Direct Log-Utility Optimization for Portfolio Allocation under Exogenous Market Dynamics

Ivan Kurnosau, Abdulrahman Altahhan
University of Leeds, Leeds, UK

***Abstract*—We propose an efficient learning framework for long–short portfolio allocation that combines an objective-driven optimization criterion with the simplicity of supervised training. Under *exogenous market dynamics*, where market transitions are independent of the agent's actions under the price-taker assumption, maximising *log-utility* reduces to a *single-step* objective; hence, trajectory rollouts and multi-step credit assignment used by policy-gradient trading agents (e.g., *AlphaStock*, *DeepTrader*) often provide limited benefit. We therefore train the allocation policy end-to-end by direct log-utility optimization of a *regularised* objective using standard mini-batch backpropagation on observed returns. We evaluate against both trajectory-based reinforcement-learning agents and supervised portfolio-allocation baselines, including *Decision by Supervised Learning* (DSL), and find that our training setup consistently delivers stronger performance in our evaluation. Empirically, our method achieves substantially higher average annual return and Sharpe ratio than the evaluated baselines, while converging in fewer training iterations and with improved stability. Overall, direct log-utility optimization under exogenous dynamics provides a practical middle ground between reinforcement learning and supervised portfolio allocation.**

I. Introduction

Portfolio management seeks to allocate capital across assets to optimize return while controlling risk. Classical approaches such as mean–variance optimization, factor models, and momentum/value strategies have provided influential foundations [7, 16, 11], but real markets are non-stationary, nonlinear, and high-dimensional, limiting the ability of hand-crafted models to capture evolving patterns and cross-asset dependencies. Recent work has increasingly adopted deep reinforcement learning (DRL) to automate decisions through representation learning and end-to-end optimization [3, 12, 24, 25, 1, 4].

Despite their promise, trajectory-based DRL methods face practical limitations in portfolio learning. First, policy-gradient and actor–critic agents require multi-step credit assignment to propagate rewards through rebalancing sequences, yielding high-variance gradients, slow convergence, and sensitivity to regime shifts. Second, stochastic policies introduce sampling variance and reduce reproducibility, while training stability depends on variance-reduction heuristics. Third, rollout-based formulations depend on trajectory-specific hyperparameters (discount factors, episode length, bootstrapping horizons) that are difficult to calibrate in finance. Most critically, under the common price-taker assumption used in historical backtesting, the influence of an agent's actions on future market states is typically negligible: market dynamics are treated as exogenously generated. Recent supervised and direct-optimization alternatives bypass trajectory learning [14, 15, 26], but focus on long-only portfolios, rely on non-differentiable target generation, or do not explicitly leverage the exogenous-dynamics structure.

We propose a direct optimization framework for long–short allocation under exogenous dynamics. Modeling rebalancing as an action-independent MDP where $p(s_{t+1} \mid s_t, a_t) = p(s_{t+1} \mid s_t)$, each rebalancing decision becomes an independent one-step optimization problem. Instead of trajectories, we perform *direct policy optimization* by maximizing a differentiable log-utility objective: the network outputs portfolio weights, the realised portfolio log-return is computed from observed market returns, and randomly sampled steps are aggregated into mini-batches for backpropagation. This eliminates rollouts, value functions, and multi-step credit assignment, enabling faster and more stable training. Our framework directly optimizes log-utility under a fixed gross-exposure constraint with ℓ_1 normalisation, yielding a fully differentiable end-to-end procedure for long–short allocation that explicitly exploits the exogenous structure of historical backtesting. To facilitate reproducibility and future research, we will make our code and data publicly available upon acceptance.

Contributions. Our main contributions are:

- **Structure-aware reformulation.** We formalise portfolio allocation under historical training as an *exogenous* (action-independent) MDP and reduce learning to single-step optimization, replacing trajectory-based RL components with immediate, low-variance gradient updates.
- **Risk-sensitive direct objective.** We optimize a growth-optimal *log-utility* objective (cumulative log-return) and incorporate a variance-based regulariser to improve stability and risk-adjusted performance.
- **Deterministic long–short policy with exposure control.** We output signed portfolio weights and enforce a fixed gross-exposure constraint through ℓ_1 normalisation, $\sum_i |w_i| = 1$, yielding a fully differentiable and reproducible allocation rule.
- **Temporal–cross-asset architecture for allocation.** We propose a hybrid model combining bidirectional LSTMs, learnable asset embeddings, and multi-head self-attention to capture temporal structure and cross-asset interactions in a manner tailored to allocation rather than forecasting.
- **Empirical evaluation across three datasets.** We evaluate across three datasets spanning different time periods and market compositions: DJIA (2009–2019), NASDAQ-

100 (2020–2025), and NASDAQ-30 (2020–2025). We observe strong risk-adjusted performance and stable optimization, and we compare against both trajectory-based DRL baselines (DeepTrader, AlphaStock) and supervised learning baselines (DSL with LSTM and Mamba backbones), noting that some baseline results are reproduced from prior work and may not be strictly comparable.

II. Related Work

Portfolio management seeks to maximize expected returns while controlling risk through dynamic reallocation ([7, 17, 18, 23]). The mean–variance framework ([17]) established portfolio optimization as a single-period problem, later extended to multi-period settings ([18, 20]), but these classical models rely on explicit assumptions about asset dynamics that are difficult to specify in practice ([10]). Empirical strategies such as momentum investing ([11]), mean reversion ([2]), and multi-factor models ([5]) have been widely explored, but often underperform in changing market regimes. Machine learning approaches have shown promise: deep learning architectures such as recurrent and graph neural networks capture temporal and cross-asset dependencies ([6]), though supervised methods face challenges in label design and hyperparameter sensitivity ([8]). Reinforcement learning (RL) offers flexibility in reward design and the ability to directly optimize portfolio weights ([19, 3, 12]), but applying RL in real markets remains difficult due to non-stationarity, which induces exploration inefficiencies and degrades learned policies.

Wang et al. ([24]), in their AlphaStock model, leverage an LSTM with hierarchical attention to encode asset representations from multiple time series and incorporate Cross-Asset Attention to capture interrelationships among assets. Building on this, Wang et al. ([25]) proposed DeepTrader, which combines asset scores and market scores using dedicated scoring modules. The Asset Scoring Unit incorporates a Temporal Convolutional Network (TCN) augmented with Spatial Attention for short-term cross-asset dependencies and a Graph Convolution Layer for long-term dependencies and industry-wide performance. The Market Scoring Unit synthesizes market-wide indicators to guide the balance between long and short allocations. Niu et al. ([21]), in their MetaTrader framework, adopt a similar state representation and contribute a two-phase meta-learning scheme: a diverse repertoire of policies is learned via imitation learning, then a meta-policy is trained through Q-learning to dynamically select the most effective strategy based on observed market state.

Reinforcement learning provides a natural framework for portfolio management, as allocation can be directly represented as a vector of asset weights. Policy-based methods—such as policy gradient and actor–critic approaches—have become dominant in this domain. Despite their flexibility, RL approaches face key challenges: non-stationarity and noise exacerbate exploration and hinder stable learning; simplified assumptions regarding execution costs limit practical applicability; high sample complexity and sensitivity make robust training difficult, raising concerns about generalisation. Critically, portfolio management environments exhibit exogenous dynamics—individual trading decisions do not materially affect subsequent market states—yet existing methods employ trajectory-based policy gradient algorithms designed for settings where actions influence future observations. This mismatch introduces unnecessary complexity and computational overhead, motivating more direct optimization procedures.

Recognising these limitations, recent work has explored alternative paradigms. Kim et al. ([14]) proposed Decision by Supervised Learning (DSL), which reframes portfolio construction as a supervised problem. Models are trained to predict optimal weights using convex surrogate losses such as cross-entropy, with target portfolios constructed by maximizing the Sharpe or Sortino ratio. By integrating Deep Ensemble methods, DSL achieves improved stability and out-of-sample performance. Kisiel and Gorse ([15]) introduced the Portfolio Transformer, an end-to-end architecture that directly optimizes the Sharpe ratio without intermediate return prediction, outputting portfolio weights that maximize risk-adjusted performance under transaction costs. Yan ([26]) demonstrated the effectiveness of Transformer networks with attention mechanisms for portfolio optimization, achieving superior risk-adjusted returns compared to traditional mean-variance methods. These supervised and direct optimization approaches share a recognition that, under exogenous market dynamics, multi-step credit assignment and trajectory rollouts can be avoided by treating portfolio allocation as a single-step decision problem. This paradigm shift aligns with the structure-aware reformulation proposed in this work, where log-utility maximization is performed through immediate gradient updates rather than policy gradient estimation over sequential episodes.

III. Methodology

A. Problem Setup

1) Trading Procedure: We adopt the trading procedure established in prior work on deep reinforcement learning for portfolio management [24, 25, 21] and described in detail by Wang et al. ([25]). Consider an investment process involving both long and short positions. At time step $t-1$, the agent holds a portfolio of long positions $\mathbf{b}_{t-1}^{+} = \{b_{t-1,1}^{+}, b_{t-1,2}^{+}, \ldots, b_{t-1,n}^{+}\} \in R^n$ and short positions $\mathbf{b}_{t-1}^{-} \in R^n$, where n is the number of assets. Given the closing prices $\mathbf{p}_t^{\text{close}} \in R^n$ and target allocation $\mathbf{a}_t = [\mathbf{a}_t^{+}; \mathbf{a}_t^{-}]$ at time t, the agent executes the following sequence: (1) at the end of period $t-1$, all long positions $\mathbf{b}_{t-1}^{+}$ are liquidated to cash; (2) simultaneously, the borrowed stocks $\mathbf{b}_{t-1}^{-}$ are repurchased and returned to the broker, yielding available capital AC_t; (3) at the beginning of period t, new stocks are borrowed according to the short allocation $\mathbf{a}_t^{-}$ and capital AC_t, and immediately sold to obtain total cash TC_t; (4) finally, long positions are established by purchasing stocks with cash TC_t according to the long allocation $\mathbf{a}_t^{+}$. All components of the short and long positions are given as percentages of the total capital (see next section for more details).

This trading procedure can be viewed as a sequential decision-making task in which the agent observes market state s_t at each decision step t, produces an allocation action a_t, and receives the subsequent market state s_{t+1}.

2) State Representation: The state at each time step t consists of market conditions derived from OHLCV (Open, High, Low, Close, Volume) data, $s_t = X_t \in R^{n\times T\times F}$, where T denotes the number of time steps in the lookback window and F the number of features per asset. The state excludes current portfolio capital, so decisions depend on market information and normalized weights rather than wealth level; therefore, weight drift caused solely by capital growth is not an issue. These features capture the fundamental drivers of asset price dynamics: momentum and trend indicators enable the model to identify directional movements, volatility measures quantify uncertainty and risk, while microstructure signals reflect intraday supply-demand imbalances and liquidity conditions. The exact features are listed below:

- **Raw price and volume dynamics:** $\log\left(\frac{\text{close}_t}{\text{close}_{t-1}}\right)$, $\frac{\text{high}_t-\text{low}_t}{\text{close}_t}$, $\frac{\text{close}_t-\text{open}_t}{\text{open}_t}$, $\log\left(\frac{\text{volume}_t}{\text{volume}_{t-1}+\varepsilon}+\varepsilon\right)$
- **Momentum and trend:** EMA_3, EMA_{30}, RSI_2, RSI_6
- **Volatility:** $\sigma_{20} = \text{Std}(\Delta\text{close}_t)_{20}$
- **Microstructure and order flow:** $\frac{\text{close}_t-\text{VWAP}_t}{\text{close}_t}$, $\frac{\text{close}_t-\text{low}_t}{\text{high}_t-\text{low}_t+\varepsilon}$
- **Other derived features:** $\text{EMA}_3(t)-\text{EMA}_{15}(t)$, $\frac{\sigma_{10}(t)}{\sigma_{20}(t)+\varepsilon}$

3) Action Space: At each time step t, the action is a portfolio allocation vector $a_t \in R^n$, where n is the number of assets, and each element $a_t^i \in [-1, 1]$ represents the proportion of capital allocated to asset i (-1 for fully short, 1 for fully long, 0 for no position), with $\sum_{i=1}^n |a_t^i| = 1$. This Target Order representation, introduced by Huang ([9]), directly specifies desired positions rather than individual trades.

4) Reward: The reward at each time step is defined as the net portfolio gain over a single trading period, accounting for both asset returns and transaction costs. Let $a_t \in R^n$ denote the agent's allocation vector at time t and $r_t \in R^n$ the vector of asset returns realised from time t to $t+1$. The return component is computed as the inner product $a_t^\top r_t$, representing the weighted portfolio return. In the general turnover-based form, the penalty is $f\|a_t - a_{t-1}\|_1$; under our trading procedure (Section 3.1.1), all positions are closed before re-opening so the pre-trade inventory is zero ($a_{t-1} = 0$), which simplifies the penalty to $f\|a_t\|_1$. The total reward used for training is then:

$$R_t = a_t^\top r_t - f\|a_t\|_1 \tag{1}$$

5) Exogenous-Dynamics MDP and Direct Policy Optimization: The portfolio allocation task can be formalized as a Markov decision process (MDP) with states, actions, transitions, and rewards. However, trading environments exhibit a critical property: *exogenous dynamics*. In a general MDP, the environment dynamics are governed by the transition kernel

$$p(s' \mid s, a) = \Pr(S_t = s' \,|\, S_{t-1} = s,\ A_{t-1} = a)\,, \tag{2}$$

which explicitly couples the agent's action to the subsequent state. In contrast, for portfolio allocation we assume the agent's trading volume is small relative to overall market liquidity. Under this standard assumption [3, 12, 24, 25, 21], market dynamics are exogenously generated and the transition distribution simplifies to

$$p(s' \mid s) = \Pr(S_t = s' \,|\, S_{t-1} = s)\,, \tag{3}$$

indicating conditional independence of S_t from A_{t-1} given S_{t-1}. Importantly, actions still affect the instantaneous portfolio reward through the allocation applied to realized returns, so there is no conditional independence assumed between rewards and actions. Even with exogenous transitions, the learning objective remains sequential because it aggregates performance over time (e.g., cumulative log-utility).

This exogenous structure has important implications for policy optimization. Since there is no closed feedback loop between actions and future states, trajectory-level credit assignment provides limited additional benefit. The entire sequence of states $\{s_1, s_2, \ldots, s_T\}$ can be pre-populated from historical data prior to training. At each time step t, the agent observes s_t, produces an allocation a_t, and immediately computes the realized return R_t using a_t and the observed next-state returns r_t. Since each tuple (s_t, a_t, R_t) is conditionally independent given the state trajectory, we can optimize the policy parameters directly via backpropagation of a differentiable objective, avoiding trajectory rollouts and reducing reliance on policy-gradient training heuristics.

We maximize cumulative wealth over multiple periods. Given realized returns $R_1, R_2, \ldots, R_T$, the final portfolio value is:

$$\text{Final Wealth} = \text{Initial Wealth} \times \prod_{t=1}^{T}(1 + R_t) \tag{4}$$

To maximize this product, we apply the logarithmic transformation, which is monotonically increasing and thus preserves the optimal solution:

$$\arg\max_\theta \prod_{t=1}^{T}(1 + R_t) = \arg\max_\theta \sum_{t=1}^{T}\log(1 + R_t). \tag{5}$$

This transforms multiplicative compounding into an additive objective—the cumulative log-return—which is numerically stable and suitable for gradient-based optimization. This formulation is equivalent to maximizing the geometric mean return and forms the basis of growth-optimal portfolio strategies (Kelly criterion [13]).

In batch training, we approximate the expectation with the empirical mean over B samples and add a variance penalty to control risk:

$$\mathcal{L}(\theta) = -\frac{1}{|B|}\sum_{i=1}^{B}\log(1 + R_i) + \lambda \cdot \text{Var}(L_1, \ldots, L_B) \tag{6}$$

where R_i is the realized return for sample i from Equation 1, $\lambda \geq 0$ is a risk-aversion coefficient, and the variance term is:

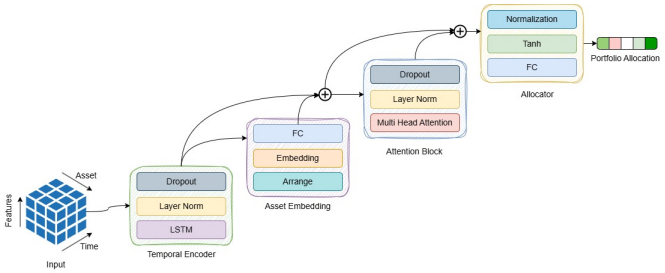

Fig. 1: The proposed network architecture for portfolio allocation.

$$\mathrm{Var}(L_1, \ldots, L_B) = \frac{1}{|B|} \sum_{i=1}^{B} (L_i - \bar{L})^2 \tag{7}$$

where $L_i = \log(1 + R_i)$ and $\bar{L} = \frac{1}{|B|} \sum_{i=1}^{B} L_i$ is the mean log return. The variance penalty discourages large fluctuations in per-period log returns, promoting risk-adjusted allocations and implicitly encouraging diversification.

This direct policy optimization approach confers several advantages over trajectory-based methods. First, it enables faster convergence by allowing parameter updates after each single-step allocation rather than requiring entire trajectory rollouts. Second, by exploiting the exogenous structure, we optimize deterministic policies directly via backpropagation, avoiding high-variance gradient estimates inherent to stochastic policy gradient estimators. Third, it eliminates the need to tune trajectory-specific hyperparameters such as episode length, discount factors, or bootstrapping horizons. By recognizing that market dynamics are exogenous to agent actions, we achieve a streamlined, sample-efficient policy optimization procedure.

B. Framework

1) Framework Overview: Figure 1 illustrates our end-to-end differentiable policy network mapping market observations to portfolio allocations. At each time step t, the input consists of sequential market features $X_t \in R^{A \times T \times F}$, where A is the number of assets, T the lookback window, and F features per asset. The model comprises four modules: (1) a *Temporal Encoder* captures per-asset time-series patterns via bidirectional LSTMs; (2) an *Asset Embedding* layer injects learnable asset-specific representations; (3) an *Attention Block* models cross-asset dependencies via multi-head self-attention; (4) a *Portfolio Allocator* produces the allocation vector $a_t \in R^A$ with $\sum_{i=1}^{A} |a_t^i| = 1$. The entire pipeline is optimized end-to-end via backpropagation of the log-utility objective (Equation 6).

2) Temporal Encoder: The temporal encoder processes each asset's sequential features independently to extract representations capturing time-series dynamics (momentum, trend reversals, volatility clustering). For asset i, the sequence $X_i = (x_{i,1}, \ldots, x_{i,T}) \in R^{T \times F}$ is fed to a bidirectional LSTM with hidden dimension H and L layers. The bidirectional architecture uses both directions only within the fixed historical lookback window ending at decision time t (no information beyond t), so no leakage occurs. We extract the final hidden state:

$$h_i^{\text{temp}} = \mathrm{BiLSTM}(X_i)_{\text{last}} \in R^{2H}, \tag{8}$$

where the factor of 2 arises from concatenating forward and backward states. Layer normalization and dropout stabilize training:

$$h_i = \mathrm{Dropout}\big(\mathrm{LayerNorm}(h_i^{\text{temp}})\big). \tag{9}$$

This produces a representation $h_i \in R^{2H}$ for each asset.

3) Asset Embedding: While the temporal encoder treats all assets uniformly, portfolios benefit from distinguishing securities with different characteristics (e.g., technology vs. financial stocks, high vs. low volatility). We introduce learnable asset embeddings providing asset identity information independent of market features. For each asset $i \in \{1, \ldots, A\}$, a learnable embedding $e_i \in R^{d_e}$ is added to the temporal encoding:

$$\tilde{h}_i = h_i + W_e e_i, \tag{10}$$

where $W_e \in R^{2H \times d_e}$ projects the embedding to match the hidden dimension. This allows subsequent layers—particularly attention—to differentiate assets and learn asset-specific biases. Embeddings enable the model to encode persistent characteristics (sector affiliation, liquidity profile) not observable in short-term feature windows.

4) Attention Block: Portfolio allocation involves cross-asset reasoning: a stock's attractiveness depends on its own dynamics and on related assets, sector trends, and market sentiment. We apply multi-head self-attention over the augmented states $\tilde{H} = [\tilde{h}_1, \ldots, \tilde{h}_A] \in R^{A \times 2H}$, computing pairwise similarity scores and re-weighting representations:

$$\tilde{H}^{\text{attn}} = \mathrm{MultiHeadAttention}(\tilde{H}, \tilde{H}, \tilde{H}), \tag{11}$$

where queries, keys, and values derive from $\tilde{H}$. We use N_h attention heads to capture diverse interaction patterns across different subspaces. Residual connections and layer normalization stabilize gradient flow:

$$H^{\text{out}} = \mathrm{Dropout}\big(\mathrm{LayerNorm}(\tilde{H}^{\text{attn}} + \tilde{H})\big). \tag{12}$$

This produces $H^{\text{out}} \in R^{A \times 2H}$ where each asset's representation is contextualized by the entire portfolio, facilitating coordinated allocations.

5) Portfolio Allocator: The final module maps attention-enriched representations to portfolio allocations. A linear projection computes raw scores:

$$z_i = w^\top H_i^{\text{out}} + b, \qquad z = [z_1, \ldots, z_A] \in R^A, \tag{13}$$

where $w \in R^{2H}$ and $b \in R$ are shared learnable parameters. To satisfy $\sum_{i=1}^{A} |a_i| = 1$ with $a_i \in [-1, 1]$, we apply tanh activation and L^1 normalization:

$$a = \frac{\tanh(z)}{\|\tanh(z)\|_1}, \qquad \sum_{i=1}^{A} |a_i| = 1. \tag{14}$$

The tanh bounds elements to $(-1, 1)$ for long-short positions, while normalization ensures unit total capital commitment; in implementation, the ℓ_1 norm uses a smooth absolute approximation (e.g., $|x| \approx \sqrt{x^2 + \varepsilon}$) for numerical stability. This deterministic policy eliminates sampling variance, enabling stable gradient flow. The entire architecture is fully differentiable, allowing end-to-end optimization of the log-utility objective (Equation 6).

IV. Experiment Results

A. Experiment Setup

1) Datasets: We evaluate our framework across three distinct asset universes:

DJIA. Daily price and volume data for the 30 constituents of the Dow Jones Industrial Average obtained from the Stooq repository (https://stooq.com/), covering January 1970 (or earliest available date) to January 2019.

NASDAQ-100. All members of the NASDAQ-100 index as described in Kim et al. ([14]), with daily data from 2020 to November 2025. This provides a broader technology-focused universe for evaluating performance in recent market conditions.

NASDAQ-30. The top 30 constituents of the NASDAQ-100 index by market capitalization, following the same temporal coverage (2020–2025) as described in Kim et al. ([14]). This subset allows evaluation on a more concentrated portfolio of large-cap technology stocks.

For all datasets, we collect OHLCV fields and compute the feature set described in Section 3.1, and adopt a rolling window evaluation protocol: at the beginning of each calendar year y, the model is trained on data up to year $y-2$, validated on year $y-1$, and tested on year y.

2) Comparative Methods: We benchmark our approach against diverse baselines adapted to each dataset. Common across all datasets are classical heuristics:

- **Market**: A passive buy-and-hold strategy on equally weighted portfolio constituents.
- **BLSW** ([22]): Buying losers and selling winners, a mean-reversion strategy capable of shorting.
- **CSM** ([11]): A cross-sectional momentum strategy that ranks assets by recent performance.

For the **DJIA** dataset, we additionally compare against trajectory-based deep reinforcement learning methods: **DeepTrader** ([25]) and **AlphaStock** ([24]), both described in Section 2. We report results as provided in Wang et al. ([25]); minor protocol differences may affect direct comparability.

For the **NASDAQ-100** and **NASDAQ-30** datasets, we compare against supervised learning baselines: **DSL-L** and **DSL-M** (Decision by Supervised Learning with LSTM and Mamba backbones, respectively, described in Section 2). We report results as provided in Kim et al. ([14]); minor protocol differences may affect direct comparability.

B. Evaluation Results

Table I summarizes out-of-sample performance across all three datasets, supporting the effectiveness of direct log-utility optimization under the exogenous-dynamics assumption.

DJIA (2009–2019). Our framework achieves the highest annualized return (APR $= 0.178$) and strongest Sharpe ratio (ASR $= 0.954$). Relative to the reported trajectory-based RL baselines, our approach attains higher APR and ASR while maintaining competitive risk (Table I). DeepTrader achieves the lowest maximum drawdown (MDD $= 13.7\%$) and highest Sortino ratio (SoR $= 0.553$), while AlphaStock attains the lowest volatility (AVOL $= 0.123$). Our framework incurs moderate volatility (AVOL $= 0.186$) and drawdown (MDD $= 15.4\%$), closely matching DeepTrader's drawdown while achieving higher returns.

NASDAQ-100 (2020–2025). Our framework achieves APR $= 0.331$ and ASR $= 1.41$, outperforming supervised learning baselines DSL-L (APR $= 0.113$, ASR $= 0.219$) and DSL-M (APR $= 0.146$, ASR $= 0.453$). These results highlight the benefit of direct log-utility optimization under the exogenous-dynamics assumption relative to target-based supervised allocation. Our framework also outperforms the passive Market strategy (APR $= 0.232$, ASR $= 1.15$).

NASDAQ-30 (2020–2025). Our framework achieves the highest return (APR $= 0.480$), while the **Ours-NA** variant (without spatial attention) achieves the best risk-adjusted performance (ASR $= 1.47$, SoR $= 0.756$). This indicates a return–risk trade-off: attention can increase returns but may amplify drawdowns in concentrated universes. Nevertheless, our full framework exceeds DSL-L (APR $= 0.380$, ASR $= 0.988$) and DSL-M (APR $= 0.249$, ASR $= 0.602$).

Figure 2 shows cumulative wealth trajectories across the three datasets. Our method exhibits strong long-term compounding and performs competitively relative to the evaluated baselines, consistent with the summary metrics in Table I.

Training efficiency. Our approach converges in 20 epochs in our setup. By leveraging the exogenous-dynamics structure and optimising the objective directly via backpropagation, we eliminate trajectory rollouts and reduce optimization variance, yielding stable training across all three datasets.

C. Ablation Study

The ablation results in Table I and Figure 3 allow us to disentangle the influence of key architectural components: **asset embeddings** that provide each ticker with a learnable identity vector, **spatial self-attention** that models cross-asset dependencies, and **variance penalty** that stabilizes returns and controls risk.

Effectiveness of asset embeddings. Removing asset embeddings (**Ours-NE**) degrades performance across all datasets. On DJIA, APR decreases from 0.178 to 0.108 and ASR from 0.954 to 0.524, with similar drops on NASDAQ-100 (APR: $0.331 \rightarrow 0.278$) and NASDAQ-30 (APR: $0.480 \rightarrow 0.278$). This indicates that explicit asset identity information is important for learning effective allocation policies.

Algorithm	DJIA (2009–2019)					NASDAQ-100 (2020–2025)					NASDAQ-30 (2020–2025)				
	APR	AVOL	MDD	ASR	SoR	APR	AVOL	MDD	ASR	SoR	APR	AVOL	MDD	ASR	SoR
Ours	**.178**	.186	15.4	**.954**	.409	**.331**	.235	27.3	**1.41**	**.618**	**.480**	.395	60.0	1.21	.515
Ours-NA	.125	.222	31.1	.564	.222	.265	.209	27.3	1.27	.520	.472	.321	28.7	**1.47**	**.756**
Ours-NE	.108	.207	37.2	.524	.223	.278	.229	27.3	1.21	.551	.278	.212	27.5	1.31	.556
Ours-NEA	.061	.254	38.4	.242	.113	.256	.206	27.3	1.24	.504	.257	.255	32.4	1.01	.362
Ours-NRT	.138	.198	23.6	.697	.289	.277	.225	27.3	1.23	.584	.410	.373	64.3	1.10	.369
DeepTrader	.094	.150	**13.7**	.631	**.553**	–	–	–	–	–	–	–	–	–	–
AlphaStock	.067	**.123**	21.8	.541	.466	–	–	–	–	–	–	–	–	–	–
DSL-L	–	–	–	–	–	.113	.518	56.7	.219	.215	.380	.384	34.2	.988	.471
DSL-M	–	–	–	–	–	.146	.324	33.2	.453	.251	.249	.414	46.5	.602	.351
CSM	.003	.250	24.5	.010	.109	.107	.242	**21.6**	.444	.331	-.117	.129	52.2	-.905	-.298
BLSW	.039	.255	37.5	.152	.325	-.116	.159	59.4	-.728	-.262	.076	**.129**	**23.6**	.591	.348
Market	.093	.162	18.7	.577	.455	.232	**.202**	28.8	1.15	.580	.285	.214	29.6	1.34	.608

TABLE I: Performance metrics across three datasets. APR = Annualised Percentage Return, AVOL = Annualised Volatility, MDD = Maximum Drawdown (%), ASR = Annualised Sharpe Ratio, SoR = Sortino Ratio. Best performance per dataset in bold. Empty cells (–) indicate baselines not evaluated on that dataset.

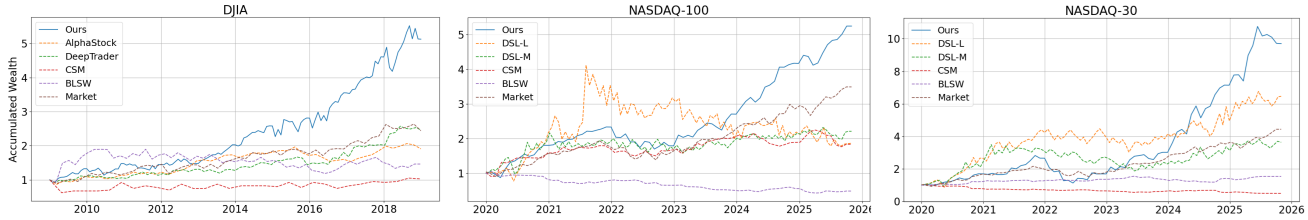

Fig. 2: Cumulative wealth trajectories across three datasets: DJIA (2009–2019), NASDAQ-100 (2020–2025), and NASDAQ-30 (2020–2025). Our framework shows strong long-term wealth accumulation across the evaluated datasets.

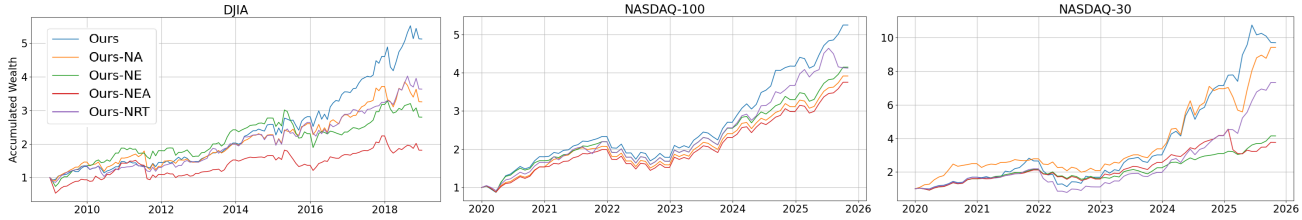

Fig. 3: Ablation study: Cumulative wealth curves for different framework variations across all three datasets (DJIA, NASDAQ-100, and NASDAQ-30), highlighting the impact of discussed key architectural components on long-term performance.

Effectiveness of spatial attention. Disabling spatial attention while retaining embeddings (**Ours-NA**) shows dataset-dependent effects. On DJIA, performance decreases (APR: $0.178 \rightarrow 0.125$) and drawdown increases ($15.4\% \rightarrow 31.1\%$). On NASDAQ-100, the impact is modest (APR: $0.331 \rightarrow 0.265$). On NASDAQ-30, **Ours-NA** achieves the best risk-adjusted performance (ASR $= 1.47$, SoR $= 0.756$) with lower drawdown (28.7% vs. 60.0%), suggesting a return–risk trade-off for attention in concentrated universes.

Combined effect. Disabling both embeddings and attention (**Ours-NEA**) yields the weakest performance across all datasets. On DJIA, this configuration underperforms most baselines (APR $= 0.061$, ASR $= 0.242$), highlighting the complementary roles of embeddings (asset-specific representation) and attention (cross-asset interaction).

Effectiveness of variance penalty. Removing the variance regularization term (**Ours-NRT**, setting $\lambda = 0$ in Equation 6) worsens downside-risk metrics and reduces returns. On DJIA, MDD increases from 15.4% to 23.6%; on NASDAQ-30, from 60.0% to 64.3%. This supports the role of the variance penalty in improving stability and risk-adjusted performance.

V. Conclusion

This work introduced a direct policy optimization framework for portfolio allocation under exogenous-dynamics assumptions that maximizes log-utility through single-step rebalancing decisions. Under the common price-taker setting, market transitions are independent of agent actions, enabling direct backpropagation of a regularized log-utility objective to produce deterministic long–short allocations under a fixed gross-exposure constraint. Our architecture combines bidirectional LSTMs for temporal pattern extraction, learnable asset embeddings for asset-specific structure, and multi-head self-attention to model cross-asset dependencies.

We evaluated across three datasets spanning different market regimes and asset universes. Results show strong risk-adjusted performance across the evaluated datasets and competitive outcomes relative to both trajectory-based RL baselines on DJIA and supervised learning approaches on NASDAQ datasets, with baseline comparability noted where results are reproduced from prior work.

The ablation study indicates that asset embeddings, attention, and the variance penalty each contribute to performance and stability. Future work includes evaluation on higher-frequency data and incorporating more realistic friction models such as market impact and slippage in settings where liquidity constraints become material.

References

[1] A. M. Aboussalah and C.-G. Lee. "Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization". In: *Expert Systems with Applications* 140, 112891 (2020).

[2] Werner F. M. De Bondt and Richard Thaler. "Does the Stock Market Overreact?" In: *The Journal of Finance* 40.3 (1985), pp. 793–805. DOI: 10.1111/j.1540-6261.1985.tb05004.x.

[3] Y. Deng et al. "Deep direct reinforcement learning for financial signal representation and trading". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.3 (2016), pp. 653–664.

[4] J. Du et al. "Deep Reinforcement Learning for Option Replication and Hedging". In: *The Journal of Financial Data Science* 2 (2020), pp. 44–57.

[5] E. F. Fama and K. R. French. "Common risk factors in the returns on stocks and bonds". In: *Journal of Financial Economics* 33.1 (1993), pp. 3–56.

[6] Thomas Fischer and Christopher Krauss. "Deep learning with long short-term memory networks for financial market predictions". In: *European Journal of Operational Research* 270.2 (2018), pp. 654–669. DOI: 10.1016/j.ejor.2017.11.054.

[7] M. Grinblatt, S. Titman, and R. Wermers. "Momentum investment strategies, portfolio performance, and herding: A study of mutual fund behavior". In: *The American Economic Review* (1995), pp. 1088–1105.

[8] S. Gu, B. Kelly, and D. Xiu. "Empirical asset pricing via machine learning". In: *The Review of Financial Studies* 33.5 (2020), pp. 2223–2273.

[9] C.-Y. Huang. "Financial trading as a game: A deep reinforcement learning approach". In: *arXiv preprint* (2018). arXiv: 1807.02787.

[10] Ravi Jagannathan and Tongshu Ma. "Risk Reduction in Large Portfolios: Why Imposing the Wrong Constraints Helps". In: *The Journal of Finance* 58.4 (2003), pp. 1651–1683. DOI: 10.1111/1540-6261.00580.

[11] N. Jegadeesh and S. Titman. "Returns to buying winners and selling losers: Implications for stock market efficiency". In: *The Journal of Finance* 48.1 (1993), pp. 65–91.

[12] Z. Jiang, D. Xu, and J. Liang. "A deep reinforcement learning framework for the financial portfolio management problem". In: *arXiv preprint* (2017). arXiv: 1706.10059.

[13] J. L. Kelly. "A New Interpretation of Information Rate". In: *Bell System Technical Journal* 35.4 (1956), pp. 917–926. DOI: 10.1002/j.1538-7305.1956.tb03809.x.

[14] Juhyeong Kim et al. "Decision by Supervised Learning with Deep Ensembles: A Practical Framework for Robust Portfolio Optimization". In: *Proceedings of 6th ACM International Conference on AI in Finance (ICAIF'25)*. ACM, 2025. arXiv: 2503.13544.

[15] Damian Kisiel and Denise Gorse. "Portfolio Transformer for Attention-Based Asset Allocation". In: *arXiv preprint* (2022). arXiv: 2206.03246.

[16] B. G. Malkiel. "Efficient market hypothesis". In: *Finance*. Springer, 1989, pp. 127–134.

[17] H. Markowitz. "Portfolio selection". In: *The Journal of Finance* 7.1 (1952), pp. 77–91.

[18] Robert C. Merton. "An Analytic Derivation of the Efficient Portfolio Frontier". In: *Journal of Financial and Quantitative Analysis* 7.4 (1972), pp. 1851–1872. DOI: 10.2307/2329621.

[19] J. Moody and M. Saffell. "Learning to trade via direct reinforcement". In: *IEEE Transactions on Neural Networks* 12.4 (2001), pp. 875–889.

[20] J. Mossin. "Optimal multiperiod portfolio policies". In: *The Journal of Business* 41.2 (1968), pp. 215–229.

[21] H. Niu, S. Li, and J. Li. "MetaTrader: A Reinforcement Learning Approach Integrating Diverse Policies for Portfolio Optimization". In: *Proceedings of the 31st ACM International Conference on Information and Knowledge Management (CIKM '22)*. 2022.

[22] J. M. Poterba and L. H. Summers. "Mean reversion in stock prices: Evidence and implications". In: *Journal of Financial Economics* 22.1 (1988), pp. 27–59.

[23] W. F. Sharpe. "Capital asset prices: A theory of market equilibrium under conditions of risk". In: *The Journal of Finance* 19.3 (1964), pp. 425–442.

[24] J. Wang et al. "AlphaStock: A Buying-Winners-and-Selling-Losers Investment Strategy using Interpretable Deep Reinforcement Attention Networks". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1900–1908.

[25] Z. Wang et al. "DeepTrader: A Deep Reinforcement Learning Approach for Risk-Return Balanced Portfolio Management with Market Conditions Embedding". In: *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*. 2021.

[26] Zetao Yan. "Investment portfolio optimization with supervised learning and attention mechanism". In: *Egyptian Informatics Journal* 32, 100839 (2025). DOI: 10.1016/j.eij.2025.