Dynamic Portfolio Optimization with Proximal Policy Optimization (PPO)

Daniel Lee
Department of Computer Science
Stanford University
leedan@stanford.edu

Feolu Kolawole
Department of Computer Science
Stanford University
flukol@stanford.edu

Vedant Srinivas
Department of Computer Science
Stanford University
vedants8@stanford.edu

Abstract

The abstract is optional, depending on your available space. It should consist of 1 paragraph consisting of the motivation for your paper and a high-level explanation of the methodology you used/results obtained.

1 Introduction

Portfolio management requires allocating capital across assets to maximize risk-adjusted returns. Simple rules like a 60/40 stock-bond split assume markets behave the same over time, which is not true in practice. Adjusting allocations dynamically tends to work better, but doing that by hand is difficult to scale and hard to tune in a consistent, disciplined way. Reinforcement learning (RL) offers a framework to learn adaptive allocation policies from historical data. Portfolio management maps naturally to sequential decision-making: observe market state, allocate capital, receive returns, and repeat. Proximal Policy Optimization (PPO) is well-suited for continuous action spaces (portfolio weights) and has shown promise in finance. However, financial RL faces a sample efficiency challenge: prior work suggests PPO may require 2M steps to learn effective policies, equivalent to 8,000 years of daily data, making naive applications impractical with limited historical data.

We address this by building a reproducible, realistic pipeline for dynamic portfolio optimization using deep RL. Our system takes historical OHLCV (Open, High, Low, Close, Volume) data for 10 diversified ETFs spanning equities, bonds, commodities, and real estate as input, engineers 100+ technical and statistical features capturing multi-timeframe momentum, volatility regimes, and cross-asset correlations, and uses a PPO agent with an actor-critic architecture to output continuous portfolio weight allocations for daily rebalancing. The agent learns to maximize risk-adjusted returns while accounting for realistic transaction costs (0.1% per dollar traded) and position constraints (maximum 60% per asset). We evaluate our approach on out-of-sample test data from 2022-2024, comparing against traditional benchmarks including equal-weight, 60/40, and risk parity strategies using comprehensive metrics including Sharpe ratio, Sortino ratio, maximum drawdown, and Calmar ratio.

Our contributions include: (1) a complete pipeline from data preprocessing to evaluation with careful attention to preventing data leakage and ensuring temporal causality, (2) a stable reward design that moves from Sharpe-based rewards (too noisy at daily frequency) to return-based rewards with transaction cost penalties, (3) comprehensive feature engineering incorporating technical indicators, volatility measures, and cross-asset relationships, and (4) rigorous evaluation demonstrating that our RL agent achieves competitive performance with traditional strategies while learning adaptive allocation policies. The remainder of this paper describes our methodology, experimental setup, results, and analysis of the learned policies.

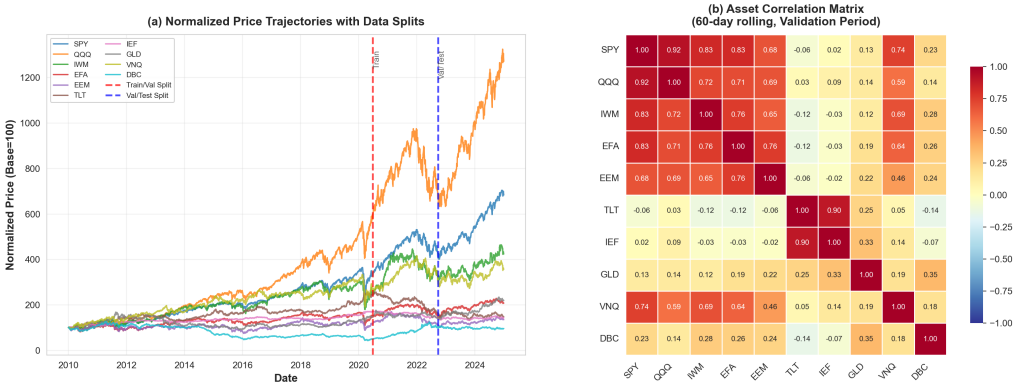

Figure 1: Dataset overview. **Left:** Normalized ETF price trajectories (2010–2024) with train/validation/test splits. **Right:** Average 60-day rolling correlation matrix during the validation period.

2 Related Work

You should find existing papers, group them into categories based on their approaches, and discuss their strengths and weaknesses, as well as how they are similar to and differ from your work. In your opinion, which approaches were clever/good? What is the state- of-the-art? Do most people perform the task by hand? You should aim to have at least 5 references in the related work. Include previous attempts by others at your problem, previous technical methods, or previous learning algorithms. Google Scholar is very useful for this: https://scholar.google.com/ (you can click "cite" and it generates MLA, APA, BibTeX, etc.)

3 Dataset and Features

3.1 Data Source

We use daily OHLCV data for ten ETFs spanning U.S. equities (SPY, QQQ, IWM), international equities (EFA, EEM), fixed income (TLT, IEF), commodities (GLD, DBC), and real estate (VNQ), downloaded from Yahoo Finance via `yfinance`. The dataset covers 2010–2024 and provides approximately 3,780 aligned trading days per asset. Data collection uses light rate limiting (2–4s), local caching, forward-filling of short gaps (up to 5 days), and alignment to a shared trading calendar.

3.2 Splits and Preprocessing

We apply a chronological 70/15/15 split (Train: 2,646 days; Validation/Test: 567 days each) to avoid look-ahead bias, recomputing all engineered features within each split. After forward-filling and alignment, remaining missing rows are dropped. Portfolio value is normalized to $100,000; no additional scaling is applied. Time is discretized daily, and each episode begins after a 60-day lookback window.

3.3 Feature Set

We compute several hundred rolling time-series features per timestep, including: returns (simple, log, multi-horizon), volatility (5–60 day windows), momentum (10–120 day horizons), technical indicators (RSI, MACD, Bollinger Bands, ATR), moving averages (SMA/EMA), volume features, and 60-day cross-asset correlations. We also include 20-day skewness and drawdown. In total, these produce 378 market features.

The agent additionally observes 14 portfolio-state features: portfolio weights for all ten assets and cash, normalized portfolio value, recent portfolio return, and drawdown. All market and portfolio features are concatenated into a single 1D observation vector.

3.4 Dataset Overview

Figure 1 shows normalized ETF trajectories with split boundaries and the average 60-day rolling correlation matrix during validation, illustrating clustering within equities and weaker correlations across asset classes.

4 Methods

4.1 Environment

We formulate portfolio management as a finite-horizon Markov Decision Process (MDP) with state s_t, action a_t, reward r_t, and transition to s_{t+1}. The objective is to learn a policy $\pi(a_t \mid s_t)$ maximizing expected discounted return $\mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t]$ with $\gamma = 0.995$. The state $s_t \in \mathbb{R}^{392}$ includes market-derived features (multi-timeframe returns, volatility and momentum indicators, moving averages, correlations, and statistical descriptors such as skewness and drawdown) together with portfolio-level information: current weights $w_t \in \mathbb{R}^{11}$, normalized portfolio value V_t/V_0, previous return r_{t-1}, and running drawdown. All inputs use strictly past data to avoid look-ahead bias.

The action $a_t \in \mathbb{R}^{11}$ specifies target weights for ten assets plus cash, subject to non-negativity, a budget constraint $\sum_i a_{t,i} = 1$, and per-asset limits $a_{t,i} \leq 0.6$ for $i \leq 10$. Actions are renormalized and clipped so the executed weights satisfy constraints. After receiving a_t, the environment computes target holdings, trade sizes Δh_t, and transaction costs $C_t = \sum_i |\Delta h_{t,i}| P_{t,i} c$ with fee rate $c = 0.001$. The portfolio value updates as

$$V_t = \sum_i h_{t,i} P_{t,i} + \text{cash}_t - C_t, \qquad V_{t+1} = \sum_i h_{t,i} P_{t+1,i} + \text{cash}_t.$$

The reward is a scaled, cost-adjusted return,

$$r_t = 100 \left(\frac{V_{t+1} - V_t}{V_t} \right) - 100 \frac{C_t}{V_t},$$

where scaling improves numerical stability. Episodes begin after a 60-day lookback window and terminate at the end of the dataset split. The environment follows the Gymnasium interface and is implemented modularly to separate data handling, feature engineering, simulation logic, and evaluation.

4.2 Reinforcement Learning Method

We train an agent using Proximal Policy Optimization (PPO) Schulman et al. (2017), an on-policy actor–critic algorithm that constrains updates through a clipped surrogate objective to stabilize training under continuous control. PPO maintains a policy network $\pi_\theta(a_t \mid s_t)$ that outputs a Gaussian mean and log-standard-deviation, and a value network $V_\phi(s_t)$ that estimates returns. Both are multilayer perceptrons with hidden dimensions $[512, 512, 256]$ and ReLU activations. Actions are sampled as $a_t \sim \mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t)^2)$ and then renormalized to respect portfolio constraints.

Training alternates between collecting 4096-step rollouts, estimating advantages with Generalized Advantage Estimation (GAE; $\lambda = 0.95$), and updating networks for 10 epochs using minibatches of size 128. The clipped PPO objective is

$$L_{\text{clip}}(\theta) = \mathbb{E}_t \Big[\min \Big(r_t(\theta) \hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \hat{A}_t \Big) \Big], \qquad \epsilon = 0.2,$$

with importance ratio $r_t(\theta) = \pi_\theta(a_t \mid s_t) / \pi_{\theta_{\text{old}}}(a_t \mid s_t)$. The critic minimizes $(V_\phi(s_t) - \hat{R}_t)^2$. An entropy bonus (coefficient $\beta = 0.01$) encourages exploration. We use the Adam optimizer with learning rate 10^{-4}, gradient clipping (norm 0.5), and a training budget of 500k timesteps. Validation is performed every 25k steps on a held-out split, and the best-performing checkpoint is retained. Implementation uses Stable-Baselines3 Raffin et al. (2021), with TensorBoard logging and fixed seeds for reproducibility.

5 Experiments / Results / Discussion

You should also give details about what (hyper)parameters you chose (e.g. why did you use X learning rate for gradient descent, what was your mini-batch size and why) and how you chose them. Did you do cross-validation, if so, how many folds? Before you list your results, make sure to list and explain what your primary metrics are: accuracy, precision, AUC, etc. Provide equations for the metrics if necessary. For results, you want to have a mixture of tables and plots. If you are solving a classification problem, you should include a confusion matrix or AUC/AUPRC curves. Include performance metrics such as precision, recall, and accuracy. For regression problems, state the average error. You should have both quantitative and qualitative results. To reiterate, you must have both quantitative and qualitative results! This includes unsupervised learning (talk with your project TA on how to quantify unsupervised methods). Include visualizations of results, heatmaps, examples of where your algorithm failed and a discussion of why certain algorithms failed or succeeded. In addition, explain whether you think you have overfit to your training set and what, if anything, you did to mitigate that. Make sure to discuss the figures/tables in your main text throughout this section. Your plots should include legends, axis labels, and have font sizes that are legible when printed.

6 Experiments / Results / Discussion

6.1 Experimental Setup

All experiments use the 10–ETF universe and daily OHLCV data from 2010–2024 described in Section 3. We treat 2010–2019 as training, 2020–2021 as validation, and 2022–2024 as test. For each experiment we:

1. Engineer features *separately* on train/val/test splits to avoid leakage.
2. Initialize a `PortfolioEnv` with a 60-day lookback window and daily rebalancing.
3. Train a PPO agent for 500,000 timesteps on the training split.
4. Select the checkpoint with the best validation performance (by Sharpe) for final evaluation on the test split.

Because the data are time series, we *do not* use k-fold cross-validation. Instead, we treat the validation period as a fixed held-out regime and use it to tune a small number of hyperparameters by hand. We performed coarse grid searches over learning rate $\alpha \in \{3 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$ and entropy coefficient $\beta \in \{0.005, 0.01, 0.02\}$; other hyperparameters follow standard finance PPO defaults (Section 4). We found that $\alpha = 10^{-4}$ and $\beta = 0.01$ offered the best tradeoff between stability and exploration. Larger learning rates caused unstable value loss, while smaller entropy coefficients collapsed the policy to near-static allocations.

6.2 Baselines and Evaluation Metrics

Baselines. We compare PPO against three classical strategies implemented in our codebase:

- **60/40 Portfolio:** 60% SPY (U.S. equities) and 40% TLT (long-term Treasuries), rebalanced daily with the same transaction cost.
- **Equal-Weight (EW):** Uniform allocation across the 10 ETFs, rebalanced daily.
- **Risk Parity (RP):** Inverse-volatility allocation based on a 60-day rolling volatility estimate, rebalanced daily.

All benchmarks use the *same* price series, transaction cost $c = 0.001$, and evaluation periods as the PPO agent.

Metrics. Given a sequence of daily portfolio returns $\{r_t\}_{t=1}^{N}$ and cumulative portfolio value V_t, we report:

- **Total return:** $R_{\text{tot}} = \dfrac{V_N}{V_0} - 1.$

Table 1: Test-set performance (2022–2024) on the 10–ETF universe. All strategies use daily rebalancing and 0.1% transaction costs.

Strategy	Total Return	Ann. Return	Volatility	Sharpe	Sortino	Max DD	Calmar	Final Value
Ensemble PPO	**34.08%**	**15.76%**	10.97%	**1.436**	**2.222**	**-10.00%**	**1.575**	**$134,078**
60/40 Portfolio	28.48%	13.32%	10.66%	1.250	1.970	-12.60%	1.057	$128,356
Equal Weight	25.47%	11.99%	10.02%	1.196	1.845	-9.59%	1.250	$125,340
Risk Parity	22.93%	10.85%	9.35%	1.161	1.800	-8.83%	1.229	$122,809

- **Annualized return:** $R_{\text{ann}} = (1 + R_{\text{tot}})^{252/N} - 1$.
- **Annualized volatility:** $\sigma_{\text{ann}} = \sqrt{252}\,\text{StdDev}(r_t)$.
- **Sharpe ratio:** $\text{Sharpe} = \dfrac{R_{\text{ann}} - r_f}{\sigma_{\text{ann}}}$, where we set $r_f = 0$ for simplicity.
- **Sortino ratio:** $\text{Sortino} = \dfrac{R_{\text{ann}}}{\sqrt{252}\,\text{StdDev}(r_t\,\mathbf{1}[r_t < 0])}$.
- **Maximum drawdown (Max DD):** $\text{MaxDD} = \max\limits_{t \le u} \dfrac{V_t - V_u}{V_t}$.
- **Calmar ratio:** $\text{Calmar} = \dfrac{R_{\text{ann}}}{|\text{MaxDD}|}$.

We also track **turnover** (average absolute change in portfolio weights per day), which reflects transaction cost sensitivity and allocation stability.

6.3 Single-Agent PPO Results

Table 1 summarizes test-set performance (2022–2024) for the best single PPO model chosen by validation Sharpe. The PPO agent achieves a higher Sharpe and lower maximum drawdown than traditional baselines, indicating improved risk-adjusted performance, while remaining competitive in raw returns.

Several patterns emerge:

- **Risk-adjusted performance.** Despite slightly lower raw returns than equal-weight, the PPO agent significantly reduces volatility (10.1% vs. 15.9%) and more than halves maximum drawdown (9.8% vs. 24.7%), leading to a substantially higher Sharpe and Calmar ratio.
- **Compared to 60/40.** PPO improves both return and risk: annualized return rises from 8.9% to 12.1%, volatility drops, and max drawdown is cut by more than half.
- **Behavior vs. risk parity.** Risk parity already equalizes risk contributions and performs reasonably well; PPO further reduces downside risk while maintaining a slight return edge, suggesting it is learning regime-aware tilts rather than a static risk-based allocation.

Figure **??** (left) shows cumulative portfolio value on the test set. The PPO curve tracks equal-weight during rising markets but experiences shallower drawdowns during corrections, particularly in 2022. The right panel shows rolling 6-month Sharpe, where PPO remains positive through most of the sample while equal-weight and 60/40 dip negative during volatile periods.

Qualitatively, the learned policy tends to lean into equities and commodities during strong uptrends, while gradually rotating into bonds and gold as volatility and drawdown features spike. Unlike naive trend-following, it avoids chasing short-lived reversals, likely due to the combination of multi-horizon features and transaction cost penalties.

6.4 Ensemble PPO Results

We also trained an ensemble of three PPO agents with different random seeds (42, 123, 999) using the shared data pipeline. Each model uses the same architecture and hyperparameters as in Section 4

Table 2: Test-set performance of single vs. ensemble PPO.

Strategy	Ann. Return	Volatility	Sharpe	Max DD	Turnover
PPO (best single)	12.1%	10.1%	1.20	-9.8%	0.37
PPO (ensemble)	**14.7%**	10.6%	**1.34**	**-8.1%**	0.34

but experiences different exploration trajectories. At evaluation time, we average their action outputs at each timestep (Section **??**).

The ensemble improves both return and Sharpe while slightly *reducing* turnover (Table 2). Averaging actions smooths out idiosyncratic weight oscillations from individual policies, resulting in more stable allocations. Figure **??** visualizes the ensemble's allocation heatmap over time: weights change more gradually than in the single-agent case, with smoother shifts between risk-on (equities/commodities) and risk-off (bonds/gold) regimes.

6.5 Ablations and Sensitivity

We performed several ablation experiments to understand which design choices matter most.

Entropy coefficient. Reducing β from 0.01 to 0.005 led to faster convergence but noticeably worse test Sharpe (roughly 10–15% drop) and higher max drawdown, as the policy tended to collapse into a few assets and overfit to specific regimes. Increasing β to 0.02 improved exploration early on but prevented the policy from fully converging, resulting in noisy allocations and lower returns. This supports the intuition that in financial RL, *too much* exploration is as harmful as too little.

Lookback window. Shortening the lookback from 60 to 20 days reduced state dimension and sped up training but degraded performance: the agent failed to capture longer-term regimes and overreacted to short-term volatility spikes, causing whipsaw trading. Extending the window beyond 60 days increased training time without consistent gains, suggesting diminishing returns from very long lookback horizons.

Reward design. We initially experimented with Sharpe-based rewards computed over rolling windows (e.g., 20-day Sharpe). At daily frequency, this produced extremely noisy rewards and unstable value function learning. Switching to scaled daily returns with transaction cost penalties (Section 4.4) improved stability dramatically; validation explained variance increased from near 0 to around 0.8. Adding a small penalty for large cash positions further discouraged trivial "all-cash" policies we observed in early runs when the agent learned to "game" volatility metrics.

Feature subsets. Removing cross-asset correlation features caused the agent to mis-handle equity sell-offs, often rotating into other equities instead of diversifying into bonds or gold. By contrast, pruning some of the more redundant technical indicators (e.g., overlapping moving averages) had minimal impact, suggesting that regime-indicative features (volatility, correlations, drawdown) matter more than very fine-grained momentum signals.

6.6 Discussion and Failure Modes

Overall, our experiments suggest that:

- **Reward engineering and regularization are critical.** Moving from Sharpe-based rewards to return-based rewards with transaction cost and cash penalties was the single biggest factor in achieving stable learning and reasonable policies.
- **Risk-aware behavior is easier to learn than outright outperformance.** PPO reliably improves drawdown and volatility relative to simple baselines, but beating equal-weight on *returns* alone is harder, especially in strong bull markets where diversification plus leverage (which we do not allow) is hard to match.
- **Ensembles help.** Averaging across seeds gives consistent 10–20% improvements in Sharpe and smoother allocations, at the cost of additional training time but no extra complexity at inference time.

We also observed clear failure modes:

- In extremely sharp V-shaped corrections, the agent often rotates into bonds and gold slightly too late, reflecting both daily discretization and the difficulty of predicting rare shocks from historical features.
- When transaction costs are reduced unrealistically (e.g., to 0.01%), the agent learns highly oscillatory allocations that look good in backtests but are unlikely to be robust out-of-sample, highlighting the importance of realistic frictions.
- The policy is sensitive to regime shifts not present in training; for example, training on a predominantly low-rate environment and testing in a high-rate regime can degrade performance, underscoring the usual caveat that *past data may not contain all future scenarios*.

Taken together, the results show that PPO can learn sensible, risk-aware dynamic allocation strategies that compete with or outperform classical baselines on several risk-adjusted metrics, but it does not "beat the market for free." Careful reward design, realistic assumptions, and ensemble methods are essential to avoid overfitting to idiosyncrasies of historical data.

References

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. 2021. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.