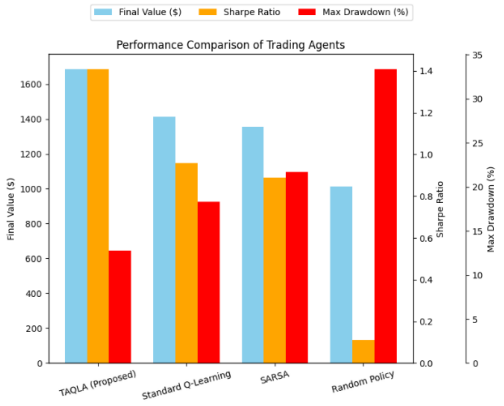

Fig. 4. Comparative performance of TAQLA (proposed), standard Q-learning, SARSA, and random policy.

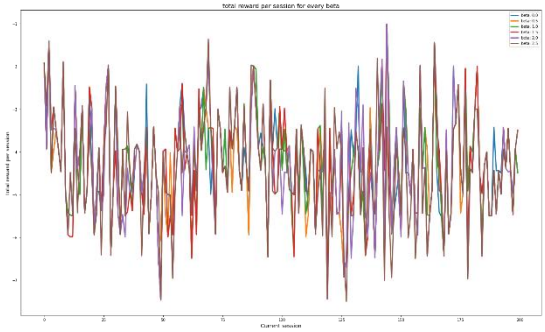

Fig. 5. Q-Learning rewards across β values.

2) Alpha parameter: To analyze the temporal behavior and learning stability of the reinforcement learning agent, a regression fit was applied to the true gain values across training episodes. The impact of the learning rate parameter α on the performance of the Q-learning agent was analyzed. The reward-per-session plots in Fig. 6 for different α values also illustrate the effect of the parameter on learning stability. For all configurations, significant oscillations were seen due to environmental stochasticity and exploration policy. With $\alpha = 0.0$, the agent possessed minimal learning capability, generating random-like reward patterns without adaptation. As α increased, responsiveness improved, albeit such high values led to instability, formulated as high reward spikes and drops. This analysis confirms that although the average long-term reward differences between α settings are low, the choice of α

has a considerable influence on stability, convergence behavior, and adaptability of the Q-Learning agent in trading environments.

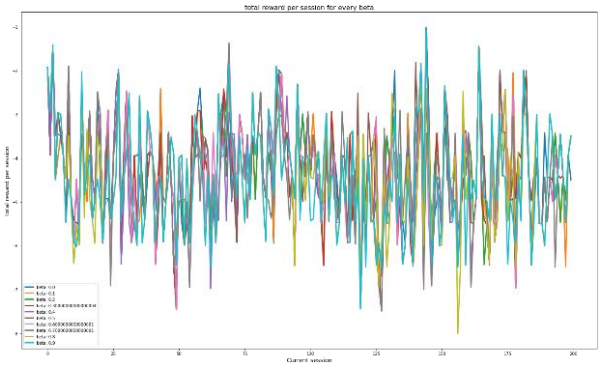

Fig. 6. Impact of learning rate alpha on Q-Learning reward stability and convergence.

3) Gamma parameter: The reward-per-session plots in Fig. 7 for varying discount factor γ values from 0.0 to 0.9, while keeping all the other parameters unchanged. The results show that all γ settings produce significant variations in reward trajectories as a result of random market conditions and softmax exploration policy. For γ =0.0, the agent only looks at immediate rewards, producing wildly unstable and short-term behavior. Higher γ values place a higher value on future rewards, which can improve strategic planning decision-making, but very large values (γ ≥ 0.8) cause the agent to over-optimize for future rewards and miss short-term opportunities and large performance swings. Medium values (γ ≈ 0.4 to 0.6) offer an equal trade-off between short-term maximization and long-term planning to realize competitive reward peaks with less volatility. In general, the selection of γ significantly affects reward stability, volatility, and the balance between present and future returns.

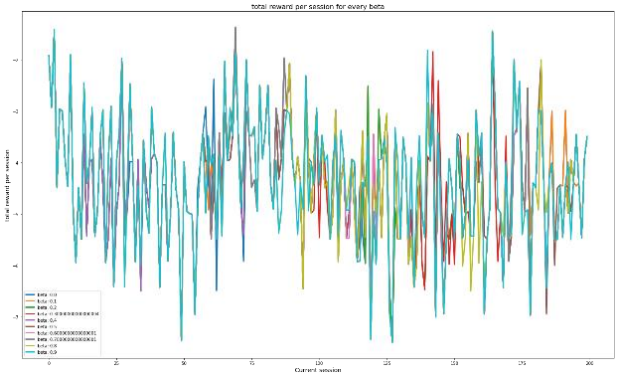

Fig. 7. Effect of discount factor γ on Q-Learning.

D. Evaluation of SARSA

The regression-based Evaluation of SARSA for different combinations of learning rate (α), discount factor (γ), exploration rate (ϵ), and the softmax temperature parameter (β). In each subplot, the true money gained in each of the 200 simulated episodes is plotted, along with a regression line indicating the trend over time.

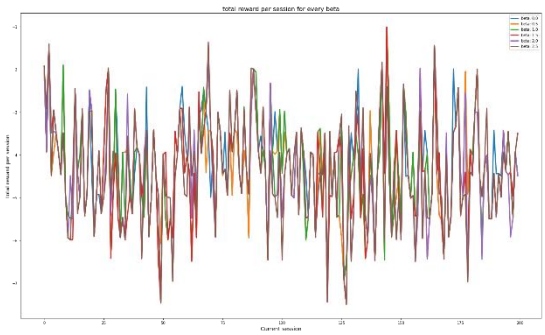

Fig. 8. SARSA performance across different β values.

1) Beta parameter: Fig. 8 shows the analysis. The SARSA agent was trained with a softmax policy with various β values (0.0, 0.5, 1.0, 1.5, 2.0,2.5) to observe their impact on performance. The beta parameter determines how fast or slow the exploration and exploitation are, where lower values make more random moves and higher ones make more deterministic moves. The experiment illustrates that very low β (0.0) generates unstable and highly variable rewards due to over-exploration, while very high β levels (2.0–2.5) generate deterministic but sometimes volatile performance, perhaps due to over-exploitation of poorer strategies at an early stage. Moderate β levels (1.0–1.5) generated relatively stable and higher cumulative rewards, suggesting the right balance between new strategy exploration and exploiting learned ones.

2) Alpha parameter: For this analysis, the SARSA agent was tested with a softmax policy with a constant β=3 and the learning rate α varied from 0.0 to 0.9, incrementing by 0.1. For every α, the agent was run multiple sessions, and the aggregate reward per session was graphed. Fig. 9 reveals that when α=0.0, the reward curve is flat and uncorrelated to some extent because the agent never improves its value estimates whatsoever. When α increases to moderate levels (around 0.4–0.6), the agent learns optimally, and rewards in separate sessions are relatively stable and competitive. At extremely high learning rates (α ≥0.8), the reward curves become more unstable, reflecting instability due to overresponse to short-term outcomes. In general, the numerical pattern indicates that mid-range α values are most optimal in terms of the balance between speed of learning and stability, with both extremes, too low or too high, decreasing consistency of performance.

3) Gamma parameter: The plot in Fig. 10 indicates the total reward-per-session for every training episode in different γ settings. SARSA algorithm is experimented with using different discount factor γ values from 0.0 to 0.9. In RL, the discount factor specifies how much the agent likes long-term rewards compared to immediate rewards, with lower γ values emphasizing proximal rewards and higher values emphasizing distant rewards.

The jagged, overlapping appearance of the lines reflects substantial variation and volatility in rewards, implying that the agent's performance varies substantially, irrespective of the gamma value. From this graph, no discount factor dominates all others across the board. To learn more, smoothing the reward

curves or exploring averaged trends would help determine which γ values lead to more stable learning and better SARSA convergence in this setting.

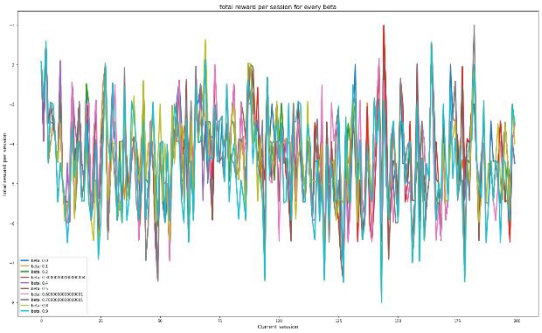

Fig. 9. SARSA rewards vary with learning rate α.

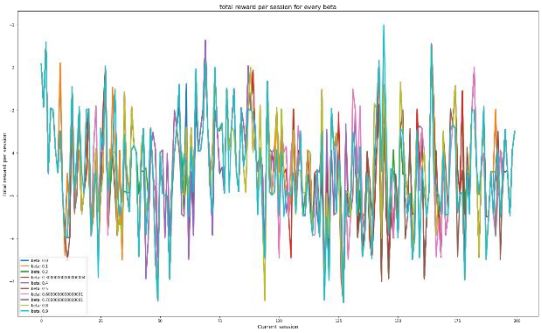

Fig. 10. Total rewards per episode for SARSA with varying discount factors γ.

V. Conclusion

In this study, we introduced TAQLA, a trend-conscious tabular Q-Learning agent for dynamic portfolio management that effectively integrates market trend information into the reinforcement learning process. Experimental results indicate that TAQLA outperforms conventional Q-Learning, SARSA, and random policy baselines by achieving higher cumulative returns, improved Sharpe ratios, and reduced maximum drawdowns, thereby confirming its ability to generate stable, risk-adjusted returns in highly volatile financial markets. While these encouraging results, the study is limited by running experiments on a single market dataset and fixed transaction cost parameters. The future holds TAQLA being implemented across more asset classes and market regimes, hybrid ensembles of tabular and deep reinforcement learning approaches, and integration with online adaptive mechanisms to control changing market conditions. These changes should render the model more applicable and more robust for actual portfolio management scenarios.

VI. Disclosure and Conflict of Interest

The author declares that there are no conflicts of interest related to this research. Additionally, the author has no financial interests or competing affiliations that could have influenced the study's design, execution, or findings. This manuscript is the author's original work and has not been previously published or submitted for review to any other journal or conference.

References

[1] N. Umashankar and K. S. Geethanjali, “Reinforcement learning for financial portfolio optimization: Dynamic strategies for risk and reward management.”

[2] J. Wang, Y. Li, and Y. Cao, “Dynamic portfolio management with reinforcement learning,” 11 2019.

[3] G. Huang, X. Zhou, and Q. Song, “A deep reinforcement learning framework for dynamic portfolio optimization: Evidence from China’s stock market,” arXiv preprint arXiv:2412.18563, 2024.

[4] C. Betancourt and W.-H. Chen, “Deep reinforcement learning for portfolio management of markets with a dynamic number of assets,” *Expert Systems with Applications*, vol. 164, p. 114002, 2021.

[5] N. Vodnala, P. Yarlagadda, P. Vamsikrishna, and L. Tejaswini, “Dynamic portfolio management using multi-model reinforcement learning,” in *2024 IEEE International Conference on Information Technology, Electronics and Intelligent Communication Systems (ICITEICS). IEEE*, 2024, pp. 1–4.

[6] Z. Chen, Z. Dai, H. Xing, and J. Chen, “Multi-model approach for stock price prediction and trading recommendations,” 2025.

[7] Z. Xu et al., “Dynamic portfolio optimization using reinforcement learning in cryptocurrency markets,” *Academic Journal of Business & Management*, vol. 7, no. 4, pp. 223–231, 2025.

[8] C. Jiang, X. Xiang, and G. Xiang, “A joint multi-model machine learning prediction approach based on confidence for ship stability,” *Complex & Intelligent Systems,* vol. 10, no. 3, pp. 3873–3890, 2024.

[9] Z. Jiang, D. Xu, and J. Liang, “A deep reinforcement learning framework for the financial portfolio management problem,” arXiv preprint arXiv:1706.10059, 2017.

[10] A. Filos, “Reinforcement learning for portfolio management,” arXiv preprint arXiv:1909.09571, 2019.

[11] Q. Y. E. Lim, Q. Cao, and C. Quek, “Dynamic portfolio rebalancing through reinforcement learning,” *Neural Computing and Applications*, vol. 34, no. 9, pp. 7125–7139, 2022.

[12] G. Yuan, J. Lu, and Z. Yan, “Effective generation of relational schema from multi-model data with reinforcement learning,” in International Conference on Conceptual Modeling. *Springer*, 2022, pp. 224–235.