Stochastic Adaptive Process State Space Model for Portfolio Management Based on Deep Reinforcement Learning

Fengchen Gu[1], Xiaotian Ren[1], Zhengyong Jiang[1], Ángel F. García-Fernández[2], Jionglong Su[1*], Huakang Li[1*]
1 *School of AI and Advanced Computing, XJTLU Entrepreneur College (Taicang)*
Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu, China
2 *IPTC, ETSI de Telecomunicación, Universidad Politécnica de Madrid*, Madrid, Spain
[*]Corresponding author(s). E-mail(s): Jionglong.Su@xjtlu.edu.cn; Huakang.Li@xjtlu.edu.cn

***Abstract*—Deep reinforcement learning (DRL) for financial portfolio management is challenged by the market's inherent volatility and non-stationary dynamics, which are often inadequately captured by models that treat financial data as generic sequences. To address this, we introduce a novel DRL framework, the Stochastic Adaptive Process State Space Model (SAP-SSM), that provides the crucial financial inductive bias lacking in prior work. The SAP-SSM explicitly models financial time series as a Hawkes jump-diffusion process, natively capturing volatility clustering and fat-tailed shocks. In contrast to prevailing models with static, deterministic mechanisms, its internal architecture features a dynamic state transition matrix that adapts to market regimes, a hierarchical state to differentiate between noise and trends, and stochastic transitions to represent market uncertainty. Evaluated on a portfolio of 30 Dow Jones Industrial Average stocks, our framework demonstrates competitive performance, achieving a cumulative return of 45.3% and a Sharpe ratio of 1.878.**

***Index Terms*—Portfolio Management, Deep Reinforcement Learning, State Space Model, Stochastic Processes, Financial Time Series**

I. INTRODUCTION

Portfolio management, the strategic allocation of capital across financial assets, is a cornerstone of modern finance that aims to optimize the risk-return trade-off [1]. This task is fundamentally challenging due to the inherent characteristics of financial markets: they are highly volatile, non-stationary, and influenced by a complex interplay of economic, political, and behavioral factors [2]. The resulting data are high-dimensional and notoriously noisy, often exhibiting well-documented stylized facts such as fat-tailed return distributions and volatility clustering, which render traditional investment methods that rely on restrictive statistical assumptions inadequate [3], [4].

Deep Reinforcement Learning (DRL) has emerged as a powerful paradigm. DRL synergizes the hierarchical feature extraction of deep learning [5] with the goal-oriented, sequential decision-making framework of reinforcement learning [6]–[8]. This combination allows an autonomous agent to learn sophisticated control policies directly from complex market data through a process of trial and error. By optimizing for a cumulative reward signal, DRL is uniquely suited to developing automated and adaptive trading strategies that can operate effectively in dynamic and uncertain environments [9], [10].

The application of DRL to portfolio management has yielded a series of promising results, demonstrating its potential to outperform traditional benchmarks. Early frameworks such as the Ensemble of Identical Independent Evaluators (EIIE) established the viability of DRL for the highly volatile cryptocurrency markets [11], while models like Investor-Imitator (IMIT) explored knowledge extraction by mimicking human investor behavior [12]. The development of comprehensive platforms, most notably FinRL [3] and TradeMaster [13], has since standardized the research pipeline and facilitated the application of various algorithms. These platforms have enabled the creation of more sophisticated ensemble strategies using algorithms like PPO [14], [15] to enhance robustness in diverse market conditions.

Despite recent advances, modeling the underlying stochastic processes of markets still faces three critical limitations. First, existing models lack financial inductive bias, as advanced architectures like Transformers [9], [16], [17] treat financial data as generic sequences. Even when hybrid models incorporate features like volatility clustering, their ad-hoc approach fails to unify market shocks, leading to policies that misprice risk. Second, they use static internal mechanisms. Despite attention's dynamic weighting, core transition functions use static parameters, preventing strategies from adapting to new market regimes and causing performance to degrade. Third, most frameworks, including multi-agent systems [8], [18], [19], use deterministic state transitions that ignore market randomness. Even Bayesian methods often fail to model the process's stochastic nature, resulting in overconfident, brittle policies that fail during unexpected events.

To address these limitations, we propose a novel DRL framework centered on the Stochastic Adaptive Process State Space Model (SAP-SSM) inspired by the Mamba architecture [20], [21]. Its key contributions are threefold. First, we introduce a novel financial process engine, the first in DRL to explicitly model financial dynamics as a Hawkes jump-diffusion process, natively capturing drift, volatility, and self-exciting shocks for realistic risk assessment. Second, our unique adaptive state dynamics architecture combines a dynamic state transition matrix with a hierarchical, gated state, allowing the model's logic to adapt to market regimes while filtering noise to preserve trends. Finally, unlike deterministic models, we embed inherent stochastic transitions directly into the state transition function. Modeling state evolution as a random process enables the agent to handle market uncertainty and foster more robust policies.

II. DRL Environment

To train an agent for dynamic portfolio management, we model the problem within a DRL framework, where an agent learns an optimal strategy through iterative interaction with a simulated market [6], [22]. This section details the construction of this simulation as a Markov Decision Process (MDP) and the specialized learning algorithm used.

A. Markov Decision Process Formulation

The agent-environment interaction is structured as an MDP [23], defined by a state-action-reward triplet $(\mathcal{S}, \mathcal{A}, \mathcal{R})$. The simulation operates on a daily timestep, assuming frictionless markets with a fixed proportional transaction cost, c.

State: The state, $s_t \in \mathcal{S}$, provides a comprehensive snapshot of market and portfolio information, defined as a concatenation of available cash (a_{t-1}), current share holdings ($\mathbf{h}_{t-1}$), and current asset prices ($\mathbf{p}_t$):

$$s_t = [a_{t-1}, \mathbf{h}_{t-1}, \mathbf{p}_t]. \tag{1}$$

Action: The agent outputs a continuous action vector $\mathbf{a}_t \in [-1, 1]^n$, representing a trading intention. This is translated into a discrete number of shares to trade, subject to cash and holding constraints.

Reward: The reward, r_t, is the change in total portfolio equity ($\mathbf{p}_t - \mathbf{p}_{t-1}$), directly reflecting capital gains while penalizing transaction costs $c = 0.1\%$ based on the total value of assets transacted, τ_t:

$$r_t = \mathbf{h}_{t-1}^T(\mathbf{p}_t - \mathbf{p}_{t-1}) - c \cdot \tau_t. \tag{2}$$

B. DRL with Risk-Aware Preference Learning

We train the agent using a specialized variant of Proximal Policy Optimization (PPO) [14], termed PPO with Risk-Aware Preference learning (PPO-RAP). The core innovation is a modified policy objective that replaces the standard advantage with a risk-aware, preference-based formulation, A_t^{RA}. This is computed by first determining the desirability of a state transition using the critic's value estimates, $\text{pref}(s_t) = \sigma(V_\phi(s_{t+1}) - V_\phi(s_t))$, and then penalizing this score by the volatility of rewards to favor stable, high-quality transitions.

The final policy objective leverages this risk-adjusted advantage within the standard PPO clipped surrogate loss structure:

$$\mathcal{L}_t^{\text{Policy}} = -\mathbb{E}_t\left[\min\left(\rho_t(\theta)A_t^{\text{RA}}, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)A_t^{\text{RA}}\right)\right], \tag{3}$$

where $\rho_t(\theta)$ is the probability ratio. The full objective function also includes standard terms for value function loss and an entropy bonus to encourage exploration, guiding the agent towards stable, risk-aware decision-making.

III. The SAP-SSM Framework

Our proposed framework (in Figure 1) is a deep recurrent network for reinforcement learning agents, centered on a novel recurrent engine: the Stochastic Adaptive Process State Space Model (SAP-SSM). The SAP-SSM is engineered to interpret and forecast financial time series by explicitly modeling their underlying stochastic and adaptive nature [24]. This section details its internal architecture, covering its foundational multi-head scan with a dynamic state matrix, its integration of a Hawkes jump-diffusion process, and its advanced hierarchical and stochastic state dynamics. The complete forward pass and its core state transition logic are summarized in Algorithm 1 and Algorithm 2, respectively.

A. Core Engine: Multi-Head Scan with Dynamic State

The SAP-SSM is fundamentally a State Space Model (SSM) that evolves a latent state vector $\boldsymbol{h} \in \mathbb{R}^{D_S}$ through time. Inspired by the Mamba architecture, its parameters are input-dependent, but we introduce several significant modifications.

Multi-Head State Scan: To capture the multifaceted nature of financial time series, the internal computations are partitioned into N_h parallel heads. The internal feature dimension, D_{inner}, is divided such that each head has a dimension of $D_{head} = D_{inner}/N_h$. Each head maintains its own independent state vector $\boldsymbol{h}^{(i)} \in \mathbb{R}^{D_{head} \times D_S}$ and performs a separate state space scan. This architecture allows different heads to specialize in learning distinct temporal patterns, such as short-term momentum, long-term mean reversion, or volatility clustering, which are then aggregated at the output.

Dynamic State Transition Matrix (A): A primary innovation of our model is that the state transition matrix $\boldsymbol{A}$, which governs the system's fundamental dynamics, is not a static, learned parameter. Instead, it is dynamically generated at each time step t as a function of the input features $\boldsymbol{x}_t$. This is achieved using a dedicated projection: a linear layer with unique learnable weights (θ_A) that maps the input features $\boldsymbol{x}_t$ directly to the parameters of matrix $\boldsymbol{A}_t$. This mapping is formally expressed as:

$$\boldsymbol{A}_t = f_A(\boldsymbol{x}_t; \theta_A). \tag{4}$$

This mechanism allows the model to adapt the rules of its internal state evolution based on observed market conditions. For instance, it can learn different state transition dynamics for high-volatility and low-volatility regimes, enabling a far more flexible and realistic representation of the market.

B. Process Modeling with Hawkes Jump-Diffusion

To capture the complex dynamics of financial markets, the output of the SAP-SSM, $\boldsymbol{y}_t$, is modeled as a synthesis of three distinct stochastic processes. This decomposition allows the model to individually address different stylized facts of financial returns:

$$\boldsymbol{y}_t = \boldsymbol{y}_{\text{drift},t} + \boldsymbol{y}_{\text{diff},t} + \boldsymbol{y}_{\text{jump},t}. \tag{5}$$

The total output is the sum of an underlying trend ($\boldsymbol{y}_{\text{drift},t}$), a continuous volatility term ($\boldsymbol{y}_{\text{diff},t}$), and a component for

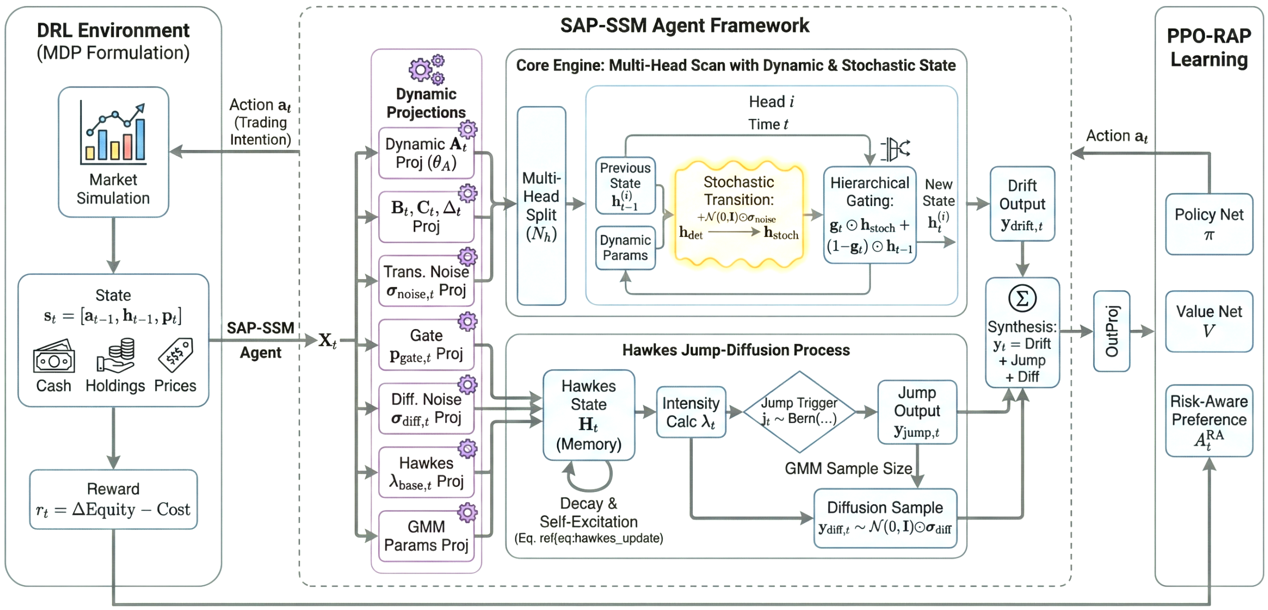

Fig. 1. The overall architecture of the SAP-SSM Agent Framework.

sudden, large shocks ($\boldsymbol{y}_{\text{jump},t}$). Each component is designed to model a specific market behavior.

Drift and Diffusion: The **Drift** component, $\boldsymbol{y}_{\text{drift},t}$, represents the underlying trend of the asset and is modeled by the main output of the state space scan. The **Diffusion** component, $\boldsymbol{y}_{\text{diff},t}$, is a continuous stochastic term that captures the regular, high-frequency volatility. Its magnitude, $\boldsymbol{\sigma}_{\text{diff},t}$, is determined by a dedicated projection from the input features $\boldsymbol{x}_t$.

Hawkes Jump-Diffusion Process: To model sudden, large price movements and their clustering behavior, we implement a sophisticated jump component governed by a Hawkes process.

1) **Volatility-Correlated Intensity:** The baseline jump intensity, $\boldsymbol{\lambda}_{\text{base},t}$, is conditioned on both the input features $\boldsymbol{x}_t$ and the magnitude of the diffusion component. This novel feature explicitly forges a link between jump risk and market volatility.
2) **Self-Excitation (Clustering):** The total jump intensity $\boldsymbol{\lambda}_t$ is the sum of the baseline intensity and a memory term $\boldsymbol{H}_t$ that decays over time but is excited by past jumps, modeling the aftershock phenomenon. The update rule for this Hawkes state is:
$$\boldsymbol{H}_t = \boldsymbol{H}_{t-1} \cdot \sigma(\boldsymbol{\theta}_{\text{decay}}) + \boldsymbol{j}_{t-1} \odot \boldsymbol{\theta}_{\text{excitement}}, \tag{6}$$
where δ is a learned decay factor, $\boldsymbol{j}_{t-1}$ is a binary indicator of a jump at the previous step, and α is a learned excitement factor.
3) **Mixture Density Network for Jumps:** To capture the fat-tailed and skewed nature of real-world jumps, the size of each jump is sampled from a Gaussian Mixture Model (GMM). The parameters of the GMM (means $\boldsymbol{\mu}_{k,t}$, standard deviations $\boldsymbol{\sigma}_{k,t}$, and mixture weights $\boldsymbol{w}_{k,t}$) are dynamically predicted by a mixture projection layer:
$$\boldsymbol{w}_{k,t}, \boldsymbol{\mu}_{k,t}, \boldsymbol{\sigma}_{k,t} = g(\boldsymbol{W}_{\text{mix}}\boldsymbol{x}_t + \boldsymbol{b}_{\text{mix}}), \tag{7}$$
where g splits the linear layer's output and applies activations ($Softmax, Softplus$) to produce the final GMM parameters.

C. Advanced State Dynamics

The evolution of the internal state vector $\boldsymbol{h}$ is further enhanced with two mechanisms that operate within the *SelectiveScanStep* function, as detailed in Algorithm 2.

Hierarchical State with Learned Gating: The state vector $\boldsymbol{h}$ within each head is partitioned into N_s scales (e.g., fast, medium, slow), each with a dimension of $D_{state\_per\_scale}$. A learned gating mechanism, implemented by an update gate projection layer, predicts an update probability $\boldsymbol{p}_{\text{gate},t}$ for each scale at each time step. A binary gate vector $\boldsymbol{g}_t$ is then sampled from a Bernoulli distribution with this probability. The state for a given scale is updated based on this gate, allowing the model to learn to preserve long-term information in the slow states while rapidly updating the fast states to react to short-term noise.

Stochastic State Transitions: To account for the inherent unpredictability of financial markets, we model the state transition itself as a stochastic process. The next state $\boldsymbol{h}_t$ is not a deterministic function. Instead, we introduce a learned noise term. The state update is a three-step process:
$$\boldsymbol{h}_{\text{det}} = \bar{\boldsymbol{A}}_t\boldsymbol{h}_{t-1} + \bar{\boldsymbol{B}}_t\boldsymbol{u}_t, \quad \boldsymbol{h}_{\text{stoch}} = \boldsymbol{h}_{\text{det}} + \boldsymbol{\sigma}_{\text{noise},t} \odot \mathcal{N}(0, \mathbf{I}), \tag{8}$$
$$\boldsymbol{h}_t = \boldsymbol{g}_t \odot \boldsymbol{h}_{\text{stoch}} + (1 - \boldsymbol{g}_t) \odot \boldsymbol{h}_{t-1}, \tag{9}$$
where $\bar{\boldsymbol{A}}_t$ and $\bar{\boldsymbol{B}}_t$ are the discretized state matrices. The covariance matrix Σ_t of the transition noise is parameterized by its standard deviation, which is predicted by a transition noise projection layer as a function of the input $\boldsymbol{x}_t$. This allows the model to express uncertainty in its own state evolution, leading to more robust and realistic dynamics.

TABLE I

PERFORMANCE METRICS OF DIFFERENT TRADING STRATEGIES. FOR ALL METRICS, HIGHER VALUES ARE BETTER. THE HIGHEST VALUE IS IN BOLD, AND THE SECOND HIGHEST IS UNDERLINED.

Category	Strategy	Cum. return	Stability	Sharpe ratio	Calmar ratio	Omega ratio	Sortino ratio	Tail ratio
Our Study (with Ablations)	SAP-SSM	**0.453**	0.887	**1.878**	**3.979**	**1.393**	**2.974**	**1.207**
	SAP-SSM_M	<u>0.279</u>	<u>0.913</u>	<u>1.414</u>	<u>3.008</u>	<u>1.271</u>	<u>2.140</u>	0.972
	SAP-SSM_F	0.151	0.684	0.778	1.876	1.148	1.109	0.800
Deep Reinforcement Learning Methods	IMIT	-0.613	**0.920**	-1.818	-0.945	0.715	-2.402	0.948
	Finrl	0.112	0.496	0.700	1.374	1.137	1.001	0.907
	ES	0.087	0.609	0.530	1.190	1.097	0.802	<u>1.138</u>
	TMP	0.073	0.562	0.440	1.108	1.076	0.622	0.997
	SARL	0.046	0.449	0.193	0.755	1.032	0.275	1.069
Traditional Statistical Methods	Best	-0.571	0.838	-1.695	-0.914	0.700	-2.188	0.951
	BNN	-0.394	0.783	-1.850	-0.906	0.719	-2.435	0.835
	CORN	-0.525	0.873	-3.164	-0.945	0.554	-3.674	0.651
	CWMR	-0.144	0.000	-0.379	-0.512	0.932	-0.546	0.881
	OLMAR	-0.021	0.241	-0.273	-0.144	0.953	-0.363	0.980
	PAMR	0.036	0.513	0.108	0.422	1.018	0.145	0.901
	RMR	0.078	0.719	0.465	0.879	1.082	0.642	0.993
	TCO	-0.193	0.696	-0.844	-0.662	0.856	-1.152	0.908
	WMAMR	0.062	0.691	0.332	0.703	1.057	0.454	1.040

IV. EXPERIMENTS

A. Experimental Setup

We evaluate the proposed SAP-SSM framework on a portfolio consisting of 30 constituent stocks of the Dow Jones Industrial Average (DJIA). The dataset is strictly partitioned into a training period (2021–2023) and a testing period (2024). This separation ensures that the model is evaluated on unseen market regimes, testing its generalization capabilities.

B. Baselines and Comparison Strategies

To rigorously validate our approach, we benchmark SAP-SSM against a comprehensive set of strategies, categorized into traditional mathematical models and state-of-the-art DRL frameworks.

a) Traditional Mathematical Strategies: These methods rely on convex optimization and statistical assumptions about market physics:

- **Mean Reversion Strategies:** These algorithms, including CWMR [25], OLMAR [26], PAMR [27], RMR [28], TCO [29], and WMAMR [30], assume that prices will eventually revert to their historical averages. They exploit short-term price fluctuations but often struggle during strong sustained trends.
- **Trend Following & Momentum:** Strategies such as BEST [11], BNN [31], and CORN [32] utilize pattern matching and correlation maximization to identify and ride market trends.

b) Deep Reinforcement Learning Baselines: We include EIIE [11], which uses direct policy evolution; IMIT [12], which mimics high-value investors; and widely used frameworks like FinRL [3], ES [15], SARL [33], and the PPO variant from TradeMaster (TMP) [13].

C. Evaluation Metrics

We employ a holistic set of metrics to evaluate both profitability and risk management:

- **Cumulative Return:** The total percentage growth of the portfolio equity over the test period.
- **Stability:** Measured as the R^2 of the log-equity curve, indicating the consistency of returns.
- **Sharpe Ratio:** The ratio of excess return to the standard deviation of returns, measuring risk-adjusted performance.
- **Calmar Ratio:** The ratio of annualized return to the maximum drawdown, highlighting the model's recovery ability after losses.
- **Omega Ratio:** The probability-weighted ratio of gains versus losses above a minimum threshold.
- **Sortino Ratio:** Similar to the Sharpe ratio but only penalizes downside deviation, differentiating harmful volatility from profitable volatility.
- **Tail Ratio:** The ratio of the 95th percentile return to the absolute value of the 5th percentile loss, measuring the capability to generate outsized gains relative to extreme losses.

D. Performance Analysis

The quantitative results are summarized in Table I, with the portfolio value evolution visualized in Fig. 2. The SAP-SSM framework exhibits dominant performance across all key metrics.

Profitability and Alpha Generation: SAP-SSM achieves a remarkable Cumulative Return of **45.3%**, significantly outperforming the best DRL baseline (FinRL, 11.2%) and far surpassing traditional methods, most of which yielded negative returns. This suggests that while traditional mean-reversion and momentum strategies failed to adapt to the non-stationary

Algorithm 1 SAP-SSM Forward Pass

Input: Input sequence $\boldsymbol{X} \in \mathbb{R}^{B \times T \times D_{model}}$
Parameters: All learnable weights θ
Initialize: Hidden state $\boldsymbol{h}_0 = \boldsymbol{0}$, Hawkes state $\boldsymbol{H}_0 = \boldsymbol{0}$
function SAP-SSM-FORWARD($\boldsymbol{X}$)
  Project $\boldsymbol{X}_t$ to get all dynamic parameters:
    $\{\boldsymbol{u}_t, \boldsymbol{z}_t, \boldsymbol{A}_{\text{log},t}, \boldsymbol{B}_t, \boldsymbol{C}_t, \boldsymbol{\lambda}_{\text{base},t}, \boldsymbol{\sigma}_{\text{diff},t}, \boldsymbol{\sigma}_{\text{noise},t}, \dots\}$
  $\boldsymbol{h} \leftarrow \boldsymbol{h}_0$, $\boldsymbol{H} \leftarrow \boldsymbol{H}_0$
  for $t = 1, \dots, T$ **do**
    Hawkes Process: Decay and Intensity Calculation
    Apply decay: $\boldsymbol{H}_t \leftarrow \boldsymbol{H}_{t-1} \cdot \sigma(\boldsymbol{\theta}_{\text{decay}})$
    Calculate total jump intensity:
    $\boldsymbol{\lambda}_t \leftarrow \text{Softplus}(\boldsymbol{\lambda}_{\text{base},t} + f(\boldsymbol{\sigma}_{\text{diff},t}) + \boldsymbol{H}_t)$
    Stochastic Component Sampling
    Sample jump trigger $\boldsymbol{j}_t \sim \text{Bernoulli}(1 - e^{-\boldsymbol{\lambda}_t \Delta t})$
    Sample jump size from GMM and get
    $\boldsymbol{y}_{\text{jump},t} \leftarrow \boldsymbol{j}_t \odot \text{GMM}(\dots)$
    Sample diffusion: $\boldsymbol{y}_{\text{diff},t} \leftarrow \text{SiLU}(\boldsymbol{\sigma}_{\text{diff},t}) \odot \mathcal{N}(0, \mathbf{I})$
    Hawkes Process: Self-Excitation
    Apply self-excitation as per Eq. (6):
    $\boldsymbol{H}_t \leftarrow \boldsymbol{H}_t + \boldsymbol{j}_t \odot \boldsymbol{\theta}_{\text{excitement}}$
    Gated Drift Process
    Predict hierarchical state gate $\boldsymbol{g}_t \sim \text{Bernoulli}(\boldsymbol{p}_{\text{gate},t})$
    Form parameter set
    $\boldsymbol{\phi}_t \leftarrow \{\boldsymbol{u}_t, \boldsymbol{A}_{\text{log},t}, \boldsymbol{B}_t, \boldsymbol{C}_t, \boldsymbol{\sigma}_{\text{noise},t}, \boldsymbol{g}_t\}$
    Evolve state and calculate drift via scan function:
    $\boldsymbol{h}, \boldsymbol{y}_{\text{drift},t} \leftarrow \text{SelectiveScanStep}(\boldsymbol{h}, \boldsymbol{\phi}_t)$ (Alg. 2)
    Synthesize Final Output
    Combine components as per Eq. (5):
    $\boldsymbol{y}_t \leftarrow (\boldsymbol{y}_{\text{drift},t} \odot \text{SiLU}(\boldsymbol{z}_t)) + \boldsymbol{y}_{\text{diff},t} + \boldsymbol{y}_{\text{jump},t}$
  end for
  return Final projected output $\text{OutProj}(\boldsymbol{Y})$
end function

Algorithm 2 Selective Scan Step with Stochastic State

Input: Previous state $\boldsymbol{h}_{\text{old}}$; current parameter set $\boldsymbol{\phi}_t$
function SELECTIVESCANSTEP($\boldsymbol{h}_{\text{old}}, \boldsymbol{\phi}_t$)
  Unpack $\{\boldsymbol{u}_t, \boldsymbol{A}_{\text{log},t}, \boldsymbol{B}_t, \boldsymbol{C}_t, \boldsymbol{\sigma}_{\text{noise},t}, \boldsymbol{g}_t\}$ from $\boldsymbol{\phi}_t$
  Discretize SSM parameters
  Calculate step size: $\Delta_t \leftarrow \text{Softplus}(\text{Project}_\Delta(\boldsymbol{u}_t))$
  Materialize dynamic state matrix: $\boldsymbol{A}_t \leftarrow -\exp(\boldsymbol{A}_{\text{log},t})$
  Discretize A via Zero-Order Hold: $\bar{\boldsymbol{A}}_t \leftarrow \exp(\Delta_t \boldsymbol{A}_t)$
  Discretize B via Zero-Order Hold: $\bar{\boldsymbol{B}}_t \leftarrow \Delta_t \boldsymbol{B}_t$
  Stochastic and Gated State Transition
  Compute deterministic and stochastic states per Eq. (8):
    $\boldsymbol{h}_{\text{det}} \leftarrow \bar{\boldsymbol{A}}_t \boldsymbol{h}_{\text{old}} + \bar{\boldsymbol{B}}_t \boldsymbol{u}_t$
    $\boldsymbol{h}_{\text{stoch}} \leftarrow \boldsymbol{h}_{\text{det}} + \boldsymbol{\sigma}_{\text{noise},t} \odot \mathcal{N}(0, \mathbf{I})$
  Update state with hierarchical gating per Eq. (9):
    $\boldsymbol{h}_{\text{new}} \leftarrow \boldsymbol{g}_t \odot \boldsymbol{h}_{\text{stoch}} + (1 - \boldsymbol{g}_t) \odot \boldsymbol{h}_{\text{old}}$
  Compute Drift Output
  $\boldsymbol{y}_{\text{drift}} \leftarrow \boldsymbol{C}_t \otimes \boldsymbol{h}_{\text{new}} + \boldsymbol{D}\boldsymbol{u}_t$
  return $\boldsymbol{h}_{\text{new}}, \boldsymbol{y}_{\text{drift}}$
end function

market dynamics of 2024, our stochastic state-space approach successfully captured the underlying market drift.

Risk-Adjusted Returns: Crucially, our model does not achieve high returns by taking excessive risks. It achieves the highest **Sharpe Ratio (1.878)** and **Sortino Ratio (2.974)**. The large gap between the Sortino and Sharpe ratios indicates that the volatility in our portfolio is predominantly on the upside (profit-generating) rather than the downside. In contrast, widely cited baselines like IMIT and traditional methods like CORN suffered from poor risk management, resulting in negative Sharpe ratios.

Drawdown Management and Resilience: The **Calmar Ratio** of **3.979** is nearly triple that of the nearest competitor (FinRL, 1.374). This metric, combined with a high **Tail Ratio (1.207)**, confirms that the SAP-SSM is exceptionally resilient to market shocks. The explicit modeling of jumps via the Hawkes process allows the agent to anticipate and mitigate extreme drawdown events better than generic LSTM or Transformer-based DRL agents.

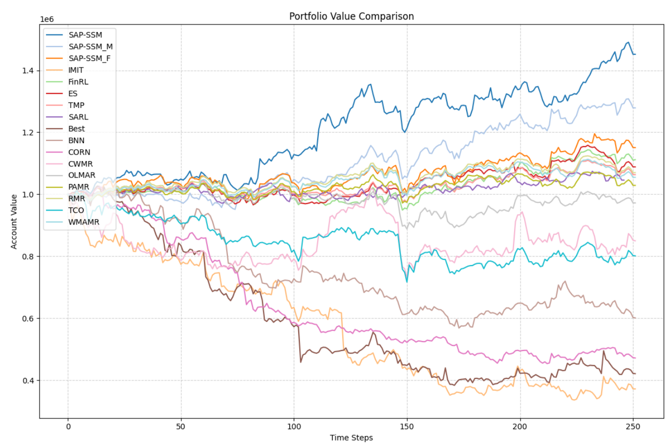

Fig. 2. Portfolio value comparison over the backtest period.

E. Ablation Study

To isolate the contribution of the proposed architecture, we conducted an ablation study (top section of Table I):

- **SAP-SSM_M (Mamba Core):** Removing the stochastic jump-diffusion output and dynamic projections reduces the return to 27.9% and the Sharpe ratio to 1.414. While still effective, it lacks the full risk-awareness of the complete model.
- **SAP-SSM_F (Feed-Forward):** Replacing the SSM core with a simple feed-forward network leads to a drastic performance drop (15.1% Return, 0.778 Sharpe), confirming that the recurrent, state-preserving nature of the SSM is critical for capturing temporal financial dependencies.

V. CONCLUSIONS

We introduced SAP-SSM, a novel DRL framework for portfolio management that marks a significant departure from the generic sequence modeling common in prior work. The model's core innovation is its state-space architecture, which is purpose-built to natively embody a sophisticated financial process model, standing in stark contrast to previous models that lack this crucial financial inductive bias. This unique integration of a Hawkes process for self-exciting shocks, a dynamic structure for market regime adaptation, and stochastic transitions for inherent randomness is a combination not previously explored in the literature for this task. The effectiveness of this novel design was confirmed through experiments on a portfolio of DJIA 30, achieving competitive performance.

Acknowledgment

This research is partially supported by Xi'an Jiaotong-Liverpool University (XJTLU) Research Development Fund (RDF-21-01-069, RDF-24-01-020), the 2024 Jiangsu Provincial Construction Science and Technology Project (No.2024ZD056), Suzhou Key Laboratory of Multi-modal Intelligent Agents for Industrial Applications (SZS2025126RC), Suzhou Multimodal Big Data Innovation Application Lab, Suzhou Science and Technology Plan Project - Innovation Consortium Project (LHT202417), Taicang Key Research and Development Program (Social Development) Project (TC2024SF09).

References

[1] D. Avramov, "Stock return predictability and model uncertainty," *Journal of Financial Economics*, vol. 64, 2002.

[2] S.-H. Poon and C. W. J. Granger, "Forecasting volatility in financial markets: A review," *Journal of economic literature*, vol. 41, no. 2, pp. 478–539, 2003.

[3] X.-Y. Liu, H. Yang, J. Gao, and C. D. Wang, "FinRL: Deep reinforcement learning framework to automate trading in quantitative finance," in *Proceedings of the second ACM international conference on AI in finance*, 2021, pp. 1–9.

[4] Z. Gao, Y. Gao, Y. Hu, Z. Jiang, and J. Su, "Application of deep q-network in portfolio management," *2020 5th IEEE International Conference on Big Data Analytics, ICBDA 2020*, 2020.

[5] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[6] R. S. Sutton and A. G. Barto, "Reinforcement learning," *A Bradford Book*, vol. volume 15, no. 7, pp. 665–685, 1998.

[7] F. Gu, Z. Zhang, Z. Jiang, Á. F. García-Fernández, J. Su, and H. Li, "Dynamic knowledge graph-guided deep reinforcement learning with hierarchical semantics transformer for portfolio management," in *2025 IEEE International Conference on Big Data (BigData)*, 2025, pp. 2150–2155.

[8] R. Sun, Y. Xi, A. Stefanidis, Z. Jiang, and J. Su, "A novel multi-agent dynamic portfolio optimization learning system based on hierarchical deep reinforcement learning," *Complex & Intelligent Systems*, vol. 11, no. 7, p. 311, 2025.

[9] X. Ren, R. Sun, Z. Jiang, A. Stefanidis, H. Liu, and J. Su, "Time series is not enough: Financial transformer reinforcement learning for portfolio management," *Neurocomputing*, p. 130451, 2025.

[10] F. Gu, H. Wang, Z. Jiang, A. F. García-Fernández, J. Su, and H. Li, "Mixture-of-experts liquid financial mamba framework for portfolio management based on deep reinforcement learning," in *2025 IEEE International Conference on Big Data (BigData)*, 2025, pp. 5756–5764.

[11] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," *2017 Intelligent Systems Conference, IntelliSys 2017*, vol. 2018-January, 2018.

[12] Y. Ding, W. Liu, J. Bian, D. Zhang, and T.-Y. Liu, "Investor-imitator: A framework for trading knowledge extraction," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1310–1319.

[13] S. Sun, M. Qin, H. Xia, C. Zong, J. Ying, Y. Xie, L. Zhao, X. Wang, B. An *et al.*, "Trademaster: A holistic quantitative trading platform empowered by reinforcement learning," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[15] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading: An ensemble strategy," in *Proceedings of the first ACM international conference on AI in finance*, 2020, pp. 1–8.

[16] F. Gu, Z. Jiang, Á. F. García-Fernández, A. Stefanidis, J. Su, and H. Li, "MTS: A deep reinforcement learning portfolio management framework with time-awareness and short-selling," *Intelligent Data Analysis*, p. 1088467X261416593, 2025.

[17] R. Sun, A. Stefanidis, Z. Jiang, and J. Su, "Combining transformer based deep reinforcement learning with black-litterman model for portfolio optimization," *Neural Computing and Applications*, vol. 36, no. 32, pp. 20 111–20 146, 2024.

[18] F. Gu, Z. Jiang, Á. F. García-Fernández, A. Stefanidis, J. Su, and H. Li, "APMPO: A portfolio management policy optimization framework with adaptive reinforcement learning algorithm," in *32nd International Conference on Neural Information Processing*, 2025.

[19] Z. Zhang, P. Sen, Z. Wang, R. Sun, Z. Jiang, and J. Su, "FinBPM: A framework for portfolio management-based financial investor behavior perception model," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024, pp. 246–257.

[20] F. Gu, Z. Zhang, Z. Jiang, Á. F. García-Fernández, J. Su, and H. Li, "Memory instance gated transformer reinforcement learning for portfolio management," in *2025 IEEE International Conference on Big Data (BigData)*, 2025, pp. 4566–4575.

[21] Z. Wang, F. Kong, S. Feng, M. Wang, X. Yang, H. Zhao, D. Wang, and Y. Zhang, "Is mamba effective for time series forecasting?" *Neurocomputing*, vol. 619, p. 129178, 2025.

[22] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, 2017.

[23] L. A. Baxter and M. L. Puterman, "Markov decision processes: Discrete stochastic dynamic programming," *Technometrics*, vol. 37, 1995.

[24] M. C. Mariani and O. K. Tweneboah, "Stochastic differential equations applied to the study of geophysical and financial time series," *Physica A: Statistical Mechanics and its Applications*, vol. 443, pp. 170–178, 2016.

[25] B. Li, S. C. Hoi, P. Zhao, and V. Gopalkrishnan, "Confidence weighted mean reversion strategy for on-line portfolio selection," *Journal of Machine Learning Research*, vol. 15, 2011.

[26] B. Li and S. C. Hoi, "On-line portfolio selection with moving average reversion," *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*, vol. 1, 2012.

[27] B. Li, P. Zhao, S. C. Hoi, and V. Gopalkrishnan, "Pamr: Passive aggressive mean reversion strategy for portfolio selection," *Machine Learning*, vol. 87, 2012.

[28] D. J. Huang, J. Zhou, B. Li, S. C. Hoi, and S. Zhou, "Robust median reversion strategy for online portfolio selection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, 2016.

[29] B. Li, J. Wang, D. Huang, and S. C. Hoi, "Transaction cost optimization for online portfolio selection," *Quantitative Finance*, vol. 18, 2018.

[30] L. Gao and W. Zhang, "Weighted moving average passive aggressive algorithm for online portfolio selection," *Proceedings - 2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics, IHMSC 2013*, vol. 1, 2013.

[31] L. Györfi, G. Lugosi, and F. Udina, "Nonparametric kernel-based sequential investment strategies," *Mathematical Finance*, vol. 16, 2006.

[32] B. Li, S. C. Hoi, and V. Gopalkrishnan, "Corn: Correlation-driven nonparametric learning approach for portfolio selection," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 2011.

[33] Y. Ye, H. Pei, B. Wang, P. Chen, Y. Zhu, J. Xiao, and B. Li, "Reinforcement-learning based portfolio management with augmented asset movement prediction states," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 1112–1119, 04 2020.