Bayesian Robust Financial Trading with Adversarial Synthetic Market Data

Haochong Xia*
Nanyang Technological University
Singapore
haochong001@e.ntu.edu.sg

Simin Li*
Beihang University
Beijing, China
lisiminsimon@buaa.edu.cn

Ruixiao Xu
Beihang University
Beijing, China
xuruixiao@buaa.edu.cn

Zhixia Zhang
Hongxiang Wang
Zhiqian Liu
Beihang University
Beijing, China
22376220@buaa.edu.cn
23371110@buaa.edu.cn
22373024@buaa.edu.cn

Teng Yao Long
Molei Qin
Chuqiao Zong
Nanyang Technological University
Singapore
yaolong001@e.ntu.edu.sg
molei001@e.ntu.edu.sg
ZONG0005@e.ntu.edu.sg

Bo An†
Nanyang Technological University
Singapore
boan@ntu.edu.sg

Abstract

Algorithmic trading relies on machine learning models to make trading decisions. Despite strong in-sample performance, these models often degrade when confronted with evolving real-world market regimes, which can shift dramatically due to macroeconomic changes—e.g., monetary policy updates or unanticipated fluctuations in participant behavior. We identify two core challenges that perpetuate this mismatch: (1) insufficient robustness in existing policy against uncertainties in high-level market fluctuations, and (2) the absence of a realistic and diverse simulation environment for training, leading to policy overfitting. To address these issues, we propose a Bayesian Robust Framework that systematically integrates a macro-conditioned generative model with robust policy learning. On the data side, to generate realistic and diverse data, we propose a macro-conditioned GAN-based generator that leverages macroeconomic indicators as primary control variables, synthesizing data with faithful temporal, cross-instrument, and macro correlations. On the policy side, to learn robust policy against market fluctuations, we cast the trading process as a two-player zero-sum Bayesian Markov game, wherein an adversarial agent simulates shifting regimes by perturbing macroeconomic indicators in the macro-conditioned generator, while the trading agent—guided by a quantile belief network—maintains and updates its belief over hidden market states. The trading agent seeks a Robust Perfect Bayesian Equilibrium via Bayesian neural fictitious self-play, stabilizing learning under adversarial market perturbations. Extensive experiments on 9 financial instruments demonstrate that our framework outperforms 9 state-of-the-art baselines. In extreme events like the COVID pandemic, our method shows improved profitability and risk management, offering a reliable solution for trading under uncertain and rapidly shifting market dynamics.

CCS Concepts

• **Computing methodologies** → *Artificial intelligence*; *Dynamic programming for Markov decision processes.*

Keywords

Quantitative Trading, Generative Model, Robust RL

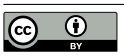
ACM Reference Format:
Haochong Xia, Simin Li, Ruixiao Xu, Zhixia Zhang, Hongxiang Wang, Zhiqian Liu, Teng Yao Long, Molei Qin, Chuqiao Zong, and Bo An. 2026. Bayesian Robust Financial Trading with Adversarial Synthetic Market Data. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770854.3780175

Resource Availability:
The source code of this paper has been made publicly available at https://github.com/XiaHaochong98/Bayesian-Robust-Financial-Trading-with-Adversarial-Synthetic-Market-Data.

1 INTRODUCTION

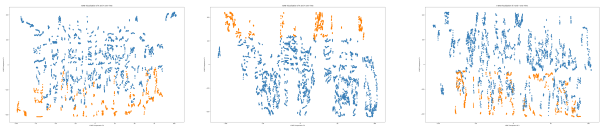
Algorithmic trading systems have become an essential component of financial markets, with reinforcement learning (RL) emerging as a promising method for making financial decisions [34]. However, while these systems often excel in learning from large volumes of historical data, they usually fail to maintain similar performance in out-of-sample data. This overfitting arises from the highly dynamic nature of financial markets, where the testing dynamics diverge from the training dynamics, as illustrated in Fig 1, because markets are continually reshaped by shifting macroeconomic indicators—such as interest rates and inflation [23]—and complex interactions among market participants.

*Both authors contributed equally to this research.
†Corresponding Author.



KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2258-5/2026/08
https://doi.org/10.1145/3770854.3780175



(a) UNG (b) QQQ (c) FXB

Figure 1: State representation (x-axis) and reward (y-axis) reduced to one dimension via t-SNE (Algorithm 4 in Appendix G). The shift in distribution between training (blue points) and testing (orange points) highlights the out-of-distribution issue during testing.

Much of the existing research in this domain centers on equity markets and cryptocurrencies [34]. By contrast, our work focuses on Exchange-Traded Funds (ETFs) of commodities, foreign exchange (FX) pairs, and stock indices—important financial instruments whose performance is more directly correlated with global economic conditions [9, 38]. These instruments exhibit substantial correlations with macroeconomic indicators, providing a strong signal for trading and risk management. In finance research, authorities like the Federal Reserve emphasize macro-driven stress testing; for example, the Dodd-Frank Act Stress Tests [27] require evaluating resilience under severe economic scenarios. While these frameworks are well-established in standard practice, they remain under-explored in algorithmic trading, highlighting the need for more robust, macro-sensitive approaches.

One way to address this is by adopting robust decision-making frameworks that explicitly incorporate uncertainty and adversarial market conditions. Machine learning methods often overfit to historical distributions and assume that future conditions will mirror the past. By contrast, robust reinforcement learning (RL) [17, 25] acknowledges that real-world environment can deviate significantly from simulations, employing a min-max paradigm in which policy maximizes return under worst-case uncertainties. However, existing robust RL approaches are not tailored for trading and fail to account for unobservable macroeconomic drivers that cause drastic regime shifts, limiting their effectiveness. Specifically, implementing robust RL in financial markets under unobservable macroeconomic shifts entails two main challenges: I) developing a data generator that produces realistic, diverse market trajectories conditioned on macro indicators; and II) designing a robust RL framework capable of optimizing trading decisions under adversarial macroeconomic scenarios.

Generating realistic and diverse adversarial conditions is nontrivial. While generative models [24, 45] can be employed to synthesize counterfactual data, it is hard to capture the complexity of financial market dynamics. We propose a macro-conditioned hybrid generator based on a Generative Adversarial Network (GAN) that captures temporal, inter-instrument, inter-feature, and feature–macro correlations in financial markets—key features that characterize the inherent structure of financial market data. By conditioning on macroeconomic indicators, our model learns from historical data while systematically injecting macro-driven variations. This design enables the synthesis of realistic market scenarios covering a broad range of potential "adverse" regimes that go beyond history. Unlike methods treating macro variables as ancillary features, we make them primary controllable conditions in data generation, yielding a more diverse training environment.

Building upon these, we introduce a Bayesian Robust Framework to systematically integrate macroeconomic uncertainties into the training of the trading policy. Conceptually, we formulate the problem as a two-player zero-sum Bayesian Markov game in which worst-case macroeconomic uncertainties are modeled as an adversarial agent that controls macroeconomic indicators in our data generator. By perturbing them away from historical norms, the adversary induces "worst-case" or stress-inducing market conditions. Meanwhile, the trading agent is formulated as a defender that maximize its profit under observed market conditions and Bayesian beliefs about the unknown macroeconomics in current market. By explicitly optimizing for worst-case robustness, the framework maximize the performance lower bound for the trading agent, ensuring its actual returns consistently exceed this threshold and remain reliable under highly volatile or adversarial conditions.

From a theoretical perspective, we aim to achieve a Robust Perfect Bayesian Equilibrium (RPBE) [11], in which the trading agent's policy is optimal given its belief over the market, and the adversary's perturbations capture worst-case macroeconomic scenarios. To solve for this equilibrium, we adopt Bayesian neural fictitious self-play (Bayesian NFSP) [16], enabling stable learning dynamics by computing max-min optimization with time-averaged opponent policy. Meanwhile, the Bayesian extension ensures that the trading agent maintains and updates a belief distribution over market states, enhancing its adaptability to adversarial macroeconomic shifts.

Overall, our key contributions are:

- **Adversarial Framework for Macroeconomic Uncertainties.** We cast trading as a two-player zero-sum game, where a trading agent makes trading decisions and an adversarial agent perturbs macroeconomic conditions.
- **Macro-Conditioned Data Generation.** We propose a deep generative model that synthesizes market data conditioned on macroeconomic indicators.
- **Bayesian Game-Theoretic Optimization.** We cast the robust trading agent and worst-case agent as a two-player zero-sum Bayesian Markov game, solving for a Robust Perfect Bayesian Equilibrium via a quantile belief network and Bayesian neural fictitious self-play.
- **Enhanced Robustness and Profitability.** Experiments across 9 ETFs against 9 baselines show our method consistently gains higher profit with lower risk, and offer robust solution in extreme events like the COVID pandemic.

2 BACKGROUND AND RELATED WORKS

2.1 Robust RL

Our research lies within the field of robust reinforcement learning (RL), which is theoretically grounded in the robust Markov Decision Process (MDP) framework [17, 25, 37, 43]. In a robust MDP, the agent (defender) is trained to be resilient against an adversary that selects worst-case scenarios within an uncertainty set. These uncertainties can arise in environment transitions [21, 29, 44], actions [39], states [47, 48], or rewards [41]. Our work extends this framework by introducing dynamic, time-varying perturbations to the environment while enabling the trading agent to identify and adapt to these attacks for improved performance. Additionally, our research is related to adversarial policy learning [12, 13], which

demonstrates that RL agents can be exploited by adversarial agents executing learned worst-case policies. In our setup, the adversary serves as a type of adversarial policy designed to maximally exploit the vulnerabilities of trading agents. However, unlike traditional adversarial policy research, our primary goal is to train a trading agent that achieves robustness without direct observation of the perturbations introduced by the adversarial policy.

2.2 Bayesian Game

First introduced by Harsanyi, Bayesian games provide a theoretical framework for analyzing games with incomplete information [14]. In a Bayesian game, each agent is unaware of the types of other agents and must act optimally based on its beliefs about others, leading to the formation of a Perfect Bayesian Equilibrium. Agents update their beliefs about others' types using Bayes' rule, incorporating observations of others' actions. Applications of Bayesian games include ad hoc coordination in multi-agent reinforcement learning (MARL), where they facilitate the coordination of agents with varying preferences and types [1, 2, 5, 33]. Bayesian games have also been applied to robust MARL [18, 44], particularly for identifying attackers with unknown types. However, these studies assume that each player's true type remains static. In our financial trading setting, the true economic states are both hidden and evolve dynamically over time. Trading agents must infer these evolving factors from their observations, introducing additional challenges.

2.3 Financial Time-series Generation

A challenge in algorithmic trading is to provide high-quality and diverse market data. Traditional agent-based methods simulate market participants to replicate the data stream [3] or integrate stochastic effects [32], are often constrained by assumptions about agent behaviors and empirical models, raising doubts about their capacity to capture real-world complexity [40]. In contrast, deep generative models provide a data-driven alternative, learning temporal and cross-sectional dependencies from data. Generative Adversarial Networks (GANs) have emerged as a popular framework: TimeGAN [45] combines autoencoder-based representations with adversarial training. In the financial realm, FIN-GAN [36] employs a vanilla GAN architecture to synthesize price features. Conditioning these generative models on macroeconomic factors offers an avenue to simulate market regimes and tail events not fully observed in historical datasets, thereby improving stress-testing and enhancing the robustness of trading. While DeepClair [6] integrates a FEDformer [50] into RL to improve portfolio returns through better trend prediction, our method fundamentally differs by using a GAN-based generator for market simulation within a Bayesian macro-adversarial game. This design is motivated by the need for an adversarial generator, rather than a forecasting backbone.

3 PROBLEM FORMULATION

We begin by introducing the Markov Decision Process (MDP) framework that defines our trading agent and its associated tasks. Subsequently, we outline the formulation of our market data generator and adversarial agent, which are designed to generate synthetic training data to enhance the robustness of the trading agent.

3.1 Financial Market Formulation

In this section, we present our formulation for generating synthetic market data. Consider a market dataset $\mathbf{X}_t$ of length L at time t. This dataset comprises observations of multiple financial instruments and their associated features (e.g., price, volume), yielding a three-dimensional tensor of (timesteps, instrument, feature). Our aim is to learn a probabilistic model that can generate synthetic samples $\mathbf{X}'_t$ with realistic statistical properties, including: I) Temporal correlations; II) Inter-instrument correlations; III) Inter-feature correlations; IV) Feature-macro correlations.

We define our generative model G as

$$G\big(\mathbf{M}_{t,t-L}, \mathbf{N}, \mathbf{X}_{t-L}\big) \;=\; \mathbf{X}'_t,$$

where:

- $\mathbf{X}'_t = [\mathbf{x}'_{t-L+1}, ..., \mathbf{x}'_t]$ is the synthetic data of length L at time t, the abbreviation for $\mathbf{X}'_{[t-L+1:t]}$.
- $\mathbf{M}_{t,t-L} == [\mathbf{m}_{t-2L+1}, ..., \mathbf{m}_t]$ is the macroeconomic state observed over the interval $[t-2L+1, t]$, which has length $2L$, the abbreviation for for $\mathbf{M}_{[t-L+1:t],[t-2L+1:t-L]}$.
- $\mathbf{X}_{t-L}$ is the historical market data of length L at time $t-L$, the abbreviation for $\mathbf{X}_{[t-2L+1:t-L]} = [\mathbf{x}_{t-2L+1}, ..., \mathbf{x}_{t-L}]$.
- $n \in \mathbf{N}$ is a random noise variable.

3.1.1 Distribution Approximation. Our generative model G is tasked with learning the conditional distribution

$$\hat{p}\big(\mathbf{X}_t \;\big|\; \mathbf{M}_{t,t-L},\, \mathbf{X}_{t-L}\big),$$

From a joint distribution perspective, we can write:

$$\begin{aligned} p\big(\mathbf{X}_t,\, \mathbf{M}_{t,t-L},\, \mathbf{N},\, \mathbf{X}_{t-L}\big) \;=\; & p\big(\mathbf{M}_{t,t-L},\, \mathbf{X}_{t-L}\big)\, p(\mathbf{N}) \\ & \times\; p\big(\mathbf{X}_t \;\big|\; \mathbf{M}_{t,t-L},\, \mathbf{N},\, \mathbf{X}_{t-L}\big), \end{aligned}$$

where $\mathbf{N}$ is assumed to be independent of the other variables.

3.1.2 Auto-Regressive Transaction Distribution. Market data for a single time step t can often be broken down into smaller "ticks," indexed by $k \in \{1, 2, \ldots, K\}$. Each tick $\mathbf{X}_{t,k}$ may correspond to a transaction, a quote update, or another micro-event within the time interval $[t, t+1)$. To capture the finer granularity and the inherently auto-regressive nature of these events, we factorize the distribution of $\mathbf{X}_t$ at each tick k conditionally on the previous ticks within the same time step. Formally,

$$\begin{aligned} p\big(\mathbf{X}_{t,1:K} \;\big|\; \mathbf{X}_{t-L},\, \mathbf{M}_{t,t-L},\, \mathbf{N}\big) = \prod_{k=1}^{K} p\Big(\mathbf{X}_{t,k} \;\Big|\; & \mathbf{X}_{t,1}, \ldots, \mathbf{X}_{t,k-1}, \\ & \mathbf{X}_{t-L},\, \mathbf{M}_{t,t-L},\, \mathbf{N}\Big). \end{aligned}$$

3.2 Robust Trading Agent

3.2.1 RL agents for trading. With the definition of data generator, we can now give the formal definition of agents trading a single financial instrument, which is an Exchange-Traded Funds (ETFs) of commodities, foreign exchange pairs, and stock indices. Conventionally, The trading problem can be framed as a sequential decision-making process, where an agent seeks to maximize total profit under uncertainties. This problem is naturally modeled as a trading Markov Decision Process (MDP) within the reinforcement learning (RL) framework [34]. In this setup, the agent interacts

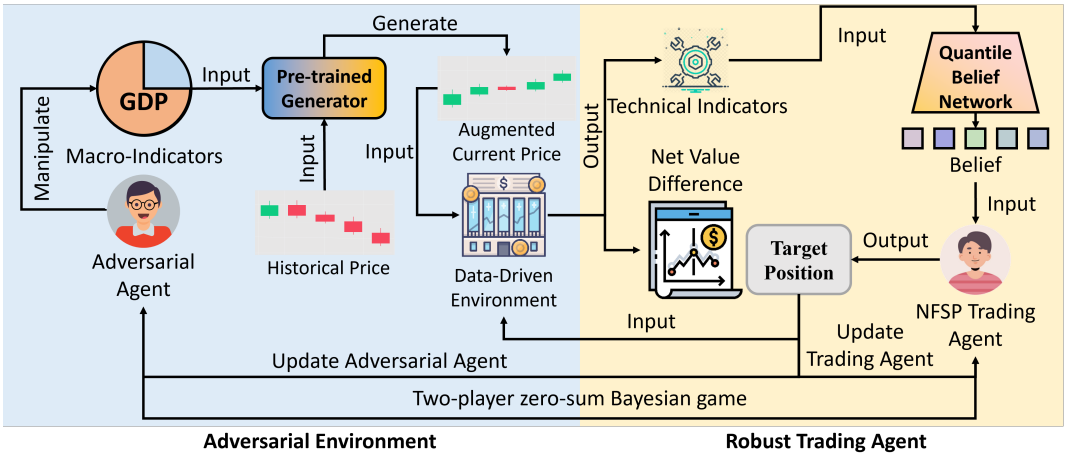


Figure 2: Overall architecture of our framework. An adversarial agent and a pre-trained generator jointly form an adversarial environment, while a trading agent is trained on observations augmented by this environment.

with the market by making trading decisions. Formally, the MDP is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathbf{M}, \mathcal{T}, \mathcal{R}, \gamma \rangle$.

- The **state space** $\mathcal{S}$ consists of a set of technical indicators and the agent's position. At time t, the state is defined as $s_t = [f(X_{[t-L+1,t]}), X_{[t]}, a_{t-1}]$, where $f(\cdot)$ is the operator that transforms a chunk of data into technical indicators and a_{t-1} is the previous action taken by the agent.
- The **action space** $\mathcal{A}$ includes three possible choices: long position, short position, or close position [8].
- The **macroeconomic indicator** $\mathbf{M}$ is a market representation, which is correlated with the state transition function.
- The **state transition function** $\mathcal{T} : \mathcal{S} \times \mathbf{M} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ describes how market states evolve over time. The transition function can be approximated by the learned data generator.
- The **reward function** $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is computed as the return of the net value, incorporating factors such as transaction fees [42].
- $\gamma \in [0, 1)$ is the discount factor.

The trading process continues in T steps. In each step $t \in T$, the agent executes an action a_t using a policy $\pi(a_t|s_t)$, and observes reward r_t. The goal is learn a policy π^* to maximize the value function $V_\pi(s_0) = \mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t r_t | s_t = s]$, which satisfies the following Bellman equation:

$$V_\pi(s) = R(s, a) + \gamma \sum_{s' \in \mathrm{S}} \hat{p}\big(f(\mathbf{X}_t) \,\big|\, \mathbf{M}_{t,t-L},\, f(\mathbf{X}_{t-L})\big) V_\pi(s').$$

3.2.2 Threat Model. In live trading, the macroeconomic data are updated at a slower frequency than the decision process of the trading agent. Such a slowly-evolving macroeconomic factor is a reflection of a longer process and does not reflect the accurate macroeconomic situation encountered by a trading agent that operates at a faster frequency. Since the ground truth macroeconomic data $\mathbf{M}^*_{t-L}$ is not known to the trading agent, we assume the slowly-evolving macroeconomic data $\mathbf{M}_{t,t-L}$ deviates from the ground truth data by $\mathbf{M}^\alpha_{t,t-L}$, satisfying the following conditions:

$$\mathbf{M}^*_{t-L} = \mathbf{M}_{t,t-L} + \epsilon \mathbf{M}^\alpha_{t-L},$$

where ϵ is a hyperparameter to measure the extent of changes in observed macroeconomic data. In our paper, we use an adversarial RL agent to learn the worst-case change in macroeconomic data that minimize the reward of the trading agent, defined as:

$$\mathbf{M}^{\alpha,*}_{t-L} \in \arg\min_{M^\alpha_{t-L}} V_\pi(s).$$

3.2.3 Solution concept. Under the worst-case perturbations applied on the macroeconomic factors, our robust financial trading agent $\pi(\cdot|s_t)$ must be able to maximize profit under such uncertainties:

$$\pi^*(\cdot|s_t) \in \arg\max_{\pi} \min_{M^\alpha_{t-L}} V_\pi(s).$$

4 METHOD

While the trading agent seeks to maximize reward despite market fluctuations, the ground-truth macroeconomic factors are rapidly changing and unobservable. As a result, there is an inherent level of incomplete information that conventional robust RL methods must incorporate to make effective trading decisions. This challenge remains unresolved in many existing approaches, which often assume stationary or fully observable market conditions.

In response to these challenges, we propose an adversarial framework, illustrated in Figure 2, that comprises a pre-trained data generator, an adversarial agent, and a robust trading agent with a quantile belief network. We begin by describing our data generator for synthetic market data generation. Subsequently, we outline the adversarial agent that perturbs the data distribution, followed by the robust trading agent designed to operate effectively under adversarial market conditions.

4.1 Data Generator

To address the absence of a realistic and diverse environment for trading, we propose our generator, following the hybrid architecture in TimeGAN [45], tailored to model temporal, inter-instrument, inter-feature, and feature-macro correlations.

4.1.1 Data Transformation. Instead of directly generating raw price and volume data, we transform the raw into a feature set comprising open-to-close return, close-to-close return, low-to-close ratio, high-to-close ratio, and log(close × volume), allowing the generator to learn a more structured distribution. Additionally, to handle missing

data in financial instruments, we implement a market-aware imputation technique that leverages pairwise correlations, detailed in the Algorithm 3 in the Appendix B. For each missing entry, we identify all tickers with valid data at that point, compute correlation-based weights, and use their weighted average values to fill in the gap. This preserves both the temporal dynamics and market structure, unlike conventional imputation methods like mean-filling.

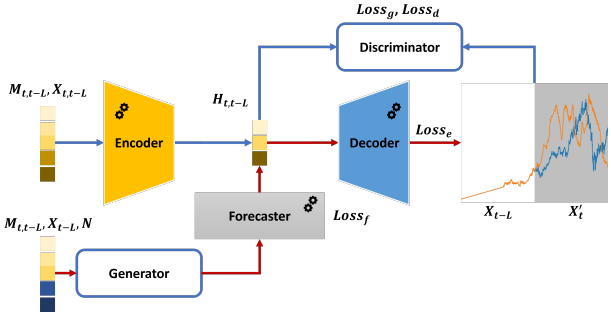


Figure 3: Overall architecture of the data generator. The encoder, decoder, and forecaster are pre-trained (marked with gear icons). The blue routes are only for training, while the red routes are for both training and inference.

4.1.2 Architecture. As shown in Figure 3, the architecture of the proposed data generator comprises three key components: the encoder, forecaster, and decoder. These components are initially pre-trained on relevant tasks to capture the underlying features of the input time-series data. Subsequently, fine-tuning is performed with the inclusion of the generator and discriminator modules, enabling adversarial training for enhanced data generation. The detailed network architecture is in section 5.2.

AutoEncoder. To capture inter-instrument, inter-feature, and macro-feature correlations, we begin by passing the processed feature through an autoencoder. This module decomposes into an encoder that transforms high-dimensional features of multiple instruments and macro indicators into a compressed latent representation and a decoder that reconstructs features from this latent space.

Formally, if $\mathbf{X}_t$ is the real market data at time t, then the encoder $E(\cdot)$ produces a latent embedding $\mathbf{H}_{t,t-L} = E(\mathbf{X}_{t,t-L}, \mathbf{M}_{t,t-L})$. From there, a decoder $D(\cdot)$ attempts to reconstruct $\mathbf{X}_{t,t-L} \approx \tilde{\mathbf{X}}_{t,t-L} = D(\mathbf{H}_{t,t-L})$. The autoencoder training objective commonly adopts a mean squared error (MSE) loss:

$$Loss_{\text{e}} = \left\|\mathbf{X}_{t,t-L} - D\big(E(\mathbf{X}_{t,t-L}, \mathbf{M}_{t,t-L})\big)\right\|^2,$$

where $\|\cdot\|$ denotes an appropriate norm (e.g., ℓ^2), possibly computed over all instruments and all features. Minimizing $Loss_e$ forces the latent representation $\mathbf{H}_{t,t-L}$ to preserve the cross-sectional and cross-feature patterns crucial for realistic data generation.

Forecaster. Financial time-series data exhibit certain temporal dynamics, including autocorrelation structures (e.g., volatility clustering and seasonality). To address this, we incorporate a forecaster that learns to predict the subsequent latent state given a sequence of past embeddings, taking the encoder's latent outputs and generating a one-step-ahead prediction $\hat{\mathbf{H}}_{t+1}$.

The corresponding loss is an MSE between forecasted and true latent representation, which is $Loss_{\text{f}} = \left\|\mathbf{H}_{t+1} - F(\mathbf{H}_t)\right\|^2$. By enforcing accurate forecasts in the latent space, the model learns to propagate temporal correlations through the generator.

Generator and Discriminator. The generator $g(\cdot)$ outputs synthetic latent representations. Meanwhile, the discriminator $d(\cdot)$ differentiates real latent embeddings from those generated by g.

Generator. The generator is trained to (1) fool the discriminator by making its synthetic outputs resemble real embeddings and (2) preserve essential market statistics:

$$g\big(\mathbf{M}_{t,t-L}, \mathbf{N}, \mathbf{X}_{t-L}\big) = \mathbf{H}'_{t,t-L},$$

The forecaster then refines these latent outputs to capture temporal correlations. Finally, the decoder maps $\mathbf{H}'_{t,t-L}$ back into $\mathbf{X}'_{t,t-L}$ and $\mathbf{X}'_t$ is used to get s' : In practice, the generator's loss $Loss_g$ comprises several terms:

- *Adversarial Terms.* Let $d(\cdot)$ output the logit that indicates "real" vs. "fake". The generator seeks to maximize d's confusion by minimizing a binary cross-entropy (BCE) loss:

$$Loss_{\text{adv}} = \text{BCE}\big(d(\mathbf{H}', M), \mathbf{1}\big) + \text{BCE}\big(d(F(\mathbf{H}'), M), \mathbf{1}\big),$$

- *Moment-Matching & Masking.* Since financial data contain missing entries and often exhibit distinct statistical signatures, we compute a masked mean and variance loss within each feature, ignoring missing values:

$$Loss_{\text{moments}} = \left|\mu_{\text{fake}} - \mu_{\text{real}}\right| + \left|\sqrt{\sigma^2_{\text{fake}}} - \sqrt{\sigma^2_{\text{real}}}\right|,$$

  while a separate term $Loss_{\text{std}}$ measures relative variance differences. Both are applied only over valid (non-missing) positions, as indicated by a mask $\mathbf{M}$.
- *Divergence and Mode-Seeking.* Additional divergence-based losses can further align synthetic and real distributions, while mode-seeking objectives encourage diversity in generated sequences. By comparing generator outputs from different noise samples $(\mathbf{N}_1, \mathbf{N}_2)$, the mode-seeking loss ensures that similar inputs do not always map to the same synthetic trajectory, fostering richer diversity.

By summing these terms with suitable weights, $Loss_g$ balances the requirements of adversarial realism, temporal coherence, and matching of key financial statistics.

Discriminator. The discriminator $d(\cdot)$ is designed to differentiate real latent representations from synthetic ones. The discriminator is trained using binary cross-entropy (BCE) loss:

$$Loss_d = \text{BCE}\big(d(\mathbf{H}),\, 1\big) + \text{BCE}\big(d(\mathbf{H}'),\, 0\big) + \text{BCE}\big(d(f(\mathbf{H}'),\, 0\big),$$

In conjunction, these modules collectively allow us to generate synthetic market scenarios that realistically reflect the multifaceted dependencies present in real-world financial data.

4.2 Bayesian Robust Trading Agent via Adversarial Training

Given the generator described above, we now focus on developing robust trading agent under adversarial market conditions. The goal is to maximize profit despite the unknown macroeconomic uncertainties. We formalize this as a Bayesian game, and define optimal solution as a robust perfect Bayesian equilibrium, which agents

Models	DBB			GLD			UNG		
	ARR%↑	SR↑	MDD%↓	ARR%↑	SR↑	MDD%↓	ARR%↑	SR↑	MDD%↓
(1) Buy and Hold	0.02	0.19	35.11	4.77	0.66	21.03	-18.71	-0.18	86.62
(2) DQN	13.59	1.08	33.33	5.35	0.82	12.81	18.55	0.93	53.86
(3) Robust Trading Agent	13.93	1.61	12.70	9.25	1.20	11.14	25.46	1.05	63.87
(4) Naïve Adversarial	2.79	0.45	15.61	7.66	1.08	12.96	23.84	1.03	52.53
(5) RoM-Q	16.26	1.22	21.38	8.06	1.13	19.30	27.17	1.09	54.92
(6) RARL	6.53	0.83	11.87	1.37	0.91	2.84	26.74	1.07	51.20
(7) DeepScalper	5.89	0.65	20.60	6.37	1.03	11.86	18.62	0.92	79.52
(8) EarnHFT	5.94	0.60	29.95	7.23	1.00	12.06	24.33	1.03	70.64
(9) CDQN-rp	4.68	0.60	30.29	6.48	0.86	21.56	18.71	0.92	42.91
(10) w/o adv agent	13.04	1.02	24.21	7.75	1.09	15.53	24.92	1.05	50.54
(11) IPG	2.55	0.37	25.33	5.81	1.01	10.10	16.79	0.88	62.99
Ours	26.03	1.86	11.00	8.96	1.20	8.33	29.12	1.10	49.37
Models	SPY			QQQ			IWM		
(1) Buy and Hold	7.72	0.82	25.36	7.92	0.71	35.62	-3.28	-0.05	33.13
(2) DQN	7.94	0.84	19.64	7.48	0.68	35.54	10.71	1.13	19.60
(3) Robust Trading Agent	9.62	1.29	13.34	8.37	0.74	34.04	12.96	1.03	23.90
(4) Naïve Adversarial	6.40	0.74	22.56	9.14	0.78	36.16	11.30	0.99	22.58
(5) RoM-Q	7.52	0.81	26.14	9.02	0.77	33.96	13.31	1.01	29.71
(6) RARL	8.05	0.89	16.13	8.53	0.83	30.68	8.16	0.76	25.71
(7) DeepScalper	5.42	0.76	16.54	7.70	0.69	35.61	9.06	0.80	26.37
(8) EarnHFT	5.53	0.73	13.72	8.17	0.72	35.62	11.25	1.22	28.42
(9) CDQN-rp	4.74	0.57	25.36	6.96	0.65	37.05	6.32	0.66	25.68
(10) w/o adv agent	7.23	0.78	17.63	11.47	0.91	28.54	15.32	1.20	27.55
(11) IPG	5.63	0.69	24.32	11.24	0.93	29.72	13.43	1.11	31.58
Ours	12.31	1.20	12.03	19.03	1.29	24.71	17.32	1.41	24.92
Models	DBC			FXY			FXB		
(1) Buy and Hold	11.78	0.99	27.78	-9.18	-1.73	31.79	-3.08	-0.49	24.97
(2) DQN	11.78	1.00	27.68	8.22	1.39	13.94	3.30	0.87	12.75
(3) Robust Trading Agent	9.27	0.98	16.78	1.63	0.38	22.32	3.21	0.67	10.42
(4) Naïve Adversarial	15.70	1.28	19.95	7.54	1.29	11.53	5.00	0.94	11.77
(5) RoM-Q	13.00	1.06	16.76	13.78	2.13	10.64	2.82	0.61	9.23
(6) RARL	12.37	1.11	29.23	8.04	1.56	9.65	0.11	0.08	7.95
(7) DeepScalper	10.21	0.93	18.58	9.67	1.85	5.74	1.54	0.46	10.21
(8) EarnHFT	8.25	0.89	27.74	11.77	1.81	11.25	3.65	0.76	10.76
(9) CDQN-rp	8.53	0.83	28.79	5.99	1.12	10.59	2.70	0.62	16.92
(10) w/o adv agent	12.77	1.31	21.83	13.15	2.43	6.55	2.70	0.62	16.92
(11) IPG	10.62	0.95	19.58	11.39	1.83	13.47	4.15	1.02	9.52
Ours	15.29	1.34	16.50	15.14	2.51	5.80	5.13	1.07	7.62

Table 1: Performance Comparison of Trading Models: The best model is in red, the second-best in yellow.

make optimal decisions based on current belief about the macroeconomic status of the market. To compute belief in highly complex market, we introduce a quantile belief network to simplify the inference of beliefs in practical trading settings. To achieve robust perfect Bayesian equilibrium, we propose Bayesian neural fictitious self-play, which provides a stable policy optimization framework that involves a robust trading agent and a worst-case adversary agent that controls the macroeconomic indicators.

4.2.1 Bayesian Game Formulation. Financial markets inherently provides incomplete information. The observable macroeconomic factor such as prices and volumes evolves at lower frequencies, and do not reflect the true macroeconomic factors at the time agents is making it decisions. Besides, many influential factors such as order flow, policy decisions remain partially hidden. Thus, while our policies are trained to be robust against worst-case uncertainties in macroeconomic factors, the policy do not have knowledge of the true macroeconomic factors at current time.

Bayesian game [14] provides a principled way of decision making under incomplete information. In a Bayesian game, the agent makes decisions based on beliefs about the incomplete information, which is updated by Bayes' rule as the game proceeds. Under the formulation of a Bayesian game, our trading agent does not treat the observed macroeconomic data $\mathbf{M}_{t,t-L}$ as ground truth and maintains a belief b over the ground truth macroeconomic data through Bayes' rule, which infers the posterior distribution over macroeconomic indicators based on current market observations.

Under the Bayesian game formulation, we treat the decision process as a *two-agent zero-sum Bayesian Markov game*, where one adversary agent selects the worst-case uncertainties on macroeconomic data, simulating the worst case the trading agent can face. The robust trading agent must maximize its profit under such uncertainties, but have to infer the true perturbations added by the

adversary. As such, the goal of the adversary is to learn a worst-case uncertainty added on macroeconomic factors, which minimize the profit gained by trading agent. In practice, we use an RL agent $\pi^{\alpha}(\mathbf{M}^{\alpha,*}_{t-L}|\mathbf{M}_{t-L}, S_{t-L})$ to learn such a worst-case agent, conditioned on the current state. Formally, this is defined as:

$$\pi^{\alpha}(\mathbf{M}^{\alpha,*}_{t-L}|S_{t-L}) \in \arg\min_{\pi^{\alpha}} V_{\pi}(s_t, b_t).$$

where $V_{\pi}(s_t, b_t)$ is the value function of the trading agent. Since in our Bayesian game formulation, the trading agent maintains a belief over the market, the adversary is required to manipulate the market subtly, such that the trading agent does not recognize an explicit change in the market but still performs suboptimally in trading. While the worst-case adversary can be optimized via any RL method, we train it via Q-learning.

As for our robust trading agent, its goal is to learn a *Robust Perfect Bayesian Equilibrium*, which maximize the value function according to its current belief $\hat{\mathbf{M}}_{t,t-L}$ over the true macroeconomic:

$$(\pi^{*}(\cdot|s_t, \hat{\mathbf{M}}_{t,t-L}), \pi^{\alpha}(\mathbf{M}^{\alpha,*}_{t-L}|S_{t-L})) \in \arg\max_{\pi} \min_{\pi^{\alpha}} V_{\pi}(s_t, b_t).$$

4.2.2 Quantile Belief Network. To handle evolving and often adversarial market conditions, the agent maintains a belief distribution over hidden states. We implement a Quantile Belief Network (QBN) to approximate this posterior distribution. For each observation s, the QBN jointly embeds both feature and temporal inputs, then passes them through an LSTM-based encoder. A normalization layer stabilizes the hidden representation, and a linear decoder outputs multiple quantiles of the market state: $b = \text{QBN}(s)$. By predicting quantiles rather than fitting a single parametric form, the QBN can capture the heavy-tailed and skewed return distributions commonly observed in financial data. In practice, selecting an inappropriate target leads QBN training to non-convergency, particularly in trading environments. To mitigate this, our QBN compares its predicted return distributions to a short-term moving average (5-day window), a strong market-state indicator. Systematic deviations from this baseline may signal regime changes or adversarial behavior, prompting a more cautious trading stance. In this way, the quantile-based representation operates as a continuous monitor of partial observability and evolving market conditions.

4.2.3 Robust Trading via Bayesian Neural Fictitious Self-play. Our trading agent seeks to maximize its cumulative reward under uncertainties, which is difficult for two reasons. First, the worst-case adversary adapts while the trading agent updates, making the environment non-stationary. Second, the agent cannot directly observe changes in these factors, so it must act optimally with its belief.

We address these challenges via *Bayesian Neural Fictitious Self-Play* (Bayesian NFSP). We use Bayesian methods to learn a policy and a value function conditioned on the agent's belief. The adversary, the trading agent, and the value function are jointly updated using NFSP for stability. We first discuss the value function conditioned on its belief b, which is updated by combining the Harsanyi transform [14] with the Bellman equation in our trading MDP:

$$Q_{\pi}(s, a, b) = R(s, a) + \gamma \sum_{s' \in \mathbf{S}} \sum_{m \in \mathbf{M}} G(x'_t|\mathbf{M}_{t,t-L}, \mathbf{N}, \mathbf{X}_{t-L}) \pi^{\alpha}(\mathbf{M}^{\alpha,*}_{t-L}|\mathbf{M}_{t-L}.X_{t-L}) Q_{\pi}(s', a', b'),$$

With belief updated via Bayes' rule, we use the QBN, ensuring faster and more stable convergence. This can be learned via TD loss:

$$\mathcal{L}_{TD} = (Q_{\pi}(s, a, b) - R(s, a) - Q_{\pi}(s', a', b'))^2,$$

This setup treats the adversary's actions as part of the environment transitions. We use the same value function to update both the trading agent and the adversary. To achieve stable learning in this setting, we adopt NFSP, which avoids divergence or convergence to suboptimal equilibria. NFSP maintains time-averaged policies for both agents. Each agent updates its policy by training against the opponent's time-averaged policy, and updates its own time-averaged policy with its current policy at each step. The complete process is given in the Algorithm 2 in the Appendix B.

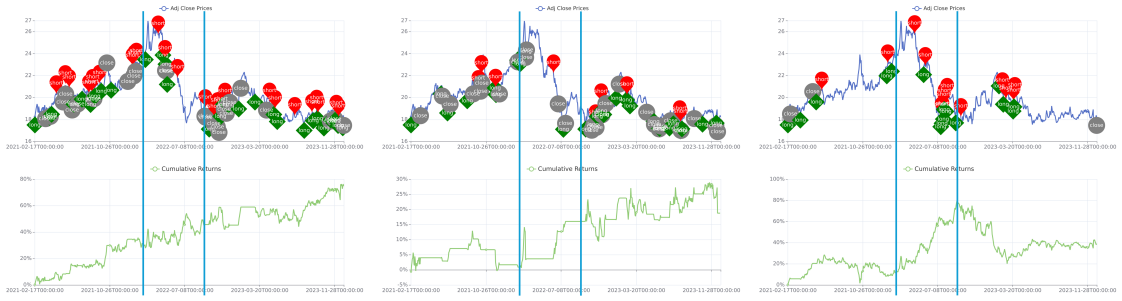
5 EXPERIMENTS

5.1 Dataset

We conduct testing on nine ETFs covering commodities, FX pairs, and stock indices, dating back to each instrument's first trading day. For dataset splitting, we use data from 2018 to 2020 for validation, 2021 to 2024 for testing, and the remaining data for training across all datasets. Both the data generator and trading agent use the same dataset setting. We use 46 macroeconomic indicators sourced from the Federal Reserve Economic Data (FRED), covering key aspects such as economic activity and interest rates. The feature selection process is described in Algorithm 5 in the Appendix.

5.2 Robust Trading Agent

Baselines. We compare our method against 9 baselines: **(1) Buy-and-Hold**: A rule-based baseline holding the instrument long throughout. **(2) DQN** [22]: Deep Q-learning. **(3) Robust Trading Agent** [16]: Uses QBN and NFSP with DQN. **(4) Naïve Adversarial** [28]: Injects random noise as an attack based on (3). **(5) RoM-Q** [26]: Minimizes the Q-value of the agent based on (3). **(6) RARL (Ours w/o generator)** [30]: Perturbs states using an adversarial agent controlling noise based on (3). **(7) DeepScalper** [35]: A risk-aware RL framework for trading. **(8) EarnHFT** [31]: An three-stage hierarchical RL framework for trading. **(9) CDQN-rp** [51]: A CDQN-based RL method with a random target update for risk-aware trading.
Ablations. We compare our method against 3 ablations: **(10) IPG (Ours w/o Bayesian NFSP)**: Use Independent Policy Gradient [7] instead of Bayesian NFSP in our method. **(11) Ours w/o adv agent**: Our method without the adversarial agent and uses random noise as the generator control. Additionally, **(6) RARL** can be seen as **Ours w/o generator**.
Network Architectures The Encoder, Decoder, Forecaster and Discriminator of the generator consist of LSTM blocks. For the trading agent, the encoder uses an embedding layer followed by Transformer blocks with MLP and Tanh. The decoder is a linear layer. The Q-function is learned via the encoder-decoder pipeline.
Metrics. We use metrics following [31, 34], including a profit metric annual return rate (ARR) as $\frac{V_T - V_0}{V_0} \times \frac{C}{T}$, where V is the net value, T is the number of trading days, and $C = 252$, a risk-adjusted profit metrics Sharpe ratio (SR) as $\frac{\mathbb{E}[\mathbf{r}]}{\sigma[\mathbf{r}]}$, where $\mathbf{r} = [\frac{V_1 - V_0}{V_0}, \frac{V_2 - V_1}{V_1}, ..., \frac{V_T - V_{T-1}}{V_{T-1}}]^T$, and a risk metric: maximum drawdown (MDD) measures the largest loss from any peak to show the worst case.



(a) Our method (b) RARL (c) DQN

Figure 4: Comparison of trading actions (top) and cumulative returns (bottom) for our method, RARL, and DQN on DBB from 2021 to 2024. Our method adapts to both high-volatility pandemic phase (between the blue lines) and calmer phases (otherwise).

5.2.1 Main Results. Table 1 compares our method with 9 baselines and 2 ablations on 9 instruments. Our model consistently outperforms across various asset classes, including Commodities (DBB, GLD, UNG, DBC) and Equities (SPY, QQQ, IWM), demonstrating strong adaptability. In Currency ETFs (FXY, FXB), while Buy-and-Hold suffers negative returns, our method delivers consistently positive ARR with lower MDD, showcasing its effectiveness in navigating macro-driven currency fluctuations. The Wilcoxon signed-rank test shows that the superiority of our result is statistically significant ($p < 0.05$) as detailed in Appendix A.

Ablations. Baseline (6), (10), (11) are ablations of our studies. **RARL (6)** can be seen as our method without generator. As shown in Table 1, RARL does not have assess to the market dynamics (our generator), learning overly conservative policy that achieving good risk control (higher MDD), but gaining lower profit. **IPG (10)** highlights the importance of our Bayesian NFSP on stablizing training dynamics, while **Ours w/o adv agent (11)** shows optimizing with random noises makes the policy still prone to fluctuations, highlighting the importance of max-min optimization and using an adversarial agent.

5.2.2 Case Study. We compare three policies—DQN, RARL, and Ours—on DBB (an ETF for metals) from 2021 to 2024. Figure 4 shows trading decisions and returns for each method. In 2021-2024, two phases emerged: I) 2022 COVID pandemic volatility peak, marked by the period between the blue lines, driven by rapid rate hikes, supply-chain disruptions, and inflation surges; II) a calmer phase other than the period in I) where markets stabilized and DBB is in sideways trends. We analyze the performance of each method:

DQN: Great in the Volatility, Weak Otherwise. DQN reacts to large price changes by maximizing returns through riskier decisions, often making "high-profit bets":

- *2022 Outperformance*: Frequent trades in volatile markets allow DQN to capture major swings, yielding high gains.
- *Losses in Calmer Markets*: Post-volatility, DQN misreads smaller fluctuations and accumulates losses with its low frequent "bet and wait" trading policy, leading to losses and drawdowns.

RARL: Conservative Under Stress, Misses Big Profits. RARL emphasizes worst-case risk management:

- *Safe in Volatility and Calmer Markets*: RARL minimized exposure to large shocks in 2022. In calmer markets, RARL does more frequent trading than QDN, thus avoiding losses.
- *Under-Exploitation of profit opportunities*: RARL avoided major losses by staying cautious but failed to capitalize on price moves. It performed consistently, though gains were the lowest among these three methods due to its conservative nature.

Ours: Combining Strengths of DQN and RARL. Our method exploits big moves (like DQN) without incurring excessive drawdowns when transitioning to the calmer market (like RARL):

- *Capturing Spikes*: Like DQN, it enters significant positions when volatility peaks, netting substantial gains.
- *Adapting to Calmer Periods*: When the market is in the calmer phase, the policy transitions to a more conservative trading style, making frequent decisions and steady profits.

Although DBB did see swings before 2021, the post-2021 macro environment introduced policy-driven fluctuations. DQN capitalized on volatility bursts but struggled in subsequent calmer markets. RARL stayed safe in extremes but sacrificed potential gains. Our method displays strong performance in both volatile and low-volatility regimes, confirming that adapting across multiple macro-driven scenarios produce more robust and profitable outcomes.

5.2.3 Time Consumption. For the 9 ETFs we evaluated, training took an average of 22.22 h (600k steps in total) for each ETF. The inference time takes an average of 2.725 ms per step. This inference time is trivial compared with decision frequency (1 action per day), making it feasible for deployment.

5.3 Generated Data Evaluation

We evaluate our generated data by comparing it against generators, including TimeGAN [45], RCWGAN [15], GMMN [19], CWGAN [46], and RCGAN [10]. We also include ablation on the architecture of our generator using Informer [49] and iTransformer [20].

5.3.1 Correlation Difference. We evaluate how well each model preserves real data correlation using three metrics, all measuring the difference in pairwise correlations between generated and real data: I) **Feature-macro**: Differences in correlations between market and macroeconomic variables; II) **Inter-instrument**: Differences in correlations among instruments; III) **Inter-feature**: Differences in correlations among features within each instrument.

Table 2 shows that our method achieves the smallest difference across all metrics, indicating superior alignment with real correlation structures and the highest level of financial fidelity.

Method	Feature-macro	Inter-instrument	Inter-feature
CWGAN	0.4542	0.3786	0.4732
GMMN	0.3134	0.2807	0.3509
RCGAN	0.4258	0.3533	0.4415
TimeGAN	0.4643	0.3672	0.4590
RCWGAN	0.4584	0.3701	0.4625
Ours(LSTM)	0.2678	0.2590	**0.3235**
Ours(Informer)	**0.2523**	0.2712	0.3342
Ours(iTransformer)	0.2819	**0.2361**	0.3371

Table 2: Comparison of Feature-macro, Inter-instrument, and Inter-feature correlation differences with real history data.

Method	ReturnsACF	AbsReturnsACF	Leverage
CWGAN	0.2088	0.1189	0.2883
GMMN	0.0825	0.0766	0.2221
RCGAN	0.1350	0.1105	0.2983
TimeGAN	0.2189	0.1320	0.2965
RCWGAN	0.1840	0.1146	0.3033
Ours(LSTM)	0.0389	0.0461	0.1725
Ours(Informer)	**0.0327**	**0.0458**	0.1937
Ours(iTransformer)	0.0357	0.0488	**0.1706**

Table 3: Comparison of ReturnsACF, AbsReturnsACF, and Leverage differences with real history data.

5.3.2 Market Stylized Facts. We assess how well each model captures essential time-series characteristics, which serve as key indicators of temporal correlations [4]: I) **ReturnsACF difference** measures the difference in the autocorrelation function of returns between real and generated data; II) **AbsReturnsACF difference** focuses on the autocorrelation of absolute returns, a primary indicator of volatility clustering; III) **Leverage** quantifies the difference in the correlation between past returns and future volatility, reflecting the asymmetry often observed in financial markets.

As shown in Table 3, our approach exhibits significantly lower differences across all stylized-fact metrics, underscoring its effectiveness in replicating fundamental market dynamics.

6 CONCLUSION

We proposed a Bayesian adversarial framework for robust algorithmic trading that combines macro-conditioned synthetic data generation and RL under adversarial conditions. A generator produces realistic market scenarios reflecting the changing market, while a two-player Bayesian Markov game—pitting an adversarial macro-perturbing agent against a trading agent—enables robust policy learning through adversarial training. Empirical results show that our approach significantly improves profitability and risk management over baselines, especially when adapting to unforeseen macroeconomic shifts. Furthermore, validation against competing generative models demonstrates the superior fidelity of our synthetic data. Overall, this scalable framework addresses both data realism and policy robustness in dynamic financial environments.

7 ACKNOWLEDGMENTS

This research is supported by the Joint NTU-WeBank Research Centre on Fintech, Nanyang Technological University, Singapore.

References

[1] Stefano V Albrecht, Jacob W Crandall, and Subramanian Ramamoorthy. 2016. Belief and truth in hypothesised behaviours. *Artificial Intelligence* 235 (2016), 63–94.
[2] Stefano V Albrecht and Subramanian Ramamoorthy. 2015. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. *arXiv preprint arXiv:1506.01170* (2015).
[3] Robert L Axtell and J Doyne Farmer. 2022. Agent-based modeling in economics and finance: Past, present, and future. *Journal of Economic Literature* (2022).
[4] Nicholas Barberis and Andrei Shleifer. 2003. Style investing. *Journal of financial Economics* 68, 2 (2003), 161–199.
[5] Samuel Barrett, Avi Rosenfeld, Sarit Kraus, and Peter Stone. 2017. Making friends on the fly: Cooperating with new teammates. *Artificial Intelligence* 242 (2017), 132–171.
[6] Donghee Choi, Jinkyu Kim, Mogan Gim, Jinho Lee, and Jaewoo Kang. 2024. DeepClair: Utilizing Market Forecasts for Effective Portfolio Selection. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 4414–4422.
[7] Constantinos Daskalakis, Dylan J Foster, and Noah Golowich. 2020. Independent policy gradient methods for competitive reinforcement learning. *Advances in neural information processing systems* 33 (2020), 5527–5540.
[8] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems* 28, 3 (2016), 653–664.
[9] Charles Engel. 2016. Exchange rates, interest rates, and the risk premium. *American Economic Review* 106, 2 (2016), 436–474.
[10] Cristóbal Esteban, Stephanie L Hyland, and Gunnar Rätsch. 2017. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633* (2017).
[11] Drew Fudenberg and Jean Tirole. 1991. Perfect Bayesian equilibrium and sequential equilibrium. *journal of Economic Theory* 53, 2 (1991), 236–260.
[12] Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. 2019. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615* (2019).
[13] Wenbo Guo, Xian Wu, Sui Huang, and Xinyu Xing. 2021. Adversarial policy learning in two-player competitive games. In *International Conference on Machine Learning*. PMLR, 3910–3919.
[14] John C Harsanyi. 1967. Games with incomplete information played by "Bayesian" players, I–III Part I. The basic model. *Management science* 14, 3 (1967), 159–182.
[15] Yan-Lin He, Xing-Yuan Li, Jia-Hui Ma, Shan Lu, and Qun-Xiong Zhu. 2022. A novel virtual sample generation method based on a modified conditional Wasserstein GAN to address the small sample size problem in soft sensing. *Journal of Process Control* 113 (2022), 18–28.
[16] Johannes Heinrich and David Silver. 2016. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121* (2016).
[17] Garud N Iyengar. 2005. Robust dynamic programming. *Mathematics of Operations Research* 30, 2 (2005), 257–280.
[18] Simin Li, Jun Guo, Jingqiao Xiu, Ruixiao Xu, Xin Yu, Jiakai Wang, Aishan Liu, Yaodong Yang, and Xianglong Liu. 2023. Byzantine robust cooperative multi-agent reinforcement learning as a bayesian game. *arXiv preprint arXiv:2305.12872* (2023).
[19] Yujia Li, Kevin Swersky, and Richard Zemel. 2015. Generative moment matching networks. arXiv:1502.02761 [cs.LG]
[20] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2023. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625* (2023).
[21] Daniel J Mankowitz, Nir Levine, Rae Jeong, Yuanyuan Shi, Jackie Kay, Abbas Abdolmaleki, Jost Tobias Springenberg, Timothy Mann, Todd Hester, and Martin Riedmiller. 2019. Robust reinforcement learning for continuous control with model misspecification. *arXiv preprint arXiv:1906.07516* (2019).
[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
[23] Sulaiman D Mohammad, Adnan Hussain, and Adnan Ali. 2009. Impact of macroeconomics variables on stock prices: empirical evidence in case of KSE (Karachi Stock Exchange). *European journal of scientific research* 38, 1 (2009), 96–103.
[24] Hao Ni, Lukasz Szpruch, Magnus Wiese, Shujian Liao, and Baoren Xiao. 2020. Conditional sig-wasserstein gans for time series generation. *arXiv preprint arXiv:2006.05421* (2020).
[25] Arnab Nilim and Laurent El Ghaoui. 2005. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research* 53, 5 (2005), 780–798.
[26] Eleni Nisioti, Daan Bloembergen, and Michael Kaisers. 2021. Robust multi-agent Q-learning in cooperative games with adversaries. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[27] Board of Governors of the Federal Reserve System. 2024. *Dodd-Frank Act Stress Test 2024: Supervisory Stress Test Results.* Technical Report. Federal Reserve Board. https://www.federalreserve.gov/publications/2024-june-dodd-frank-act-stress-test-results.htm Accessed: 2025-01-29.
[28] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. 2017. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632* (2017).
[29] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. 2017. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2817–2826.
[30] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. 2017. Robust adversarial reinforcement learning. In *International conference on machine learning*. PMLR, 2817–2826.
[31] Molei Qin, Shuo Sun, Wentao Zhang, Haochong Xia, Xinrun Wang, and Bo An. 2024. Earnhft: Efficient hierarchical reinforcement learning for high frequency trading. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 14669–14676.
[32] Zijian Shi and John Cartlidge. 2023. Neural Stochastic Agent-Based Limit Order Book Simulation: A Hybrid Methodology. *arXiv preprint arXiv:2303.00080* (2023).
[33] Peter Stone, Gal Kaminka, Sarit Kraus, and Jeffrey Rosenschein. 2010. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 24. 1504–1509.
[34] Shuo Sun, Molei Qin, Wentao Zhang, Haochong Xia, Chuqiao Zong, Jie Ying, Yonggang Xie, Lingxuan Zhao, Xinrun Wang, and Bo An. 2023. TradeMaster: a holistic quantitative trading platform empowered by reinforcement learning. *Advances in Neural Information Processing Systems* 36 (2023), 59047–59061.
[35] Shuo Sun, Wanqi Xue, Rundong Wang, Xu He, Junlei Zhu, Jian Li, and Bo An. 2022. DeepScalper: A risk-aware reinforcement learning framework to capture fleeting intraday trading opportunities. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 1858–1867.
[36] Shuntaro Takahashi, Yu Chen, and Kumiko Tanaka-Ishii. 2019. Modeling financial time-series with generative adversarial networks. *Physica A: Statistical Mechanics and its Applications* 527 (2019), 121261.
[37] Aviv Tamar, Huan Xu, and Shie Mannor. 2013. Scaling up robust MDPs by reinforcement learning. *arXiv preprint arXiv:1306.6189* (2013).
[38] Ke Tang and Wei Xiong. 2012. Index investment and the financialization of commodities. *Financial Analysts Journal* 68, 6 (2012), 54–74.
[39] Chen Tessler, Yonathan Efroni, and Shie Mannor. 2019. Action robust reinforcement learning and applications in continuous control. In *International Conference on Machine Learning*. PMLR, 6215–6224.
[40] Svitlana Vyetrenko, David Byrd, Nick Petosa, Mahmoud Mahfouz, Danial Dervovic, Manuela Veloso, and Tucker Balch. 2020. Get real: Realism metrics for robust limit order book market simulations. In *Proceedings of the First ACM International Conference on AI in Finance*. 1–8.
[41] Jingkang Wang, Yang Liu, and Bo Li. 2020. Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 6202–6209.
[42] Rundong Wang, Hongxin Wei, Bo An, Zhouyan Feng, and Jun Yao. 2021. Commission fee is not enough: A hierarchical reinforced framework for portfolio management. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 626–633.
[43] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. 2013. Robust Markov decision processes. *Mathematics of Operations Research* 38, 1 (2013), 153–183.
[44] Annie Xie, Shagun Sodhani, Chelsea Finn, Joelle Pineau, and Amy Zhang. 2022. Robust policy learning over multiple uncertainty sets. In *International Conference on Machine Learning*. PMLR, 24414–24429.
[45] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. 2019. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems* 32 (2019).
[46] Ying Yu, Bingying Tang, Ronglai Lin, Shufa Han, Tang Tang, and Ming Chen. 2019. CWGAN: Conditional wasserstein generative adversarial nets for fault data generation. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2713–2718.
[47] Huan Zhang, Hongge Chen, Duane Boning, and Cho-Jui Hsieh. 2021. Robust reinforcement learning on state observations with learned optimal adversary. *arXiv preprint arXiv:2101.08452* (2021).
[48] Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Mingyan Liu, Duane Boning, and Cho-Jui Hsieh. 2020. Robust deep reinforcement learning against adversarial perturbations on state observations. *Advances in Neural Information Processing Systems* 33 (2020), 21024–21037.
[49] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 11106–11115.
[50] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*. PMLR, 27268–27286.
[51] Tian Zhu and Wei Zhu. 2022. Quantitative trading through random perturbation Q-network with nonlinear transaction costs. *Stats* 5, 2 (2022), 546–560.

A Wilcoxon Signed-Rank Tests

Table 4: Directional Wilcoxon Signed-Rank Test on Annualized Return Rate (ARR)

Comparison	p-value
Ours vs Buy and Hold	0.00195
Ours vs DQN	0.00195
Ours vs Robust Trading Agent	0.00391
Ours vs Naïve Adversarial	0.00586
Ours vs RoM-Q	0.00195
Ours vs RARL	0.00977
Ours vs DeepScalper	0.00195
Ours vs EarnHFT	0.00195
Ours vs CDQN-rp	0.00195
Ours vs w/o adv agent	0.00195
Ours vs IPG	0.00586

Table 5: Directional Wilcoxon Signed-Rank Test on Sharpe Ratio (SR)

Comparison	p-value
Ours vs Buy and Hold	0.00195
Ours vs DQN	0.00195
Ours vs Robust Trading Agent	0.01758
Ours vs Naïve Adversarial	0.01367
Ours vs RoM-Q	0.00977
Ours vs RARL	0.00195
Ours vs DeepScalper	0.00195
Ours vs EarnHFT	0.00195
Ours vs CDQN-rp	0.00195
Ours vs w/o adv agent	0.00195
Ours vs IPG	0.00586

Table 6: Directional Wilcoxon Signed-Rank Test on Maximum Drawdown (MDD)

Comparison	p-value
Ours vs Buy and Hold	0.00195
Ours vs DQN	0.00977
Ours vs Robust Trading Agent	0.00391
Ours vs Naïve Adversarial	0.00391
Ours vs RoM-Q	0.00195
Ours vs RARL	0.00195
Ours vs DeepScalper	0.00391
Ours vs EarnHFT	0.00195
Ours vs CDQN-rp	0.00391
Ours vs w/o adv agent	0.00391
Ours vs IPG	0.00586

The Wilcoxon signed-rank test on ARR and SR shows that our approach yields significantly better values than each baseline, while for MDD, our method achieves significantly lower drawdowns.

B Algorithms

B.1 NFSP with Adversarial Observations and Quantile Belief

Algorithm 1: NFSP with Adversarial Observations and Quantile Belief

Input: *total_num_step*, initial networks: `network1`, `network2`, `network3`
Output: Trained NFSP agent, belief network, and adversarial agent.
Initialize:
  $agent \leftarrow agent\_nfsp \leftarrow$ network1
  $belief\_network \leftarrow$ network2
  $adv\_agent \leftarrow$ network3
  adv_buffer, nfsp_buffer, buffer $\leftarrow$ [], [], []
for $i \leftarrow 1$ **to** *total_num_step* **do**
  $obs \leftarrow adv\_agent(obs) + obs$
  `// adversarial modification of observation`
  $belief \leftarrow belief\_network(obs)$
  `// quantile belief from belief network`
  $use\_avg\_policy \leftarrow$ (random().rand() $> \tau$)
  if *use_avg_policy* **then**
    $action \leftarrow agent\_nfsp(obs, belief)$
  else
    $action \leftarrow agent(obs, belief)$
  end
  $(obs, reward) \leftarrow env$.step$(action)$
  buffer.insert$\big((obs, action, reward, next_obs)\big)$
  adv_buffer.insert$\big((obs, adv_agent(obs), reward)\big)$
  if $\neg use\_avg\_policy$ **then** `// Only store in NFSP buffer if not average-policy`
    nfsp_buffer.insert$\big((obs, action)\big)$
  end
  `// Update each component`
  update_adv_agent(adv_agent, adv_buffer)
  `// policy gradient for adversarial obs generator`
  update_nfsp_agent($agent\_nfsp$, $nfsp\_buffer$)
  `// MSE loss for the average-policy branch`
  update_quantile_belief($belief\_network$, $buffer$)
  `// quantile regression for belief network`
  update_agent($agent$, $buffer$)
  `// DQN update for main Q-function`
end

Algorithm 1 outlines the training procedure for our robust trading framework, which combines Bayesian NFSP with adversarial observation perturbations and quantile-based belief modeling. The agent interacts with a perturbed environment where an adversarial agent modifies the observations. A belief network estimates quantile-based market beliefs, which are fed into the NFSP agent to guide action selection. The agent alternates between using its best-response policy and an average-policy branch, controlled by a mixing parameter τ. Transitions are stored in dedicated buffers to update the adversarial agent, NFSP policy, belief network, and Q-function separately.

B.2 Bayesian Neural Fictitious Self-Play

Algorithm 2: Bayesian Neural Fictitious Self-Play

Input: Q function of trading agent $Q(s_t, a_t, b_t)$, time-averaged policy $\overline{\pi}$, trained market simulator $G(x_t|\mathbf{m}_{t,t-L}, \mathbf{n}, \mathbf{x}_{t-L})$, circular buffer $\mathcal{M}_{RL}$ and reservoir buffer for time-averaged policy $\mathcal{M}_{SL}$.
Output: Robust trading policy $Q(s_t, a_t, b_t)$.

Initialize Q function of trading agent, initialize the network of time-averaged policy $\overline{\pi}$, circular buffer $\mathcal{M}_{RL}$, reservoir buffer $\mathcal{M}_{SL}$, and belief b_0;
for *each training iteration* **do**
  for *each episode* **do**
    for *each timestep t* **do**
      Adversary update macroeconomic factors $\mathbf{M}_{t-L}^{\alpha,*}$, environment proceeds via G ;
      Trading agent updates belief b_t. Samples an action from $Q(s_t, a_t, b_t)$ with probability η with epsilon greedy exploration, sample an action from $\overline{\pi}$ with probability $1 - \eta$;
    Store transition in circular buffer $\mathcal{M}_{RL}$;
    if *Action is sampled from* $Q(s_t, a_t, b_t)$ **then**
      Store transition in circular buffer $\mathcal{M}_{RL}$;
  Update value function via TD loss:

$$\mathcal{L}_{TD} = (Q_\pi(s_t, a_t, b_t) - r_t - \gamma Q_\pi(s_{t+1}, a_{t+1}, b_{t+1}))^2$$

  Update time-averaged trading policy $\overline{\pi}$ via supervised learning ;
return Optimized trading policy π^θ;

Algorithm 2 presents the training procedure of our proposed Bayesian Neural Fictitious Self-Play (BNFSP) framework. This method extends traditional NFSP by incorporating a belief modeling component and adversarial market simulation. During training, the agent interacts with a market simulator G perturbed by adversarial macroeconomic factors, and makes decisions based on both its Q-function and a time-averaged policy. A quantile-based belief b_t is updated at each step to capture latent market states. Transitions are stored in two separate replay buffers: a circular buffer for reinforcement learning updates and a reservoir buffer for supervised learning of the average policy. The agent's Q-function is optimized via temporal-difference learning, while the average policy is updated through supervised learning. This joint training improves the robustness and adaptability of the trading agent in dynamic and uncertain environments.

B.3 Correlation-Weighted Imputation

To handle missing values in cross-sectional time-series data, we propose a correlation-weighted imputation method that leverages

Algorithm 3: Correlation-Weighted Imputation

Input: Ticker set $T = \{t_1, \ldots, t_N\}$, feature datasets $\{D_t\}_{t \in T}$, correlation matrices $\{C^{(f)}\}_{f \in \mathcal{F}}$.

Output: Imputed datasets $\{D_t\}_{t \in T}$.

For each feature $f \in \mathcal{F}$, process all tickers $t \in T$. For a given ticker t, every time index i where $D_t(i, f)$ is missing;

Identify the set of valid tickers:

$$V = \{t' \in T \setminus \{t\} : D_{t'}(i, f) \text{ is available}\}$$

if $V \neq \emptyset$ **then**

Compute weights $w_{t'} = \exp(C^{(f)}(t, t'))$ for all $t' \in V$;

Normalize weights:

$$\tilde{w}_{t'} = \frac{w_{t'}}{\sum_{s \in V} w_s}$$

Impute missing value:

$$D_t(i, f) = \sum_{t' \in V} \tilde{w}_{t'} \cdot D_{t'}(i, f)$$

return Updated datasets $\{D_t\}_{t \in T}$;

the structural similarity across assets. As detailed in Algorithm 3, the method imputes missing feature values for each ticker by taking a weighted average of the corresponding values from other tickers, where the weights are derived from an exponential transformation of their historical correlation. This approach ensures that more correlated tickers contribute more significantly to the imputed value, preserving consistency across assets while mitigating the noise introduced by unrelated instruments.

B.4 t-SNE Plot Generation

Algorithm 4: t-SNE Plot Generation

Input: Raw OHLCV time series data

Parameters: Window size $w = 21$, t-SNE output dimension $d = 1$, perplexity = 50, iterations = 3000

Step 1 Compute derived features from raw OHLCV, such as:
- Rolling returns
- Moving averages
- Standard deviations
- Volume-based indicators

Result: feature matrix F

Step 2 Construct sliding windows:
- For each time $t = w$ to T:
  - Flatten the window $F_{t-w+1:t}$ into a vector X_t
  - Assign corresponding target value Y_t

Result: windowed feature vectors X, aligned targets Y

Step 3 Apply t-SNE separately:
- Compute $Z_x \leftarrow$ t-SNE$(X, d = 1)$
- Compute $Z_y \leftarrow$ t-SNE$(Y, d = 1)$

Result: 1D embeddings for features and targets

Step 4 Plot:
- Use Z_x as x-axis and Z_y as y-axis
- Color each point according to chronological order (e.g., train vs. test)

To visualize the distributional shift between training and testing states, we construct a 1D embedding of both state features and reward targets using t-SNE. As described in Algorithm 4, we first extract meaningful technical features from raw OHLCV data and organize them into overlapping windows. Each window is flattened into a vector representation, aligned with its corresponding future return. We then apply t-SNE separately to the feature and target spaces to obtain low-dimensional embeddings that preserve local structure. By plotting these embeddings against each other and coloring by temporal phase, we can effectively illustrate the dynamics shift faced by the agent between training and testing periods.

B.5 Feature Selection

Algorithm 5: Feature Selection Procedure

Input:

Raw feature set $\mathcal{F} = \{f_1, f_2, ..., f_n\}$

Future return series $\mathbf{r}_{t+\Delta}$

Correlation threshold τ_{corr}

Redundancy threshold τ_{red}

Output:

Selected feature set $\mathcal{F}_{\text{selected}}$

Procedure:

(1) Initialize $\mathcal{F}_{\text{candidate}} \leftarrow \emptyset$

(2) For each feature f_i in $\mathcal{F}$:
- Compute Pearson correlation $\rho_i = \text{corr}(f_i, \mathbf{r}_{t+\Delta})$
- If $|\rho_i| > \tau_{\text{corr}}$, add f_i to $\mathcal{F}_{\text{candidate}}$

(3) Initialize $\mathcal{F}_{\text{selected}} \leftarrow \emptyset$

(4) For each f_i in $\mathcal{F}_{\text{candidate}}$:
- Compute correlation $\rho_{ij} = \text{corr}(f_i, f_j)$ for all $f_j \in \mathcal{F}_{\text{selected}}$
- If $\max_j |\rho_{ij}| < \tau_{\text{red}}$, add f_i to $\mathcal{F}_{\text{selected}}$

(5) Return $\mathcal{F}_{\text{selected}}$

To ensure that the input features are both informative and non-redundant, we employ a two-stage correlation-based feature selection procedure, as detailed in Algorithm 5. In the first stage, we compute the Pearson correlation between each candidate feature and the target future return, retaining only those with correlation magnitude above a threshold τ_{corr}. In the second stage, we iteratively filter out redundant features by enforcing a maximum pairwise correlation constraint τ_{red} with already selected features. This procedure results in a compact feature set that preserves predictive relevance while mitigating multicollinearity.