# REGENT: A Regime-Guided Equity Neural Trading System with Differentiable Graph-Based Portfolio Optimization

Szalay Péter

June 2026

**Abstract**

I present **REGENT** (Regime-guided Equity Neural Trading System), an end-to-end research pipeline for daily, long-only US equity portfolio management. Traditional portfolio optimisation pipelines decouple alpha generation, cross-sectional ranking, and portfolio sizing, leading to misaligned objectives. REGENT addresses this by coupling a temporal-transformer Vector-Quantised Variational Autoencoder (VQ-VAE) for macroeconomic regime classification with **GRASP** (Graph-based Risk-Aware Spatio-temporal Portfolio Agent). GRASP is a four-module differentiable graph neural network that maps per-stock OHLCV and cross-sectional features directly to constrained portfolio weights. The system enforces strict allocation constraints (e.g., long-only, sector bounds, gross exposure limits) natively via a differentiable convex quadratic-program (QP) projection layer, eliminating the need for soft penalty tuning. The entire architecture is trained end-to-end against a composite financial objective that combines Sharpe, Sortino, Conditional Value-at-Risk ($\text{CVaR}_{0.08}$), turnover, and diversification metrics, with no reinforcement-learning reward shaping. I detail the system's mathematical formulations, a walk-forward evaluation protocol with a purged $K$-fold cross-validation ensemble per window, and an execution-fidelity live trading research loop. Across an eight-window non-overlapping walk-forward (2022–2026) the deployed ensemble achieves a mean test Sharpe of 1.16 (median 1.28), a median Calmar of 2.99, and a positive Sharpe in 8/8 windows; chained across all eight test slices the out-of-sample equity compounds to +100.2% against +61.0% for SPY and +83.6% for an equal-weight book of the same 120-stock universe.

**Disclaimer.** This paper and the accompanying software are for research and educational purposes only. They are *not* investment advice and *not* a recommendation to trade any security. All performance figures are historical backtests and walk-forward *simulations* on past data; simulated results have inherent limitations and do not represent actual trading. **Past performance does not predict future results**, and future performance may differ materially, including total loss of capital. Any use of this work is entirely at the reader's own risk.

## Contents

## Glossary of Terms

The architecture is acronym-dense by necessity. The following terms recur throughout; each is also defined at first substantive use.

| Term | Meaning |
|---|---|
| REGENT | Regime-guided Equity Neural Trading System (the full pipeline). |
| GRASP | Graph-based Risk-Aware Spatio-temporal Portfolio agent (the GNN allocator). |
| VQ-VAE | Vector-Quantised Variational Autoencoder (the discrete macro-regime encoder). |
| GNN | Graph Neural Network. |
| BiLSTM | Bidirectional Long Short-Term Memory recurrent layer. |
| GCN | Graph Convolutional Network [12] (the spatial encoder). |
| GAT / GATv2 | Graph Attention Network [11] (the learnable-graph module). |
| BAR | Block-Attention-Residual aggregation [3]. |
| QP | Quadratic Program; here, the convex projection solved in the portfolio head. |
| KKT | Karush–Kuhn–Tucker optimality conditions of the projection QP (used by the `cvxpylayers` reference that validates the deployed head, which itself differentiates by unrolling). |
| SAM | Sharpness-Aware Minimization [6]; an optimiser wrapper that biases training toward flat minima. |
| MHA / FFN | Multi-Head Attention / Feed-Forward Network (transformer sublayers). |
| OHLCV | Open/High/Low/Close/Volume daily bars. |
| ADR | American Depositary Receipt (foreign equity traded on US exchanges). |
| VaR / CVaR | Value-at-Risk / Conditional Value-at-Risk (tail-loss measures). |
| MaxDD | Maximum drawdown. |
| HHI | Herfindahl–Hirschman Index (concentration measure used in the diversification term). |
| EMA | Exponential Moving Average. |
| TE / IR | Tracking Error / Information Ratio (vs. a benchmark). |
| SPA / HAC | Superior Predictive Ability test [17] / Heteroskedasticity-and-Autocorrelation-Consistent covariance. |
| VSA | Vertical Slice Architecture (the codebase organisation). |

# 1 Introduction

## 1.1 Motivation

End-to-end optimisation of long-only equity portfolios has historically been decomposed into three stages: (i) feature engineering and forecasting, (ii) cross-sectional ranking, and (iii) portfolio sizing under risk constraints. Each stage typically uses a separate objective (mean-squared error, listwise ranking loss, mean–variance utility) that is at best a proxy for end-investor performance. These intermediate decompositions create training-time misalignments and inherently constrain the system's capacity to capture complex inter-asset dynamics that span the three stages.

REGENT abandons that decomposition. The full pipeline from per-stock indicators and macro state to constrained portfolio weights is differentiable end-to-end; the loss is a direct convex combination of Sharpe, Sortino, CVaR, drawdown, turnover, diversification, and graph-entropy terms, all computed on the realised portfolio return path. The architecture is inspired by Li and Fan [1] but adapted to (i) a fixed, sector-balanced 120-stock graph universe, (ii) hard convex-QP constraint enforcement at the portfolio head, and (iii) an explicit market-regime stream produced by a separately-trained temporal-transformer VQ-VAE.

## 1.2 Contributions

1. **Macro-state encoder.** A temporal-transformer VQ-VAE encodes five macro series (VIX, TNX, DXY, SPY, GLD) over a 60-day rolling window into one of $K$=12 discrete regimes per trading day. The codebook is updated by exponential moving average with K-means warm-start, dead-code revival, the Rotation Trick for gradient flow [2], and a Switch-Transformer-style load balance loss to prevent codebook collapse. The decoder reconstructs the last-day macro vector under Huber loss.

2. **Spatio-temporal portfolio agent (GRASP).** A four-module graph neural network: BiLSTM with two stacked multi-head self-attention blocks (temporal), two-layer GCN on a sector and defensive↔cyclical cross-sector prior (spatial), two stacked GATv2 blocks with EMA-stabilised top-$k$ edge sparsification (learnable graph), and a differentiable QP portfolio head. All encoder modules use Block-Attention-Residual (BAR) aggregation [3] and DropPath stochastic depth.

3. **Differentiable portfolio head.** The deployed portfolio head emits raw scores that a convex program projects onto the feasible set, enforcing long-only weights, per-stock and sector caps, and unit gross exposure. The projection is solved on the GPU by an unrolled dual-decomposition solver (a closed-form capped-simplex inner step under dual ascent on the sector multipliers) and is differentiated by unrolling rather than through implicit KKT conditions. It matches the `cvxpylayers` solve it replaces to $\sim 10^{-6}$ on the forward pass while running about six times faster at batch 96 [4, 1], and it keeps the exact-feasibility property that removes the cap-pinning trap of soft-penalty formulations. Running on-device also makes the second forward/backward of sharpness-aware training affordable (section 6.4).

4. **Composite financial objective.** A multi-term loss directly optimises annualised risk-adjusted return (Sharpe, Sortino), tail-risk and drawdown ($\text{CVaR}_{0.08}$, MaxDD), one-sided diversification (HHI excess over an effective-$N$ target), one-way turnover, GAT edge entropy, and sector dispersion. No reinforcement learning or reward shaping.

5. **Execution-fidelity backtester and live loop.** The backtester implements Alpaca's exact commission-free fee schedule (SEC, FINRA TAF capped, CAT both-sides, 2 bps slippage) with per-execution cent-ceiling rounding, and the live-trading service re-uses the same constants and rounding logic verbatim.

6. **Walk-forward protocol.** A rolling 504/252/126 day train/validation/test splitter with 20-day embargo and a per-window *purged $K$-fold cross-validation ensemble* ($K$=4): the pre-test span is carved into $K$ contiguous embargoed folds, each held out as one member's regime-disjoint validation set, and the $K$ members are averaged in prediction space (a convex combination, hence feasibility-preserving). The VQ-VAE encoder is fixed across windows; only GRASP is retrained.

### 1.3 Scope and limitations

REGENT is a research codebase, not a deployed product, and its current iteration focuses strictly on daily close-to-close trading. I leave intraday execution modelling, stop-loss safety circuits, short-selling, leverage, and options derivatives for future work. The graph universe is fixed at $N = 120$ liquid US equities and ADRs and is not rebalanced for delistings inside a training window. The fee model is calibrated to emulate execution via Alpaca paper/live accounts.

**Survivorship bias.** The 120-name roster is selected *once* and then frozen for the entire backtest. The selection criteria (`config/symbols.yaml`) are S&P 500 membership or large-cap ADR status, market capitalisation > \$10B, average daily dollar volume > \$300M, a *full nine-year price history*, and direct tradability on Alpaca; they are evaluated against the universe as constituted at the time the research codebase was built (mid-2026) rather than point-in-time at the start of each window. The full-history requirement in particular admits only names that survived continuously across the 2013–2026 sample, so the roster structurally excludes firms that were liquid early in the period but later delisted, were acquired, or fell below the liquidity floor. This induces a survivorship (and look-ahead-membership) bias that can flatter the reported figures relative to a fully point-in-time, delisting-aware universe. I disclose this as a known limitation rather than correct for it; a reconstituted point-in-time universe with delisting handling is the cleanest way to quantify the gap and is left to future work.

**Static slippage.** Execution slippage is modelled as a flat 2 bps of notional on both sides (table 7), independent of order size, name liquidity, and market state. This is a deliberately simple, and in the tail *optimistic*, assumption: in the systemic-selloff windows the system explicitly encounters (e.g. the 2022 rate shock), realised spreads and market impact widen well beyond 2 bps, so the true drag in exactly the crisis windows that bound the drawdown tail is understated. A volatility- or regime-conditional slippage model — for instance scaling the 2 bps floor by the VQ-VAE regime or by realised volatility — would tighten this and is a natural extension of the existing fee-parity contract.

All reported results are historical simulations; they include modelled commissions and slippage but cannot capture every real-world friction (queue position and live fill prices, borrow availability, broker or data outages, corporate actions, and macro regimes absent from the 2013–2026 sample). They are therefore an estimate of historical strategy behaviour under the stated assumptions, not a forecast of, or a claim about, realisable future returns.

## 2 Related Work

REGENT sits at the intersection of three lines of work: deep portfolio optimisation, graph-based asset representation learning, and discrete regime modelling. I briefly position the system against each.

**Deep portfolio optimisation.** Differentiable portfolio construction has progressed from simple softmax-projected scoring heads [1] toward exact convex constraint enforcement via implicit differentiation. OptNet [5] established the QP-layer template, and `cvxpylayers` [4] generalised it to disciplined-convex programs with conic backends. Closest in spirit, Li and Fan [14] apply a QP projection layer to a recurrent forecaster with mean–variance utility. REGENT differs in that (i) the upstream encoder is a graph network with an explicit regime stream rather than a per-asset RNN, and (ii) the training loss is a direct composite of realised Sharpe, Sortino, $\text{CVaR}_{0.08}$, drawdown, turnover, and diversification, with no mean–variance proxy.

**Graph neural networks for asset modelling.** A growing literature uses GNNs to capture cross-asset structure: HATS and RSR encode hierarchical relations among stocks; HIST adds latent concept graphs. These works typically optimise a return-forecasting or ranking proxy and feed the prediction into a downstream allocator. GRASP's contribution is to keep the entire chain (GCN $\rightarrow$ GATv2 $\rightarrow$ portfolio head) differentiable end-to-end against the financial objective, removing the proxy and the post-hoc cross-sectional ranker.

**Discrete regime models.** Regime detection has historically been Hidden-Markov-Model territory (Nystrup et al. and successors). Vector-quantised autoencoders [13] offer a learnable codebook alternative with discrete latents trainable by backprop. REGENT's contribution here is small but practical: a temporal-transformer VQ-VAE on macro inputs with the Rotation Trick [2] for gradient flow, Switch- Transformer load-balance for codebook utilisation, and an explicit causal pre-processing stack designed so that no future information leaks into the regime stream consumed by GRASP at training time.

**Reinforcement learning approaches.** FinRL, AlphaPortfolio, and similar RL frameworks treat portfolio construction as a Markov decision problem with reward shaping. REGENT's loss is computed analytically over a finite realised return window and is fully supervised; it avoids the credit-assignment and reward-design pathologies typical of RL portfolio agents at the cost of giving up the ability to model multi-step transaction strategies. The single-period (daily-rebalance) framing matches the deployment cadence and is identical at train and test time.

## 3 System Overview

### 3.1 Vertical Slice Architecture

The codebase follows the Vertical Slice Architecture (VSA) pattern: each feature owns its data directories, repositories, services and CLI entry point under `features/<slice>/`; slices do not import each other's internals. Cross-cutting concerns (logging, configuration, retry, sector mapping, macro alignment) live in `shared_kernel/`.

### 3.2 End-to-end data flow

**Table 1.** The seven vertical slices and one shared kernel of REGENT.

| Slice | Entry point | Owns |
|---|---|---|
| market_data | download_command.py | data/raw/1d/ |
| data_processing | process_command.py | data/features/ |
| regime_detection | training_command.py, generate_market_regimes.py | data/models/regime_encoder/, data/regimes/ |
| grasp_agent | train_grasp.py | data/models/grasp_agent/ |
| backtesting | backtest_command.py | data/backtesting/ |
| live_trading | scripts/run_live_trading.py | data/live_trading/ |
| walk_forward | scripts/run_walk_forward.py | data/walk_forward/ |
| shared_kernel | n/a | logging, config, retry, macro alignment, symbol loader |

```
data/raw/1d/*.parquet                          (market_data)
  -> data/features/*.parquet                     (data_processing)
  -> data/regimes/market_regimes.parquet         (regime_detection)
  -> data/models/grasp_agent/                    (grasp_agent training)
  -> orders submitted via Alpaca Markets API     (live_trading)
```

### 3.3 Symbol universe

The universe is partitioned into three tiers in `config/symbols.yaml`:

- **Regime indicators** (5, not tradable): `^VIX`, `^TNX`, `DX-Y.NYB`, `SPY`, `GLD`. These are the macro features of the VQ-VAE. The runtime constraint `input_dim=5` is validated on load and any deviation raises `ValueError`.
- **Active research universe** (`symbols`, 120 entries): downloaded and feature-engineered. After retirement of the CatBoost forecasting layer this list equals the GRASP universe; an additional `storage` reserve pool of ∼317 swap-in candidates is documented but not loaded.
- **GRASP graph universe** (`grasp_universe`, 120): fixed $N = 120$ tensor input to the graph NN. Sector breakdown is Information Technology (30), Financials (15), Health Care (15), Industrials (12), Consumer Discretionary (9), Communication Services (7), Consumer Staples (7), Real Estate (5), Energy (5), International ADRs (5), Materials (4), Utilities (4), Benchmarks (2).

The loader `shared_kernel/reference_data/symbol_loader.py` validates that every `grasp_universe` entry is present in `symbols` and that no list contains duplicates or empty entries.

## 4 Data Pipeline

### 4.1 Market data ingestion

The `market_data` slice ingests OHLCV data from Yahoo Finance via `yfinance`. There is no Alpaca ingestion path; Alpaca is used only at order execution. Calls are issued with split- and dividend-adjusted back-history (`auto_adjust=True`). Because adjustment factors propagate

backwards over the full series, incremental appends are disabled and an update performs a full re-download. Retries use a fixed 2 s back-off with at most three attempts and a 0.5 s inter-symbol sleep. Any symbol that returns fewer than 250 rows is rejected outright (this exceeds the downstream 200-row warmup cutoff with a 50-row safety margin for null drops).

Files are written as `data/raw/{interval}/{SYMBOL}_{interval}.parquet` using PyArrow with Snappy compression. The default download window is nine calendar years.

### 4.2 Feature engineering

`data_processing` computes more than 150 features per symbol using Polars lazy evaluation. Strategic `.collect()` checkpoints are placed between phases to prevent deep-window expression nesting errors. After all phases complete, the first 200 rows per symbol are filtered out so that SMA-200 and other long-lookback indicators are fully populated. A 34-column allowlist of volatility, momentum, deviation, velocity, volume and raw candlestick columns is winsorised to the [1%, 99%] cross-sectional range. The phased pipeline is summarised in table 2.

**Table 2.** Feature engineering phases (output schema `data/features/`).

| Phase | Indicator family |
|---|---|
| 0–0.7 | SMA (20/50/200), EMA (12/26/50/200), HMA, VWMA20, ATR/NATR, annualised 20-d volatility, ±DI, ADX, MACD + signal + hist, Stochastic %K/%D, Bollinger upper/middle/lower/width/%B, OBV, CMF, volume ratio + SMA20, RSI (Wilder), ROC |
| 0.8–1 | log-returns 1/5/20 d, support/resistance, price position, Ichimoku 5 lines, volume force, MFI, ADL, Elder bull/bear power, trend strength, composite buy/sell signals |
| 1–1.5 | price velocity & acceleration (raw + smoothed), multi-timeframe momentum (5–3, 10–5, 20–10), leak-free alpha signals (momentum divergence, price-vs-SMA50/200, volatility breakout, volume–price alignment, bullish-pattern strength, mean-reversion signal), per-symbol 60-d return $z$-score, candlestick decomposition, microstructure proxies (spread, price efficiency, volume–price impact, vol-of-vol, high-vol regime flag) |
| 2 | discrete Kelly position (100-d window, capped 25%), vol-adjusted position, 5- and 3-class trading signals, ATR stop-loss/take-profit at $2\sigma/3\sigma$, fixed 1.5 risk-reward ratio |
| 2.5–3 | calendar (weekday, day-of-month, month-/quarter-end), $\mathrm{VaR}_{5\%}$ (60-d), drawdown 20/60-d, downside deviation, relative volatility 5-d/20-d, risk parity weight, Sharpe/Sortino/Calmar proxies, volatility spike, volume shock, gap size, overall risk score |

### 4.3 Macro alignment

Two macro-join paths exist. The *processing-time* join builds a single in-memory cache from the five macro raw parquets, gap-fills weekday-only calendar holes with forward fill, and left-joins `{vix,tnx,dxy,spy,gld}_close` into every stock feature parquet. A fail-fast check raises `ValueError` listing the missing macros and the exact command to remedy the situation. The *inference-time* `MacroAlignmentService` in `shared_kernel/market_context/` performs a richer 15-column join (close + high + low for all five macros) and additionally computes the regime flag set `vix_regime`, `rate_regime`, `dollar_regime`, 5/20-d momentum changes, and a composite `risk_regime` flag. This is consumed by the backtester and the live-trading warmup pipeline.

**Ordering constraint.** Macro symbols must be processed strictly before any stock symbol. The constraint is enforced both at runtime (fail-fast `ValueError`) and in the staleness check, which forces re-processing of any stock whose feature mtime is older than the maximum macro raw mtime.

# 5 VQ-VAE Regime Detection (v3.2)

The regime encoder is designed around a simple observation: macroeconomic stress propagates slowly. A single-day VIX spike may be noise; a month of rising yields, a weakening dollar, falling equities, and rising gold, all at once, is a regime shift. To recover that structure I encode sixty-day rolling windows of five macro indicators into one of twelve discrete codewords, retrained jointly with a small reconstruction decoder. The discrete codebook gives downstream modules a categorical signal they can embed and condition on, while the temporal-transformer backbone with K-means warm-start and load-balanced quantisation prevents the well-known VQ-VAE collapse mode in which a handful of codewords absorb the entire distribution.

## 5.1 Inputs and preprocessing

The encoder receives sliding windows of shape ($T$=60, $F$=5) over five macro series: `vix_close`, `tnx_close`, `dxy_close`, `spy_close`, `gld_close`. A shared transform stack (`apply_macro_feature_transforms`) is applied identically at training and inference time:

1. SPY, GLD: 20-day percentage return, $x_t/x_{t-20} - 1$.
2. TNX: 20-day absolute yield change, $x_t - x_{t-20}$.
3. All five features: 5-day rolling mean.
4. Causal rolling $z$-score with 126-day window, shifted by one day to enforce strict causality, with $\epsilon = 10^{-6}$ in the denominator.
5. Clipping to $[-4.0, 4.0]$ to bound the COVID-era VIX spike at approximately $+5.5\sigma$.

The bundled `feature_scaler.pkl` is an identity transform; the causal rolling $z$-score is the normalisation. `RegimeEncoder` refuses to load any non-identity scaler.

Windows are constructed with `np.lib.stride_tricks.sliding_window_view` at stride 1, dropping the first 59 rows. A `use_recent_years` restriction of 9 keeps at most $9 \times 252 = 2268$ windows from the tail of the series, preventing distribution contamination from the zero-interest-rate years.

## 5.2 Architecture

The model is summarised in table 3. The temporal encoder uses $L$=2 stacked *Attention Residual Blocks* (the variant from [3]) each with pre-LayerNorm multi-head attention, a feed-forward sublayer, and two learnable residual gates $\alpha_{\text{attn}}, \alpha_{\text{ffn}}$ initialised to 1.0 and excluded from weight decay. Dropout is single-application per sublayer: no in-MHA dropout, no inter-linear FFN dropout. After encoder output the model pools the last $K$=5 time steps with a mean to smooth daily macro noise while preserving causality.

## 5.3 Mathematical formulation

**Reconstruction.** Huber loss with $\delta = 0.5$ (robust to post-standardisation $\geq 3\sigma$ outliers):

$$\mathcal{L}_{\text{recon}} = \frac{1}{B} \sum_{i=1}^{B} \mathcal{H}_{\delta=0.5}(\hat{x}_i,\, x_{i,T}), \quad \hat{x}_i = \text{Decoder}(e_{k_i^*}).$$

**Table 3.** VQ-VAE architecture summary.

| Component | Specification |
|---|---|
| Input window | $(B, T{=}60, F{=}5)$ |
| Input projection | Linear$(5 \to 32)$ |
| Positional encoding | Sinusoidal, maxlen=60 |
| Encoder blocks | $L{=}2$ pre-LN BAR (MHA $d{=}32$, $h{=}4$; FFN $32 \to 64 \to 32$, GELU) |
| Encoder dropout | 0.22 single-application per sublayer |
| Encoder pooling | Mean over last 5 tokens |
| Latent projection | Linear$(32 \to 16)$ then L2-normalise |
| Vector quantiser | $K{=}12$ codewords in $\mathbb{R}^{16}$, EMA-updated, L2-normalised |
| Decoder | Linear$(16 \to 16) \to$ LN $\to$ GELU $\to$ Dropout$(0.30) \to$ Linear$(16 \to 5)$ |
| Reconstruction target | Last-day macro vector $\in \mathbb{R}^5$ |

**Commitment loss with Rotation-Trick gradient routing.** Codebook vectors $\{e_k\}_{k=1}^{12} \subset \mathbb{R}^{16}$ are EMA-only; they have `requires_grad=False`. The commitment loss is

$$\mathcal{L}_{\text{commit}} = \beta \left\| \text{sg}(e_{k^*}) - z \right\|_2^2, \quad \beta = 1.0.$$

Forward quantisation uses a Householder reflection that maps $z$ onto $e_{k^*}$:

$$q = H\, z, \qquad H = I - \frac{2\, v\, v^\top}{v^\top v}, \qquad v = \text{sg}\big(z - e_{k^*}\big),$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator (the reflection vector $v$ is detached so gradients flow only through the ambient $z$, not the reflection axis itself). The backward pass routes $\partial q / \partial z = H$, encoding the angular distance between encoder output and chosen codeword into the gradient.

**Diversity regulariser.** Combined soft-entropy and Switch-Transformer load-balance:

$$\mathcal{L}_{\text{div}} = \lambda_d \left[ \log K + \sum_k \bar{p}_k \log \bar{p}_k + \text{ReLU}\Big( K \sum_k \ell_k \bar{p}_k - 1 \Big) \right],$$

where $\bar{p}_k = \frac{1}{B} \sum_i \text{softmax}(-d_i/T)_k$ with temperature $T{=}0.5$, $\ell_k = \frac{1}{B} \sum_i \mathbf{1}[k_i^* = k]$ is the hard usage proportion (detached), $\lambda_d = 0.30$, and $K{=}12$. The load-balance term couples hard argmin concentration to the soft probability gradient, preventing decoupled collapse.

**Total objective.** Auxiliary heads (`aux_loss_weight`, `return_sign_aux_weight`) are disabled in v3.2; only the three terms remain:

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{commit}} + \mathcal{L}_{\text{div}}.$$

### 5.4 Training procedure

The $\beta_2{=}0.95$ choice shortens the second-moment horizon to roughly 20 steps, which is appropriate for the $\sim$1000-window dataset (the default $\beta_2{=}0.999$ produces an effectively static second-moment estimate).

**Table 4.** VQ-VAE training hyperparameters (`config/vqvae_config.yaml`, authoritative).

| | |
|---|---|
| Optimiser | AdamW, $\beta = (0.9, 0.95)$, $\varepsilon = 10^{-8}$, lr $1.5\times10^{-4}$ |
| Weight decay (matrices) | $7\times10^{-4}$; biases/LN/$\alpha$-gates 0.0 |
| Scheduler | 15-epoch linear warmup $\rightarrow$ cosine to $\eta_{\min} = 10^{-6}$, $T_{\max} = 135$ |
| EMA codebook decay | 0.97; Laplace smoothing $\varepsilon = 10^{-5}$ |
| K-means warm-start | 12 clusters, 10 inits, fit on 2048 latents; reset at epoch 15 |
| Dead-code revival | threshold 0.10, warmup 200 steps; Gaussian jitter $\sigma$=0.05 |
| Early stopping | metric `val_recon_loss`, patience 15, min-delta $5\times10^{-4}$ |
| Max epochs | 150 |
| Gradient clip | global norm 2.0 |
| Input noise (train) | Gaussian $\sigma = 0.07$ on standardised features |
| Determinism | seed 42, math-only SDPA, TF32 off |
| Split | 70 / 15 / 15 chronological |

### 5.5 Inference and regime stream

`RegimeEncoder.batch_encode_dataframe` slides the stride-1 window of shape $(60, 5)$ over the aligned macro DataFrame and calls `VQVAE.encode_with_soft_probs(x, top_k=1)` to obtain the argmin codeword on L2-normalised latents. Per-sample confidence is reported as plain MSE, $\text{conf}_i = 1/(1 + \frac{1}{F}\|\hat{x}_i - x_{i,T}\|_2^2)$, *not* the Huber objective used at training time. The asymmetry is deliberate: Huber's piecewise-linear tail buys robustness during optimisation (a single $\geq 3\sigma$ outlier macro day cannot dominate the recon-loss gradient), but at inference we want *exactly* that quadratic sensitivity in the confidence signal: a macroeconomic extreme should drive confidence sharply downward so that downstream adaptive risk multipliers (`features/regime_detection/adaptive_multipliers.py`) revert to the neutral 0.9 prior rather than committing to a regime label inferred from a far-OOD window. The two numbers are not on the same scale and should not be compared directly with the trainer's logged `val_recon_loss`. Two strictly causal post-processing passes are then applied:

- **Causal mode filter:** at time $t$ assign the mode of `regimes[max(0,t-6) : t+1]`, with most-recent tie-break. No look-ahead.
- **Minimum dwell enforcement:** a regime change is committed only after the new regime persists for 10 consecutive days; until then the previously committed regime is broadcast.

Output is written to `data/regimes/market_regimes.parquet` with columns `date`, `regime_id`, `regime_changed`, `regime_age`, and `regime_confidence` (one minus the trailing 20-day switch rate). An entropy warning is logged if the regime distribution entropy falls below $0.75\log_2(12) \approx 2.77$ bits.

## 6 GRASP Portfolio Agent (v1.0)

GRASP is the part of the system that does the actual portfolio construction, and it is the part that has to satisfy the hardest constraints: 120 stocks to weight long-only every day, each at most 7%, each sector at most 30%, gross exposure exactly 100%, and ideally with stable turnover, real diversification, and a tolerable drawdown profile. The classical approach splits this into three stages (forecast, rank, size) and pays an alignment cost between them. GRASP collapses the entire chain into one differentiable network and trains it directly against the realised portfolio return path. The four modules below (temporal encoder, spatial encoder, learnable graph, and a differentiable convex-QP head) each contribute a distinct inductive bias, and the loss function

tells them how to negotiate.

### 6.1 Input tensor

The GRASP model consumes a tensor of shape

$$x \in \mathbb{R}^{B \times N \times T \times F}, \quad N = 120,\ T = 20,\ F = 44.$$

Channel 25 originally holds the integer regime ID (normalised to $[0, 1]$); before the temporal encoder receives the tensor, `GRASPModel.forward` extracts this scalar, normalises it back, and looks up a learned 8-dimensional embedding. This is then spliced back into the tensor in place of the single scalar, expanding the core 37-channel raw feature width to an internal $F$=44 (see Section 6.2.5).

**Table 5.** GRASP input feature channels (schema v3, authoritative `features/grasp_agent/constants.py`).

| Ch | Symbol | Description |
|---|---|---|
| 0 | `CH_MEAN_RETURN` | Rolling mean log-return |
| 1 | `CH_STD_RETURN` | Rolling std of log-returns |
| 2 | `CH_SKEWNESS` | Rolling return skewness |
| 3 | `CH_KURTOSIS` | Rolling return kurtosis |
| 4 | `CH_VAR_5PCT` | Value-at-Risk, 5% |
| 5 | `CH_CVAR_5PCT` | Conditional VaR, 5% |
| 6 | `CH_DOWNSIDE_DEV` | Downside deviation |
| 7 | `CH_DRAWDOWN` | Rolling drawdown |
| 8 | `CH_CUMULATIVE_RETURN` | Cumulative return over window |
| 9 | `CH_VOLUME_RANK` | Cross-sectional volume rank |
| 10 | `CH_AMIHUD` | Amihud illiquidity ratio |
| 11 | `CH_VOLUME_STABILITY` | Volume stability measure |
| 12 | `CH_VOLUME_TREND` | Volume trend |
| 13 | `CH_MOMENTUM` | Price momentum |
| 14 | `CH_MOMENTUM_5D` | 5-day momentum |
| 15 | `CH_ACCELERATION` | Momentum acceleration |
| 16 | `CH_RSI` | Relative Strength Index |
| 17 | `CH_MA_RATIO_50` | Close / 50-day MA |
| 18 | `CH_MA_RATIO_200` | Close / 200-day MA |
| 19 | `CH_VOLATILITY_PCT` | Volatility (percent units) |
| 20 | `CH_VOLATILITY` | Realised volatility |
| 21 | `CH_MARKET_CORRELATION` | Rolling correlation with SPY |
| 22 | `CH_MARKET_BREADTH` | Market breadth indicator |
| 23 | `CH_DEFENSIVE_INDICATOR` | Defensive stock score |
| 24 | `CH_CRISIS_BETA` | Crisis-regime beta |
| 25 | `CH_REGIME_ID` | VQ-VAE regime ID, $\rightarrow$ 8-d embedding |
| 26 | `CH_MOM_VOL_INTERACTION` | Momentum $\times$ volatility |
| 27 | `CH_RSI_DRAWDOWN_INTERACTION` | RSI $\times$ drawdown |
| 28 | `CH_VOL_TREND_INTERACTION` | Volatility $\times$ trend |
| 29 | `CH_MACD` | MACD signal |

| Ch | Symbol | Description |
|---|---|---|
| 30 | CH_BB_WIDTH | Bollinger band width |
| 31 | CH_STOCH_K | Stochastic %K |
| 32 | CH_CMF | Chaikin Money Flow |
| 33 | CH_SECTOR_REL_MOMENTUM | Sector-relative momentum (cross-sectional) |
| 34 | CH_SECTOR_REL_VOLATILITY | Sector-relative volatility (cross-sectional) |
| 35 | CH_ADX | Average Directional Index (trend-strength magnitude) |
| 36 | CH_BB_PCT_B | Bollinger %B $= (\text{close} - L)/(U - L)$ (continuous mean-reversion) |

### 6.2 Four-module architecture

The forward pass is summarised as a dataflow schematic in fig. 1: the input tensor flows top to bottom through the four modules, with tensor shapes annotated at each stage and the operative hyperparameters shown inline. Each module is then specified formally in the subsections that follow.

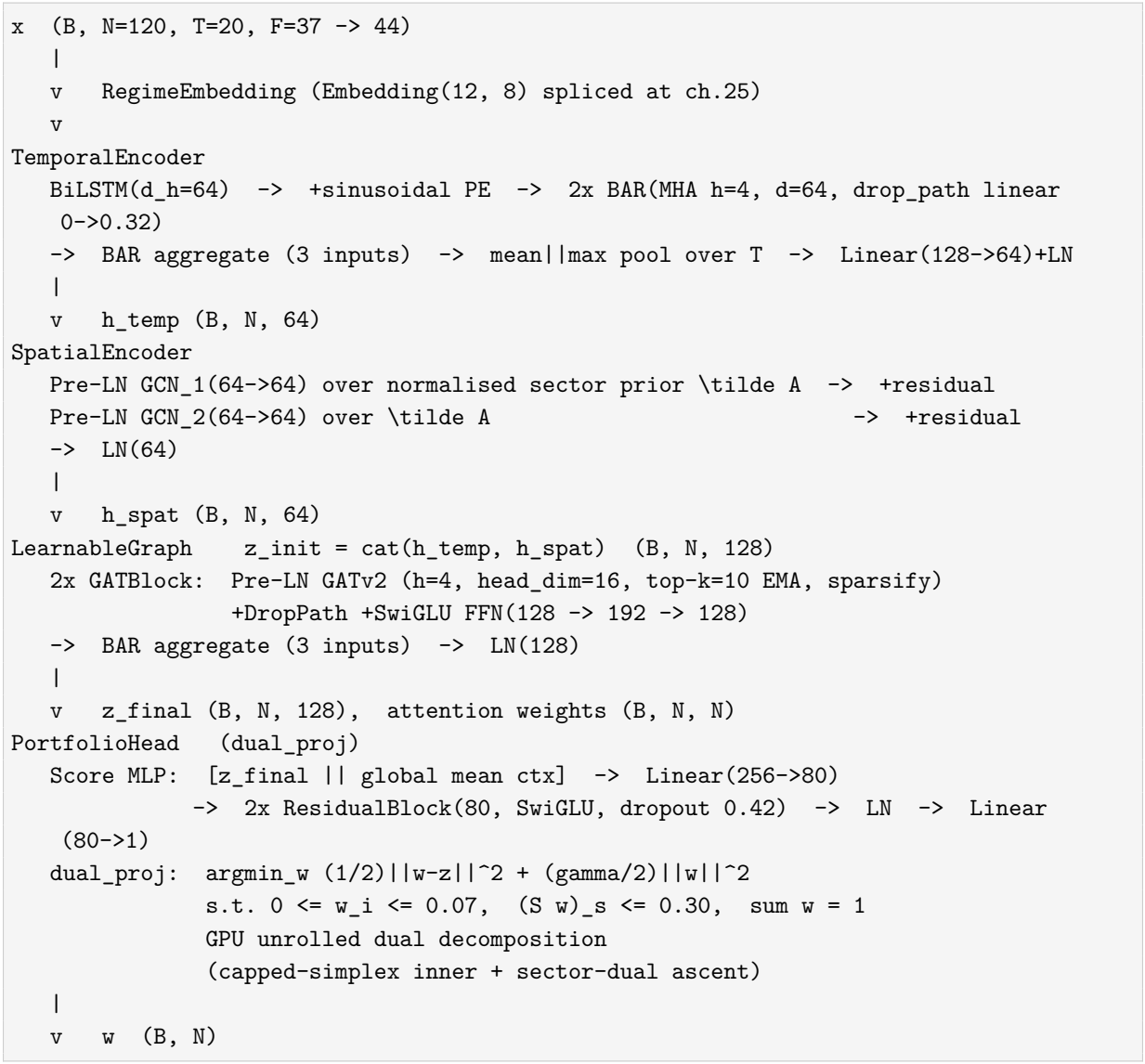



**Figure 1.** GRASP forward-pass schematic. Data flows top to bottom through the four modules (temporal $\rightarrow$ spatial $\rightarrow$ learnable-graph $\rightarrow$ portfolio head); tensor shapes are annotated as $(B, N, \dots)$ at each stage and module hyperparameters are shown inline. "$\rightarrow$" denotes a transform, "$\|$" concatenation, and "$\tilde{A}$" the normalised sector-prior adjacency.

### 6.2.1 TemporalEncoder

Per-stock, the BiLSTM with hidden size $d_h$=64 (32 per direction) maps $X_n \in \mathbb{R}^{T\times F}$ to $H_n^{(0)} \in \mathbb{R}^{T\times 64}$. A sinusoidal positional encoding is added:

$$P_{t,2k} = \sin\Big(\tfrac{t}{10000^{2k/d_h}}\Big), \quad P_{t,2k+1} = \cos\Big(\tfrac{t}{10000^{2k/d_h}}\Big).$$

Two pre-LayerNorm attention blocks follow:

$$H^{(\ell)} = H^{(\ell-1)} + \mathrm{DropPath}\Big(\mathrm{Dropout}\big(\mathrm{MHA}(\mathrm{LN}(H^{(\ell-1)}))\big)\Big), \quad \ell = 1, 2.$$

DropPath rates are linearly spaced: {0, 0.32}. A Block-Attention- Residual module aggregates $\{H^{(0)}, H^{(1)}, H^{(2)}\}$:

$$H_t^{\mathrm{BAR}} = \sum_{k=0}^{L} \alpha_k H_t^{(k)}, \quad \alpha = \mathrm{softmax}\Big(q_L^\top K / \sqrt{d_{\mathrm{proj}}}\Big),$$

where the query is the projected *current* (deepest) block output $q_L = W_q H^{(L)}$ and the keys are the projected stack of all candidate blocks $K = W_k\,[\,H^{(0)}; H^{(1)}; \ldots; H^{(L)}\,]$, with learned $W_q, W_k \in \mathbb{R}^{d_h \times d_{\mathrm{proj}}}$ (no bias), $L$=2, and $d_{\mathrm{proj}} = 32$. The aggregation thus lets the deepest representation choose, per forward pass, how much of each shallower block to retain. The output is mean‖max-pooled over $T$ and projected to 64 dimensions, yielding $h_{\mathrm{temp}} \in \mathbb{R}^{N \times 64}$.

### 6.2.2 SpatialEncoder

A static prior graph $A \in \{0,1\}^{N \times N}$ encodes (i) intra-sector membership and (ii) defensive↔cyclical cross-sector edges (Staples / Utilities / Healthcare ↔ Technology / Financials / Energy). After adding self-loops, the symmetric normalisation

$$\tilde{A} = D^{-1/2}(A + I)D^{-1/2}, \qquad D_{ii} = \sum_j A_{ij} + 1,$$

is applied. Each GCN layer is pre-LayerNormed with an identity residual:

$$H^{(\ell)} = \mathrm{res\_proj}(H^{(\ell-1)}) + \mathrm{Dropout}\Big(\mathrm{ReLU}\big(\tilde{A}\ \mathrm{LN}(H^{(\ell-1)})\,W_\ell\big)\Big).$$

Two such layers, each $64 \to 64$, produce $h_{\mathrm{spat}} \in \mathbb{R}^{N \times 64}$.

### 6.2.3 LearnableGraph (GATv2 + BAR)

The combined node feature is $Z^{(0)} = [h_{\mathrm{temp}} \,\|\, h_{\mathrm{spat}}] \in \mathbb{R}^{N \times 128}$. Two GAT blocks each contain a pre-LN GATv2 attention sublayer with four heads, head dimension 16, learnable attention vector $a_k \in \mathbb{R}^{16}$, and per-head weight matrices $W_{\mathrm{src}}^{(k)}, W_{\mathrm{dst}}^{(k)}, W_{\mathrm{val}}^{(k)} \in \mathbb{R}^{128 \times 16}$; followed by a SwiGLU FFN with hidden dimension 192 and DropPath rate linearly interpolated 0–0.32. Raw attention logits:

$$e_{ij}^{(k)} = \frac{a_k^\top\ \mathrm{LeakyReLU}_{\alpha=0.2}\big(W_{\mathrm{src}}^{(k)} z_i + W_{\mathrm{dst}}^{(k)} z_j\big)}{\sqrt{d_{\mathrm{head}}}}.$$

An EMA of batch-mean logits $\bar{E}^{(k)}$ (decay 0.9) is maintained *per head*, because each head's attention vector $a_k$ places its logits on a different numeric scale. A shared buffer would mask incomparable ranges and destabilise the selected edge set. At each forward pass, only the top-$k$=10 columns of $\bar{E}^{(k)}$ are retained per row and all other entries are masked to $-\infty$ prior to softmax. The edge *selection* is non-differentiable (a `detach`ed top-$k$ on the EMA buffer, persisted in the state dict so it survives checkpoint/restore); gradients flow only through the softmax over the retained edges, so the mask gates information flow without contributing a gradient path of its own. The masked attention is

$$\alpha_{ij}^{(k)} = \mathrm{softmax}_j(\tilde{e}_{ij}^{(k)}), \qquad \hat{z}_i^{(k)} = \sum_j \alpha_{ij}^{(k)}\, W_{\mathrm{val}}^{(k)} z_j.$$

Heads are concatenated and projected back to 128 dimensions; BAR aggregation over $\{Z^{(0)}, Z^{(1)}, Z^{(2)}\}$ followed by a final LayerNorm produces $z_{\text{final}} \in \mathbb{R}^{N\times 128}$ together with the last-block head-averaged attention $\bar{\alpha} \in \mathbb{R}^{N\times N}$.

#### 6.2.4 PortfolioHead

The score MLP consists of a per-node Linear($256 \to 80$) projection of $[z_i \,\|\, \bar{z}]$ (where $\bar{z}$ is the global mean), two SwiGLU residual blocks with dropout 0.42, a LayerNorm, and a Linear($80 \to 1$) output. The output scores $z \in \mathbb{R}^N$ are mean-centred per batch and projected onto the feasible set defined by

$$w^\star = \underset{w\in\mathbb{R}^N}{\arg\min}\ \tfrac{1}{2}\|w-z\|_2^2 + \tfrac{\gamma}{2}\|w\|_2^2 \quad \text{s.t.} \quad \begin{cases} \sum_i w_i \in [\,0.99995,\ 1.00005\,] & \\ 0 \le w_i \le 0.07 & \forall i \\ (Sw)_s \le 0.30 & \forall s \in \{1,\dots,12\} \end{cases} \tag{1}$$

with sector-membership matrix $S \in \{0,1\}^{12\times 120}$ and ridge coefficient $\gamma = 0.03$ (`qp_ridge`). The ridge makes the objective strongly convex and softly biases towards equal weight (interior scores compress by $1/(1+\gamma) \approx 0.971$).

The deployed head (`head_type: dual_proj`) solves (1) on the GPU rather than calling a CPU convex-program solver. Folding the ridge into a pre-scale $z' = z/(1+\gamma)$ turns the problem into a Euclidean projection of $z'$ onto the feasible polytope, which a dual decomposition handles in two nested loops. The inner loop projects onto the capped simplex $\{0 \le w \le 0.07,\ \sum_i w_i = 1\}$, which has the closed form $w = \text{clamp}(v-\theta,\, 0,\, 0.07)$ with the single scalar $\theta$ found by bisection; it is a custom autograd function whose backward is the exact capped-simplex Jacobian (a masked mean-subtraction), so its memory cost is $O(N)$ regardless of bisection depth. The outer loop dualises the only coupling constraint, the twelve sector caps, and runs projected dual ascent on the multipliers $\mu \ge 0$,

$$w(\mu) = \text{capped\_simplex}(z' - S^\top\mu), \qquad \mu \leftarrow \big[\,\mu + \eta\,(S\,w(\mu) - c)\,\big]_+,$$

with cap $c = 0.30$ and step $\eta = 1.2$ (40 outer, 25 inner iterations). The whole loop is unrolled, so autograd differentiates the fixed point directly: there is no implicit-KKT linear solve and none of the active-set ill-conditioning that appears near tight caps. A final clamp-and-renormalise pins $\sum_i w_i = 1$ and $w \ge 0$ exactly, leaving the sector caps satisfied to the dual tolerance ($< 10^{-3}$). The forward pass matches the reference `cvxpylayers`/SCS solve [4, 5, 1] to about $10^{-6}$ in mean and $10^{-4}$ in maximum weight, and runs roughly six times faster at batch 96 because it is fully on-device and batch-parallel rather than an $O(B)$ Python loop over per-sample solves. A `cvxpylayers`/SCS solve of the same projection ($\varepsilon = 10^{-6}$, `max_iters=5000`), which backpropagates through the KKT conditions, is kept only as a test-time correctness reference for the GPU head.

#### 6.2.5 Regime embedding

Raw regime IDs in channel 25 are stored as regime_id$/(K-1) \in [0,1]$ with $K = 12$. Inside `GRASPModel.forward`, before the temporal encoder receives the tensor:

1. Extract `regime_raw` of shape $(B, N, T)$ from channel 25.

2. De-normalise: ids = clamp(round(11 · regime_raw), 0, 11).

3. Look up nn.Embedding(12, 8) to obtain an 8-dimensional regime embedding $(B, N, T, 8)$.

4. Splice back into the original tensor at channel 25, expanding the full feature tensor dimensionality from $F$=37 to an internal $F$=44.

This captures the regime dynamics internally across 8 channels before sequence modeling rather than coercing it through a rank-1 scalar bottleneck. *Any GRASP checkpoint predating the 2026-05-26 schema update to an 8-dim embedding is structurally incompatible.*

**Historical note.** An earlier version used a collapsed projection $\text{Linear}(8, 1) \circ \text{Embedding}(12, 8)$ that was rank-1-equivalent to a single $\text{Embedding}(12, 1)$ table. The recent changes remove this intermediate capacity bottleneck entirely by supplying the full 8-dimensional regime vector directly to the temporal network.

### 6.3 Loss function

The composite training loss is, with weights from `config/grasp_config.yaml` authoritative:

$$\mathcal{L} = \underbrace{0.38\,\mathcal{L}_{\text{Sharpe}} + 0.17\,\mathcal{L}_{\text{Sortino}} + 0.17\,\mathcal{L}_{\text{Risk}} + 0.18\,\mathcal{L}_{\text{Div}} + 0.10\,\mathcal{L}_{\text{Turn}}}_{\text{convex objective, weights sum to 1.0}} + \underbrace{0.03\,\mathcal{L}_{\text{GATEnt}} + 50\,\mathcal{L}_{\text{SecDiv}}}_{\text{one-sided barriers, } \approx 0 \text{ at the operating point}} . \tag{2}$$

The two coefficient groups are not commensurable and should not be read as a single weighting. The first five terms are the convex objective and their weights sum to 1.0. The last two are *one-sided barrier* penalties of the form $\max(\cdot - \tau, 0)^2/\tau^2$ (defined below): each is normalised to $\mathcal{O}(1)$ at a 2× constraint violation and is *identically zero* once the constraint is satisfied, with zero gradient there. The large $\lambda_s$=50 is therefore a constraint-activation stiffness, not an objective weight: it sets how hard the barrier pushes *while violated*, and contributes nothing at the converged operating point where the sector spread already clears its target. Comparing 50 against the diversification weight 0.18 conflates a barrier stiffness with a convex coefficient; the operative comparison is that $\mathcal{L}_{\text{SecDiv}} \to 0$ at convergence, so the term shapes the feasible region the optimiser settles into rather than biasing the objective it minimises within it. All return-based terms are computed on the batch-flattened chronological sequence $r_{\text{flat}} \in \mathbb{R}^{B \cdot R}$ of net portfolio log-returns; the transaction-cost charge of 5 basis points (`transaction_fee` = 0.0005) applied to the per-window one-way turnover is amortised evenly across the $R$ days.

**Scope of the batch-flatten.** The flatten is defined *within* each minibatch only: `port.flatten()` concatenates the $(B, R)$ return tensor of a single batch into a 1D sequence of length $B \cdot R$, and the cumulative sum used for MaxDD is initialised at $V_0 = 1$ for every batch independently. There is *no* cross-batch state: the optimiser sees a fresh portfolio path per gradient step. This construction is sound at the deployed configuration (shuffle = False, train_stride = 1, $R$ = rebalance_freq = 1), where the $B = 96$ rows of a batch are exactly the next 96 consecutive trading days. At $R > 1$ with stride 1 the same flatten would over-count overlapping forward windows; the trainer hard-enforces stride = 1 to keep the daily-rebalance configuration honest. The reported per-batch MaxDD is therefore a 96-day rolling drawdown

statistic, not a full-trajectory maximum drawdown; the latter is computed separately over the entire concatenated validation path inside `trainer._evaluate` (the Calmar entry in table 8).

**Sharpe.**

$$\mathcal{L}_{\text{Sharpe}} = -\frac{(\bar{r} - r_f/252)\sqrt{252}}{\sqrt{\text{Var}(r_{\text{flat}}) + \epsilon_\sigma^2}}, \quad \epsilon_\sigma = 10^{-3}.$$

**Sortino.**

$$\mathcal{L}_{\text{Sortino}} = -\frac{(\bar{r} - r_f/252)\sqrt{252}}{\sqrt{\mathbb{E}[\min(r - r_f/252,\, 0)^2] + \epsilon_d^2}}, \quad \epsilon_d = 5\times10^{-4}.$$

**Tail risk and drawdown.** Using Rockafellar–Uryasev [8] on the loss $L = -r_{\text{flat}}$:

$$\text{CVaR}_\alpha(L) = \min_\xi \Big[\xi + \tfrac{1}{\alpha}\mathbb{E}[\max(L - \xi,\, 0)]\Big], \quad \alpha = 0.08;$$

the tail fraction is the mean of the worst 8% of returns; the wider window (relative to a 5% tail) lowers the per-rare-crash penalty and thus the model's defensive mega-cap avoidance. The empirical minimiser $\xi^\star$ is the $(1-\alpha)$-quantile of $L$ and is detached, so gradients flow through every tail sample above VaR rather than through a discrete top-$K$ cutoff (which drops the fractional sample when $\alpha T \notin \mathbb{N}$ and introduces a sort-boundary discontinuity that destabilises the optimiser).

The risk term itself is *not* an unconditional level penalty. Earlier versions used $\mathcal{L}_{\text{Risk}} = \text{CVaR}_{0.08}(L) + 0.5\ \text{MaxDD}$, whose gradient never shuts off: a synthetic-factor audit found $\text{corr}(\partial\mathcal{L}/\partial w,\, \beta) \approx 0.996$, i.e. a persistent de-risking pull with essentially zero alpha content, which (at 0.17 core weight) was the structural engine of the defensive utilities/staples tilt and the $\beta_{\text{SPY}} \approx 0.8$ lean. We replace it with a *one-sided tail-budget hinge* measured against the equal-weight $(1/N)$ universe basket:

$$\mathcal{L}_{\text{Risk}} = \frac{\max(\text{CVaR}_p - \kappa_c\ \text{CVaR}_{\text{ew}},\, 0)^2}{\max(\kappa_c\ \text{CVaR}_{\text{ew}},\, \epsilon_c)^2} + 0.5\,\frac{\max(\text{MaxDD}_p - \kappa_d\ \text{MaxDD}_{\text{ew}},\, 0)^2}{\max(\kappa_d\ \text{MaxDD}_{\text{ew}},\, \epsilon_d)^2},$$

with $\text{MaxDD} = \max_t(\text{peak}_t - V_t)/\text{peak}_t$, $V_t = \exp(\sum_{s\le t} r_s)$, and the equal-weight budget legs $\text{CVaR}_{\text{ew}}$, $\text{MaxDD}_{\text{ew}}$ computed on a detached, cost-free $1/N$ buy-and-hold basket each batch ($\kappa_c = \kappa_d = 1.0$ in `grasp_config.yaml`). The term is zero — gradient and value — while the portfolio's tail is no worse than the universe's, removing the calm-regime de-vol pressure; above budget it ramps quadratically, normalised by the budget so a 30%-worse tail costs the same in any regime. Because the budget widens in crisis batches, the only way a high-$\beta$ book is penalised there is by being worse than the basket itself: regime-conditional defensiveness stays learnable through Sharpe/Sortino while the unconditional low-vol carry trade stops being rewarded. This makes the term consistent with every other guard in the loss (one-sided, normalised, zero inside the acceptable region).

*Softplus-smoothed CVaR.* At $\alpha = 0.08$ over a 96-sample batch the hard indicator $\max(L-\xi, 0)$ fires on only $\lfloor \alpha T \rfloor \approx 7$ tail samples. An optional smoothing replaces it with $\tau \cdot \text{softplus}((L-\xi)/\tau)$, giving every sample gradient weight $\sigma((L-\xi)/\tau) \in (0,1)$ for lower gradient variance. Since $\text{softplus}(x) - \max(x, 0) = \log(1 + e^{-|x|}) \le \ln 2$, the smoothed estimator upper-bounds the hard CVaR with bias $\le \tau \ln 2/\alpha \to 0$ as $\tau \to 0$; $\tau$ is in return units ($\tau = 0.002 \approx 20$ bp in `cvar_smoothing_tau`, applied to both the portfolio and the EW-budget legs). This is deliberately

*not* the log-sum-exp smooth-max, which drops $\alpha$ and converges to the single worst loss. $\tau = 0$ recovers the exact hard path and is the back-compatible default; the smoothing takes effect only on the next retrain.

**One-sided diversification.** With $\mathrm{HHI} = \sum_i w_i^2$ and effective-$N$ target 30:

$$\mathcal{L}_{\mathrm{Div}} = \mathbb{E}_B\left[\frac{\max(\mathrm{HHI} - 1/30,\, 0)^2}{(1/30)^2}\right].$$

Zero when $1/\,\mathrm{HHI} \geq 30$ and quadratic above; gradient $2w_i$ attacks the largest holdings preferentially.

**Turnover.** With one-way turnover $T_{\mathrm{ow}} = \frac{1}{2}\|w_t - w_{t-1}\|_1$,

$$\mathcal{L}_{\mathrm{Turn}} = \mathbb{E}_B[\max(T_{\mathrm{ow}} - 0.04,\, 0)^2 + 50\, \max(T_{\mathrm{ow}} - 0.25,\, 0)^2].$$

A 4% free zone plus a 50× quadratic barrier above 25%.

**GAT edge entropy.** With effective edges per node $\mathrm{eff\_edges} = \exp(-\sum_j \bar{\alpha}_{ij} \log \bar{\alpha}_{ij})$ averaged over $(B, N)$:

$$\mathcal{L}_{\mathrm{GATEnt}} = \frac{\max(\mathrm{eff\_edges} - 20,\, 0)^2}{20}.$$

Fires only when the head-averaged effective edge count exceeds the target.

**Sector diversification.** With $\mathrm{sec\_HHI} = \sum_{s=1}^{12} \left(\sum_{i \in s} w_i\right)^2$ and target 1/7:

$$\mathcal{L}_{\mathrm{SecDiv}} = \mathbb{E}_B\left[\frac{\max(\mathrm{sec\_HHI} - 1/7,\, 0)^2}{(1/7)^2}\right].$$

This term is active ($\lambda_s = 50$) because the QP enforces only per-sector caps; without this penalty the policy could place 90% of the book in three defensive sectors at 30% each.

### 6.4 Training procedure

Three AdamW parameter groups are used (table 6). Per-module gradient clips are applied before the global clip: `temporal_encoder` and `spatial_encoder` at norm 2.0, `learnable_graph` at 1.0, `portfolio_head` at 1.5, with a final global clip at 2.0.

**Sharpness-aware minimisation.** Each optimiser step takes two forward/backward passes rather than one. The first ascends to the worst-case point in an $\ell_2$ ball of radius $\rho = 0.05$ around the current weights, $w \leftarrow w + \rho\, g/\|g\|$; the second evaluates the loss and its gradient there and applies the AdamW update from the original weights [6]. Minimising the loss over a neighbourhood rather than at a point favours flat minima, which is the property that matters here: the binding constraint on GRASP is out-of-sample behaviour under regime shift, and in-sample Sharpe overfits readily (section 11.4). The same effect has been reported for time-series transformers [7]. The trainer runs the two passes explicitly so it can reuse the per-module gradient clips between them, and the step requires `gradient_accumulation_steps`=1. The cost is one extra forward/backward per step, which only became affordable once the portfolio head moved off the CPU convex-program solve onto the GPU projection of section 6.2.4.

**Table 6.** GRASP training hyperparameters (authoritative `config/grasp_config.yaml`).

| | |
|---|---|
| Optimiser | AdamW, lr $5\times10^{-5}$, $\varepsilon = 10^{-5}$, $\beta = (0.9, 0.999)$ |
| Weight decay (encoder) | 0.048; (portfolio head) 0.045; biases/LN 0.0 |
| Scheduler | 5-epoch linear warmup $\rightarrow$ CosineAnnealingLR($T_{\max} = 15$, $\eta_{\min} = 10^{-6}$) |
| EMA decay | 0.985 ($\sim$67-step smoothing window, $\sim$4 epochs at batch 96) |
| Batch size | 96 |
| Max epochs | 130 |
| Train stride | 1 (hard-enforced, not configurable) |
| Ensemble | purged $K$-fold CV, $K$=4 (`n_folds`); members averaged in prediction space, equal weight over a val-Sharpe>0 gate |
| Validation | per fold: a contiguous regime-disjoint block (no shuffle), 20-day purge |
| Dropout | 0.42 |
| DropPath rate | 0.32 (max, linear per block) |
| Feature noise | Gaussian $\sigma = 0.040$ on all channels except CH_REGIME_ID (train) |
| Gradient clip (global) | 2.0 |
| Early stopping | patience 18 on the EMA-smoothed composite (trailing-5 Sharpe with a drawdown term); patience must exceed the 30-epoch cosine LR period so a member sitting in the LR trough is not killed prematurely |
| Drawdown veto | disabled in $K$-fold CV mode (an absolute DD threshold across regime-disjoint folds measures regime difficulty, not skill) |
| Sharpness-aware update | SAM [6], two-step ascent/descent, $\rho = 0.05$, non-adaptive; per-module clips reused between the two passes; requires `gradient_accumulation_steps`=1 |

### 6.5 Constraint enforcement

The portfolio head enforces all portfolio constraints in the forward pass through its convex projection:

- **Long-only:** $w_i \geq 0$ (pinned exactly by the final clamp)
- **Per-stock maximum weight:** $w_i \leq 0.07$ (held by the capped-simplex inner step)
- **Sector maximum weight:** $\sum_{i \in s} w_i \leq 0.30$ (satisfied to the dual-ascent tolerance, $< 10^{-3}$)
- **Gross exposure:** $\sum_i w_i = 1$ (pinned exactly by the final renormalise)

Because the constraints are enforced in the forward pass and the projection is differentiated by unrolling the dual-decomposition solver, the backward pass returns a gradient consistent with the binding constraint set, with no soft-penalty barrier terms in the loss. This avoids the cap-pinning pathologies of soft formulations and lets the gradients optimise the financial objectives directly, rather than trading off incommensurate penalty coefficients at the boundaries. A `cvxpylayers`/SCS solve of the same projection is retained only as a test-time correctness reference.

## 7 Backtesting Engine

### 7.1 Fee model

The fee constants are defined as module-level scalars in `features/backtesting/backtest_service.py` and are mirrored in `config/live_trading_config.yaml`; both must be kept in lockstep.

**Table 7.** Alpaca-equivalent fee schedule (per execution unless noted).

| Item | Value | Side |
|---|---|---|
| Commission | 0 bps | both |
| CAT fee | \$0.000216 / share | both |
| Slippage | 2 bps of notional | both |
| SEC fee | \$22.90 / \$1,000,000 | sell |
| FINRA TAF | \$0.000166 / share, cap \$8.30 | sell |

For an execution of notional $D = S \cdot p$ at price $p$ with $S$ shares:

$$\begin{aligned} C_{\text{slip}} &= D \cdot 0.0002 \\ C_{\text{CAT}} &= \lceil S \cdot 0.000216 \cdot 100 \rceil / 100 \\ C_{\text{SEC}} &= \lceil D \cdot 0.0000229 \cdot 100 \rceil / 100 \\ C_{\text{TAF}} &= \min(\lceil S \cdot 0.000166 \cdot 100 \rceil / 100,\ 8.30) \\ C_{\text{buy}} &= C_{\text{slip}} + C_{\text{CAT}} \\ C_{\text{sell}} &= C_{\text{slip}} + C_{\text{CAT}} + C_{\text{SEC}} + C_{\text{TAF}}. \end{aligned}$$

All ceiling operations round up to the nearest cent ($\lceil x \cdot 100 \rceil / 100$), exactly matching Alpaca's billing.

Portfolio-level deduction on rebalance day is applied multiplicatively to the *current* portfolio value, not to the initial capital:

$$\text{PV}_{t+1} = \text{PV}_t \cdot \left(1 - \frac{C_{\text{total}}(t)}{D_t}\right), \qquad D_t \ = \ \text{PV}_t \cdot C_0,$$

where $C_0$ is the initial capital, $D_t$ is the current portfolio value in dollar terms (`port_dollars` in code), and $C_{\text{total}}(t)$ is the dollar fee total for the rebalance (sum of $C_{\text{buy}}$ / $C_{\text{sell}}$ across all executed trades). $D_t$ does *not* cancel: $C_{\text{total}}(t)$ itself is computed from notionals $D_i = |\Delta w_i| \cdot D_t$, so the ratio $C_{\text{total}}(t)/D_t$ is approximately constant in $D_t$ (up to the per-execution cent-ceiling) and the percentage cost scales correctly as the portfolio compounds or draws down. Equivalently, in code:

```
tc_percent      = tc_dollars / port_dollars   # ratio scales with portfolio
portfolio_value *= (1.0 - tc_percent)         # multiplicative compounding
```

## 7.2 Execution model

The simulator rebalances every `REBALANCE_FREQ` = 1 trading day, matching the deployed live cadence (`features/grasp_agent/constants.py`) and the GRASP training objective which is computed at the same daily horizon. The operator-side `rebalance_freq: 5` field in `config/live_trading_config.yaml` is informational only. It denotes the expected number of trading days between cron-driven live trading invocations and has no effect on the per-cycle portfolio update. All trades execute at end-of-day close. The return earned on day $t$ uses the previous day's weights (lookahead-bias fix). Weight drift between rebalances is tracked with

$$w_{i,t} = \frac{w_{i,t-1}(1 + R_{i,t})}{\sum_j w_{j,t-1}(1 + R_{j,t})}.$$

The portfolio begins fully in cash so the initial entry cost is modelled. If a close is zero or missing, that symbol holds its prior weight and its rebalance is skipped.

### 7.3 Metrics

With excess returns $e_t = r_t - r_f/252$ (annual risk-free 4% by default), periods per year 252:

**Table 8.** Backtest metrics produced by `performance_metrics.py`.

| Metric | Formula |
|---|---|
| Sharpe | $(\bar{e}/\sigma_e)\sqrt{252}$ |
| Sortino | $(\bar{e}/\sigma_{e^-})\sqrt{252}$, threshold 0 |
| Calmar | annualised return / \| MaxDD \| |
| Max drawdown | $\min_t(\mathrm{PV}_t - \mathrm{cummax}_t)/\mathrm{cummax}_t$ |
| MaxDD duration | run-length of underwater periods |
| Turnover | one-way per rebalance, $\frac{1}{2}\sum_i \lvert w_{i,t} - w_{i,t-1}\rvert$ |
| Win rate | $\#\{r_t > 0\}/n$ (excludes day 0) |
| Profit factor | gross gains / gross losses |
| Beta vs SPY | $\mathrm{Cov}(r_p, r_m)/\mathrm{Var}(r_m)$ (ddof=1) |
| Jensen $\alpha$ | $r_p - [r_f + \beta(r_m - r_f)]$, annualised |
| Information ratio | $(r_p - r_m)/\mathrm{TE} \cdot \sqrt{252}$ |

Risk-adjusted metrics are reported as `insufficient_history` when fewer than 20 return periods are available.

### 7.4 Out-of-sample significance tests

Point estimates of Sharpe and excess return are reported alongside a battery of assumption-light significance tests in `performance_metrics.py`, wired into the per-window OOS diagnostics (`oos_diagnostics.assemble_diagnostics`). All operate on the non-overlapping daily OOS series and are designed for the regime of short, autocorrelated, fat-tailed financial samples in which the i.i.d. Sharpe standard error of Lo [9] is optimistic.

**Automatic block length.** `politis_white_block_size` implements the Politis–White [18] optimal block length for the circular block bootstrap with the Patton–Politis–White [19] correction, selecting the lag-window cutoff from the empirical autocorrelation flat-spot and combining it into $\hat{D}_{\mathrm{CB}} = \frac{4}{3}\hat{g}(0)^2$. This block length feeds both the moving-block Sharpe bootstrap and the paired test below.

**HAC covariance.** `hac_covariance` computes a Bartlett-kernel Newey–West estimator; when no bandwidth is supplied it uses the Andrews [20] AR(1) plug-in automatic bandwidth (the standard practical choice), so the autocorrelation correction is data-driven rather than a hand-picked lag.

**Sharpe-difference test.** `sharpe_difference_test` implements the Ledoit–Wolf [16] test of $H_0 : \mathrm{SR}_{\mathrm{portfolio}} = \mathrm{SR}_{\mathrm{benchmark}}$ against SPY and the EW-120 basket, reporting both a HAC delta-method $z$ (Andrews bandwidth) and a paired circular moving-block bootstrap $p$-value over the same return pairs — the matched-pairs structure cancels the common market factor, giving far more power than comparing two one-sample Sharpe intervals.

**Reality-check / SPA.** `hansen_spa_test` implements Hansen's Superior Predictive Ability test [17], the studentised, recentred refinement of White's Reality Check, with the null distribution from a Politis–Romano stationary bootstrap. It returns the consistent $p$-value bracketed by its lower (most-favourable) and upper (White RC / least-favourable) bounds, and is the correct instrument for the *data-snooping* incurred by the configuration search set (seed, overlay, and ablation sweeps) — distinct from the cross-window aggregation in section 11.4. Each estimator is covered by unit tests against analytic or i.i.d.-asymptotic baselines in `tests/backtesting/test_performance_metrics.py`, and the softplus CVaR of eq. (2) by the $\tau \to 0$ hard-path and $\leq \tau \ln 2/\alpha$ bias bounds in `tests/grasp_agent/test_loss_functions.py`.

# 8 Walk-Forward Evaluation

## 8.1 Window splitter

`WalkForwardWindowSplitter` partitions the daily regime stream into rolling train/validation/test triplets with embargo gaps at both intra-window split boundaries:

$$[\,\text{train } 504\text{d}\,|\,\text{embargo}_1\ 20\text{d}\,|\,\text{val } 252\text{d}\,|\,\text{embargo}_2\ 20\text{d}\,|\,\text{test } 126\text{d}\,].$$

The outer embargo (`val_test_embargo_days`) is enforced by the window splitter directly; the inner train→val embargo is enforced inside `features.grasp_agent.training.data_setup._prepare_splits`, which slices the validation tensor starting at index $n_{\text{train}} + \text{embargo\_steps}$ with $\text{embargo\_steps} = T_{\text{window}} + R - 1 \;=\; 20 + 1 - 1 \;=\; 20$. The same 20-day gap is applied at the val→test boundary by the trainer when a held-out test slice is passed in directly (the non-walk-forward training path). Both gaps remove the structural overlap between the $T$=20-day feature window of one split and the forward-return prices of the next: without them the last $T_{\text{window}} + R - 1$ feature windows of train include val-period prices (look-ahead leakage), and the last $R - 1$ forward-return rows of val are computed from prices that lie inside the test period (label leakage). Empirical impact of the embargo was a roughly 5× collapse of the val/test Sharpe gap on a representative window (val 4.76 → test −0.36 in the no-embargo configuration).

The stride equals the test window (126 days), so concatenated OOS test returns have no overlapping days, giving a well-defined non-overlapping standard error for the aggregate Sharpe. A regime-diversity gate requires the train slice to contain at least 3 unique VQ-VAE regime IDs; the Shannon entropy

$$H = -\sum_{k=1}^{K} p_k \log_2 p_k, \qquad H_{\max} = \log_2 K \approx 3.58\,\text{bits}$$

is recorded for diagnostics. With a ∼1850-day regime stream this yields roughly eight non-overlapping windows.

## 8.2 Per-window retraining: purged $K$-fold ensemble

The orchestrator invokes `features.grasp_agent.train_grasp` as a subprocess per window, passing window-specific date boundaries; the VQ-VAE encoder is fixed across windows. Within each window the pre-test span [train_start, val_end] (train + val = 756 trading days) is carved into $K$=4 contiguous, embargoed folds. Each fold is held out in turn as one member's *regime-disjoint* validation set while the remaining folds train that member (seeds 42–45); a 20-day purge

at each fold boundary removes the feature/forward-return overlap. The $K$ members are deployed together: at inference each emits a feasible weight vector, and the ensemble weight is their convex combination $\bar{w} = \sum_k c_k \, w^{(k)}$ with $\sum_k c_k = 1$, $c_k \geq 0$, which is itself feasible (the constraint set is convex). The combination coefficients $c_k$ are equal weight over a val-Sharpe>0 gate.

This replaces the earlier single-checkpoint, median-of-seeds selector entirely. The motivation is variance reduction without a selection step: a regime-disjoint $K$-fold average has no single epoch/seed "pick" whose selection noise (per-window test Sharpe SE at 126 days is $\approx 0.55$) needs attacking, and it forces every member to have been validated on a distinct macro regime. The per-member checkpoint drawdown veto is *disabled* in this mode: an absolute DD threshold applied across regime-disjoint folds measures the difficulty of each fold's regime, not the member's skill, and would collapse the ensemble onto the calmest folds. After retraining, the per-window backtest reads a precomputed feature parquet to avoid redundant indicator computation.

### 8.3 Artefact layout

```
data/walk_forward/<run_id>/
  metadata.json
  windows/
    window_001/
      grasp_agent/           # checkpoint dir (path-only by default)
      metrics.json
      window_metadata.json
    ...
  aggregate_results.json
```

A master metadata file is written after every window (`checkpoint_frequency=1`); `artifact_retention=null` keeps all window directories.

## 9 Live Trading

### 9.1 Cycle

A single trading cycle (`LiveTradingService.run_cycle`) executes the following ordered steps:

1. Market-hours guard via `AlpacaRepository.get_clock` (skipped on dry-run).
2. Portfolio snapshot: account, positions, latest prices.
3. `WarmupPipeline.run()`: yfinance download of 300 trading days for all 120 GRASP symbols plus the five macros; Alpaca gap-fill for the last `alpaca_lag_days=3` days; in-progress bar scrub; indicator computation; `MacroAlignmentService` alignment; regime encoding and causal post-processing; feature tensor and graph construction.
4. GRASP inference: `GRASPPortfolioGenerator.generate_weights` evaluates the QP head and returns a Dict[symbol, weight].
5. `OrderBuilder.build_orders`: target–current weight deltas are converted to `TradeOrder` objects with fee estimates identical to the backtester.
6. Order submission via `AlpacaRepository.submit_market_order`; sells are sorted before buys to free cash.

7. Persistence: trade log, portfolio history parquet, running metrics JSON.

Intermediate files use `tempfile.TemporaryDirectory` and are cleaned up after the feature tensor is built; no writes touch `data/raw/` or `data/features/`.

### 9.2 Credentials and outputs

Credentials are loaded from `.env` via `shared_kernel/infrastructure/config_pydantic.py`; if either of `ALPACA_API_KEY`, `ALPACA_API_SECRET` is missing and `dry_run=False`, the service raises `ValueError` on construction. The paper/live endpoint is selected by `ALPACA_PAPER` (default true).

**Table 9.** Live trading outputs.

| Path | Contents |
|---|---|
| `data/live_trading/trade_log.jsonl` | per-order JSONL (`cycle=trade`) + per-cycle summary (`cycle=summary`) |
| `data/live_trading/portfolio_history.parquet` | per-cycle NAV snapshots with weights and fee accumulation |
| `data/live_trading/metrics.json` | latest Sharpe, Sortino, Calmar, MaxDD, annualised return |

### 9.3 Order filtering

Orders pass through a deterministic waterfall before submission: notional below $10 is dropped (except orphan positions which are force- liquidated regardless), zero or negative prices are dropped, integer share counts of zero are dropped, sells are issued before buys. Constraints are enforced upstream by the QP head; `OrderBuilder` only emits drift warnings, with the per-stock warning threshold sourced from `config/grasp_config.yaml` (`constraints.max_weight`) at startup.

The live config field `constraints.max_position_pct` defaults to `null`: `live_trading_command._load_config` reads the GRASP cap (`max_weight`) and uses it plus a 50 basis-point slack ($w_{\text{cap}}^{\text{warn}} = w_{\text{cap}}^{\text{QP}} + 0.005$) as the warning threshold, so the live drift detector is always tighter than the QP cap by exactly one fixed slack. An explicit override is honoured but produces a startup warning when the override exceeds the QP cap by more than 100 bp, eliminating the previous silent gap in which a runaway upstream weight up to twice the QP boundary would never trip the warning. The sector warning threshold is 30% (same as QP). The $\varepsilon = 10^{-4}$ tolerance on both warnings absorbs floating-point rounding in the QP solve output (typically $10^{-6}$–$10^{-7}$) without admitting materially-out-of-spec weights.

### 9.4 Fee parity

`OrderBuilder._compute_fees` re-uses the exact constants and cent-ceiling rounding from the backtester's `BacktestService._run_simulation`. The `FeeConfig` dataclass and `config/live_trading_config.yaml` are the single authoritative sources; updates must be made in both files at once.

### 9.5 Volatility-managed gross overlay

A volatility-managed cash sleeve (Moreira–Muir [15]) scales the gross book down — never up — when realised volatility runs hot:

$$g_t = \mathrm{clip}\left( \frac{\sigma_{\text{target}}}{\hat{\sigma}_t^{\text{EWMA}}},\ g_{\min},\ 1 \right), \qquad w_t \leftarrow g_t\, w_t,$$

with the un-invested $(1 - g_t) \sum_i w_i$ fraction held in cash. $\hat{\sigma}_t^{\text{EWMA}}$ is the annualised exponentially-weighted realised volatility of the portfolio's own daily returns at half-life `vol_halflife`= 20, held at $g_t = 1$ until `vol_min_obs`= 20 returns accrue (so the warm-up window is identical to the un-overlaid baseline). Deployed parameters are $\sigma_{\text{target}} = 0.10$, $g_{\min} = 0.70$.

The overlay is a second *parity contract* alongside the fee model: the Moreira–Muir math (`vol_target_gross`, the half-life `EwmaVolEstimator`, and `apply_gross_factor`) lives in `shared_kernel/risk/gross_overlay.py` so neither the backtesting nor the live slice imports the other's internals (VSA rule) and the two cannot drift. The backtester re-exports the function for back-compatibility; the live cycle applies it in `LiveTradingService._compute_gross_factor` — EWMA over the persisted `portfolio_history.parquet` returns, mapped through the shared function — after GRASP inference and before `OrderBuilder`, which never renormalises, so the $(1 - g_t)$ remainder lands in cash. The `gross_overlay:` blocks in `backtesting_config.yaml` (`simulation.gross_overlay`) and `live_trading_config.yaml` must stay in sync, exactly as the fee schedule does. It is a *no-retrain* lever: a pure post-inference rescale that requires no change to the trained policy. It is shipped `enabled: false` in both configs — it is the cheapest candidate for cutting the −13% to −14% drawdown tail (section 11.4), but walk-forward crisis-window validation is the deploy gate, and a single calm OOS window leaves it correctly inert at the designed $\sigma_{\text{target}} = 0.10$. The backtest↔live parity is locked by dedicated tests (`tests/shared_kernel/test_gross_overlay.py`, `tests/backtesting/test_gross_overlay.py`, `tests/live_trading/test_live_gross_overlay.py`): the same return path yields the same $g_t$ and the same cash residual on both code paths, mirroring the fee-model contract.

## 10 Configuration Files

**Table 10.** All authoritative configuration files.

| File | Purpose |
|---|---|
| `config/symbols.yaml` | Tier-0/1/2 universe definitions and reserve pool. |
| `config/vqvae_config.yaml` | VQ-VAE v3.2 architecture, loss weights, training. |
| `config/grasp_config.yaml` | GRASP architecture, loss weights, training (authoritative over `constants.py`). |
| `config/walk_forward_config.yaml` | Window sizes, stride, embargo (fold count $K$ inherited from `grasp_config.yaml`). |
| `config/backtesting_config.yaml` | OOS symbols, simulation settings, and the `simulation.gross_overlay` block. |
| `config/live_trading_config.yaml` | Constraints, fee schedule, warmup days, output paths, and the `gross_overlay` block (parity-mirrored with backtesting). |
| `.env` | `ALPACA_API_KEY`, `ALPACA_API_SECRET`, `ALPACA_PAPER`. Never committed. |

# 11 Empirical Results

This section reports the deployed system's behaviour on four fronts: the VQ-VAE regime stream, the GRASP $K$-fold ensemble training trajectory, a single-window out-of-sample backtest on the first five-and-a-half months of 2026, and an eight-window non-overlapping walk-forward spanning 2022–2026. The single-window slice is strictly disjoint from both the GRASP training window (ending 2025-12-31) and the VQ-VAE training window (ending 2025-06-30); the walk-forward is the proper instrument for distinguishing genuine out-of-sample signal from window-specific luck, and section 11.4 reports it in full.

## 11.1 Regime stream diagnostics

The VQ-VAE was trained on 1,852 trading days (2019–2026) of aligned macro inputs and produces a regime label per day after the causal post-processing stack described in Section 5.5. All twelve codewords are active (perplexity 10.6, entropy 2.36 bits against a $\log_2 K \approx 3.58$ ceiling), well clear of the 0.75-of-maximum warning floor. Reconstruction MSE has a heavy right tail (mean 0.758, p95 2.340, p99 4.960) with 6.7% of days exceeding the threshold of 2.0; this tail concentrates in the March–April 2020 COVID dislocation and in three 2024–2026 volatility episodes, exactly the kind of OOD windows for which the inference-time confidence signal (Section 5.5) is designed to suppress downstream regime commitment.

After the causal mode filter and 10-day minimum-dwell enforcement, the regime stream contains 73 commits with a median dwell of 22 days and a 30-day switch rate of 1.18, consistent with the persistent macro regimes assumed by the training objective (the raw, pre-smoothing sequence had 190 switches). The most-utilised regime carries 17.0% of days; the least 2.2%; no codeword is dead. The per-regime macro fingerprints decompose cleanly: high-VIX / low-SPY regimes (R3, R9) map to drawdown periods with annualised Sharpe of $-1.53$ and $-0.48$ respectively in the realised SPY return stream, while low-vol / positive-momentum regimes (R1, R4) show Sharpe $+2.19$ and $+1.46$. This is not a calibration target (the encoder is trained on reconstruction, not on returns), but it confirms the codebook has discovered an axis with macroeconomic meaning. Figure 2 collects the full VQ-VAE diagnostic dashboard: codeword utilisation, the regime timeline against SPY, the reconstruction-error tail, and the per-regime macro fingerprints.

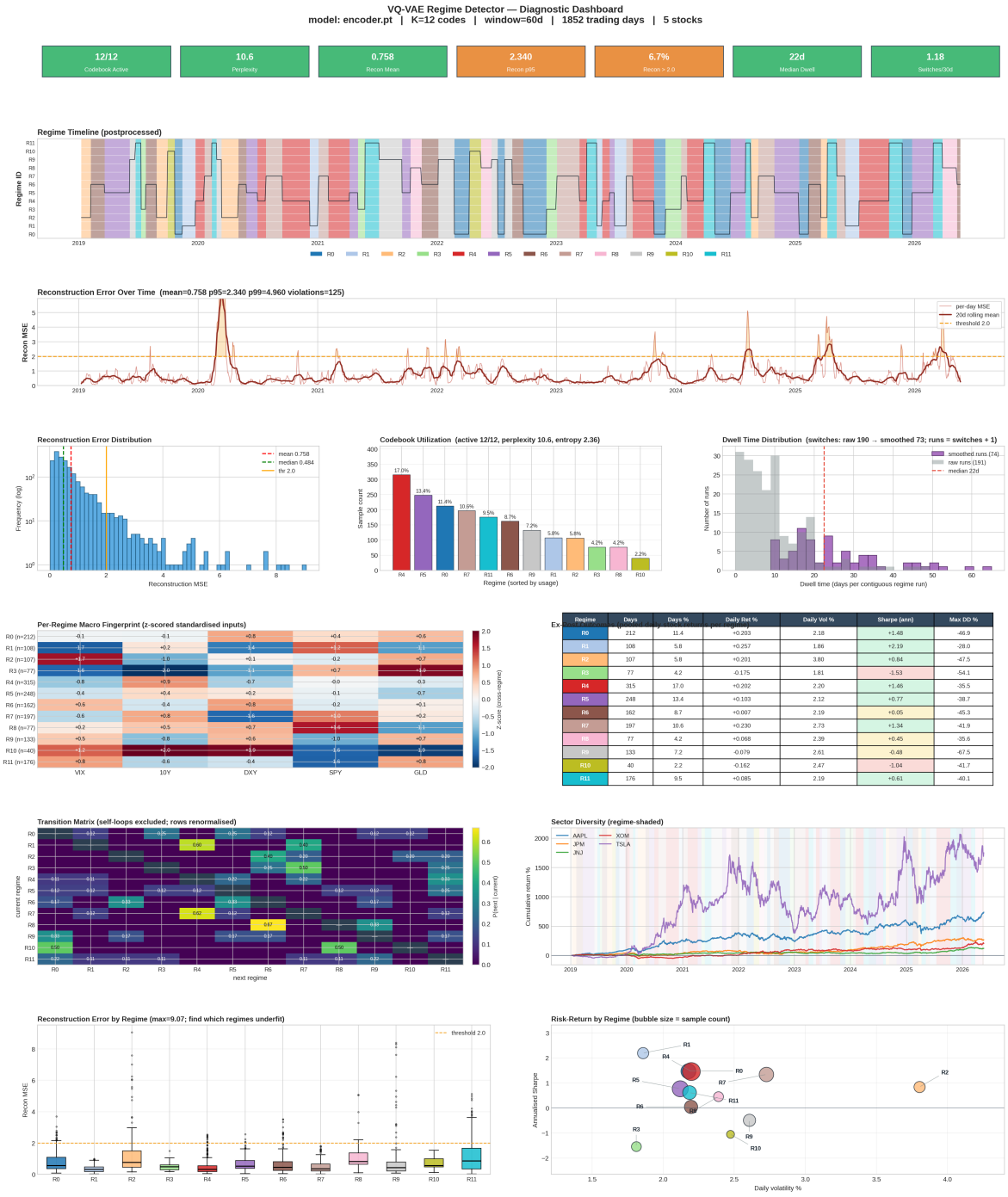



**Figure 2.** VQ-VAE regime-encoder diagnostic dashboard. Codebook utilisation and perplexity (all twelve codewords active), the causal post-processed regime timeline overlaid on SPY, the reconstruction-error distribution with its heavy right tail concentrated in macro dislocations, and the per-regime macro fingerprints (VIX/TNX/DXY/SPY/GLD).

## 11.2 GRASP $K$-fold ensemble training

The deployed model is the purged $K$=4-fold cross-validation ensemble (section 8) trained on the full history through 2025-12-31. Each member (298,337 parameters) trains under the deployed configuration (batch 96, AdamW $5\times10^{-5}$, dropout 0.42, drop-path 0.32, feature noise $\sigma = 0.040$, cosine-$T_{\max} = 15$ with 5-epoch warmup, EMA decay 0.985, sharpness-aware updates at $\rho = 0.05$, max 130 epochs, early-stopping patience 18) on three of the four regime-disjoint folds and

validates on the fourth; all four members are then averaged in prediction space with equal weight over a val-Sharpe>0 gate. On an RTX 5070 the four members train in ∼18 minutes total. Per-member validation results are in table 11; the ensemble's held-out (sealed-test) metrics are in table 12.

**Table 11.** Per-member validation at the EMA-best-composite checkpoint. Each fold validates on a distinct, regime-disjoint macro block, so the spread in val Sharpe and val drawdown reflects regime difficulty, not relative skill.

| Member | Seed | Val Sharpe | Val MaxDD | Epochs |
|---|---|---|---|---|
| 0 | 42 | 0.79 | 35.1% | 33 |
| 1 | 43 | 1.35 | 21.4% | 56 |
| 2 | 44 | 1.33 | 14.9% | 54 |
| 3 | 45 | 1.32 | 12.2% | 63 |
| Ensemble (mean) | | 1.20 | 20.9% | n/a |

**Table 12.** Deployed $K$-fold ensemble on the sealed-test windows (76 windows held out from all four folds).

| Metric | Value |
|---|---|
| Sharpe (ann.) | 2.40 |
| Sortino (ann.) | 3.77 |
| Calmar | 4.30 |
| Annualised return | 26.4% |
| Annualised volatility | 11.0% |
| Maximum drawdown | 6.14% |

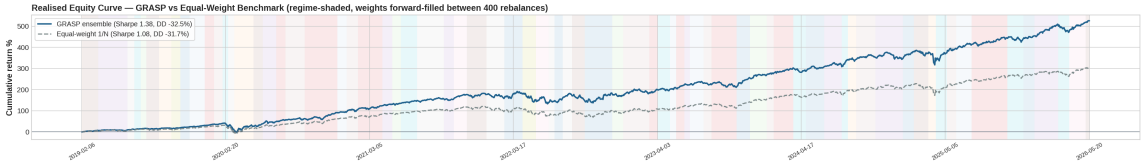
Two observations. First, the ensemble mean is capped by its hardest fold: member 0's validation block (val Sharpe 0.79, drawdown 35.1%) is a crisis-dominated regime on which every member would struggle, while the three calmer folds sit at 1.32–1.35. Equal-weight prediction-space averaging deliberately keeps the crisis fold in the deployed mix rather than selecting it away, since dropping it would only flatter the in-sample number. Second, because the checkpoint is selected on the *EMA-smoothed validation* composite, the early-stopping epoch varies widely across folds (member 0 stops at 33, member 3 runs to 63, none reaches the 130-epoch cap) without affecting which weights deploy: each member contributes its best-validation snapshot, not its final overfit one. The full GRASP diagnostic dashboard (realised equity versus the equal-weight benchmark, concentration and turnover traces, the learned GATv2 attention topology, and per-member health) is shown in fig. 3.

# GRASP Portfolio Agent — Diagnostic Dashboard

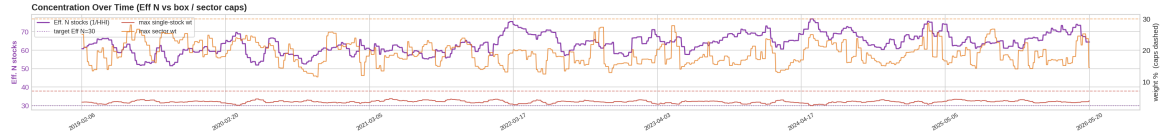
**model: grasp_agent | 4 K-fold members | N=120 stocks | 400 rebalances (1832 trading days) | window=20d**

| 4 | 1.38 | +28.7% | -32.5% | 64 | 3.6% | 19% | 14.5% | 54.2 |
|---|---|---|---|---|---|---|---|---|
| Ensemble Members | Sharpe (ann) | CAGR | Max DD | Eff. N stocks | Avg max wt | Avg max sector | Turnover | GAT edges/node |

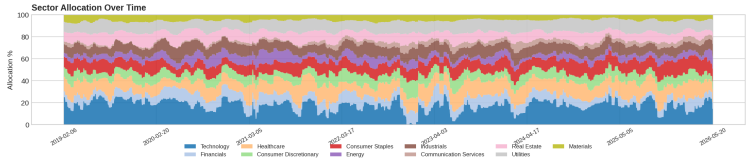
## Realised Equity Curve — GRASP vs Equal-Weight Benchmark (regime-shaded, weights forward-filled between 400 rebalances)



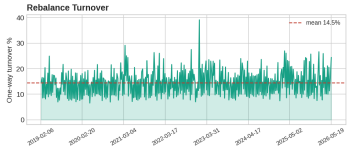

## Concentration Over Time (Eff N vs box / sector caps)



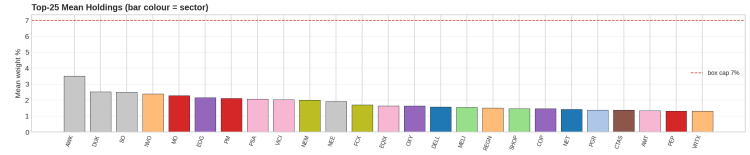

## Sector Allocation Over Time



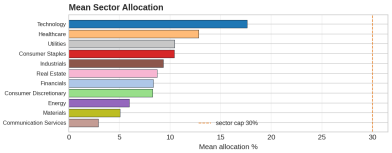

## Rebalance Turnover



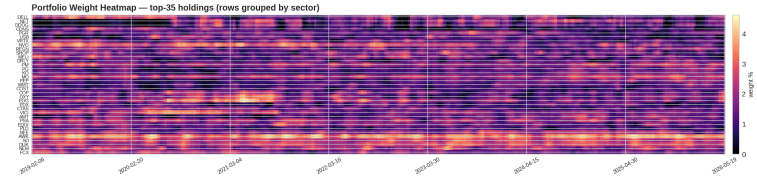

## Top-25 Mean Holdings (bar colour = sector)




## Mean Sector Allocation



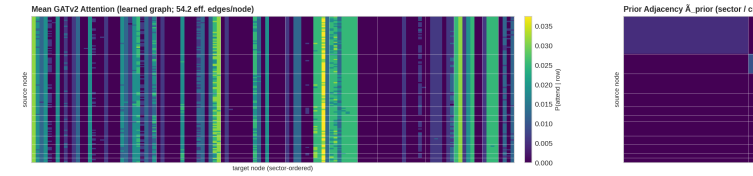

## Portfolio Weight Heatmap — top-35 holdings (rows grouped by sector)



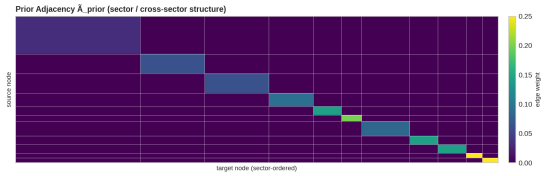

## Ensemble Member Health (grey = dropped by val-Sharpe gate)

| Member | Val Sharpe | Val DD % | Epochs | Ens. wt |
|---|---|---|---|---|
| m0 | 0.79 | 35.1 | 33 | 25% |
| m1 | 1.35 | 21.4 | 56 | 25% |
| m2 | 1.33 | 14.9 | 54 | 25% |
| m3 | 1.32 | 12.2 | 63 | 25% |

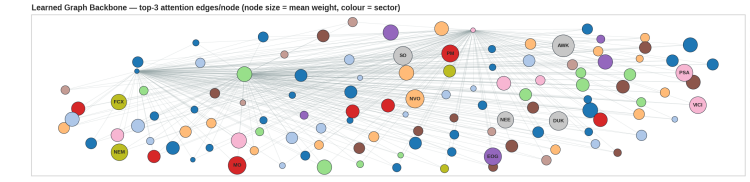
## Mean GATv2 Attention (learned graph; 54.2 eff. edges/node)



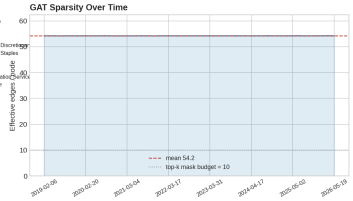

## Prior Adjacency Ã_prior (sector / cross-sector structure)




## Learned Graph Backbone — top-3 attention edges/node (node size = mean weight, colour = sector)



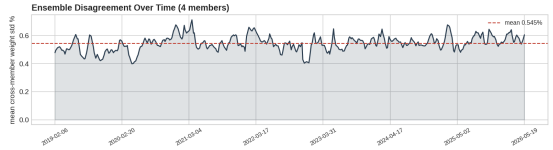

## GAT Sparsity Over Time



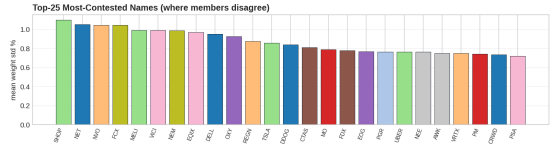

## GRASP Book Drawdown (underwater from peak)


## Constraint Binding Over Time (how hard the QP shapes the book)

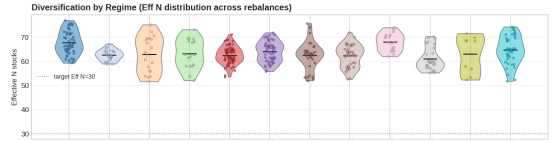

## Ensemble Disagreement Over Time (4 members)



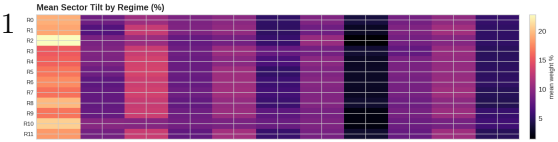

## Top-25 Most-Contested Names (where members disagree)




## Diversification by Regime (Eff N distribution across rebalances)




## Mean Sector Tilt by Regime (%)



### 11.3 Single-window out-of-sample backtest

This single-window OOS backtest covers 2026-01-02 through 2026-06-26 (121 trading days), one trading day after the deployed ensemble's training cutoff (2025-12-31) and ∼185 days after the VQ-VAE training cutoff (2025-06-30). All 120 stocks are GRASP-universe constituents; the SPY total-return series is the benchmark. The simulator applies the same Alpaca fee schedule (table 7) with cent-ceiling rounding identical to the live execution path; capital starts at $100,000 and rebalances daily at close. Headline metrics are in table 13.

**Table 13.** Single-window out-of-sample backtest, 2026-01-02 to 2026-06-26 (121 trading days), deployed $K$-fold ensemble. "Excess" is REGENT total return minus SPY total return over the same window; IR uses the daily excess-return path.

| Metric | Value |
|---|---|
| Total return | +13.23% |
| Annualised return | +29.53% |
| Annualised volatility | 9.73% |
| Sharpe | +2.30 |
| Sortino | +3.30 |
| Calmar | +4.90 |
| Maximum drawdown | −6.03% |
| Max-DD duration | 42 trading days |
| Win rate | 56.67% |
| Profit factor | 1.58 |
| Daily turnover (one-way) | 2.19% |
| Transaction cost paid | $145.68 |
| SPY benchmark total return | +8.06% |
| Excess return vs SPY | +5.17% |
| Information ratio | +0.91 |
| Beta vs SPY | +0.48 |
| Jensen's $\alpha$ (annualised) | +19.04% |

On this window the ensemble both out-returns and out-risk-adjusts the benchmark: total return +13.23% versus SPY's +8.06% (excess +5.17%) at a Sharpe of 2.30, with a contained maximum drawdown (−6.03%) and a moderate $\beta = 0.48$. The positive information ratio (+0.91) and annualised Jensen's $\alpha$ of +19.04% say the active return stream is favourable per unit of its own volatility and well above what $\beta$ alone would predict. Against the harder EW-120 control (the identical 120 names, equal-weighted) the edge is narrower (+1.89% excess, EW information ratio 0.50, $\beta_{\text{EW}} = 0.67$): holding the universe fixed, the learned tilt clears equal-weight by a smaller margin and buys drawdown control alongside it. The caveat is statistical, not directional: at 121 days the bootstrap 95% Sharpe CI is still wide ([+0.02, +5.54], P(Sharpe>0)= 97.8%), so the walk-forward of section 11.4, not this window, is the instrument that should be trusted.

Figure 4 shows the portfolio NAV against both SPY buy-and-hold and the EW-120 equal-weight book: REGENT leads SPY throughout (its cumulative excess widens sharply through the March SPY drawdown), while EW-120 catches up only in the late-May/June rally where the constrained per-stock and sector caps keep REGENT from concentrating into the leaders. Figure 5 plots the rolling 63-day Sharpe, which holds above the +1.0 target across the window and strengthens through the spring as absolute returns accelerate. The position heatmap (fig. 6) shows the portfolio head's behaviour: a long tail of ∼0.3–0.8% satellite weights under the 7% cap, with the effective-$N$ trace (red overlay) holding well above the $1/\text{HHI} \geq 30$ diversification floor.

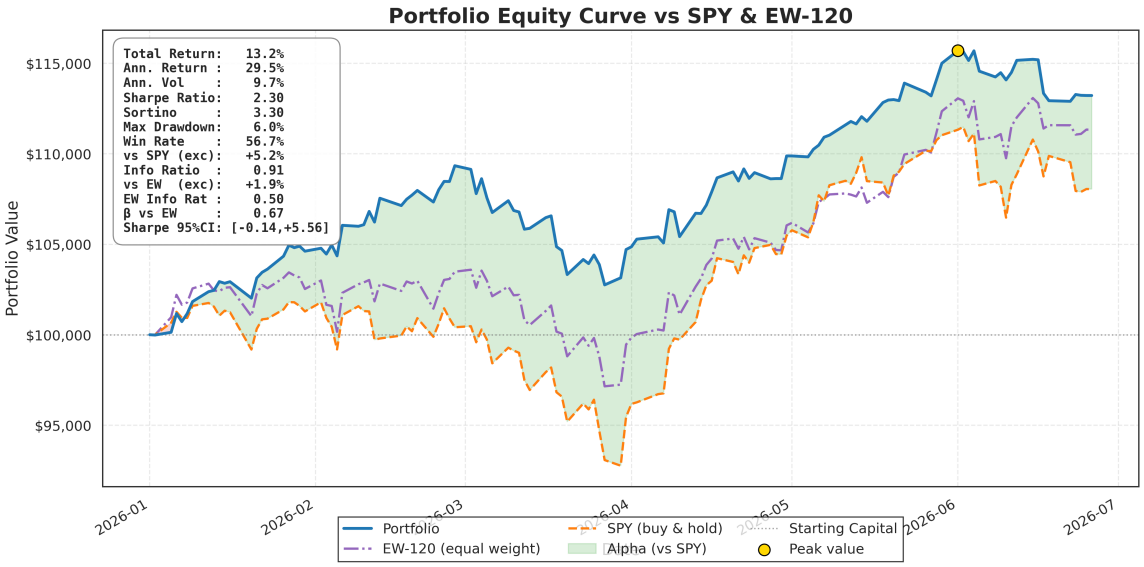



**Figure 4.** Single-window out-of-sample portfolio equity curve versus SPY buy-and-hold and the EW-120 equal-weight book, 2026-01-02 to 2026-06-26. Green shading marks REGENT's lead over SPY; red shading SPY's lead over REGENT.

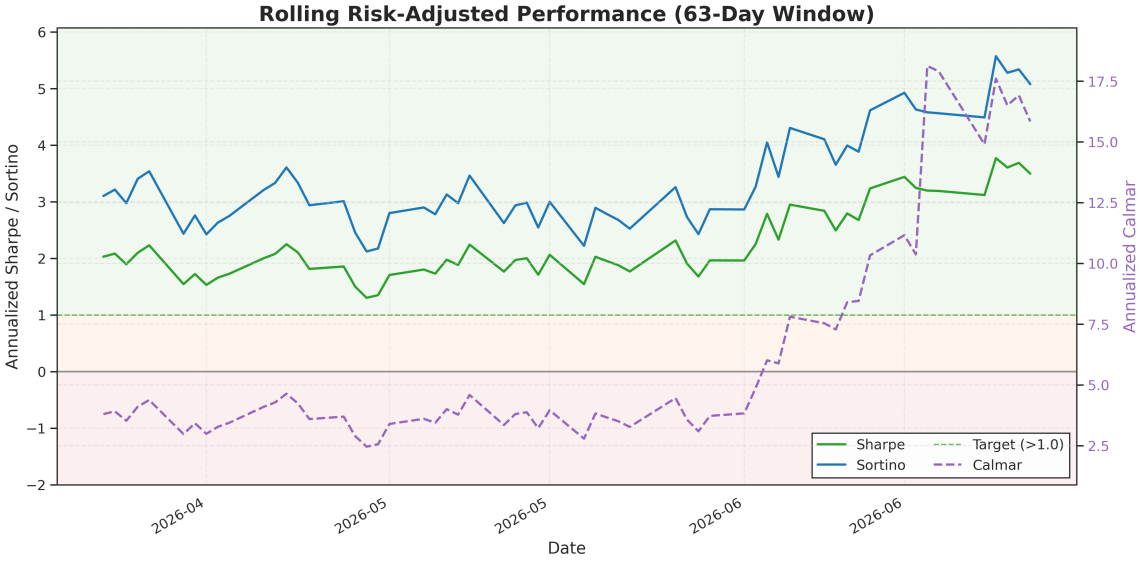



**Figure 5.** Rolling 63-day Sharpe ratio over the OOS window. The metric stays in the strategy-acceptable band ($>+1.0$) across the window; the cumulative excess over SPY peaks in early spring and compresses as SPY rallies into summer.

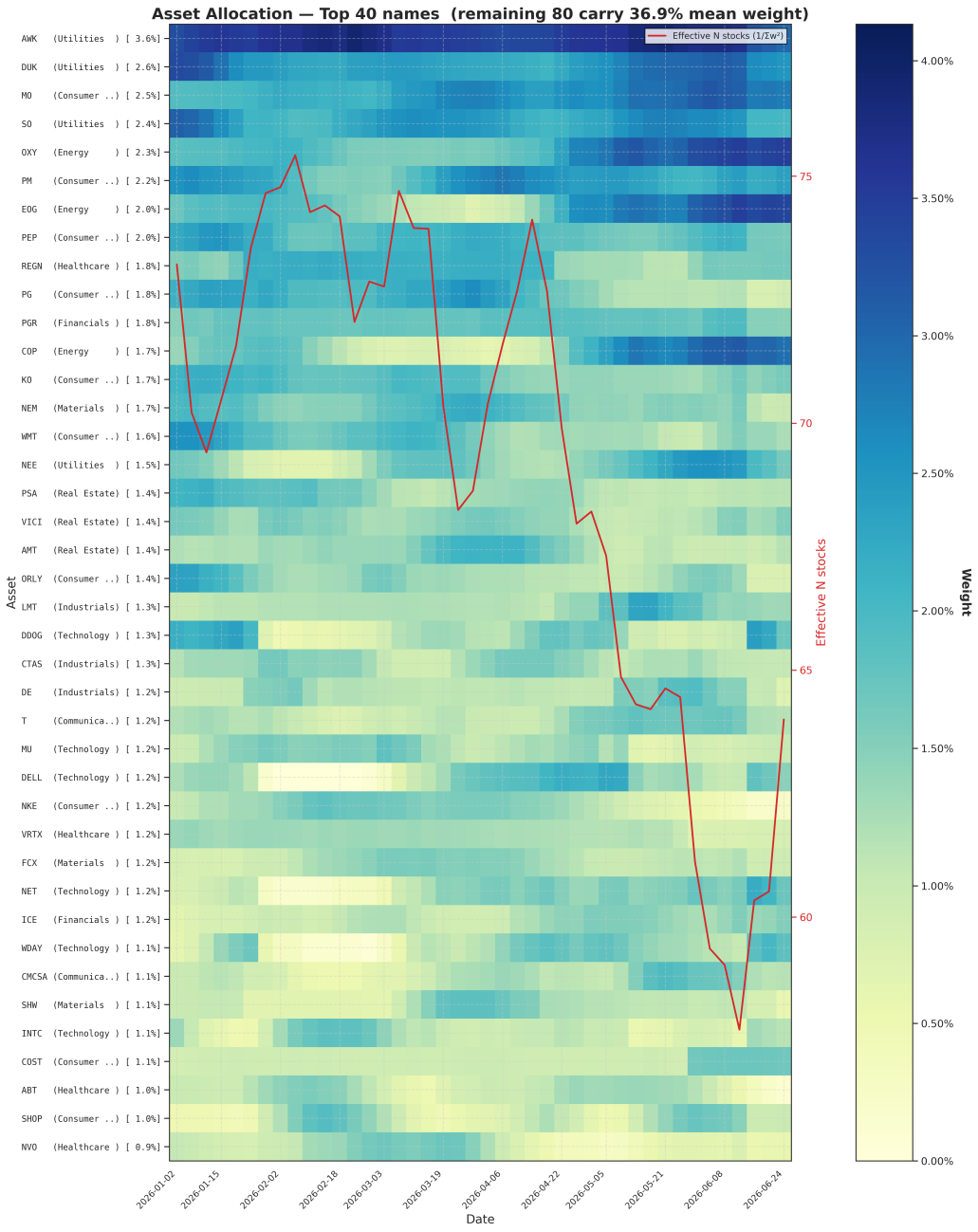



**Figure 6.** Asset allocation over the single-window OOS, top 40 names by mean weight. The red overlay shows the effective number of stocks ($1/\sum w^2$), which stays well above the diversification target of 30.

Figure 7 collects the OOS backtest diagnostic dashboard: the monthly-return heatmap, the bootstrap Sharpe distribution (realised +2.30, P(Sharpe$>0$) = 97.8%, 95% CI $[+0.02, +5.54]$), per-regime return attribution against the benchmark (the one negative contribution is the high-volatility regime 11), and the rolling market $\beta$/correlation that the scalar $\beta = 0.48$ summarises.


**Figure 7.** Single-window OOS backtest diagnostic dashboard, 2026-01-02 to 2026-06-26. Headline metrics, monthly-return heatmap, bootstrap Sharpe distribution with analytic and resampled 95% confidence intervals, per-regime return attribution (portfolio vs benchmark vs excess), and the rolling 63-day market $\beta$ and correlation versus SPY.

## 11.4 Walk-forward results

The single window above is suggestive but underpowered. The proper test is the eight-window non-overlapping walk-forward of section 8: each window retrains a fresh $K$=4-fold ensemble on its own 504/252-day train+val span and backtests on the following embargoed 126-day test slice, so the eight test slices tile 2022-02 through 2026-02 with zero overlap. Table 14 reports the per-window test metrics; the aggregate is in table 16.

The benchmark panel settles two questions the aggregate table alone leaves open. First, on market exposure: the realised $\beta_{\text{SPY}}$ spans 0.60–0.95 (mean 0.80). The book is *moderately* defensive, not the $\beta \approx 0.5$ the single calm-2026 window (table 13, $\beta$=0.47) suggested in isolation; that window is the low tail of the distribution, not its centre. The deepest drawdowns (W1,

**Table 14.** Per-window walk-forward test metrics. Each row is a fresh ensemble retrained on data ending 20+ days before the test slice. Turnover is one-way per rebalance.

| W | Test period | Sharpe | Sortino | Calmar | Return | MaxDD | Turn |
|---|---|---|---|---|---|---|---|
| 1 | 2022-02-07 / 2022-08-08 | +0.06 | +0.09 | 0.18 | +1.46% | −16.36% | 2.13% |
| 2 | 2022-08-09 / 2023-02-07 | +0.51 | +0.95 | 0.93 | +6.56% | −14.55% | 2.12% |
| 3 | 2023-02-08 / 2023-08-09 | +1.07 | +1.59 | 2.63 | +8.21% | −6.50% | 1.92% |
| 4 | 2023-08-10 / 2024-02-08 | +1.89 | +3.07 | 3.80 | +14.61% | −8.25% | 1.87% |
| 5 | 2024-02-09 / 2024-08-09 | +1.86 | +2.62 | 5.38 | +13.38% | −5.31% | 2.50% |
| 6 | 2024-08-12 / 2025-02-11 | +1.74 | +2.45 | 3.34 | +11.07% | −6.99% | 1.72% |
| 7 | 2025-02-12 / 2025-08-13 | +0.67 | +0.87 | 1.17 | +8.39% | −15.00% | 2.00% |
| 8 | 2025-08-14 / 2026-02-12 | +1.49 | +2.21 | 3.81 | +9.40% | −5.17% | 1.81% |

**Table 15.** Per-window benchmark panel against SPY buy-and-hold, replayed on the identical test slice through the same fee/slippage engine. Returns are totals; $\beta/\alpha$/IR are against SPY ($\alpha$ annualised, IR on the daily excess-return path). The aggregate comparison against the composition-matched EW-120 book is reported at the stitched-equity level in the text.

| W | Model | SPY | $\beta_{\text{SPY}}$ | $\alpha_{\text{SPY}}$ | $\text{IR}_{\text{SPY}}$ |
|---|---|---|---|---|---|
| 1 | +1.46% | −6.98% | 0.86 | +14.0% | +2.53 |
| 2 | +6.56% | +1.81% | 0.90 | +9.8% | +1.58 |
| 3 | +8.21% | +9.37% | 0.82 | +0.3% | −0.42 |
| 4 | +14.61% | +12.60% | 0.95 | +5.8% | +0.60 |
| 5 | +13.38% | +7.02% | 0.73 | +16.8% | +1.46 |
| 6 | +11.07% | +14.24% | 0.60 | +3.4% | −0.72 |
| 7 | +8.39% | +7.52% | 0.83 | +3.8% | +0.12 |
| 8 | +9.40% | +6.24% | 0.69 | +9.6% | +0.84 |
| mean | +9.14% | +6.48% | 0.80 | +7.9% | +0.75 |

W2, W7, all $\leq -14.5\%$) occur in the systemic-selloff windows, consistent with a book that damps but does not escape broad selloffs. Second, on the source of return: against SPY the model out-returns in 6/8 windows and posts a positive Jensen $\alpha_{\text{SPY}}$ in all eight (mean +7.9% annualised), with the two raw-return shortfalls (W3, W6) being late-cycle melt-ups where a $\beta<1$ book mechanically trails a rallying index even while its risk-adjusted $\alpha$ stays positive. Against the harder, composition-matched EW-120 control (the identical 120 names, equal-weighted, replayed through the same engine) the honest read is at the chained-equity level: stitched across all eight test slices REGENT compounds to +100.2% versus EW-120's +83.6% and SPY's +61.0% (fig. 8). Holding universe and sector composition fixed, the model therefore clears equal-weight by ∼17 percentage points cumulatively while running materially below it in drawdown, which is the correct rebuttal to the concern that the result is a sector-composition artefact: the EW-120 book shares that composition exactly, so the residual edge over it is a learned-tilt and risk-control effect, not a universe-selection one.

**Table 16.** Walk-forward aggregate over the eight non-overlapping test windows.

| Metric | Mean | Median | Min | Max |
|---|---|---|---|---|
| Sharpe | +1.16 | +1.28 | +0.06 | +1.89 |
| Sortino | +1.73 | +1.90 | +0.09 | +3.07 |
| Calmar | +2.66 | +2.99 | +0.18 | +5.38 |
| Total return | +9.14% | +8.89% | +1.46% | +14.61% |
| Annualised return | +19.25% | +18.58% | +2.94% | +31.35% |
| Maximum drawdown | −9.77% | −7.62% | −5.17% | −16.36% |
| Win rate | 55.8% | 56.8% | 48.8% | 60.8% |


**Figure 8.** Walk-forward diagnostic dashboard (run `wf_20260628_083811`), eight non-overlapping windows tiling 2022-02 through 2026-02. Top: stitched out-of-sample equity (GRASP +100.2%) chained across the eight test slices against SPY (+61.0%) and an equal-weight book of the same 120 names (+83.6%), with alternating bands marking window boundaries. Middle: per-window test Sharpe, total return, maximum drawdown and turnover; the in-sample member-validation-vs-realised-test *generalisation gap* (mean +0.19); and per-fold member health. Bottom: the SPY benchmark panel ($\beta/\alpha$/IR per window).

The aggregate is positive and consistent rather than spectacular: a positive test Sharpe in 8/8 windows, a mean of 1.16 (median 1.28), and a median Calmar of 2.99 that clears the 2.0 internal bar. Five of the eight windows are "deploy-grade" (Sharpe>1). The dispersion is informative and honest: performance tracks regime difficulty almost monotonically. The two weakest windows are W1 (Sharpe 0.06, the 2022-H1 rate-shock drawdown) and W7 (Sharpe 0.67, the early-2025 correction), and the deepest drawdowns (−16.36% W1, −15.00% W7, −14.55% W2) coincide with the broad-market selloffs of those periods: the model damps but does not escape systemic risk, consistent with the moderate $\beta_{\text{SPY}} \approx 0.80$ profile (range 0.60–0.95, table 15) the loss induces. The strong windows (W4, W5) reach Sharpe $\approx 1.9$ with Calmar 3.8–5.4, and W6/W8 sit at Sharpe 1.5–1.7. The full per-window diagnostic dashboard — stitched OOS equity against SPY and EW-120, the per-window Sharpe/return/drawdown/turnover bars, the in-sample-vs-realised generalisation gap, per-member health, and the SPY benchmark panel — is collected in fig. 8.

**Aggregate significance.** The naive non-overlapping estimate $\sigma/\sqrt{8} = 0.65/\sqrt{8} \approx 0.23$ treats the eight window Sharpes as i.i.d. and so understates dispersion: the windows share one fixed VQ-VAE encoder and one fixed universe, which induces positive cross-window dependence the $\sqrt{8}$ scaling ignores. We therefore report assumption-lighter tests on the concatenated daily OOS series ($T = 1008$ non-overlapping days, equity chained across windows). The pooled daily Sharpe annualises to 1.16, in line with the per-window mean (1.16). Its Lo [9] i.i.d. standard error gives $t = 2.33$; a Newey–West HAC correction (automatic lag $L = 6$) on the mean daily return gives $t = 2.42$ ($p \approx 0.016$ two-sided), so the autocorrelation adjustment does not erase significance. Independently, a sign test on the eight window outcomes (8/8 positive) has $p = 2^{-8} = 0.0039$ under the no-skill null. The single 121-day window could establish none of these on its own; the walk-forward can.

### 11.5 Read-out

The walk-forward is the load-bearing result: a positive Sharpe in every window and a median above the deploy bar, earned on regime-disjoint retrains with no window-specific tuning. The system is, by construction, a risk-adjusted-return strategy rather than a market-tracker (the loss trades a fraction of upside for tail-loss control), and the per-window record confirms it, with the strongest results in trending regimes and the weakest in the systemic-selloff windows it can only damp. Whether that trade is worth taking is a function of the investor's volatility budget, not the model's correctness.

Three honest open questions remain. (i) The drawdown tail is the binding constraint: three windows still print −14.5% to −16.3%, and cutting that tail without surrendering the trending-window Sharpe is the active line of work (the volatility-managed gross overlay of section 9.5 is implemented as a shared-kernel no-retrain cash sleeve, disabled pending crisis-window validation). (ii) Whether the moderate $\beta_{\text{SPY}} \approx 0.80$ defensive lean (table 15) should be closed under a different objective hierarchy (an information-ratio-against-equal-weight term is prototyped but not deployed). The EW-120 panel shows the residual edge over a composition-matched book is risk-adjusted rather than raw-return, which is exactly what that term would target. (iii) Whether the encoder justifies its capacity in windows where macro state is uninformative for the cross-section. The walk-forward is the instrument for adjudicating each, and the figures above are reported on its deployed ensemble.

## 12 Discussion

Four design choices in REGENT were not obvious in advance and deserve discussion now that the empirical record (section 11) is on the table.

**Soft penalties versus a differentiable projection.** The original GRASP head used a softmax projection with soft-penalty terms in the loss for the per-stock cap, the sector cap, and the gross exposure budget. Two failure modes appeared consistently. First, *cap-pinning under non-differentiable boundaries*: once a name hit the 7% cap the penalty gradient saturated, the score gradient was clipped at the boundary, and the optimiser routed its conviction into secondary names whose ordering relative to the pinned name became meaningless. Second, *incommensurate coefficients*: the per-stock, sector, and gross penalties had penalty surfaces with different curvatures, and tuning their coefficients amounted to picking which constraint the optimiser should prefer to violate. The differentiable projection removes both pathologies by enforcing the constraints in the forward pass and returning a gradient consistent with the binding constraint set, so the gradient on the score $z_i$ correctly attributes the marginal effect of a unit increase in stock $i$'s desirability. The qp_ridge $\gamma = 0.03$ is what makes this tractable: it gives the projection a unique minimiser and a Lipschitz Jacobian without materially perturbing the unconstrained solution (interior compression $1/(1+\gamma) \approx 0.971$). The deployed head solves that projection on the GPU by unrolling a dual-decomposition solver rather than calling `cvxpylayers` on the CPU; the two agree to about $10^{-6}$ on the forward pass, and moving the solve on-device is what made the second pass of sharpness-aware training (below) affordable.

**Sharpness-aware training.** GRASP's binding constraint is generalisation, not in-sample fit: with 298,337 parameters and a few thousand overlapping windows, a member can push its training Sharpe well past its validation Sharpe inside a single fold. Sharpness-aware minimisation attacks this directly by minimising the loss over an $\ell_2$ ball of radius $\rho = 0.05$ around the weights instead of at a single point, which steers the optimiser toward flatter minima that degrade less under the distribution shift between folds. It is not free: every step costs a second forward/backward, so it was off the table while the portfolio head ran on the CPU. Once the head moved to the GPU projection, the second pass became cheap enough to keep on by default. The same idea has improved out-of-sample accuracy for time-series transformers [7], whose failure mode, sharp minima that fit one regime and break on the next, is the one that hurts here.

**Why the 20-day embargo matters more than expected.** The most surprising empirical finding during development was the $\sim 5\times$ collapse in the val-test Sharpe gap once the dual embargo (train→val and val→test, $T_{\text{window}} + R - 1 = 20$ days each) was enforced. Without it, the val Sharpe on representative windows climbed past 4.0, an obvious red flag that, before the fix, was simply attributed to "a good window." The mechanism is mechanical: the last 19 feature windows of train include prices that extend into val, and (under $R > 1$ rebalancing) the last $R - 1$ forward-return rows of val are computed using prices inside test. Across walk-forward folds this was sufficient to inflate val Sharpe above 4.0 and pull test Sharpe to $-0.36$ on the same window, because the model selected on val ended up overfitted to information that does not exist at deployment. The same embargo is enforced by the window splitter at the val→test boundary and by `_prepare_splits` at the train→val boundary; deleting either reproduces the failure.

**Inference-time confidence asymmetry.** The decision to compute inference-time regime confidence under plain MSE while training reconstruction under Huber loss is a deliberate asymmetry. Huber's piecewise-linear tail prevents a single $\geq 3\sigma$ macro day from dominating the recon gradient during training; MSE's quadratic penalty at inference is what gives a far-OOD window (e.g., a COVID-style spike whose post-standardisation magnitude exceeds $4\sigma$) a sharply collapsed confidence signal. The downstream adaptive risk multipliers then revert to the neutral 0.9 prior rather than committing to a possibly-spurious regime label. The 6.7% tail of recon $> 2.0$ in section 11 is exactly the population of days for which this asymmetry is load-bearing.

**What is enforced and where.** Several pipeline-integrity invariants are encoded as runtime `ValueError`s rather than passive convention: macro symbols must be processed before stock symbols (otherwise feature joins miss columns silently), the GRASP universe must be exactly 120 symbols (graph topology is fixed), the train stride is hard-enforced at 1 even if the config requests otherwise (any value $> 1$ collapses Sharpe through the batch-flatten amortisation in eq. (2)), and the VQ-VAE input dimension is checked at load time. These are not stylistic preferences; each enforcement marks a specific past failure where the silent version was a multi-day debugging session.

## 13 Conclusion

On an eight-window, non-overlapping walk-forward spanning 2022–2026, the deployed purged $K$-fold ensemble delivers a positive test Sharpe in 8/8 windows, a mean test Sharpe of 1.16 (median 1.28), and a median Calmar of 2.99; chained across all eight test slices the OOS equity compounds to +100.2% against +61.0% for SPY and +83.6% for the composition-matched equal-weight book. The drawdown tail ($-14.5\%$ to $-16.4\%$ in the three systemic-selloff windows) is the binding weakness. On the most recent single OOS window (2026, 121 days) it returns +13.23% against SPY's +8.06% at a Sharpe of 2.30, a maximum drawdown of $-6.03\%$, and an annualised Jensen's $\alpha$ of +19.04%. The strategy is, by construction, a risk-adjusted-return strategy rather than a market-tracker: the loss function trades absolute upside for tail-loss control and the portfolio head trades concentration for diversification, both in ways the realised metrics confirm.

The contribution behind those numbers is structural: an end-to-end differentiable pipeline from macro state and per-stock indicators to constrained portfolio weights, with all hard constraints (long-only, per-stock cap, sector cap, gross exposure) enforced exactly through a convex QP layer rather than approximately through soft penalties. The classical forecast–rank–size decomposition does not appear anywhere, and there is no RL reward shaping; the model is trained on the realised portfolio return path under a composite financial objective. The most non-obvious empirical finding is the magnitude of the val–test Sharpe gap induced by missing temporal embargo: roughly 5× on representative windows, and large enough to make the difference between a model that looks publishable on val and one that loses money on test. That mechanism is not specific to this system.

Three lines of follow-up are open. First, the drawdown tail is the binding constraint on the walk-forward aggregate: three windows still draw down $-13\%$ to $-14\%$, and a volatility-managed gross overlay (a no-retrain cash sleeve that scales gross exposure down in high-vol regimes, section 9.5) is implemented as a shared-kernel parity contract and is the cheapest lever to cut that tail without surrendering the trending-window Sharpe; it is shipped disabled pending

crisis-window validation. Second, the moderate $\beta_{\text{SPY}} \approx 0.75$ defensive lean (table 15; not the $\approx 0.2$ a single calm window suggests) was previously induced by the unconditional $\text{CVaR}_{0.08}$ + MaxDD risk term; that term has since been refactored into the one-sided equal-weight-budgeted tail hinge of eq. (2), which removes the always-on $-\beta$ de-risking gradient that drove the lean, and an explicit information-ratio-against-equal-weight component (prototyped but not deployed at the time of writing) is the natural next experiment. Third, the fixed $N = 120$ graph and the daily-rebalance framing both constrain the model; expanding the universe with delisting handling, and extending the action space to a multi-day execution horizon, would test the architecture's generality.

## References


[1] J. Li and X. Fan, *Crisis-Resilient Portfolio Management via Graph-Based Spatio-Temporal Learning*, arXiv:2510.20868, 2025.

[2] C. Fifty, R. G. Junkins, D. Duan, A. Iger, J. Liu, E. Amid, S. Thrun, and C. Ré, *Restructuring Vector Quantisation with the Rotation Trick*, arXiv:2410.06424, 2024.

[3] *Block Attention Residuals for Deep Transformers*, arXiv:2603.15031, 2026 (preprint).

[4] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, *Differentiable Convex Optimisation Layers*, NeurIPS 2019.

[5] B. Amos and J. Z. Kolter, *OptNet: Differentiable Optimisation as a Layer in Neural Networks*, ICML 2017.

[6] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, *Sharpness-Aware Minimization for Efficiently Improving Generalization*, ICLR 2021. arXiv:2010.01412.

[7] R. Ilbert, A. Odonnat, V. Feofanov, A. Virmaux, G. Paolo, T. Palpanas, and I. Redko, *SAMformer: Unlocking the Potential of Transformers in Time Series Forecasting with Sharpness-Aware Minimization*, ICML 2024. arXiv:2402.10198.

[8] R. T. Rockafellar and S. Uryasev, *Optimisation of Conditional Value-at-Risk*, Journal of Risk, 2(3), 2000.

[9] A. W. Lo, *The Statistics of Sharpe Ratios*, Financial Analysts Journal, 58(4), 2002.

[10] A. Vaswani et al., *Attention Is All You Need*, NeurIPS 2017.

[11] P. Veličković et al., *Graph Attention Networks*, ICLR 2018.

[12] T. N. Kipf and M. Welling, *Semi-Supervised Classification with Graph Convolutional Networks*, ICLR 2017.

[13] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, *Neural Discrete Representation Learning*, NeurIPS 2017.

[14] J. Li and X. Fan, *Differentiable Portfolio Constraints via Convex QP Projection*, arXiv:2412.20679, 2024.

[15] A. Moreira and T. Muir, *Volatility-Managed Portfolios*, Journal of Finance, 72(4), 2017.

[16] O. Ledoit and M. Wolf, *Robust Performance Hypothesis Testing with the Sharpe Ratio*, Journal of Empirical Finance, 15(5), 2008.

[17] P. R. Hansen, *A Test for Superior Predictive Ability*, Journal of Business & Economic Statistics, 23(4), 2005.

[18] D. N. Politis and H. White, *Automatic Block-Length Selection for the Dependent Bootstrap*, Econometric Reviews, 23(1), 2004.

[19] A. Patton, D. N. Politis, and H. White, *Correction to "Automatic Block-Length Selection for the Dependent Bootstrap"*, Econometric Reviews, 28(4), 2009.

[20] D. W. K. Andrews, *Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation*, Econometrica, 59(3), 1991.