Deep Portfolio Optimization with Soft Actor–Critic and GA-Driven Feature Selection: A Comparison with LSTM/Transformer and Classical Baselines

NYU MSCS Deep Learning – Final Project

Aranya Aryaman[1] and Prashanth Kumar[2]

[3]Department of Computer Science, New York University
{aa12939,pk3047}@nyu.edu

Abstract

This project studies deep learning methods for data–driven portfolio optimization. An end–to–end Soft Actor–Critic (SAC) agent learns a stochastic allocation policy that maps technical indicators and macro–financial features directly to daily portfolio weights under a reward that combines a smoothed Sharpe–like objective with penalties for volatility, drawdown, tail risk, and transaction costs. As deep learning baselines, LSTM and Transformer models forecast next–day returns; these predictions feed a convex optimizer that approximately maximizes Sharpe ratio subject to long–only, fully–invested constraints. We further compare against standard classical portfolios, including equal weight, risk parity, minimum variance, and Markowitz mean–variance optimization. On a multi–year equity dataset, the SAC agent achieves the strongest risk–adjusted performance, with an annualized return of about 42% and Sharpe ratio near 1.0, while both LSTM–Sharpe and Transformer–Sharpe remain close to 9% annualized return and Sharpe around 0.18. Equal weight and risk–parity baselines perform competitively but still underperform SAC on Sharpe and Calmar ratios. These results highlight the potential of end–to–end reinforcement learning for robust portfolio allocation in non–stationary markets.

Keywords: deep learning, reinforcement learning, portfolio optimization, sequence modeling

1 Introduction

Deep learning is widely used to model complex financial time series. A common approach is a two–stage pipeline: first predict future returns with neural networks, then use classical optimization to construct a portfolio from those forecasts. This decouples predictive accuracy from the investment objective and often ignores transaction costs and sequential decision structure.

Reinforcement learning (RL) provides an alternative framework in which a policy is trained end–to–end to map market states directly to portfolio weights by optimizing a task–specific reward. In this project, a Soft Actor–Critic (SAC) agent is compared against prediction–based baselines and classical portfolios in a unified experimental setup.

Our contributions:

- Implement an attention–style SAC agent that produces long–only portfolio weights from per–asset technical indicators and macro features, using a reward combining Sharpe–like return, volatility, drawdown, CVaR, entropy, and transaction–cost terms.
- Implement LSTM and Transformer models that predict next–day returns; their forecasts feed a Sharpe–maximizing convex optimizer, yielding LSTM–Sharpe and Transformer–Sharpe portfolios.
- Benchmark SAC, LSTM–Sharpe, Transformer–Sharpe, equal weight, risk parity, minimum variance, and Markowitz portfolios on a common test period and analyze risk–adjusted metrics and allocation behavior.

2 Background and Related Work

Classical portfolio theory originates from Markowitz mean–variance optimization, where investors trade off expected return and variance under linear constraints, and has since been extended with risk–parity, minimum–variance, and robust covariance ideas [1, 2].

Feature selection has been extensively studied in machine learning and finance, with surveys covering filter, wrapper, and embedded methods, and their role in high–dimensional prediction problems [2–7]. Genetic algorithms and other metaheuristics are widely used for feature selection and model optimization, including financial applications such as stock prediction and technical–indicator design [8–16].

Deep reinforcement learning for portfolio allocation has been explored in several recent works [17, 18], often building on the Soft Actor–Critic algorithm and entropy–augmented objectives [19, 20]. Our approach follows this line by combining SAC with GA–driven feature selection and comparing it against prediction–based LSTM/Transformer pipelines and classical portfolios.

3 Problem Statement and Goals

3.1 Problem Description

Let $p_{t,i}$ denote the daily close price of asset i on day t and $r_{t,i} = (p_{t+1,i} - p_{t,i})/p_{t,i}$ the corresponding return. At each day t the allocator observes a state vector s_t that aggregates:

- Per–asset technical indicators (momentum, volatility, trend, volume–based features).
- Market–level features such as implied volatility and broad index levels.

The allocator then outputs a weight vector $w_t \in \mathbb{R}^N$ with $w_t \geq 0$ and $\sum_i w_{t,i} = 1$. After proportional transaction costs, the portfolio return is

$$r_t^{\text{port}} = w_t^\top r_t - c(\Delta w_t),$$

and the portfolio value evolves as $V_{t+1} = V_t(1 + r_t^{\text{port}})$.

3.2 Objectives and Scope

The main objective is to learn allocation rules or predictive models that maximize risk–adjusted performance over a held–out test period. We restrict attention to:

- Long–only, unlevered portfolios with daily rebalancing.
- A fixed universe of liquid large–cap equities.
- Backtesting on historical data without modeling microstructure or live execution effects.

Performance is evaluated using daily mean return, volatility, Sharpe ratio, Sortino ratio, Calmar ratio, maximum drawdown, and win rate.

4 Approach

4.1 Overall Design

All methods share a common feature pipeline that computes technical indicators for each asset and combines them with macro factors to form state vectors. From this shared representation, two types of methods are built:

1. An SAC agent that directly outputs portfolio weights.
2. LSTM and Transformer models that predict next–day returns; these predictions feed a Sharpe–ratio optimizer that produces weights.

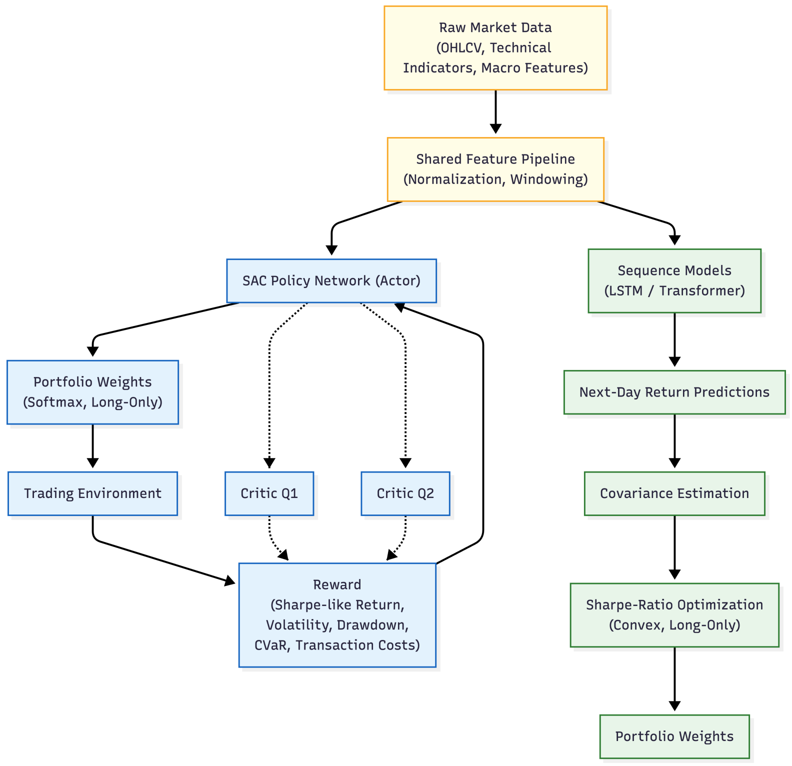
Raw Market Data (OHLCV, Technical Indicators, Macro Features)
Shared Feature Pipeline (Normalization, Windowing)
SAC Policy Network (Actor)
Sequence Models (LSTM / Transformer)
Portfolio Weights (Softmax, Long-Only)
Next-Day Return Predictions
Trading Environment
Critic Q1
Critic Q2
Covariance Estimation
Reward (Sharpe-like Return, Volatility, Drawdown, CVaR, Transaction Costs)
Sharpe-Ratio Optimization (Convex, Long-Only)
Portfolio Weights

Figure 1: Shared feature pipeline with two heads: an SAC policy for end–to–end allocation, and LSTM/Transformer predictors whose forecasts drive Sharpe–ratio optimization.

4.2 SAC Portfolio Agent

The state s_t concatenates per–asset feature blocks and macro features. The SAC actor applies linear encoders to each asset's feature vector, adds learned asset embeddings, fuses macro features, and passes the result through fully connected layers. A temperature–scaled Softmax transforms Gaussian samples into simplex weights.

Two Q–networks form the critic and estimate $Q_1(s, a)$ and $Q_2(s, a)$. Target networks are updated with Polyak averaging. The one–step reward blends a smoothed Sharpe–like term with penalties for volatility, drawdown, and CVaR, plus entropy and stability bonuses. SAC maximizes the discounted sum of reward plus an entropy term, using stochastic gradient updates for actor, critic, and temperature.

4.3 LSTM and Transformer Baselines

For each asset, windows of length T of its features form sequences. The LSTM baseline uses stacked LSTM layers followed by a dense head to predict next–day returns. The Transformer baseline replaces the recurrent backbone with a small encoder–only Transformer that uses self–

attention over time; a pooled representation is mapped through an MLP to a return forecast. Both models are trained with mean–squared error loss on realized returns.

Predicted returns $\hat{r}_{t+1}$ and an empirical covariance matrix estimated from recent data are passed to a convex program that approximates Sharpe maximization under long–only, fully–invested constraints, producing LSTM–Sharpe and Transformer–Sharpe portfolios.

4.4 Classical Baselines

Classical baselines include:

- Equal weight (EW) portfolios with $w_i = 1/N$.
- Risk parity portfolios with weights proportional to inverse asset volatility.
- Minimum variance portfolios using a shrinkage covariance estimator.
- Markowitz mean–variance portfolios with long–only constraints.

4.5 Implementation Details

All models are implemented in Python using modern deep learning and optimization libraries. SAC is trained for a few hundred episodes with mini–batch updates from a replay buffer and gradient clipping. The LSTM and Transformer predictors are trained for 15–20 epochs with AdamW and learning–rate scheduling. Experiments run on a single GPU; training times are on the order of a few hours.

5 Data and Experimental Setup

5.1 Dataset Description

The dataset consists of several years of daily OHLCV data for a basket of large–cap equities, together with macro variables such as index levels and implied volatility indices. Technical indicators (e.g., moving averages, MACD, RSI, CCI, ATR, Bollinger Bands, OBV, MFI) are computed for each asset and normalized. These indicators form the per–asset feature block within each state.

5.2 Preprocessing and Splits

All price and feature series are aligned on trading days, forward– and backward–filled where appropriate, and scaled using training–period statistics. Data are split temporally into training, validation, and test periods. The same splits are used across SAC, LSTM, Transformer, and classical baselines to enable fair comparison.

5.3 Evaluation Protocol

Each method generates a sequence of daily portfolio weights over the test period. From the resulting portfolio returns, the following metrics are computed:

- Mean daily return and daily volatility.
- Annualized return and annualized volatility.
- Sharpe, Sortino, and Calmar ratios.
- Maximum drawdown, win rate, VaR, and CVaR.

A constant annual risk–free rate is assumed for Sharpe and Sortino calculations.

6 Results

6.1 Quantitative Results

Table 1 summarizes the main test–period results for SAC, the deep learning baselines, and classical portfolios. Values are computed from the out–of–sample daily return series.

Method	Sharpe	Sortino	Calmar	Max DD
Equal Weight	0.804	1.080	1.064	-0.240
Risk Parity	0.663	0.854	0.908	-0.196
Min Variance	0.230	0.277	0.482	-0.160
LSTM–Sharpe	0.181	0.248	0.549	-0.162
Transformer–Sharpe	0.179	0.266	0.664	-0.133
SAC (ours)	0.998	1.898	2.104	-0.198

Table 1: Out–of–sample performance on the test period. Values are computed from daily returns: SAC clearly attains the highest Sharpe, Sortino, and Calmar ratios, at the cost of slightly larger maximum drawdown than some classical baselines.

SAC achieves the best risk–adjusted performance, with Sharpe ratio close to 1.0 and Calmar ratio above 2.0, alongside an annualized return around 42%. LSTM–Sharpe and Transformer–Sharpe produce similar annualized returns near 9% and Sharpe ratios around 0.18, while equal weight and risk parity deliver competitive but lower Sharpe than SAC.

6.2 Qualitative Results

Figure 2 shows training curves for the SAC agent, including episode rewards, Sharpe–like rewards, entropy temperature, critic and actor losses, and excess returns versus a Markowitz reference.

Figure 3 displays the evolution of SAC portfolio weights across episodes together with a heatmap across assets, showing how the policy gradually concentrates on a subset of assets while maintaining diversification.

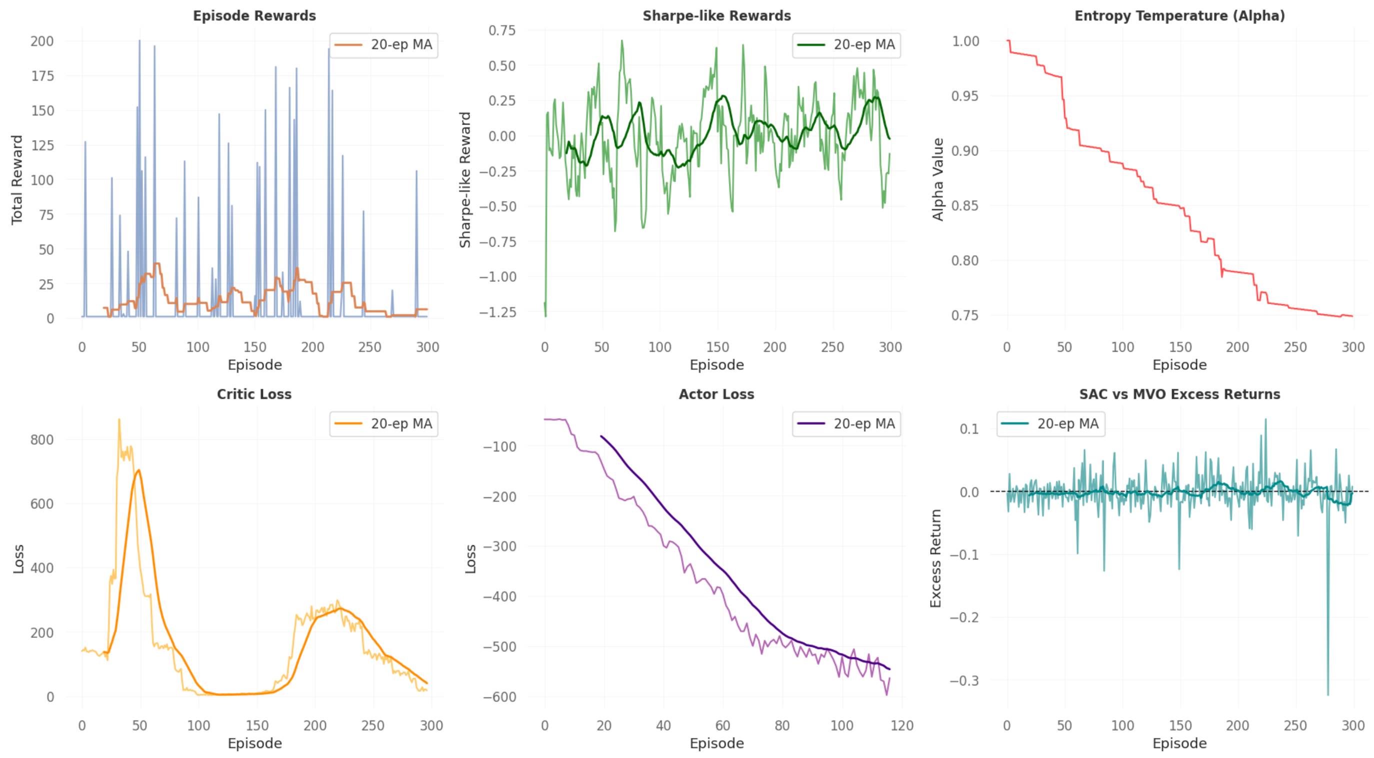

Figure 2: SAC training diagnostics: episode rewards and smoothed Sharpe–like rewards, entropy temperature, critic and actor losses, and excess returns versus a mean–variance baseline.

Figure 4 compares cumulative returns, drawdowns, rolling Sharpe, and key risk–adjusted metrics across SAC, LSTM–Sharpe, Transformer–Sharpe, and classical portfolios.

7 Analysis and Discussion

Quantitative results indicate that SAC improves substantially over prediction–based baselines in Sharpe and Sortino ratios while remaining competitive in drawdown. The LSTM and Transformer pipelines perform similarly, suggesting that for this horizon and feature set, temporal modeling capacity is not the limiting factor.

Classical portfolios exhibit smoother behavior and lower drawdown but lower risk–adjusted returns. SAC's reward shaping appears effective in balancing return, volatility, and drawdown, and the entropy term helps avoid overly concentrated allocations.

8 Limitations and Ethical Considerations

The study is limited to a specific historical period, asset universe, and daily frequency; performance may not generalize to other regimes or markets. Backtests are also subject to survivorship bias and potential data–snooping. In practice, deploying autonomous trading agents raises questions around market impact, transparency, and systemic risk that are beyond the scope of this work.

9 Conclusion and Future Work

This project compares an end–to–end SAC portfolio agent with prediction–based deep learning baselines and classical portfolios on a common equity dataset. SAC achieves the highest risk–adjusted performance, demonstrating the potential of RL for portfolio optimization when reward design and training are carefully controlled.

Future directions include richer macro inputs, multi–frequency data, distributional or risk–sensitive RL objectives, and tighter integration of Transformer–style temporal modeling within RL policies.

Reproducibility and Artifacts

- https://github.com/AranyaAryaman/Portfolio-Optimization

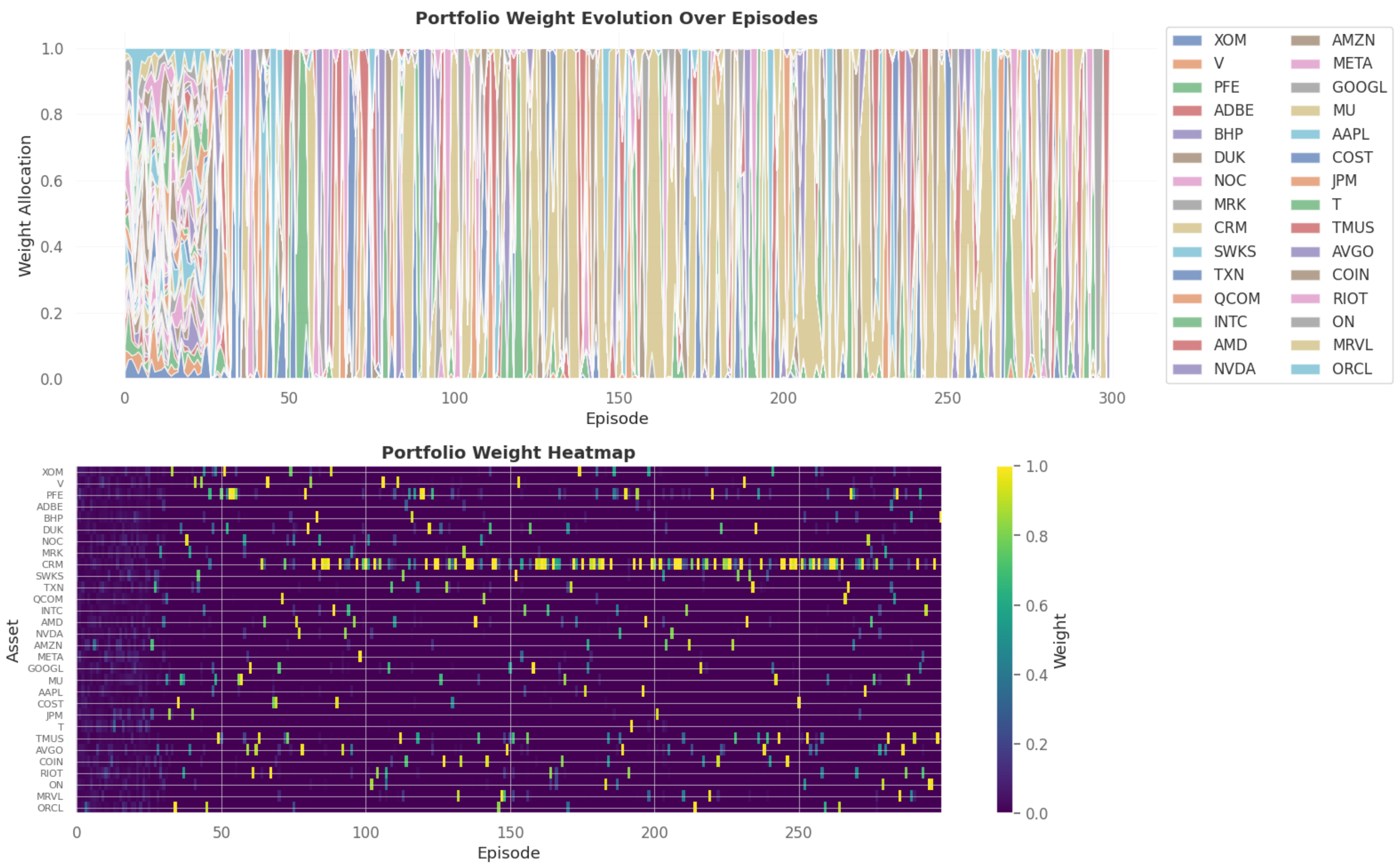

Figure 3: Portfolio weight evolution and heatmap for the SAC agent over training episodes.

References

[1] J. Ayala, M. García-Torres, J. L. V. Noguera, F. Gómez-Vela, and F. Divina. Technical analysis strategy optimization using a machine learning approach in stock market indices. *Applied Intelligence*, 2021.

[2] Y. Xia et al. A review of feature selection and optimization algorithms for superior-dimensional data analysis. *Information Sciences*, 2023.

[3] Nirajan Jha. Understanding feature selection techniques in machine learning. *Medium*, 2024. Online article.

[4] Htet Htet Htun, Michael Biehl, and Nicolai Petkov. Survey of feature selection and extraction techniques for stock market prediction. *Journal of Financial Data Science*, 2023.

[5] Xueyi Cheng. A comprehensive study of feature selection techniques in machine learning models. *Machine Learning Review*, 2024.

[6] Yuqing He, Kamaladdin Fataliyev, and Lipo Wang. Feature selection for stock market analysis. *International Journal of Computer Applications*, 2013.

[7] C. Wan. Feature selection paradigms. *Pattern Recognition Letters*, 2019.

[8] Seyedali Mirjalili. *Genetic Algorithm*. Springer, 2019.

[9] T. Dokeroglu, A. Deniz, and H. E. Kiziloz. A comprehensive survey on recent metaheuristics for feature selection. *Expert Systems with Applications*, 2022.

[10] M. A. Albadr, S. Tiun, M. Ayob, and F. Al-Dhief. Genetic algorithm based on natural selection theory for optimization problems. *Applied Soft Computing*, 2020.

[11] Zhila Yaseen Taha, Abdulhady Abas Abdullah, and Tarik A. Rashid. Optimizing feature selection with genetic algorithms: A review of methods and applications. *Artificial Intelligence Review*, 2025.

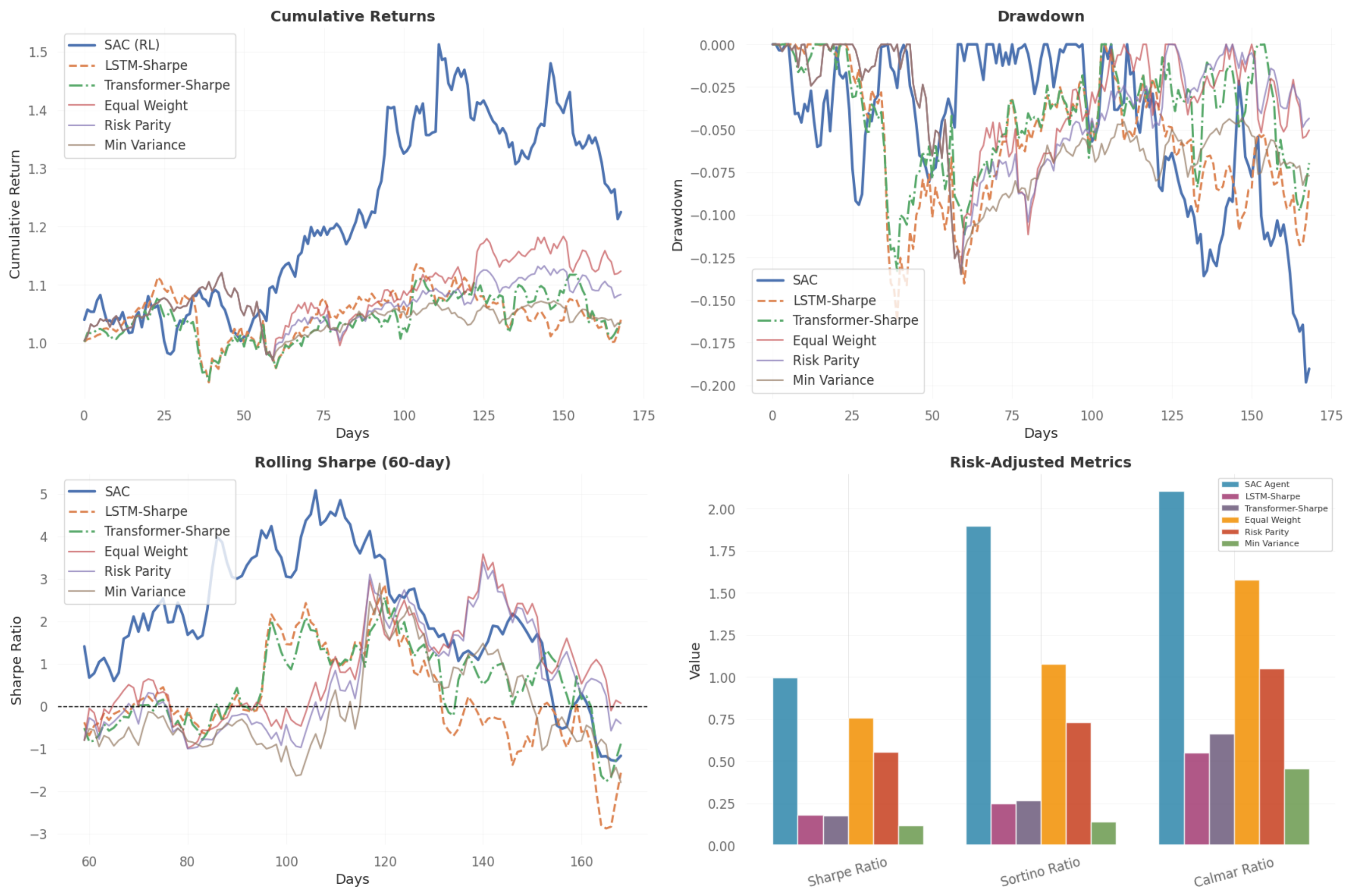

Figure 4: Test–period comparison of SAC, LSTM–Sharpe, Transformer–Sharpe, and classical baselines: cumulative returns, drawdowns, rolling Sharpe (60–day), and risk–adjusted metrics.

[12] A. Sharma and N. Rajput. Feature selection using genetic algorithms for credit scoring and stock market predictions. *Financial Innovation*, 2023.

[13] D. H. Cho, S. H. Moon, and Y. H. Kim. Genetic feature selection applied to kospi and cryptocurrency price prediction. *IEEE Access*, 2021.

[14] A. Suryavanshi and A. Joshi. Hybrid feature selection using genetic algorithms for improved prediction in high-dimensional data. *Expert Systems*, 2022.

[15] Y. Sun and R. Chen. Optimizing financial models using genetic algorithms. *Quantitative Finance*, 2023.

[16] Z. Li and Q. Zhao. Improving forecasting accuracy with genetic algorithms. *Journal of Forecasting*, 2024.

[17] Eric Benhamou, Daniel Saltiel, Jean Ohana, Jamal Atif, and Rida Laraki. Deep reinforcement learning for portfolio allocation. *Applied Mathematical Finance*, 2021.

[18] Ruyu Yan, Jiafei Jin, and Kun Han. Reinforcement learning for deep portfolio optimization. *Finance and Stochastics*, 2024.

[19] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[20] J. Ma. Entropy augmented reinforcement learning. *Neurocomputing*, 2022.

[21] K. K. Yun, S. W. Yoon, and D. Won. Prediction of stock price direction using a hybrid ga-xgboost algorithm with a three-stage feature engineering process. *Expert Systems with Applications*, 2022.

[22] Sali Rasoul, Sodiq Adewole, and Alphonse Akakpo. Feature selection using reinforcement learning. *International Journal of Machine Learning and Cybernetics*, 2021.