Graphs and Meta Reinforcement Learning for Portfolio Management

Dhruv Manani
dmanani@stanford.edu

Churan He
churanhe@stanford.edu

Extended Abstract

In this report, we study daily portfolio allocation as a deep reinforcement learning problem. We begin with the policy optimization question: PPO is our main on-policy backbone, SAC is our off-policy contrast, and a reward shaping sweep tests whether explicit drawdown and turnover penalties can control risk. We then move from optimizer and reward choices to architecture. The PPO training path adds graph structure through GATv2 and temporal memory through a GRU over rolling daily embeddings, while the recurrent PPO path adds an LSTM for the meta-learning experiments. A graph encoder captures which stocks move together, while meta-reinforcement learning captures which market regime the agent is in. The central question is whether these spatial and temporal structures are complementary enough to improve risk-adjusted return.

Method. We build on the GATv2 plus PPO portfolio agent of Aresh (CS224R 2025) and extend it along three axes that prior surveys flag as open. We enrich the state from 8 to 26 per-stock features, we scale the universe from 8 names to 29 and then 99, and we add temporal structure through recurrence. Before adding structure, we test whether a different optimizer or a different scalar reward is enough: SAC reuses transitions through a replay buffer, and the reward sweep varies drawdown and turnover weights. We then implement the architectural ladder in PPO: a flat MLP baseline, a GATv2 graph feature extractor, a GRU temporal extractor over rolling windows, and a GNN plus GRU variant where the GRU runs over graph embeddings. The meta-learning branch replaces this plain rolling-window memory with an LSTM policy trained with the RL^2 recipe, where each task is a random 60-day market window and the observation carries the previous action and reward. The combined agent places the GATv2 spatial encoder underneath the meta-trained LSTM, so one policy sees both the graph and the regime. We hold the PPO hyperparameters of Aresh fixed across every PPO agent, which makes architecture the main variable in the ablation.

Results. On the held-out 2023 to 2025 window the combined agent is our best result. Across three seeds it reaches a mean Sharpe of 1.66 ± 0.26 and a mean return of $102\% \pm 32\%$, which is higher on both axes than the meta-RL agent (1.45 ± 0.16) and the graph agent (1.34 ± 0.41). The three-seed study also corrects our own poster, where single-seed numbers turned out to be optimistic top draws. The graph agent alone is the most seed-sensitive of the three and on the mean it only ties the equal-weight benchmark. The meta-RL agent is the most stable. The combined agent inherits the stability of meta-RL and the upside of the graph.

Insights and limitations. The headline message is that spatial and temporal structure stack rather than compete. The graph reduces concentration risk by reasoning over co-movement, and the meta-learning recipe, not the recurrent cell by itself, is what makes memory transfer to unseen regimes. We also report honest negative results. An off-policy SAC agent over traded and lost to simple buy-and-hold, and a plain recurrent agent trained on one long timeline overfit that timeline. The main limitations are a backtest cost of one basis point that is lower than live trading, trailing twelve-month fundamentals that carry some noise, and survivorship in the chosen universe.

1 Introduction

Portfolio management means allocating capital across a set of assets over time to maximize risk-adjusted return, the return earned per unit of risk taken. It is a natural sequential decision problem. An agent observes a noisy view of the market, chooses how to weight each asset, and learns from the realized outcome. Classical mean-variance optimization [Markowitz, 1952] treats the task as a single static period and assumes return distributions that rarely hold in practice. Markets are non-stationary and the relationships between assets are nonlinear, so a method that can adapt over time and reason about structure is a better fit.

Reinforcement learning offers exactly that adaptivity, and recent work shows that a graph view of the market helps. Aresh (CS224R 2025) encoded inter-stock correlations as a dynamic graph and fed the resulting embeddings to a PPO agent, beating buy-and-hold on a held-out window [Aresh, 2025]. That work also named three open problems. The reward optimized raw return with no control of drawdown (the fall from the portfolio's running peak) or churn (how much the portfolio trades), the graph used a fixed window that reacts slowly to regime shifts, and the universe was only 8 similar mega-cap names. Surveys of the field reach the same conclusion and list reward design, non-stationarity, and cross-asset modeling as the most open questions [Bai et al., 2024, Hoque et al., 2025].

We organize the study in two steps. First, we ask what can be gained without changing the representation. PPO is our main optimizer, and SAC is the off-policy contrast: its replay buffer improves sample efficiency in many domains, but in a non-stationary market those old transitions may describe a regime that has already passed. We also sweep drawdown and turnover penalties in the PPO reward, evaluating each setting on the held-out window and comparing Sharpe and Calmar changes against the unshaped same-seed baseline. This tells us whether risk control can be solved by the optimizer or scalar objective alone.

Second, we ask which representation changes matter. The PPO trainer lets us add a GATv2 graph encoder, a GRU over rolling windows of daily embeddings, or both together. This gives a plain temporal-memory baseline before we move to the stronger LSTM meta-learning setup. A graph captures the spatial fact of which stocks co-move at the moment. A meta-trained recurrent policy captures the temporal fact of which regime the market is in. We build both, measure each in isolation, and then combine them. Our contributions are the following.

- We extend the graph plus PPO agent with a richer 26-feature state that adds macro signals, cross-sectional signals, sector identity, and fundamentals pulled from SEC filings, and we scale the universe from 8 to 29 and then 99 names.
- We implement a meta-reinforcement learning agent using the RL^2 recipe, and we show that the random task distribution, not the recurrent cell on its own, is what makes memory useful out-of-sample.
- We introduce a combined agent that places a GATv2 spatial encoder beneath a meta-trained LSTM, and we show across three seeds that it beats both parent agents on Sharpe and return.
- We run a clean architecture ablation with all PPO hyperparameters held fixed, plus SAC and reward shaping diagnostics that explain why the final gains come from spatial and temporal structure rather than only from the optimizer or reward.

2 Related Work

Deep RL for trading. EIIE [Jiang et al., 2017] introduced a modern deep RL framework for portfolio management with a per-asset convolutional feature extractor. It ignores inter-asset dependence, which leads to concentration risk. FinRL [Liu et al., 2022] provides a standard environment with PPO, SAC, A2C, and DDPG baselines and sets strong flat-feature references, but it does not model cross-asset structure. Kabbani and Duman [Kabbani and Duman, 2022] used self-attention to pool signals across many names and showed that cross-asset reasoning raises the Sharpe ratio, though the attention is implicit and does not produce an interpretable graph.

Graph and temporal extensions. Aresh [Aresh, 2025] built a GATv2 encoder that produces dynamic correlation graph embeddings for a PPO actor-critic and identified reward design, graph adaptivity, and universe size as open limitations. Sarlakifar et al. [Sarlakifar et al., 2025] showed that temporal

sequence modeling over price states improves trading performance, which motivates a memory layer over graph embeddings. Nixon Raj [Nixon Raj, 2025] showed that conditioning on a latent market regime improves out-of-sample Sharpe, which supports a meta-learning view of the temporal axis.

3 Method

3.1 Problem formulation

We frame the task as a partially observed Markov decision process. The true market state, which includes positioning and macro regime, is never observed. We only see noisy prices and volume. We follow the standard MDP notation of the course, with states $\mathbf{s}_t$, actions $\mathbf{a}_t$, a policy $\pi_\theta(\mathbf{a} \mid \mathbf{s})$, and a reward $r(\mathbf{s}, \mathbf{a})$. The objective is the expected return

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\Big[\textstyle\sum_t r(\mathbf{s}_t, \mathbf{a}_t)\Big], \qquad p_\theta(\tau) = p(\mathbf{s}_1) \textstyle\prod_t \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)\, p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t). \tag{1}$$

The state at day t holds the per-stock feature matrix and a correlation graph over the assets. We grew the feature set from the 8 technical signals of Aresh to 26 per-stock features. The additions are two macro signals (the VIX and the 10-year Treasury yield), two cross-sectional signals (beta to the market and distance to the 52-week high), eight sector indicators, and six fundamentals pulled from SEC filings. The action is a long-only weight vector $\mathbf{w}_t \in [0,1]^N$ with $\sum_i w_t^i = 1$, where long-only means the agent can only buy assets and never short sell them. We approximate the partial observability in two ways that the course motivates, namely state enrichment through rolling features and temporal memory through recurrence.

The base reward is the clipped daily log return. We also study a shaped reward that adds a drawdown term and a turnover term,

$$r_t = \mathrm{clip}\Big(\log \frac{V_{t+1}}{V_t}, -\delta, \delta\Big) + \lambda_{\mathrm{dd}}\, \mathrm{drawdown}_t - \lambda_{\mathrm{to}}\, \mathrm{turnover}_t, \tag{2}$$

where V_t is portfolio value, $\delta = 0.05$ caps the daily move, $\mathrm{drawdown}_t = V_t / \max_{\tau \le t} V_\tau - 1 \le 0$ measures the current loss from the running peak, and $\mathrm{turnover}_t = \sum_i |w_t^i - w_{t-1}^i|$ measures how much the agent traded. Setting both weights to zero recovers the base reward.

3.2 Policy optimization backbone

Every PPO agent shares the same optimization backbone. We start from the policy gradient, which the course derives as

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\Big[\textstyle\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t)\, \hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t)\Big], \tag{3}$$

where the advantage $A^\pi(\mathbf{s}, \mathbf{a}) = Q^\pi(\mathbf{s}, \mathbf{a}) - V^\pi(\mathbf{s})$ says how much better an action is than the policy average. A critic V_ϕ supplies the baseline. The course estimates the advantage with bootstrapped returns. The one-step form is

$$\hat{A}^\pi(\mathbf{s}_t, \mathbf{a}_t) \approx r(\mathbf{s}_t, \mathbf{a}_t) + \gamma V_\phi(\mathbf{s}_{t+1}) - V_\phi(\mathbf{s}_t), \tag{4}$$

and we use the generalized advantage estimator with $\lambda = 0.95$ to trade bias for variance, which the course presents as one of the practical tricks on top of the actor-critic baseline. To keep each update stable, PPO maximizes the clipped surrogate the course derives. With the probability ratio $\rho_t(\theta) = \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) / \pi_{\theta_{\mathrm{old}}}(\mathbf{a}_t \mid \mathbf{s}_t)$,

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(\rho_t(\theta)\, \hat{A}_t,\ \mathrm{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\, \hat{A}_t\big)\Big]. \tag{5}$$

The clip keeps each update inside a trust region, which is the idea the course motivates so a freshly collected batch can be reused for several epochs without the policy drifting too far. We use the

hyperparameters of Aresh for every PPO variant, namely 1024 steps per rollout, a batch of 2048, a learning rate of 10^{-4}, an entropy coefficient of 5×10^{-3}, $\gamma = 0.99$, a clip of 0.2, ten epochs per update, and 500 thousand training steps.

SAC contrast. SAC [Haarnoja et al., 2018] is our off-policy contrast to this PPO backbone. Instead of optimizing only expected return, SAC maximizes a maximum entropy objective,

$$J_{\mathrm{SAC}}(\pi) = \mathbb{E}_{\tau\sim\pi}\Big[\sum_t \gamma^t \big(r(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi(\mathbf{a}_t \mid \mathbf{s}_t)\big)\Big], \tag{6}$$

where α controls the strength of the entropy bonus. In the implementation, SAC uses the same portfolio environment and MLP observation interface, but replaces rollouts with a replay buffer of 200 thousand transitions, starts learning after 10 thousand transitions, samples batches of 256, and tunes α automatically. This makes SAC a useful diagnostic: if the main limitation were sample efficiency, replay should help. In our setting, however, the buffer can reuse transitions from old market regimes, and the entropy objective encourages extra exploration in a domain where exploration appears as real trading turnover. The SAC experiment therefore tests whether an off-policy optimizer is enough before we add graph or recurrent structure.

3.3 Reward shaping diagnostic

The second diagnostic changes the objective but keeps the PPO architecture fixed. We sweep the reward shaping weights λ_{dd} and λ_{to} on a grid, train one PPO policy for each setting, and evaluate each policy on the held-out window. For a metric M such as Sharpe or Calmar, the reported sweep value is the same-seed delta from the unshaped run,

$$(\lambda_{\mathrm{dd}}, \lambda_{\mathrm{to}}) \in \{0, 0.1, 0.5\} \times \{0, 0.05, 0.1\}, \qquad \Delta M_s(\lambda_{\mathrm{dd}}, \lambda_{\mathrm{to}}) = M_s(\lambda_{\mathrm{dd}}, \lambda_{\mathrm{to}}) - M_s(0, 0). \tag{7}$$

This isolates whether explicit drawdown and turnover penalties can match what the graph achieves through representation. If reward shaping alone controlled risk, this would be the simplest fix. If it does not, the case for spatial structure is stronger.

3.4 Spatial structure: a GATv2 graph encoder

A flat policy scores each stock from its own features and can pile weight into names that crash together. A graph addresses exactly this. Each day we compute a 60-day rolling correlation matrix across the assets and keep the top $K = 4$ strongest edges per node, which gives a fresh graph that adapts as correlations move. A two-layer GATv2 encoder [Brody et al., 2022] then mixes information across neighbors. For node i with feature $\mathbf{h}_i$ and neighbor j with edge feature $\mathbf{e}_{ij}$, the attention score and the update are

$$\alpha_{ij} = \mathrm{softmax}_j\Big(\mathbf{a}^\top \mathrm{LeakyReLU}\left(\mathbf{W}[\mathbf{h}_i \,\|\, \mathbf{h}_j \,\|\, \mathbf{e}_{ij}]\right)\Big), \qquad \mathbf{h}'_i = \sigma\Big(\textstyle\sum_{j\in\mathcal{N}(i)} \alpha_{ij}\, \mathbf{W}\mathbf{h}_j\Big). \tag{8}$$

GATv2 applies the linear map before the nonlinearity, so the attention is a genuine function of both endpoints rather than a fixed ranking, which the original GAT could not express [Veličković et al., 2018]. The encoder output replaces the flat features that feed the PPO actor and critic. We use an embedding width of 64 on the 29-name universe and 32 on the 99-name universe. The encoder output is flattened to a vector of size N times the width before it reaches the policy head, so this dimension grows with the universe. At width 64 the head input would jump from about 1900 values on the 29-name universe to about 6300 on the 99-name universe. Halving the width to 32 on the larger universe keeps the head input and its parameter count close to the smaller setting, which matches the scale Aresh used and limits overfitting on the larger graph.

3.5 Temporal structure: recurrence and the RL^2 recipe

Markets shift between calm and volatile regimes, and a memoryless policy cannot tell them apart from a single day. We first add a GRU temporal memory layer to the PPO feature extractor as a direct

test of whether short history helps. For each asset i, the GRU reads a trailing window of $W = 20$ daily embeddings and returns the last hidden state,

$$\mathbf{z}_{i,t} = \mathrm{GRU}\left(\mathbf{x}_{i,t-W+1}, \ldots, \mathbf{x}_{i,t}\right), \qquad \mathbf{x}_{i,t} = \begin{cases} \mathbf{h}_{i,t}, & \text{raw GRU,} \\ \mathrm{GATv2}(\mathbf{h}_{i,t}, \mathcal{G}_t), & \text{GNN + GRU.} \end{cases} \tag{9}$$

The raw GRU path uses per asset features directly, while the GNN + GRU path first embeds each day's correlation graph and then runs the same GRU over the graph embeddings. These agents still train on the one long 2015 to 2022 trajectory, so they test ordinary memory rather than meta adaptation. We also add an LSTM [Hochreiter and Schmidhuber, 1997] through recurrent PPO. As we report below, plain recurrence over one timeline can overfit position in that timeline and fail to transfer.

The fix is meta-reinforcement learning. We follow the course formulation, where a task $\mathcal{T}_i = \{\mathcal{S}_i, \mathcal{A}_i, p_i(\mathbf{s}_1), p_i(\mathbf{s}' \mid \mathbf{s}, \mathbf{a}), r_i\}$ is drawn from a distribution $p(\mathcal{T})$, and the meta objective averages return over that distribution,

$$\max_{\theta} \; \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \Big[\mathbb{E}_{\pi_\theta} \big[\textstyle\sum_t \gamma^t r_i(\mathbf{s}_t, \mathbf{a}_t) \big] \Big]. \tag{10}$$

We implement the RL^2 construction [Duan et al., 2016]. Each task is a random 60-day market window sampled from the training period, and we replay each window four times. The recurrent input carries the previous action and reward,

$$\text{input}_t = (\mathbf{s}_t, \; \mathbf{a}_{t-1}, \; r_{t-1}), \tag{11}$$

and the hidden state persists across the sub-episodes of a task. This lets the recurrent network perform task inference inside its hidden state, producing an inferred regime code $\hat{\mathbf{z}}$ that the policy acts on. Because the agent sees hundreds of different windows, it cannot memorize any one timeline and must learn the general skill of reading the recent data and inferring the regime. At evaluation we run a single continuous pass over 2023 to 2025 so the hidden state grows over the whole window.

3.6 The combined agent

The combined agent puts both structures in one policy. A feature extractor parses the meta observation, which is laid out as node features, then a date index, then the previous action and reward. It runs the GATv2 encoder on the node features and that day's graph, then passes the previous action and reward straight through to the recurrent policy. The LSTM therefore reasons over graph embeddings rather than flat-features, while still receiving the RL^2 signal it needs for task inference. Figure 1 shows where this agent sits in the family we explored.

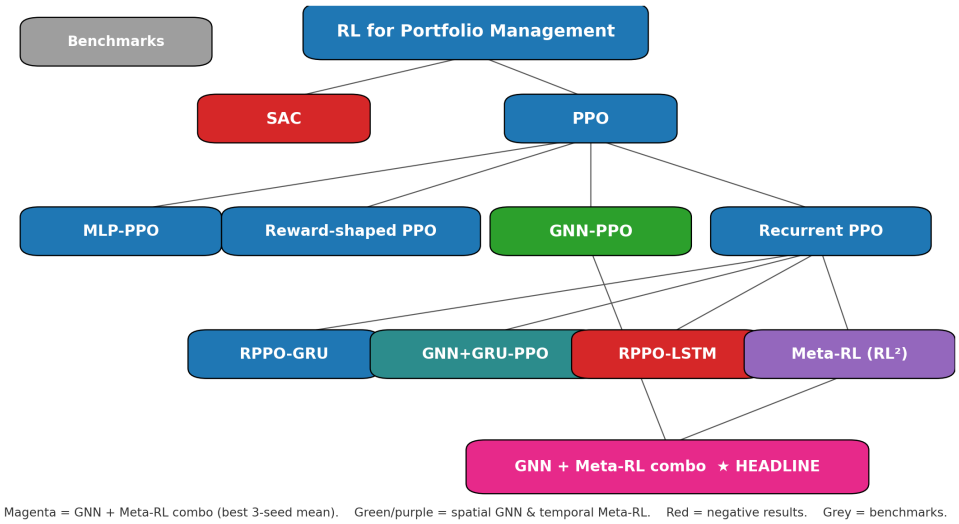

Figure 1: The family of agents we explored. PPO is the spine, with all hyperparameters held fixed. The graph agent adds spatial structure, the recurrent family adds temporal structure, and the combined agent draws from both the graph branch and the meta-RL branch.

4 Experimental Setup

Data and environment. The primary universe is the current Dow Jones 30 minus one name that lacks clean history before 2019, which leaves 29 stocks. We use the S&P 100 minus three recent listings, which leaves 99 stocks, for the scaling study. Prices come from yfinance and fundamentals come from SEC EDGAR filings. We train on 2015 to 2022 and evaluate on the held-out 2023 to 2025 window, matching Aresh for a direct comparison. The agent rebalances daily, starts with one million dollars, pays a transaction fee of one basis point (0.01 percent) on traded notional (the dollar value of shares bought and sold), and is long-only. The base reward clips the daily log return at plus or minus five percent.

Architectures. The graph encoder is a two-layer GATv2 over the daily top-four correlation graph. The recurrent agents use an LSTM or a GRU with a hidden size of 128. The meta-RL agents use 60-day window tasks with four replays per task and the augmented observation. Holding the PPO hyperparameters fixed across all of these makes architecture the only variable.

Implementation. The base RL algorithms, namely PPO, SAC, and recurrent PPO, come from Stable-Baselines3 and its contrib package [Raffin et al., 2021]. Our own work is everything that makes the study novel, which is the GATv2 graph encoder, the combined agent feature extractor that joins the graph to the recurrent policy, the meta environment wrapper that builds the RL^2 task distribution, the shaped reward, and the full data and feature pipeline. We did not reimplement the base optimizers.

Metrics and baselines. We report the annualized Sharpe ratio (average return divided by its volatility) as the primary risk-adjusted metric, along with total return, maximum drawdown, Sortino (like Sharpe but penalizing only downside moves), Calmar (return divided by the worst drawdown), and turnover (the fraction of the portfolio traded each day). Baselines are equal-weight buy-and-hold on the Dow, the SPY market index, and the off-the-shelf FinRL flat-feature agent. The headline agents are re-run on three seeds so we can report a mean and a standard deviation rather than a single draw.

5 Results

5.1 Headline comparison

Figure 2 overlays the equity curves of the Dow 29 agents against the two buy-and-hold benchmarks and the FinRL reference. The combined agent is the top line, ending well above every other agent.

Table 1 lists the single-seed numbers for every agent we built, sorted by return. Every structured RL agent beats both benchmarks on return, with one exception. The plain recurrent LSTM agent lands just below the equal-weight Dow at 69.2 percent against 70.2 percent. The clear floors are the off-the-shelf FinRL agent, the off-policy SAC agent, and the flat agent on the larger universe, which is the expected ordering.

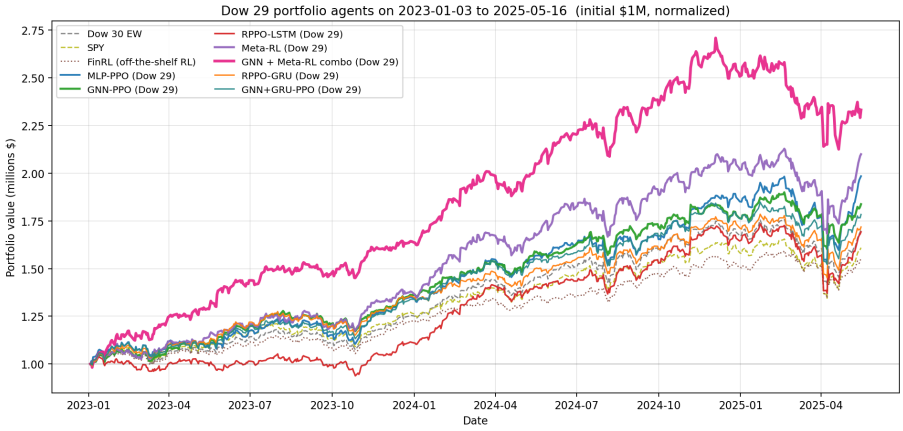

Figure 2: Equity curves on the held-out 2023 to 2025 window, normalized to one million dollars at the start. The combined agent (bold magenta) is the strongest, followed by the meta-RL agent. Benchmarks are dashed.

Table 1: Single-seed results on the held-out window. Learned agents are Dow 29 unless marked. This is the seed 0 view. The three-seed view in Table 2 is the one to trust for the headline agents.

Agent	Universe	Sharpe	Max drawdown	Total return
GNN + Meta-RL combo	Dow 29	**1.877**	−21.5%	**133.1%**
Meta-RL (RL^2, LSTM)	Dow 29	1.631	−20.9%	109.9%
Flat MLP	Dow 29	1.510	−23.1%	98.4%
Reward-shaped (λ_{dd}=0.5)	Dow 29	1.614	−21.8%	90.1%
GNN (GATv2)	Dow 29	1.714	−15.7%	83.7%
GNN + GRU	Dow 29	1.510	−17.7%	78.4%
GNN (GATv2)	S&P 99	1.646	−14.9%	75.8%
Recurrent (GRU)	Dow 29	1.400	−18.7%	71.8%
Recurrent (LSTM)	Dow 29	1.291	−20.9%	69.2%
Flat MLP	S&P 99	1.399	−14.2%	57.1%
SAC	Dow 29	1.040	−21.2%	55.0%
Dow 30 equal-weight	Dow 30	1.373	−19.5%	70.2%
SPY market index	S&P 500	1.245	−18.8%	60.9%
FinRL flat agent	Dow 29	1.376	−15.6%	55.4%

5.2 Three-seed robustness

A single seed is especially risky in this setting. PPO training is stochastic: weight initialization, rollout sampling, and action noise all change which local policy the optimizer settles on. In a non-stationary market with a short held-out window, those differences can flip whether an agent looks strong or weak even when the architecture and hyperparameters are identical. A lucky seed can therefore land on an aggressive policy that rides one favorable regime, while an unlucky seed learns a timid policy that gives up return. Reporting one draw hides that spread and can reverse the ranking between agents.

We saw this firsthand on the graph agent. Our poster quoted a single-seed Sharpe of 1.714, which beat the equal-weight benchmark and made the graph look like the clear winner. That number came from seed 0. Seeds 1 and 2 tell a different story. To get an honest read, we retrained the three headline agents on seeds 0, 1, and 2 and report the mean and standard deviation in Table 2 and Figure 3.

Table 2: Three-seed robustness on Dow 29. Mean and standard deviation over seeds 0, 1, 2.

Agent	Sharpe (mean ± sd)	Return (mean ± sd)	Per seed Sharpe
GNN + Meta-RL combo	$\mathbf{1.656 \pm 0.262}$	$\mathbf{102.0\% \pm 31.8\%}$	1.877 / 1.367 / 1.724
Meta-RL (RL^2)	1.451 ± 0.160	$76.1\% \pm 29.7\%$	1.631 / 1.398 / 1.324
GNN (GATv2)	1.335 ± 0.406	$63.4\% \pm 20.4\%$	1.714 / 0.907 / 1.384
Dow 30 equal-weight	1.373	70.2%	baseline

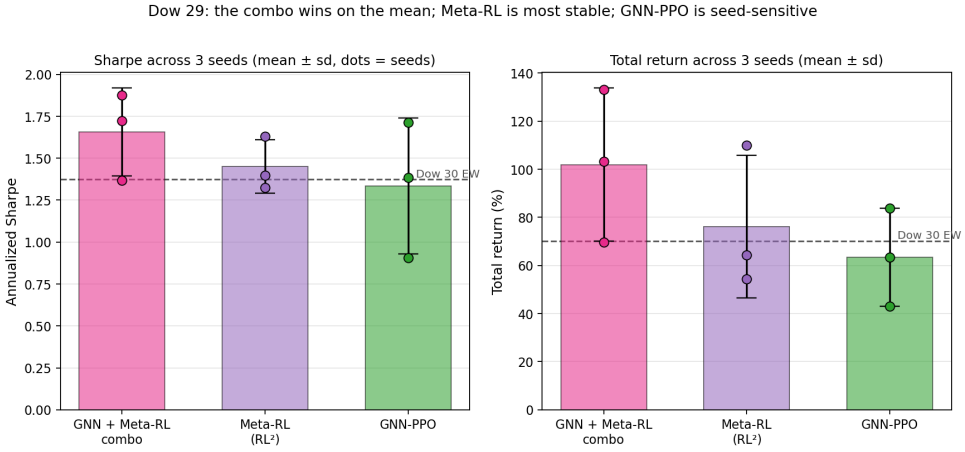

Figure 3: Sharpe and return across three seeds, with the mean as a bar, the standard deviation as an error bar, and the individual seeds as dots. The dashed line is the equal-weight benchmark. The combined agent wins on the mean, the meta-RL agent is the most stable, and the graph agent is the most seed-sensitive.

The three-seed view changes the conclusion. The combined agent wins on the mean for both Sharpe and return, and it is not a lucky draw because it leads on the average over three seeds. Its worst seed still roughly matches the benchmark. The meta-RL agent has the tightest spread, and every one of its seeds beats the benchmark, which makes it the most reliable single structure. The graph agent is the most seed-sensitive. Its mean Sharpe only ties the equal-weight benchmark, and one seed dropped to 0.907 while another stayed at 1.714. That 0.6-point gap within the same agent is the side effect of single-seed reporting in RL: one run captured its upside, another captured a failure mode, and neither alone is representative.

The combined agent therefore earns its place on evidence that survives re-seeding. It pairs the upside of the graph with the stability of meta-RL and lands the best mean at a moderate spread. For the rest of the paper we treat the headline three-seed aggregates as the primary result and the single-seed table as a snapshot that can overstate or understate any one agent.

5.3 Reward shaping outcomes

Before adding graph or recurrent structure, we swept the shaped reward on the flat MLP PPO agent. The grid spans $\lambda_{\mathrm{dd}} \in \{0, 0.1, 0.5\}$ and $\lambda_{\mathrm{to}} \in \{0, 0.05, 0.1\}$, with three seeds per setting. Each run is evaluated on the held-out window and compared against the same-seed unshaped baseline, so the heatmaps in Figure 4 show the change in Sharpe and Calmar rather than raw levels.

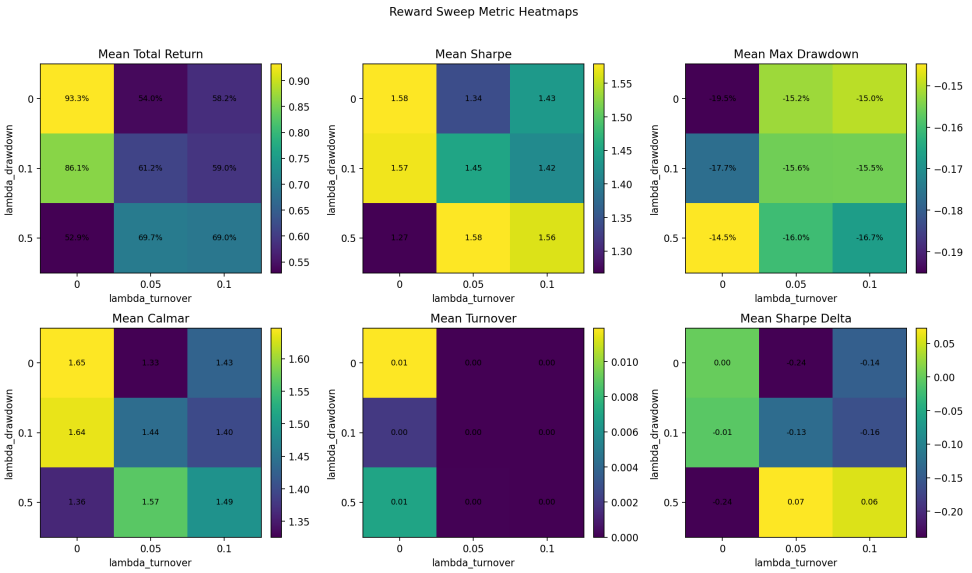

Figure 4: Reward shaping sweep on the flat MLP agent. Each cell is the mean same-seed delta in Sharpe (left) or Calmar (right) relative to the unshaped baseline on that seed. Warmer colors are better. Turnover penalties hurt on both metrics, and only a light drawdown weight with no turnover penalty gives a modest Sharpe lift.

The sweep does not find a scalar reward that replaces architectural risk control. The best mean Sharpe is $\lambda_{\mathrm{dd}} = 0.1$ and $\lambda_{\mathrm{to}} = 0$, which reaches 1.59 ± 0.29 and beats the unshaped baseline by only $+0.06$ on average. The unshaped reward still wins on mean Calmar at 1.64, and every setting that turns on a turnover penalty lowers both Sharpe and Calmar relative to the same-seed baseline. A heavy drawdown weight of 0.5 without turnover does shallow the mean loss a little, to about -16.4% versus -17.2% for the unshaped baseline, but mean Sharpe falls to 1.51 and mean return drops from 81% to 74%, so the trade is modest at best.

The single-seed row in Table 1 shows the $\lambda_{\mathrm{dd}} = 0.5$ agent at 1.614 Sharpe, above the flat baseline, but that is the seed 0 draw. Averaged over three seeds the lift disappears, the same single-seed effect we saw with the graph agent, so the table value overstates what reward shaping actually buys.

The pattern is therefore negative but informative. Penalizing turnover in the reward does not translate into better evaluated turnover or better risk-adjusted return, and drawdown shaping alone does not deliver a clear win on the metrics we care about. That is why we treat reward design as a diagnostic in this study rather than the main lever for risk control.

5.4 What each structure adds

Reading down Table 1, the flat agent already beats buy-and-hold on return but takes the deepest drawdown, since it chases return without controlling co-movement. Adding the graph trades a little return for the shallowest drawdown of any Dow agent, which is the smarter diversification we expected. Adding the meta-RL recipe instead lifts absolute return, the highest among the single-structure agents. The combined agent then sits above both, taking the strongest Sharpe and the strongest return overall. As the reward sweep above shows, shaping the objective helped only modestly on the flat agent, which says that risk control here is more a property of architecture than of the scalar reward alone.

5.5 Negative results

Two agents underperformed and both are informative. SAC reused stale transitions from old regimes through its replay buffer, and its entropy bonus encouraged constant trading, which costs money. It churned the book far more than PPO and finished below both benchmarks. This is direct evidence that on-policy PPO is the better fit for a non-stationary series with trading costs. The plain recurrent LSTM agent fell below the flat agent and just under the equal-weight benchmark on return. Trained

on one long timeline, it encoded position along that timeline rather than a transferable notion of regime. The contrast with the meta-RL agent is sharp. Same architecture, same hyperparameters, opposite result. The only change was the training distribution, which tells us that the random task distribution is the active ingredient, not the recurrent cell.

5.6 Universe scaling

The graph advantage should grow with more assets, because the value comes from relationships and there are more of them. It does. The gap in Sharpe between the graph agent and the flat agent widens from about 0.20 on the 29-name universe to about 0.25 on the 99-name universe. On absolute return the larger flat agent actually slips below the benchmarks while the larger graph agent stays above them, which is another sign that the graph, rather than RL on its own, is what beats the market here. Figure 5 places every agent on the return versus drawdown plane and shows the same ordering visually.

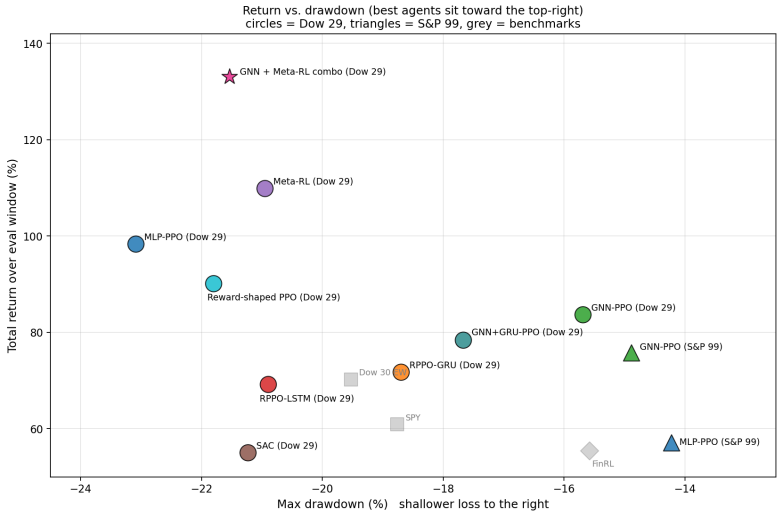

Figure 5: Return against maximum drawdown. The ideal corner is the top right, with high return and a shallow loss. The combined agent (magenta star) sits at the top. Circles are Dow 29 agents, triangles are S&P 99 agents, and grey markers are benchmarks.

6 Discussion

The clearest insight is that spatial and temporal structure are complementary and that combining them helps. The graph answers which assets move together and tightens drawdown. The meta-learning recipe answers which regime we are in and lifts return. Neither subsumes the other, and the combined agent that holds both is our strongest result across seeds. A second insight is methodological. The three-seed study changed our own conclusion. Our single-seed poster numbers were optimistic top draws, the graph agent in particular is high variance, and only a multi-seed view tells the honest story. We think this is a useful caution for the many single-seed results in this application area.

Limitations. The backtest charges one basis point per traded notional, which is lower than the five to seven basis points a live book (a real trading account) would pay, so the high-turnover agents would look worse under realistic costs. A live deployment would also have to handle taxes and wash-sale rules (which block claiming a loss when the same asset is rebought within a short window), neither of which the simulator models, and it would need to turn the learned policy into an actionable set of daily target weights for a real account. The fundamentals use a trailing twelve-month window (the most recent four quarters of company financials) that carries some fourth quarter noise. The universe is chosen from current index members, which introduces survivorship bias, since the set leaves out companies that dropped out of the index and flatters returns.

Future work. The natural next step is to run the combined agent on the larger 99-name universe to confirm that the stacking result holds where the graph advantage is strongest. Beyond that, a turnover penalty on the high-churn larger graph agent and point-in-time fundamentals would both sharpen the realism of the study.

7 Conclusion

This report studied whether graph-based spatial structure and meta-trained temporal structure are complementary for daily portfolio allocation when all PPO hyperparameters are held fixed. On the held-out 2023 to 2025 evaluation window, the combined agent that stacks a GATv2 encoder beneath an RL^2-trained LSTM is the strongest across three random seeds, with a mean annualized Sharpe ratio of 1.66 ± 0.26 and a mean total return of $102\% \pm 32\%$. This beats either part on its own, meta-reinforcement learning alone (1.45 ± 0.16 Sharpe, 76% return) and graph attention alone (1.34 ± 0.41 Sharpe, 63% return). Meta-RL is the more stable single structure, with every seed above the equal-weight benchmark, while the graph agent is the most seed-sensitive and only matches the benchmark on average. Spatial and temporal structure therefore add up rather than replace each other.

Our ablations and negative results show that the gains come from representation, not from the optimizer or the reward alone. An off-policy SAC baseline underperforms buy-and-hold, which shows replay adds little in a non-stationary market with trading costs. A reward-shaping sweep on the flat MLP agent improves mean Sharpe by at most $+0.06$ over the unshaped objective, and turnover penalties consistently reduce both Sharpe and Calmar. Plain LSTM training on a single historical trajectory overfits and fails to transfer, while the same architecture succeeds under the RL^2 task distribution, which points to the training recipe, not the recurrent cell. The Sharpe advantage of the graph agent over a flat policy also grows as the universe expands from 29 to 99 names.

The main result is that stacking spatial and temporal structure beats either one alone. The three-seed study is also a reminder that a high-variance agent can look clearly better on a single seed, so multi-seed evaluation should be the default for RL results in finance.

Team Contributions

Both members contributed to analysis, the poster, and this report. The division of labor stayed close to the proposal, with the adjustments noted below.

- **Dhruv Manani.** Data pipeline and feature engineering, graph construction and the daily correlation snapshots, the GATv2 encoder and the combined agent feature extractor, the meta-RL trainer and the combined agent, the three-seed study, and experiment coordination.
- **Churan He.** Reward function design and the shaping sweep, the GRU temporal memory layer and the recurrent agents, the evaluation pipeline and the metrics, and the FinRL and SAC baselines.

Changes from the proposal. The proposal framed the temporal axis as a GRU memory layer. After the plain recurrent agent overfit, we shifted the temporal axis to the RL^2 meta-learning recipe, which became the temporal half of our headline combined agent. The combined agent itself was listed as future work in the proposal and is now a completed result.

AI Tools Disclosure

We used AI coding assistants for boilerplate and scaffolding, such as argument parsing, data loading, and configuration files, and for writing the plotting and analysis scripts that produce the result figures in this report. The reinforcement learning components that make this study novel were designed and implemented by us. This includes the GATv2 graph encoder, the combined agent feature extractor, the meta-environment that builds the RL^2 task distribution, and the shaped reward, along with the data and feature pipeline. The base PPO, SAC, and recurrent PPO optimizers come from the Stable-Baselines3 library rather than from AI generated code. We also used an assistant for light editing of this report.

References

Nevin Aresh. Graph-based stock market rl agent, 2025. CS224R Final Project, Stanford University.

Yifan Bai et al. A review of reinforcement learning in financial applications. *arXiv preprint arXiv:2411.12746*, 2024.

Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? In *International Conference on Learning Representations (ICLR)*, 2022.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Md. Rashedul Hoque et al. Reinforcement learning in financial decision making: A systematic review. *arXiv preprint arXiv:2512.10913*, 2025.

Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017.

Taylan Kabbani and Ekrem Duman. Deep reinforcement learning approach for trading automation in the stock market. *IEEE Access*, 10:93564–93574, 2022.

Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*, 2022.

Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

G. Nixon Raj. Adaptive and regime-aware rl for portfolio optimization. *arXiv preprint arXiv:2509.14385*, 2025.

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

Faezeh Sarlakifar, Mohammadreza Mohammadzadeh Asl, Sajjad Rezvani Khaledi, and Armin Salimi-Badr. A deep reinforcement learning approach to automated stock trading using xLSTM networks. *arXiv preprint arXiv:2503.09655*, 2025.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.