A Learnable Wavelet Transformer for Long-Short Equity Trading and Risk-Adjusted Return Optimization

Anonymous authors
Paper under double-blind review

Abstract

We propose WaveLSFormer, a learnable wavelet Transformer for intraday long-short trading. A trainable filter bank produces low/high-frequency components and a low-guided high-frequency injection module fuses them stably. The model outputs risk-budgeted positions and is trained with a trading-aware objective combining soft-label supervision with ROI and Sharpe oriented regularization. On five years of hourly U.S. equity data across six industries and ten seeds, WaveLSFormer improves both ROI and Sharpe over matched MLP/LSTM/Transformer baselines.

Track: Research Track

1 Introduction

Financial time series are noisy, non-stationary, and heavy-tailed, and profitable structure may only emerge at particular temporal scales. With large historical datasets, deep learning has become a standard tool for modeling nonlinear market dependencies Kohzadi et al. (1996); Ozbayoglu et al. (2020); Zhang et al. (2024a); Giantsidi & Tarantola (2025); Li et al. (2025). A persistent obstacle is *objective mismatch*: many pipelines optimize point-wise forecasting losses (e.g., MSE/MAE), whereas trading performance is determined by sequential position decisions and risk exposure, so predictive accuracy does not necessarily translate into robust out-of-sample returns Krauss et al. (2017). Moreover, assets within the same segment co-move and exhibit lead–lag effects, motivating group-level modeling rather than treating each series in isolation Ozbayoglu et al. (2020); Giantsidi & Tarantola (2025).

Wavelet transforms provide a natural multi-scale decomposition and have been widely used as fixed preprocessing for financial forecasting Peng et al. (2021); Rao et al. (2022); Zhang et al. (2024b); Ziółkowski (2025); Wu et al. (2025). In parallel, Transformers have become a dominant architecture for sequential modeling like natural language processing Li et al. (2026), and wavelet–Transformer hybrids combine multi-scale inputs with attention for improved forecasting Vaswani et al. (2017); Zhou et al. (2021); Sasal et al. (2022); Wei et al. (2025); Ping et al. (2025). For trading, two limitations remain: many hybrids rely on *fixed* wavelets and evaluate mainly under forecasting metrics, and cross-frequency fusion is often handled implicitly, allowing unstable high-frequency components to dominate learning under market noise.

To address these gaps, we propose *WaveLSFormer*, an end-to-end intraday trading policy that outputs continuous long/short positions for a group of related assets. WaveLSFormer couples a *learnable* wavelet front-end with a Transformer backbone: the filter bank is optimized jointly with a trading-aware objective to learn task-adaptive frequency bands, and we introduce **low-guided high-frequency injection (LGHI)** that computes attention from the low-frequency branch and injects high-frequency cues through a gated residual pathway for stable fusion. Predicted positions are rescaled to satisfy a fixed risk budget, aligning learning with practical portfolio construction.

1) We introduce *WaveLSFormer*, replacing fixed wavelet preprocessing with an end-to-end learnable wavelet filter bank coupled with a Transformer backbone for intraday long/short trading.

2) We propose stable cross-frequency fusion via LGHI with gated residual control, and integrate risk-budget rescaling to produce practical long/short positions.
3) On five years of hourly U.S. equity data (29.10.2020–29.10.2025) across six industries and ten random seeds, WaveLSFormer improves overall ROI from 0.225 ± 0.056 to 0.607 ± 0.045 and Sharpe from 1.024 ± 0.122 to 2.157 ± 0.166.

2 Problem formulation

At hour t, the policy observes a rolling window of multi-asset log returns $\mathbf{X}_t \in \mathbb{R}^{d\times L}$ and outputs a long/short position for a target asset $j^\star$. We compute a raw logit $p_t = f_\theta(\mathbf{X}_{t-1})$ and map it to $w_t = \tanh(p_t/2) \in [-1, 1]$. A fixed validation-calibrated scaling $\tilde{w}_t = h(w_t)$ enforces a constant risk budget (Appendix A.4.2). The gross return is $R_t = 1 + \tilde{w}_t(\exp(\ell_{j^\star,t}) - 1)$ and the ROI over horizon $\mathcal{T}$ is $\mathrm{ROI}_\mathcal{T} = \exp\big(\sum_{t\in\mathcal{T}} \log R_t\big) - 1$. We also report a Sharpe-style score on R_t (Appendix A.4).

3 Architecture

WaveLSFormer combines a learnable wavelet front-end with a standard Transformer. It decomposes prices into low-frequency trends and high-frequency residuals, fuses them, and feeds a Transformer encoder for position prediction. We therefore detail the front-end-filter parameterization, regularization, and cross-frequency integration that injects high-frequency cues into the low-frequency path.

3.1 Neural Wavelet Filters

As widely known, financial data contains a low info-noise ratio that causes the training process to always be misled by high-frequency noise gradient. To deal with this issue, we implement neural wavelet blocks as roles of low-pass and high-pass filters to re-balance the proportion of input info and noise. Low/high-pass filters are learnable 1D FIR convolution kernels. We regularize their spectra. We regularize the learned filter bank in the frequency domain to encourage complementary low/high responses and reduce band overlap. Implementation details are deferred to Appendix A.5.1.

3.1.1 Low-guided High-frequency Injection

To fuse low-frequency and high-frequency information while maintaining training stability, we introduce a low-guided high-frequency injection (LGHI) module. Let $L \in \mathbb{R}^{T\times d}$ denote the low-frequency representations and $H \in \mathbb{R}^{T\times d}$ the corresponding high-frequency representations. Different from standard cross-attention, where Q comes from one sequence and K, V come from the other, we compute the attention map *only* from the low-frequency branch and use it to inject high-frequency values. Concretely, we define

$$A(L) = \mathrm{softmax}\left(\frac{(LW_Q)(LW_K)^\top}{\sqrt{d_k}}\right),$$
$$Z(L, H) = A(L)\,(HW_V)W_O,$$

where $W_Q, W_K \in \mathbb{R}^{d\times d_k}$ and $W_V \in \mathbb{R}^{d\times d_v}$ are projection matrices and W_O is the output projection. The final fused representation is given by a gated residual injection:

$$Y = L + \beta\, Z(L, H), \qquad \beta = \sigma(\gamma), \tag{1}$$

where γ is a learnable scalar controlling the overall scale of the injected high-frequency residual. This design encourages the model to preserve the stable low-frequency backbone while selectively incorporating high-frequency cues, because the attention weights are determined by L and are therefore less sensitive to noisy high-frequency fluctuations.

4 Training Design

Training choices, such as loss, optimizer, and learning-rate schedule, strongly influence trading outcomes. Because convenient objectives may misalign with deployment metrics, low training loss can

still generalize poorly. We therefore present: (i) a differentiable surrogate objective, (ii) a stabilized optimization setup for heavy-tailed financial noise, and (iii) validation/checkpoint selection based on trading performance.

4.1 Loss Function Design

We train WaveLSFormer to output continuous long/short positions while aligning optimization with realized return and risk (details in Appendix A.7).

4.2 Trading-aware objective

We use a soft-label position loss to avoid overly sharp supervision when returns are near zero. Let $P_t = \sigma(p_t)$ and set the probabilistic target $y_t = \sigma(k\,\ell_{j^\star,t})$ with a fixed calibration constant $k > 0$. The per-step loss is

$$\mathcal{L}_{trade}(t) = -y_t \log P_t - (1 - y_t)\log(1 - P_t). \tag{2}$$

4.3 Return and risk regularization

To discourage unrealistic overfitting with extremely large training returns, we impose a soft cap on batch-level ROI. For a mini-batch $\mathcal{B}$ spanning $H_\mathcal{B}$ steps, define

$$L_\mathcal{B} = \sum_{t\in\mathcal{B}} \ell_{j^\star,t}, \qquad R_\mathcal{B} = \exp(L_\mathcal{B}) - 1. \tag{3}$$

With an annual cap $R_{\text{ann}}^{\max}$ and H_{year} steps per year, we set the batch threshold

$$T_\mathcal{B} = (1 + R_{\text{ann}}^{\max})^{H_\mathcal{B}/H_{\text{year}}} - 1, \tag{4}$$

and use a one-sided quadratic penalty

$$\mathcal{L}_{penalty} = \lambda_{\text{roi}}\big[\max(R_\mathcal{B} - T_\mathcal{B}, 0)\big]^2. \tag{5}$$

We further encourage risk-adjusted return with a differentiable Sharpe-style regularizer computed over the mini-batch,

$$\mathcal{L}_{sharpe} = \exp\left(-\alpha \cdot \min\left(\frac{3}{\sqrt{K}},\ \frac{\mathbb{E}[R_p]}{\sigma(R_p) + \varepsilon}\right)\right), \tag{6}$$

where K is the annualization factor in Equation 12, $\alpha > 0$ controls strength, and ε stabilizes training.

4.3.1 Neural Wavelet Loss

The neural wavelet loss is a weighted sum

$$\mathcal{L}_{wavelet} = \lambda_{\text{spec}}\big(\mathcal{L}_{low} + \mathcal{L}_{high}\big) + \mathcal{L}_{overlap} + \mathcal{L}_{parseval} + \mathcal{L}_{ratio},$$

and the overall training objective is

$$\mathcal{L}_{train} = \mathcal{L}_{trade} + \mathcal{L}_{penalty} + \mathcal{L}_{sharpe} + \mathcal{L}_{wavelet}. \tag{7}$$

see Appendix A.7.4 for formulae. In experiments, we select checkpoints based on the best ROI as shown in Appendix A.7.7

5 Experiment

5.1 Experiment Setup

5.1.1 Data Split, Leakage Prevention, and Model Selection

We use five years of hourly U.S. equity data (29.10.2020–29.10.2025) with a temporal 70%/10%/20% train/val/test split (5292/756/1512 hours). Hyperparameters are manually tuned due

Table 1: Overall average performance and model complexity of all backbones with and without the wavelet front-end. Numbers are averaged over 6 industry groups and 10 random seeds. Model complexity is reported as parameter count and FLOPs per forward pass. Best ROI and Sharpe are in **bold**, second best are underlined.

Model	Params (M)	FLOPs (G)	Avg. ROI	Avg. Sharpe
MLP	8.146	3.468	0.075±0.023	0.813±0.311
MLP + Wavelet	8.151	10.412	0.165±0.045	1.079±0.139
LSTM	12.538	492.614	0.191±0.029	1.656±0.495
LSTM + Wavelet	13.298	530.862	0.317±0.049	1.879±0.221
Transformer	15.928	665.921	0.225±0.056	1.024±0.122
WaveLSformer (ours)	15.943	659.742	**0.607±0.045**	**2.157±0.166**

to limited resources. To prevent leakage, all preprocessing is fit on the training period only, including ARR-based industry selection, DTW similarity filtering, and Granger-causality filtering with BH-FDR correction; all models share the resulting per-industry datasets. After forming industry-level asset groups, we compare WaveLSFormer with LSTM and MLP baselines. Unless noted, WaveLSFormer uses $d_{\text{model}} = 512$, $d_{\text{ff}} = 1024$, $n_{\text{heads}} = 128$, input length 96, 6 encoder layers and a 128-dim time2vec embedding. Transformer variants use ProbSparse attention with distillation for efficiency. We train each model for 80 epochs over 10 seeds (batch 256, lr 10^{-5}) with a two-phase wavelet schedule: 30 epochs to stabilize learnable filters under spectral regularization, then checkpoint selection by validation trading metrics. For each seed, we select the phase-2 checkpoint with the highest validation ROI and evaluate on the test set, reporting mean±std over 10 runs on the held-out test set.

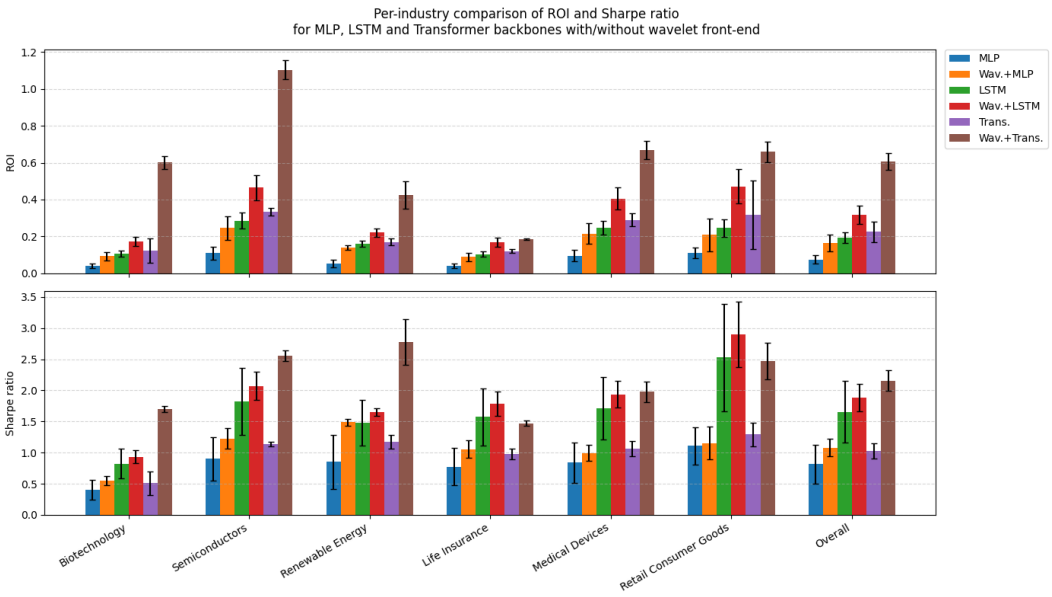


Figure 1: Per-industry ROI (a) and Sharpe ratio (b) for MLP, LSTM, and Transformer backbones with and without the neural wavelet front-end. Bars show mean over ten seeds and error bars denote one standard deviation, including six industries and the overall average.

5.2 Overall Results

Among the 14 candidate industries in Table 4, we compute the training-only annualized rate of return (ARR) of an intraday strategy and retain those with ARR $\geq 10\%$. The resulting six industries are fixed for all experiments to prevent leakage and to focus on potentially profitable, sufficiently liquid sectors; we then highlight Renewable Energy and Retail Consumer Goods (the lowest ARR) for detailed analysis. For each selected industry, constituent stocks are fed into the model via the pipeline

above; we select the checkpoint with the highest validation return and evaluate under three execution modes, "Long & Short, Long Only, and Short Only", with additional per-industry curves in the appendix (Figs. 10–15). We compare WaveLSFormer against parameter-/FLOP-matched MLP, LSTM, and Transformer backbones, each with and without the learnable wavelet front-end. Fig. 1 shows consistent gains across industries, and Table 1 reports mean ROI/Sharpe over six industries and ten seeds, where WaveLSFormer performs best without disproportionate scaling (Appendix A.10).

5.3 Ablation Study

The ablation study in Appendix A.9 shows that learnable filters and LGHI account for most gains.

6 Conclusion

In this paper, we proposed *WaveLSFormer*, a learnable wavelet-based Transformer for intraday long-short equity trading that combines a neural wavelet front-end with a Transformer backbone and outputs continuous long/short positions under a fixed risk budget. WaveLSFormer optimizes a trading-aware objective that couples soft-label supervision with ROI- and Sharpe-oriented regularization, targeting risk-adjusted returns rather than point-wise forecasting accuracy. Across five years of hourly U.S. equity data from 29.10.2020 to 29.10.2025 over six industry groups and ten random seeds, WaveLSFormer consistently outperforms matched MLP, LSTM, and Transformer baselines with and without wavelet front-ends. Overall, WaveLSFormer achieves 0.607 ± 0.045 ROI and 2.157 ± 0.166 Sharpe, while the plain Transformer attains 0.225 ± 0.056 ROI and 1.024 ± 0.122 Sharpe. Across backbones, the neural wavelet module improves profitability with modest computational overhead, and ablations confirm that learnable filters and gated cross-frequency injection drive the gains rather than backbone capacity alone.

REFERENCES

Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7):e0180944, 2017. doi: 10.1371/journal.pone.0180944.

Cees Diks and Valentyn Panchenko. A new statistic and practical guidelines for nonparametric granger causality testing. *Journal of Economic Dynamics and Control*, 30(9–10):1647–1669, 2006. doi: 10.1016/j.jedc.2005.08.008.

Sofia Giantsidi and Claudia Tarantola. Deep learning for financial forecasting: A review of recent trends. *International Review of Financial Analysis*, 2025. doi: 10.1016/j.iref.2025.104719. in press.

Md R. Kabir, Dipayan Bhadra, Moinul Ridoy, and Mariofanna Milanova. LSTM-transformer-based robust hybrid deep learning model for financial time series forecasting. *Sci*, 7(1):7, 2025. doi: 10.3390/sci7010007.

Nowrouz Kohzadi, Michael S. Boyd, Bahram Kermanshahi, and I. Kaastra. A comparison of artificial neural network and time series models for forecasting commodity prices. *Neurocomputing*, 10(2):169–181, 1996. doi: 10.1016/0925-2312(95)00020-8.

Christopher Krauss, Xuan Anh Do, and Nicolas Huck. Deep neural networks, gradient-boosted trees, random forests: Statistical arbitrage on the S&P 500. *European Journal of Operational Research*, 259(2):689–702, 2017. doi: 10.1016/j.ejor.2016.10.031.

Shuozhe Li, Zachery B. Schulwol, and Risto Miikkulainen. Transformer based time-series forecasting for stock, 2025. URL `https://arxiv.org/abs/2502.09625`.

Shuozhe Li, Vaishnav Tadiparthi, Kwonjoon Lee, Nakul Agarwal, Hossein Nourkhiz Mahjoub, Ehsan Moradi Pari, Lizhang Chen, Amy Zhang, and Liu Leqi. Learning robust reasoning through guided adversarial self-play. *arXiv preprint arXiv:2602.00173*, 2026.

Abdelmadjid Moghar and Mohamed Hamiche. Stock market prediction using LSTM recurrent neural network. *Procedia Computer Science*, 170:1168–1173, 2020. doi: 10.1016/j.procs.2020.03.049.

Leila Mozaffari and Jianhua Zhang. Predictive modeling of stock prices using transformer model. In *Proceedings of the 9th International Conference on Machine Learning Technologies (ICMLT 2024)*, pp. 1–8, Oslo, Norway, 2024. ACM. doi: 10.1145/3674029.3674037.

David M. Q. Nelson, Adriano C. M. Pereira, and Renato A. de Oliveira. Stock market's price movement prediction with LSTM neural networks. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1419–1426, 2017. doi: 10.1109/IJCNN.2017.7966019.

Ahmet Murat Ozbayoglu, Mehmet Ufuk Güdelek, and O.Özgür Sezer. Deep learning for financial applications: A survey. *Applied Soft Computing*, 93:106384, 2020. doi: 10.1016/j.asoc.2020.106384.

Jialong Peng, Yu Huang, Wei Jiang, and Lei Wang. Predicting stock movements: Using multiresolution wavelet reconstruction and deep learning in neural networks. *Information*, 12(10):388, 2021. doi: 10.3390/info12100388.

Yu Ping, Hoiio Kong, and Zijun Li. Wavelet-enhanced transformer for adaptive multi-period time series forecasting. *Preprint*, 2025. available on preprints.org.

Syamala Rao, Souvik Mitra, and Rakesh Kumar. Financial time series forecasting using optimised wavelet transform and neural network models. *International Journal of Information Technology*, 14(4):2231–2240, 2022. doi: 10.1007/s41870-022-00924-x.

Lena Sasal, Tanujit Chakraborty, and Abdenour Hadid. W-transformers: A wavelet-based transformer framework for univariate time series forecasting. In *2022 IEEE 21st International Conference on Machine Learning and Applications (ICMLA)*, pp. 671–676, 2022. doi: 10.1109/ICMLA55696.2022.00111.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. doi: 10.48550/arXiv.1706.03762.

Wei Wei, Zi'ang Wang, Bowen Pang, Jiannan Wang, and Xue Liu. Wavelet transformer: An effective method on multiple periodic decomposition for time series forecasting. *IEEE Transactions on Neural Networks and Learning Systems*, 36(8):14063–14077, 2025. doi: 10.1109/TNNLS.2025.3525502.

Jiangtao Wu, Jiaqi Li, Jie Yang, and Shuli Mei. Wavelet-integrated deep neural networks: A systematic review of applications and synergistic architectures. *Neurocomputing*, 657:131648, 2025. doi: 10.1016/j.neucom.2025.131648.

Chang Zhang, Zhi Xu, Yan Song, and Jing Wang. Deep learning models for price forecasting of financial time series: A review of recent advancements. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 14(1):e1519, 2024a. doi: 10.1002/widm.1519.

Junting Zhang, Haifei Liu, Wei Bai, and Xiaojing Li. A hybrid approach of wavelet transform, ARIMA and LSTM model for the share price index futures forecasting. *The North American Journal of Economics and Finance*, 69:102022, 2024b.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115, 2021. doi: 10.1609/aaai.v35i12.17325.

Krzysztof Ziółkowski. Enhancing financial time series forecasting: A comparative study of discrete wavelet transform and LSTM models for selected global indices. *Quality & Quantity*, 2025. doi: 10.1007/s11135-025-02325-1. online first.

A Appendix

A.1 Additional Details for Problem Formulation

A.2 Input Windows and Model Outputs

We consider the instruments in a given industry, including related ETFs and sector indices. We index instruments by $j \in \{1, \dots, d\}$ and hourly bars by $t \in \{1, \dots, T\}$. For instrument j, let $Open_{j,t}$ and $Close_{j,t}$ denote the open and close prices at time t. The simple return and log return of bar t are

$$r_{j,t} := \frac{Close_{j,t}}{Open_{j,t}} - 1, \qquad \ell_{j,t} := \log(1 + r_{j,t}). \tag{8}$$

In our experiments, we use the log price return $\ell_{j,t}$ defined in Eq. equation 8 and collect

$$\mathbf{r}_t = (r_{1,t}, \dots, r_{d,t})^\top, \qquad \boldsymbol{\ell}_t = (\ell_{1,t}, \dots, \ell_{d,t})^\top \in \mathbb{R}^d.$$

We adopt a rolling-window formulation: at decision time t the model observes

$$\mathbf{X}_t = \left[\boldsymbol{\ell}_{t-L+1}, \dots, \boldsymbol{\ell}_t\right] \in \mathbb{R}^{d \times L},$$

with columns ordered from oldest to most recent. Additional channels are absorbed into d.

Given $\mathbf{X}_{t-1}$, the model outputs a scalar trading signal

$$p_t = f_\theta(\mathbf{X}_{t-1}) \in \mathbb{R},$$

where f_θ is WaveLSFormer. We map p_t to a bounded position via

$$w_t = 2\sigma(p_t) - 1 = \tanh\left(\frac{p_t}{2}\right), \tag{9}$$

so $w_t > 0$ corresponds to long and $w_t < 0$ corresponds to short. $|w_t|$ controls position magnitude.

As discussed in Section A.4.2, using w_t directly can confound signal quality with aggressiveness. We therefore apply a global risk-budget scaling calibrated on the validation set and define the executable position as

$$\tilde{w}_t = h(w_t) \in [\tilde{w}_{\min}, \tilde{w}_{\max}],$$

where $h(\cdot)$ is a fixed piecewise-linear mapping that enforces a prescribed average leverage and clips extremes. In long-short experiments we set $\tilde{w}_{\min} = -L$ and $\tilde{w}_{\max} = L$, while in long-only or short-only settings one bound is set to zero. Section A.4.2 gives the exact form of $h(\cdot)$.

A.3 Return Representation

In each experiment we select one target asset $j^\star \in \{1, \ldots, d\}$ whose trading performance we evaluate, and treat the remaining instruments as auxiliary series providing contextual information. This reflects a common use case in industry-level trading, where a trader is primarily interested in a particular stock or ETF, but leverages signals from other stocks within the same sector. The single-period realized return of the trading strategy between t and $t+1$ is then

$$R_{t+1} \;=\; 1 + \tilde{w}_t \cdot r_{j^\star,t+1} \tag{10}$$

The return on investment (ROI) of the trading strategy over an evaluation period $\mathcal{T}$ is

$$ROI_{\mathcal{T}} \;=\; \exp\Big(\sum_{t\in\mathcal{T}} \log(R_t)\Big) - 1. \tag{11}$$

A.4 Optimization Target

From a trading perspective, the goal is not to minimize a point-wise prediction error, but to control the distribution of realized strategy returns. In our setting the single–period return of the strategy between t and $t+1$ is Eq. equation 10 and over an evaluation horizon $\mathcal{T}$, the corresponding ROI is Eq. equation 11.

Maximizing return alone is not sufficient in practice, since extremely volatile strategies with large draw-downs are usually unacceptable. A standard risk-adjusted performance measure in finance is the annualized Sharpe ratio, defined on simple returns R_t as

$$\mathcal{S}_{\text{ann}} = \sqrt{K}\,\frac{\mathbb{E}[R_t - r_f/K]}{\sqrt{\text{Var}(R_t - r_f/K)}}, \qquad K \approx 252 \times 6.5, \tag{12}$$

where r_f is the annual risk-free rate. In our hourly–bar backtests, R_t is already a small return per bar. For model comparison on a fixed dataset we (i) treat the risk–free rate as negligible, like $r_f \approx 0$. (ii) drop the constant $\sqrt{K}$, which only rescales all Sharpe values by the same factor. (iii) calculate the expectation and variation over $\mathcal{T}$. Throughout the paper we therefore report a raw Sharpe ratio of the form

$$\mathcal{S}_{\mathcal{T}} \;=\; \frac{\mathbb{E}[R_t]}{\sqrt{\text{Var}(R_t)}}, \tag{13}$$

and interpret it as a scale–free, risk–adjusted score rather than a directly tradable quantity.

Ideally, we would like the learned policy to achieve a good trade–off between cumulative return and Sharpe ratio. Conceptually, this can be formulated as a multi–objective optimization problem over the model parameters θ,

$$\max_{\theta} \Big(ROI_{\mathcal{T}}(\theta),\; \mathcal{S}_{\mathcal{T}}(\theta)\Big),$$

The exact loss components and their implementation details are given in Section 4.1.

A.4.1 Wavelet-based Signal Noise Decomposition

At short horizons, financial returns exhibit low signal-to-noise ratios. Following wavelet denoising, we decompose the target log return into a slowly varying component and a rapidly fluctuating residual,

$$\ell_{j^\star,t} = s_t + n_t,$$

where s_t captures low-frequency structure (e.g., trend) and n_t aggregates high-frequency, largely unpredictable variations. Over an interval $\mathcal{T}$, we define the effective SNR as

$$\mathrm{SNR}_{\mathcal{T}} \;=\; \frac{\mathrm{Var}_{t\in\mathcal{T}}[s_t]}{\mathrm{Var}_{t\in\mathcal{T}}[n_t]+\varepsilon}, \tag{14}$$

with a small $\varepsilon > 0$ for numerical stability.

In WaveLSFormer, a neural wavelet front-end performs a learnable time-frequency decomposition of the raw log-return sequence $\{\ell_{j^\star,t}\}$ into approximations of $\{s_t\}$ and $\{n_t\}$ using finite-impulse-response low-/high-pass filters. We learn filters that increase SNR relative to the raw series while producing features that benefit the downstream trading objective. To this end, we add frequency-domain regularizers that promote low-/high-frequency separation and encourage the filter pair to behave approximately as a tight frame; details are given in Section 3.

Given multivariate histories $\mathbf{X}_{t-1}$ and the next-step return $\ell_{j^\star,t}$, our goal is to learn a mapping f_θ that outputs trading signals p_t whose induced positions w_t maximize risk-adjusted strategy performance, aided by the wavelet front-end that improves the effective SNR of the input.

A.4.2 Position Rule Under a Fixed Risk Budget

Equation 9 bounds the trading signal, so a single trade never exceeds full long/short exposure. Yet different models can induce very different distributions of w_t: some outputs saturate near ± 1, while others concentrate around 0. Using w_t directly would therefore confound signal quality with aggressiveness, since strategies with larger average $|w_t|$ effectively take larger bets and thus realize larger gains and losses.

To normalize risk across models, we estimate a global scale on the validation set $\mathcal{D}_{\mathrm{val}}$ with length $T_{\mathrm{val}} = |\mathcal{D}_{\mathrm{val}}|$. We compute the mean absolute magnitude s_{val} and rescale the signal as $\hat{w}_t$

$$s_{\mathrm{val}} = \frac{1}{T_{\mathrm{val}}} \sum_{t\in\mathcal{D}_{\mathrm{val}}} |w_t|, \qquad \hat{w}_t = \frac{w_t}{s_{\mathrm{val}}}, \tag{15}$$

so the validation-period average of $|\hat{w}_t|$ is close to one.

To suppress tiny and noise-dominated trades, we apply a dead zone with threshold $\tau > 0$ and set $\tau = 0.01$ in all experiments. We then enforce an optional leverage cap and scale by a target average leverage $L > 0$ (we use $L = 1$), yielding the executable position in Equation 16:

$$\tilde{w}_t = \begin{cases} 0, & |\hat{w}_t| < \tau, \\ \max(-L, \min(L, \hat{w}_t)), & \text{otherwise.} \end{cases} \qquad \frac{1}{T_{\mathrm{val}}} \sum_{t\in\mathcal{D}_{\mathrm{val}}} |\tilde{w}_t| \approx L, \tag{16}$$

By constructing a fixed risk budget, up to sparsification from the dead zone, so models are evaluated under a comparable risk budget. The scale s_{val} is estimated once on $\mathcal{D}_{\mathrm{val}}$ and kept fixed for testing and online simulation, ensuring a consistent and implementable mapping from network outputs to positions at a pre-specified risk level.

A.5 Additional Details of Neural Wavelet Filters

A.5.1 FIR Convolution Kernels and Classical Discrete–time Filters

In our neural wavelet front-end, each learnable filter is implemented as a one-dimensional finite impulse response (FIR) convolution kernel. Let $x[n]$ denote a discrete-time input signal and let $\boldsymbol{\theta} = (\theta_0, \ldots, \theta_{L-1})$ be the convolution kernel of length L. A standard `conv1d` layer with stride 1 and no dilation computes

$$y[n] = \sum_{k=0}^{L-1} \theta_k \, x[n-k], \tag{17}$$

which is exactly the input and output relation of a linear time-invariant (LTI) discrete-time filter.

For an LTI system, the sequence $h[n]$ is defined as the response to a discrete-time unit impulse

$$\delta[n] = \begin{cases} 1, & n = 0, \\ 0, & n \neq 0 \end{cases}$$

Substituting $x[n] = \delta[n]$ into Equation 17 yields

$$y[n] = \sum_{k=0}^{L-1} \theta_k \, \delta[n-k] = \theta_n,$$

for $n = 0, \ldots, L-1$ and $y[n] = 0$ otherwise. Hence the impulse response of the filter is $h[n] = \theta_n$, so the FIR convolution kernel and the time-domain impulse response are the same object that the trainable parameters $\{\theta_k\}$ directly define the filter's behavior in the time domain.

The discrete-time frequency response of an LTI filter is defined as the complex gain applied to a complex exponential input $x[n] = e^{\mathrm{j}\omega n}$. Using Equation 17 we obtain

$$y[n] = \sum_{k=0}^{L-1} \theta_k \, e^{\mathrm{j}\omega(n-k)} = e^{\mathrm{j}\omega n} \sum_{k=0}^{L-1} \theta_k \, e^{-\mathrm{j}\omega k}.$$

Thus $e^{\mathrm{j}\omega n}$ is an eigenfunction of the system and the corresponding eigenvalue

$$H(e^{\mathrm{j}\omega}) = \sum_{k=0}^{L-1} \theta_k \, e^{-\mathrm{j}\omega k} = \sum_{k=0}^{L-1} h[k] \, e^{-\mathrm{j}\omega k} \tag{18}$$

is the frequency response. Equation 18 is exactly the discrete-time Fourier transform (DTFT) of the impulse response $h[n]$. Therefore, for an FIR filter, the frequency response is simply the Fourier transform of the convolution kernel. This provides a direct bridge between the learnable parameters in the neural network and the classical time–frequency interpretation of digital filters.

A.5.2 Differentiable Spectral Regularization via rFFT

Let $\boldsymbol{\theta} \in \mathbb{R}^L$ be a real-valued FIR kernel and $\hat{\boldsymbol{\theta}} = \mathcal{F}(\boldsymbol{\theta})$ its DFT. In matrix form, $\hat{\boldsymbol{\theta}} = \mathbf{F}\boldsymbol{\theta}$ with a fixed $\mathbf{F} \in \mathbb{C}^{L \times L}$. Hence $\mathcal{F}$ is linear and differentiable everywhere, with constant Jacobian $\mathbf{F}$. In practice we use rFFT, which is an optimized implementation of the same linear map specialized to real inputs (exploiting conjugate symmetry).

This enables differentiable frequency-domain regularization. A generic spectral regularizer is

$$\mathcal{L}_{spec}(\boldsymbol{\theta}) = \sum_k \rho\Big(|\hat{\theta}_k|\Big), \qquad \frac{\partial \mathcal{L}_{spec}}{\partial \boldsymbol{\theta}} = \mathbf{F}^{\mathsf{H}} \, \frac{\partial \mathcal{L}_{spec}}{\partial \hat{\boldsymbol{\theta}}}, \tag{19}$$

where $\rho(\cdot)$ is differentiable and $\mathbf{F}^{\mathsf{H}}$ the Hermitian transpose (i.e., inverse DFT up to scaling). Equivalently, back-propagation through rFFT amounts to applying an inverse FFT to the frequency-domain gradient. Therefore, adding Equation 19 on rFFT-transformed learnable kernels preserves differentiability and allows end-to-end training with standard optimizers.

To enforce complementary low/high-frequency coverage, we penalize high-frequency energy in the low-pass filter and low-frequency energy in the high-pass filter, add an overlap term to sharpen separation, and use Parseval and shape losses to constrain energy and frequency-response form; details are in Section 4.3.1.

A.6 Data Construction Pipeline

Our model is designed to directly optimize a trading objective rather than minimize a point-wise forecasting error. Therefore, we focus on economic performance, such as annualized return and Sharpe ratio, as the primary evaluation metrics, rather than MSE/MAE.

A.6.1 Data Collection

We construct an intraday U.S. equity dataset from two sources. We obtain price and volume data from the commercial API provider *polygon.io*, and download one-hour OHLCV bars for each stock over roughly the past ten years, providing multi-year coverage at intraday resolution.

We obtain sector and industry classifications from *StockAnalysis*[1], which publishes curated U.S. stock lists by sector and industry. We extract symbols for each industry and use this stock-industry mapping to define the cross-sectional universe and the industry-group experiments in Section 5.

[1] `https://stockanalysis.com/`

A.6.2 Data Preprocessing

We consider hourly bar-based intraday trading. Each stock is represented by OHLCV bars, and at the beginning of hour t the model observes past bars and selects the position to hold over hour $t+1$, making within-bar price change a natural primitive. Because price levels vary widely across securities, we use scale-free inputs by converting prices to intraday returns: Equation 8 computes the log percentage change of the close relative to the open, which stabilizes variance, partially symmetrizes gains and losses, and yields smoother dimensionless features for hourly modeling.

A.6.3 Stock Selection

We build industry-level stock groups as multi-asset inputs. For each industry with a liquid U.S.-listed ETF, we collect constituent stocks traded on major U.S. exchanges and treat the ETF as an economic proxy for the sector. We quantify similarity in recent price dynamics by computing Dynamic Time Warping (DTW) distances between candidates' historical log-change series over a fixed look-back window, where DTW is robust to local time shifts. Using the training period, we set the empirical median (50th percentile) of DTW distances as a threshold and retain stocks whose distance to the industry reference is below this value, filtering idiosyncratic outliers and preserving coherent trend shapes as illustrated in Fig. 2. We then apply non-parametric Granger causality tests to the DTW-selected set and remove stocks that show no predictive relation with the target asset.

A.6.4 Financial Securities with Granger-type Causality

We assume that same-industry stocks exhibit co-movements and lead-lag effects. We quantify such dependencies using nonparametric Granger causality Diks & Panchenko (2006), where x_t Granger-causes y_t if adding lags of x_t significantly improves predicting y_t beyond using lags of y_t alone.

For each industry universe, we run pairwise tests on the log-change series of candidate securities to obtain a directional p-value matrix. We then apply Benjamini-Hochberg FDR control and retain securities whose adjusted p-values with respect to the target asset are below 0.05 in at least one direction. The retained securities form the final multi-asset input set. Fig. 3 illustrates the selected stocks in Renewable Energy, and Table 2 reports the full adjusted p-value matrix, which reveals the directed dependence structure within each industry universe.

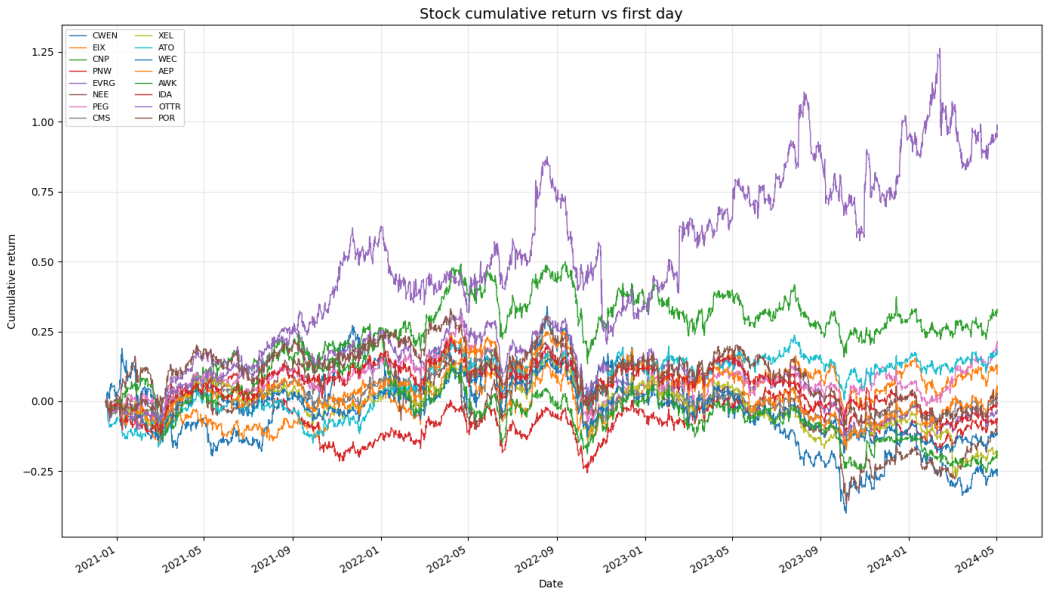


Figure 2: Stock codes selected under DWT.

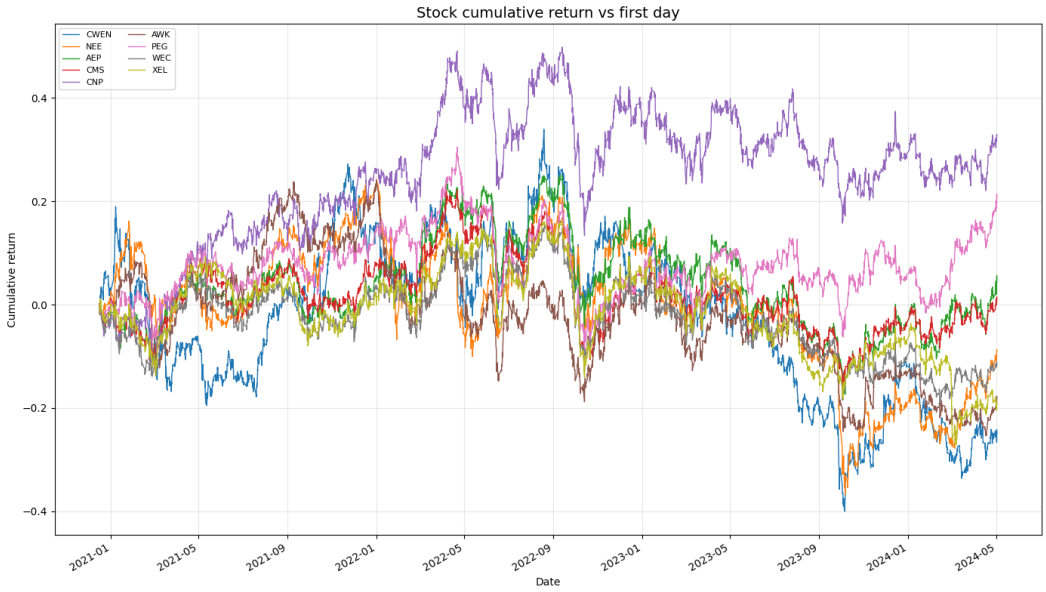


Figure 3: Stock selected after Granger causality filter in the renewable energy industry.

Table 2: BH-FDR-adjusted p-value matrix for pairwise tests on hourly log-percent-change series. Smaller values indicate stronger evidence of dependence. Bold entries denote significant pairs under FDR control ($p_{\text{adj}} < 0.05$); bold tickers indicate the retained assets.

	AEP	ATO	**AWK**	**CMS**	**CNP**	CWEN	EIX	EVRG	IDA	**NEE**	OTTR	**PEG**	PNW	POR	**WEC**	**XEL**
AEP	1	0.8107	0.7066	0.7824	1	0.3241	0.8397	0.9305	0.8041	0.8434	0.3984	0.4637	0.6562	0.9759	0.9506	0.8379
ATO	0.3984	1	0.3984	0.3241	0.3984	0.6317	1	1	1	0.8041	0.3812	1	0.7737	0.9759	0.3812	0.5568
AWK	1	0.7002	1	1	0.9759	0.0846	1	0.7393	0.8379	0.4262	0.456	0.8379	0.9305	0.8041	0.8379	0.6723
CMS	0.0521	1	0.3716	1	0.9506	0.3984	0.8379	1	0.8379	0.3835	0.8379	1	0.8813	0.6592	0.3812	0.3487
CNP	0.0505	0.3984	0.3241	0.4262	1	0.4262	0.9612	0.4838	0.7276	0.3984	0.3984	0.8107	0.3798	0.3984	0.3241	0.2143
CWEN	**0.0124**	0.0846	**0.0326**	**0.0221**	**0.0297**	1	0.0576	0.0648	0.1123	**0.0001**	0.7909	**0.0326**	0.3812	0.1703	**0.0326**	**0.0345**
EIX	0.3812	0.8107	0.3487	0.5568	0.296	0.9955	1	1	0.8107	0.3984	0.7025	0.8379	0.3487	0.4262	0.4262	0.4701
EVRG	0.0297	0.2143	0.0149	0.0558	0.1048	0.8107	0.52	1	0.3716	0.1719	0.3241	0.3467	0.2443	0.3812	0.0505	0.0912
IDA	0.4181	0.7737	0.4262	0.3812	0.1836	0.6062	0.8379	0.6235	1	0.3812	0.596	0.9759	0.3984	0.3342	0.3984	0.7409
NEE	0.7909	0.3449	0.3812	0.4262	0.6396	0.8107	0.8731	0.503	0.2482	1	0.6235	0.8409	0.5043	0.3812	0.5805	0.4397
OTTR	1	0.6062	0.9759	0.942	0.8379	0.9506	0.8434	0.296	0.6235	1	1	0.6396	0.6592	0.9759	0.7096	0.8107
PEG	0.0687	0.3812	0.0326	0.0846	0.4262	0.9421	0.5308	0.5006	0.604	0.0846	0.4106	1	0.3812	0.296	0.3984	0.296
PNW	0.8379	0.9759	0.7409	1	1	0.5985	0.7393	0.9759	1	0.3241	1	1	1	0.9759	0.9305	0.9633
POR	0.3984	0.8813	0.3161	0.3984	0.4713	0.9791	0.9759	0.8321	0.9759	0.2213	0.6988	0.8107	0.7224	1	0.3812	0.8321
WEC	0.531	1	0.4262	0.4262	1	0.3487	0.7909	0.8379	0.7409	0.4838	0.8379	0.8713	0.9421	0.9759	1	0.2443
XEL	1	0.9305	0.9573	0.9759	0.9123	0.3812	0.4397	0.8813	0.9305	0.8379	0.8321	0.3812	0.9791	0.9759	1	1

A.7 Additional Details of Loss Design

In stock trading, the ultimate objective is to maximize realized profit rather than to minimize point-wise prediction error. Therefore, the training loss should be a simple, differentiable surrogate that aligns with return optimization and provides stable gradients. In generic time-series forecasting, models are typically trained to predict the next-step target value, where MAE or MSE are standard choices. However, in financial settings, minimizing MAE/MSE on prices or returns is often weakly correlated with trading performance and thus is not well suited for learning profit-seeking decisions. The discussion about drawbacks of regression losses in trading optimization is in Appendix A.7.1.

A.7.1 Drawbacks of Regression Losses

Many financial time-series models train neural networks as return forecasters by minimizing point-wise regression losses such as MSE or MAE. Given a future log return $\hat{\ell}_t$, they learn $\ell_t = f_\theta(x_t)$ via

$$\mathcal{L}_{reg} = \mathbb{E}\big[loss(\ell_t, \hat{\ell}_t)\big], \qquad loss \in \{\text{MSE, MAE}\}. \tag{20}$$

While statistically convenient, $\mathcal{L}_{reg}$ is misaligned with trading: performance is driven by the position $\tilde{w}_t$ and the resulting P&L $R_t = \tilde{w}_t \, \ell_t$, not by the numeric forecast error of ℓ_t. Consequently, lower regression error does not necessarily translate to higher ROI/Sharpe. The mismatch is exacerbated by the asymmetric economic cost as a wrong sign under high exposure is far more damaging than a small magnitude error on a correct trade, whereas MSE/MAE penalize deviations symmetrically.

Moreover, under noisy and non-stationary returns, minimizing $\mathcal{L}_{reg}$ mainly encourages approximating $\mathbb{E}[\ell_t \mid x_t]$, which offers no guarantee of improved R_t or risk-adjusted performance. Therefore, we do not treat the network as a pure forecaster. Instead, we interpret its output as a continuous trading position $\tilde{w}_t \in [-1, 1]$ (flat if $\tilde{w}_t \in [-0.01, 0.01]$) and optimize a trading-aligned objective,

$$\mathcal{L}_{trade} = -\,\mathbb{E}\big[\tilde{w}_t y_t\big] + \lambda \, \Omega(\tilde{w}_t),$$

where $\Omega(\cdot)$ regularizes leverage, turnover, or excessive position variability.

A.7.2 Stock Direction and Tanh Position Loss

The *Stock Direction* loss treats the sign of the model output as a long/short signal: go long if $p_t > 0$, short if $p_t < 0$, and optionally trade only when $|p_t| > \tau$ to filter low-confidence positions. The resulting position in Equation 11 is defined in Equation 9. Following Equation 8, let r_t and ℓ_t denote the simple and log returns.

To turn multiplicative ROI into additive form, we use the small-return approximation. For $|r_t| < 0.1$,

$$\log(1 + r_t) = r_t - \frac{r_t^2}{2} + O(r_t^3),$$

so $\ell_t \approx r_t$ with leading error $r_t^2/2 \leq 0.005$ when $|r_t| \leq 0.1$. Hence we approximate

$$\mathrm{ROI} \approx \exp\Big(\sum_t \mathrm{sign}(p_t) \, \ell_t\Big) - 1,$$

and define the corresponding trade loss as

$$\mathcal{L}_{trade} = -\exp\Big(\sum_t \mathrm{sign}(p_t) \, \ell_t\Big).$$

The *Tanh position* loss replaces the all-in direction decision by a continuous position size on the target asset using $\tanh(p_t) \in [-1, 1]$ as

$$\mathcal{L}_{trade} = -\exp\Big(\sum_t \tanh(p_t) \, \ell_t\Big). \tag{21}$$

A.7.3 Sigmoid Position Loss

Direct position optimization with the Stock Tanh loss often suffers from gradient saturation: as outputs approach the bounds, $\tanh'(\cdot)$ becomes small and updates vanish. We also avoid $\log(\tanh(\cdot))$ objectives, whose logit gradient

$$\frac{\mathrm{d}}{\mathrm{d}p_t} \log(\tanh(p_t)) = \frac{1 - \tanh^2(p_t)}{\tanh(p_t)},$$

is ill-conditioned near the origin and still decays as $|p_t|$ grows. Instead, we treat p_t as the logit of a Bernoulli variable indicating willingness to go long and perform the signed position mapping only at inference.

Ignoring transaction costs and position constraints, the ROI-optimal policy is fully long for $r_t \geq 0$ and fully short otherwise, so we cast trading as a binary classification problem:

$$y_t = \begin{cases} 1, & r_t \geq 0, \\ 0, & r_t < 0. \end{cases}$$

We train in probability space with a weighted logistic loss, so gradients depend on the confidence gap between $\sigma(p_t)$ and y_t and remain informative; at inference, we map $\sigma(p_t)$ to a smooth exposure in $[-1, 1]$ via Equation 9.

To emphasize economically significant moves, we weight each sample by the absolute log return $|\ell_t|$ (e.g., $+100\%$ and -50% have equal $|\ell_t|$). The resulting Stock Sigmoid loss is

$$\mathcal{L}_{trade}(t) = \begin{cases} -\log\big(\sigma(p_t)\big)\,|\ell_t|, & r_t \geq 0, \\ -\log\big(1-\sigma(p_t)\big)\,|\ell_t|, & r_t < 0, \end{cases} \tag{22}$$

which mitigates saturation in Stock Tanh while aligning optimization with the sign and magnitude of future returns.

A.7.4 Neural Wavelet loss Formulae

Our low-/high-pass filters are implemented as 1D FIR convolutions. As discussed in Section A.5.1, the convolution weights W correspond to the impulse responses, whose discrete Fourier transform gives the frequency responses. Therefore, the rFFT of the filter parameters directly yields the spectrum on the grid

$$\frac{k}{n_{\text{fft}}}, \qquad k \in \Big[0, 1, \ldots, \frac{n_{\text{fft}}}{2} + 1\Big].$$

With hourly data, we set $n_{\text{fft}} = 81$ to cover month-level frequencies. We regularize the learnable filter bank in the frequency domain to encourage clear low-/high-pass separation and avoid degenerate solutions. Let $|G_{\text{low}}|^2 := \sum_\omega |G_{\text{low}}(\omega)|^2$ and $|G_{\text{high}}|^2 := \sum_\omega |G_{\text{high}}(\omega)|^2$ be the total spectral energies on a discrete grid $\omega \in [0, \pi]$. We penalize out-of-band energy using a frequency-weighted power term with $p = 2$ in all experiments:

$$\mathcal{L}_{low} = \sum_\omega (\omega/\pi)^p\,|G_{\text{low}}(\omega)|^2, \qquad \mathcal{L}_{high} = \sum_\omega (1-\omega/\pi)^p\,|G_{\text{high}}(\omega)|^2,$$

$$\mathcal{L}_{overlap} = |G_{\text{low}}|^2 \cdot |G_{\text{high}}|^2, \qquad \mathcal{L}_{parseval} = \big(|G_{\text{low}}|^2 + |G_{\text{high}}|^2 - 2\big)^2.$$

To balance the two branches, we further impose an energy-ratio hinge loss. Define ρ as following and penalize ρ outside $[\rho_{\min}, \rho_{\max}]$

$$\rho = \frac{|G_{\text{high}}|^2}{|G_{\text{low}}|^2 + \varepsilon}, \qquad \mathcal{L}_{ratio} = \max\big(\rho - \rho_{\max},\, 0\big) + \max\big(\rho_{\min} - \rho,\, 0\big). \tag{23}$$

A.7.5 Regression Loss vs. Soft-label Loss

Motivated by the above discussion, we replace regression with a soft-label classification objective that focuses on return direction and economically relevant magnitude, while treating the output as a proxy for position.

With regression, the model minimizes Equation 20 to predict the future log return. Trading then relies on a separate hand-designed mapping $h : \mathbb{R} \to [\tilde{w}_{\min}, \tilde{w}_{\max}]$, such as the clamping rule in Equation 16, and sets $\tilde{w}_t = h(\ell_t)$. As a result, L_{reg} optimizes only return accuracy, whereas ROI/Sharpe depends on the untrained composite decision rule and can differ substantially even when L_{reg} is similar.

To remove this layer, we form a soft target using a scaled sigmoid,

$$y_t = \sigma(k\,\hat{\ell}_t),$$

where $k > 0$ controls saturation. The network outputs a logit p_t with $P_t = \sigma(p_t)$ and is trained with soft-label cross-entropy. At inference, we convert p_t to an executable position using the same squashing and risk-budget mapping described in Section 2, so the optimized scalar becomes the trading signal up to a fixed monotone transform.

Soft labels offer three advantages: they penalize sign errors on large moves more than small deviations, learn a decision signal rather than a return forecast, and integrate naturally with the Sharpe-oriented regularizer in Section 4.1.

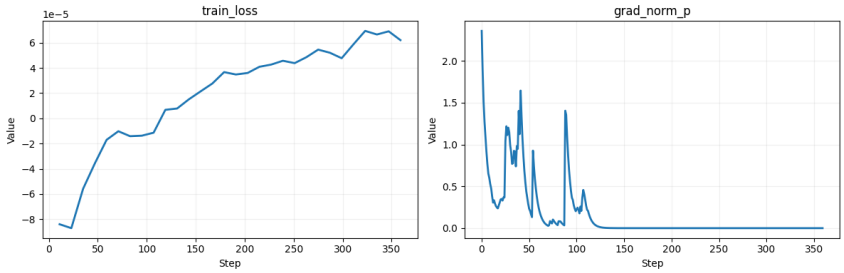
Table 3 shows Renewable Energy results with identical architecture and optimization. We disable the Sharpe regularizer and only change the supervised loss (MSE, MAE, or soft-label). The soft-label objective yields higher ROI and Sharpe than regression despite comparable error on ℓ_t, supporting soft-label training for trading signals.

Table 3: Comparison of regression losses and soft-label loss on Renewable Energy with 10 seeds. All models share the same WaveLSFormer architecture and optimization setup, the Sharpe regularizer is disabled for all rows to isolate the effect of the supervised loss.

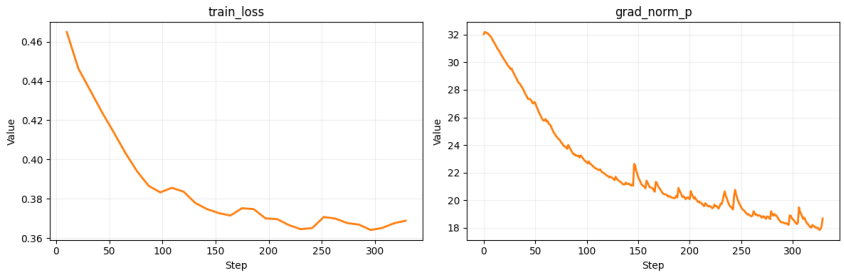
Loss	ROI (test)	Sharpe (test)
MSE	0.078 ± 0.039	0.586 ± 0.275
MAE	0.125 ± 0.061	0.899 ± 0.355
Soft-label	0.377 ± 0.045	1.943 ± 0.311

A.7.6 Tanh Loss vs. Soft-label Loss

We observed that optimizing Transformers for U.S. equity trading is highly sensitive to the loss design. When using the tanh-based objective in Equation 21, training often stagnates because the tanh nonlinearity saturates, producing near-zero gradients and effectively "deadlocking" the optimizer. We diagnosed this behavior by monitoring the ℓ_2 norm of parameter gradients, which frequently collapsed toward zero, and found that an overly small initial learning rate further increases the likelihood of getting trapped in poor local optima. However, the soft-label loss overcomes this drawback and produces a stable training process.



(a) Tanh loss function leads to gradient vanishing.



(b) Soft-label loss function has stable and continuous gradients.

Figure 4: Loss curve and gradient of tanh and soft-label loss functions

A.7.7 Validation Metric and Model Selection

We train the network by minimizing the loss in Equation 7, but our goal is to maximize out-of-sample trading performance rather than validation cross-entropy. Using the probability-to-position rule in Equation 16, the cumulative strategy return on the validation set is

$$R_{\text{val}} = \sum_{t \in \mathcal{D}_{\text{val}}} \tilde{w}_t r_t,$$

or, in the risk-adjusted setting, a Sharpe-like ratio computed from the sequence $\{\tilde{w}_t r_t\}$.

Crucially, L_{val} and R_{val} are not necessarily monotonic. The threshold-based decision rule in Equation 16 can flip actions near boundaries with minimal change in L_{val} yet a large change in R_{val}. Moreover, cross-entropy weights samples roughly uniformly, whereas R_{val} is dominated by a small

Table 4: Industry-level annualized rate of return (ARR) used for sector selection. Industries with ARR $\geq 10\%$ are retained for subsequent experiments.

Industry	ARR (%)	Selected
Biotechnology	33.98505	Yes
Regional Banks	5.706405	No
Engineering Construction	8.208481	No
Electronic Components	8.899873	No
Information Technology Services	8.718001	No
Medical Devices	12.48685	Yes
Semiconductors	14.11355	Yes
Software Application	7.635932	No
Specialty Industrial Machinery	4.412272	No
Utilities Electric	4.132072	No
Real Estate REITs	2.451229	No
Renewable Energy	10.14711	Yes
Life Insurance	12.77445	Yes
Retail Consumer Goods	11.83257	Yes

number of large moves: improving calibration on many small-return samples may reduce L_{val} with little impact on R_{val}, while better capturing large profitable moves can slightly worsen L_{val} but substantially improve R_{val}.

Therefore, we use soft-label cross-entropy as a differentiable surrogate for optimization, but perform early stopping and hyperparameter selection solely based on R_{val}. In practice, we select the checkpoint with the highest validation ROI, using L_{val} only as a diagnostic for optimization stability. We use a soft-label objective for position learning, as it consistently yields higher ROI/Sharpe than regression losses under identical architectures Appendix A.7.5.

A.8 Hyperparameter Sensitivity Analysis

A.8.1 λ_{roi} in Overfitting Penalty

The hyperparameter λ_{ROI} controls the strength of the ROI-aware penalty and thus how aggressively the model is encouraged to maximize profit during training: small values make the constraint nearly inactive, whereas large values strongly discourage ROI-constraint violations.

To analyze its effect, we sweep

$$\lambda_{\text{roi}} \in \{0.0, 0.1, 0.3, 0.5, 0.7, 1.0, 1.2\},$$

train the model under the same configuration for each setting, and record the final training ROI over the full 4-year horizon together with the corresponding val ROI and Sharpe ratio. Results are shown in Fig. 5.

We observe a clear dependence on λ_{roi}. Too small a value yields weak regularization and overly aggressive policies that can produce inflated training ROI, while too large a value makes training overly conservative and substantially reduces ROI. In a mid-range of λ_{roi}, training ROI stabilizes around 6 over the 4-year horizon and val ROI/Sharpe achieve their best values, which we adopt as a stable operating regime.

A.8.2 γ in Low-guided High-frequency Injection

This subsection analyzes the scalar gate that controls the strength of LGHI. Empirically, overly large gate values cause vanishing gradients for LSTM backbones and exploding gradients for Transformer backbones, motivating a small-gate initialization.

Let $L_t \in \mathbb{R}^d$ be the low-frequency representation at time t and H_t the corresponding high-frequency input. LGHI produces a refinement $Z_t(L_t, H_t) \in \mathbb{R}^d$, and the fused output follows Equation 1. Here, γ is a learnable scalar and $\beta = \sigma(\gamma) \in (0, 1)$ controls the overall contribution of LGHI: $\beta \approx 0$ yields an almost purely low-frequency representation, while $\beta \approx 1$ heavily relies on the high-frequency refinement.

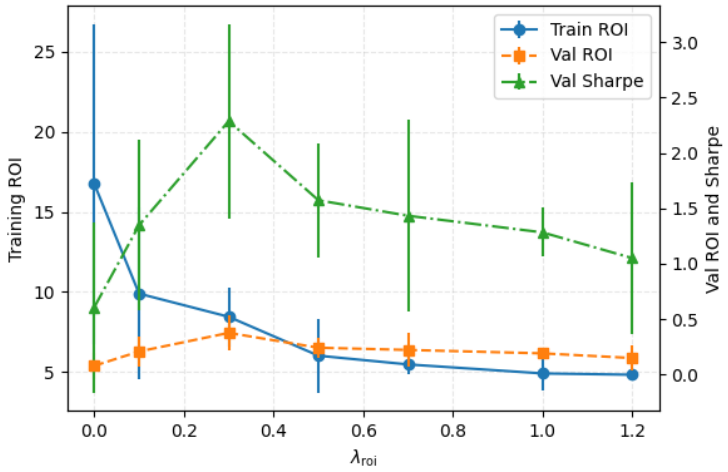


Figure 5: Sensitivity of cumulative ROI to the ROI-aware penalty coefficient λ_{roi} on the Renewable Energy industry. The left y-axis shows training cumulative ROI over the training horizon, and the right y-axis shows validation ROI and Sharpe. All values are reported as mean$\pm$std over ten random seeds.

The gate scales not only the forward contribution but also the backward signal to LGHI parameters θ_Z. By the chain rule,

$$\frac{\partial \mathcal{L}}{\partial \theta_Z} = \frac{\partial \mathcal{L}}{\partial Y_t} \frac{\partial Y_t}{\partial Z_t} \frac{\partial Z_t}{\partial \theta_Z} = \beta \frac{\partial \mathcal{L}}{\partial Y_t} \frac{\partial Z_t}{\partial \theta_Z},$$

where $\mathcal{L}$ denotes the training objective. Thus, larger β linearly amplifies gradients w.r.t. θ_Z, while also increasing the scale of Y_t fed into the backbone.

Effect with LSTM backbones With an LSTM backbone, Y_t enters recurrent transitions whose gates use sigmoid and $\tanh$. As β increases, the magnitude of Y_t tends to grow, pushing gate pre-activations into saturation where $\sigma'(x)$ and $1 - \tanh^2(x)$ become small. Consequently, the recurrent-step Jacobian tends to have a spectral radius below 1, and the backpropagated gradients decay through time:

$$\frac{\partial \mathcal{L}}{\partial (h_t, c_t)} = \left(\prod_{k=t}^{T-1} J_k \right) \frac{\partial \mathcal{L}}{\partial (h_T, c_T)},$$

leading to vanishing gradients and early training plateaus for moderately large β.

Effect with Transformer backbones In contrast, a Transformer backbone forms a deep stack of residual blocks with Layer Normalization:

$$X^{(\ell+1)} = \mathrm{LN}\big(X^{(\ell)} + F^{(\ell)}(X^{(\ell)})\big).$$

A first-order approximation yields a layer Jacobian of the form

$$J^{(\ell)} \approx I + \beta\, \tilde{J}^{(\ell)}.$$

Thus, increasing β raises the effective layer-wise gain, and the backward signal can grow through the product of Jacobians:

$$\frac{\partial \mathcal{L}}{\partial X^{(0)}} = \left(\prod_{\ell=0}^{L-1} J^{(\ell)} \right) \frac{\partial \mathcal{L}}{\partial X^{(L)}},$$

which may cause exploding gradients when the spectral radius exceeds 1 across many layers. In our experiments, setting $\beta \in \{0.01, 0.1, 0.5, 1.0\}$ shows stable training for small/moderate gates, while $\beta = 0.5$ and $\beta = 1.0$ quickly become numerically unstable as shown in Fig. 6.

Initialization strategy Across backbones, overly large gates can stall optimization by saturating LSTM gates and causing vanishing gradients, or by over-amplifying deep residual stacks in Transformers and triggering exploding gradients. Therefore, we initialize β to a very small value so

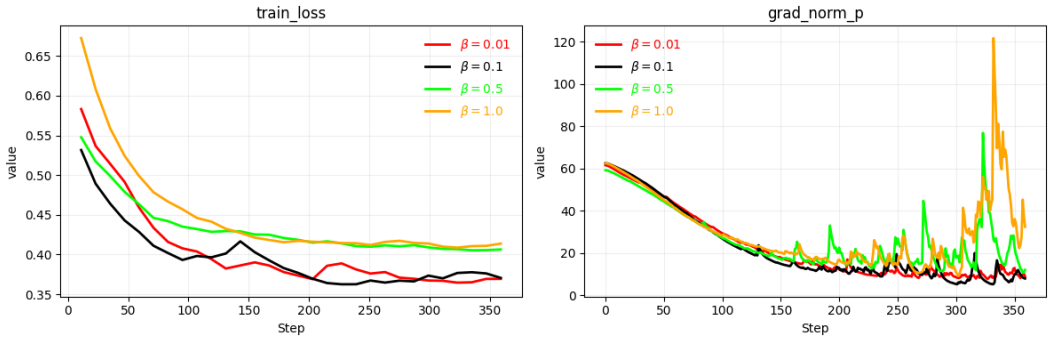


Figure 6: Effect of the LGHI module β on training loss and gradient norms for Transformer backbones. Curves correspond to $\beta = 0.01$ (red), $\beta = 0.1$ (black), $\beta = 0.5$ (green), and $\beta = 1.0$ (orange).

that LGHI starts as a weak perturbation of the low-frequency path. Specifically, we parameterize $\beta = \sigma(\gamma)$ and initialize $\gamma = -5$, yielding $\beta_0 = \sigma(-5) \approx 0.0067$. This keeps the mapping close to the identity at initialization and avoids both LSTM saturation and excessive amplification in deep Transformer stacks. Since γ is learnable, the optimizer can increase β when the refinement $Z_t(L_t, H_t)$ is beneficial; otherwise, β remains small and the model behaves close to a purely low-frequency baseline.

A.8.3 λ_{spec} IN NEURAL WAVELET LOSS

The hyperparameter λ_{spec} weights the spectral-shaping term in the wavelet loss. Smaller values relax frequency-domain constraints and allow more task-driven filters, while larger values enforce cleaner low-/high-pass behavior but can reduce flexibility.

We sweep

$$\lambda_{\text{spec}} \in \{0,\ 0.3,\ 1,\ 3,\ 10,\ 30,\ 100\},$$

keeping all other settings fixed, and evaluate validation ROI and Sharpe. Fig. 7 summarizes the results on Renewable Energy. We select λ_{spec} using validation performance to avoid test-set leakage, and report test results with the selected value in the main results section.

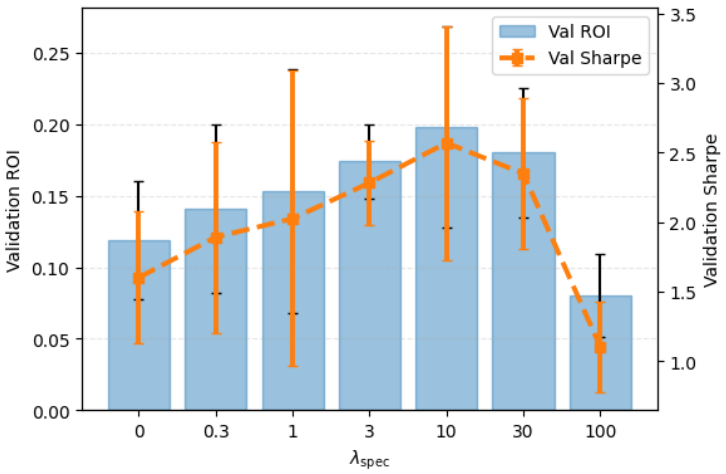


Figure 7: Validation-set sensitivity of λ_{spec} in the wavelet loss on the Renewable Energy industry. Bars report validation ROI on the left y-axis, and the dashed line reports validation Sharpe ratio on the right y-axis. All values are reported as mean$\pm$std over ten random seeds.

Overall, performance peaks in a moderate range. Too small λ_{spec} weakens spectral shaping and can yield less stable filters, while too large λ_{spec} over-regularizes optimization and degrades ROI and Sharpe. We use $\lambda_{\text{spec}} = 10$ as the default unless otherwise stated.

A.9 Additional Details of Ablation Study

A.9.1 Effect of Using Only a Single Frequency Branch

We evaluate whether both frequency branches are necessary by comparing the full model (Low+High) with two ablations: *Low-Freq only* and *High-Freq only*. Table 5 shows that removing either branch consistently hurts performance. Across most industries, *Low-Freq only* outperforms *High-Freq only* in ROI and Sharpe, indicating that slowly varying trends are more informative than high-frequency residuals alone. However, Low+High achieves the best average ROI/Sharpe with lower seed variance, suggesting that the high-frequency branch provides complementary short-term refinements, whereas relying solely on high-frequency signals reduces mean performance and increases training instability.

Table 5: Effect of two frequency branches. Results are mean ± std over ten seeds. Best in each row is red.

Industry	Metric	Full Low+High	Only Low-Freq	Only High-Freq
Biotechnology	ROI	0.601 ± 0.034	0.305 ± 0.028	0.133 ± 0.050
	Sharpe	1.695 ± 0.050	0.902 ± 0.117	0.429 ± 0.185
Semiconductors	ROI	1.104 ± 0.053	0.561 ± 0.052	0.244 ± 0.092
	Sharpe	2.555 ± 0.081	1.360 ± 0.176	0.646 ± 0.279
Renewable Energy	ROI	0.423 ± 0.074	0.225 ± 0.021	0.098 ± 0.037
	Sharpe	2.775 ± 0.365	1.477 ± 0.191	0.702 ± 0.303
Life Insurance	ROI	0.185 ± 0.004	0.094 ± 0.009	0.041 ± 0.015
	Sharpe	1.472 ± 0.039	0.783 ± 0.101	0.372 ± 0.161
Medical Devices	ROI	0.669 ± 0.049	0.340 ± 0.032	0.148 ± 0.056
	Sharpe	1.979 ± 0.167	1.053 ± 0.136	0.501 ± 0.216
Retail Consumer Goods	ROI	0.659 ± 0.055	0.334 ± 0.165	0.145 ± 0.286
	Sharpe	2.465 ± 0.293	1.312 ± 0.161	0.624 ± 0.329
Overall (avg.)	ROI	0.607 ± 0.045	0.310 ± 0.051	0.135 ± 0.089
	Sharpe	2.157 ± 0.166	1.148 ± 0.147	0.546 ± 0.246

A.9.2 Sharpe Regularizer Ablation

To quantify the benefit of explicitly optimizing a trading-oriented objective, we ablate the Sharpe loss term by training two WaveLSFormer variants with identical architectures, optimization settings, and data splits: (i) Soft-label only and (ii) Soft-label with a Sharpe loss computed from per-step P&L. For each industry and seed, both variants share the same training schedule and differ only in whether the Sharpe term is included. Table 8 reports industry-level test metrics averaged over 10 seeds with checkpoints selected by validation ROI. Adding the Sharpe loss improves risk-adjusted performance and often increases ROI while reducing maximum draw-down, suggesting that prediction-only training is insufficient and that directly regularizing the mean-volatility trade-off yields more favorable risk-return profiles.

Beyond ROI and Sharpe ratio, we report maximum draw-down (MDD) on the test set as a complementary risk measure. Let W_t denote the cumulative wealth process. The draw-down at time t and the maximum draw-down over the test period are

$$D_t = 1 - \frac{W_t}{\max_{s \le t} W_s}, \qquad \text{MDD} = \max_t D_t.$$

A.9.3 Effect of Learnable Wavelet Filters

We evaluate the proposed neural wavelet layer by replacing fixed filters in the same Transformer backbone. We compare three front-ends: (i) *Neural Wavelet*, which learns the filters jointly with the

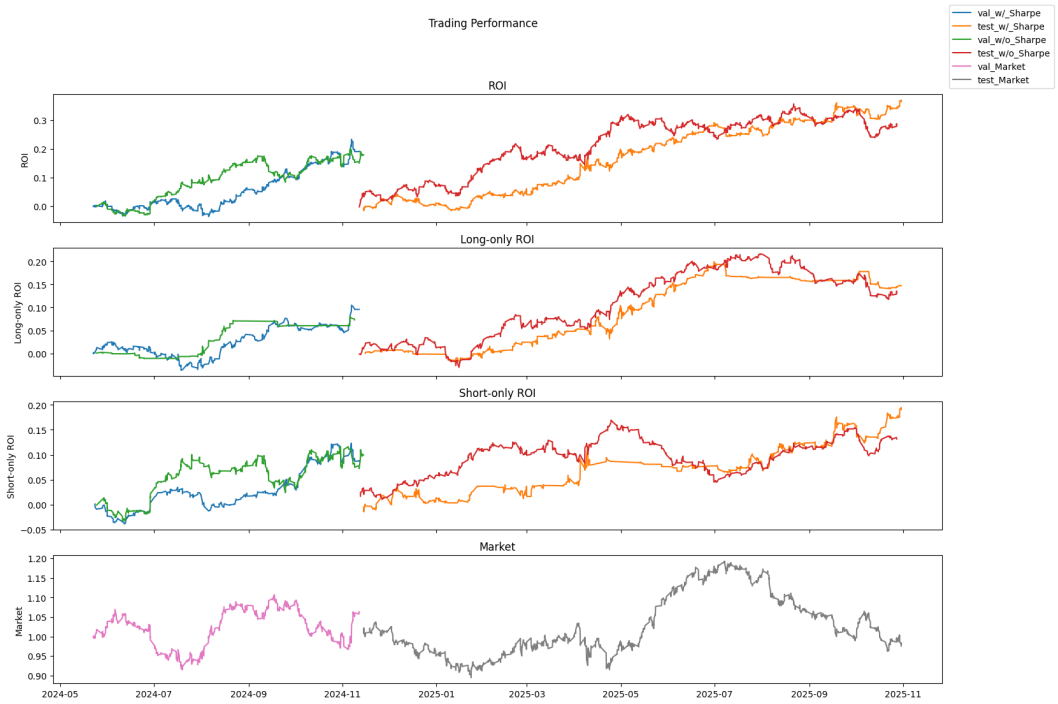


Figure 8: Equity curves with and without the Sharpe loss for Renewable Energy. Top: cumulative ROI on validation and test. Bottom: cumulative market return. The model without Sharpe loss (red) shows deeper and more frequent test draw-downs, whereas the Sharpe-aware model (orange) produces a smoother equity curve with smaller draw-downs at a similar final ROI, reflecting more conservative exposure in risky periods.

trading objective using Parseval, overlap, and pass-band regularizers; (ii) *Classic Wavelet*, a fixed wavelet transform; and (iii) *Transformer only*, which uses raw price features. As shown in Table 6, Neural Wavelet attains the best test ROI and Sharpe in almost all industries under matched capacity, and also yields smaller maximum draw-down and more stable performance across seeds, indicating that learnable, regularized wavelet filters extract more tradable signals than fixed transforms or no decomposition.

A.9.4 Low-guided High-frequency Injection

We further examine how the low and high frequency branches should be fused. In the proposed design, the low-frequency representation attends to the high-frequency branch via a LGHI block. As a comparison, we remove the LGHI module and simply concatenate the two branches followed by a linear projection, while keeping the overall model size comparable. The multi-industry, multi-seed results are reported in Table 7.

Across all industry groups, the LGHI design consistently outperforms simple concatenation in terms of both ROI and Sharpe ratio, and also exhibits smaller variance across seeds on average. This confirms that allowing the low-frequency branch to selectively attend to high-frequency details is more effective than treating the two branches as independent features.

A.10 Additional Details for Baseline Comparisons

A.10.1 Per-Industry Performance

We compare MLP, LSTM, and Transformer backbones with and without the neural wavelet front-end, using $d_{\text{model}} = 512$ for all models. WaveLSFormer uses $d_{\text{ff}} = 1024$, $n_{\text{heads}} = 128$, $L = 6$ encoder layers, sequence length 96, temporal embedding dimension 128, and pre-layer normalization. The MLP baseline has 10 fully connected layers with 512 hidden units, and the LSTM baseline

Table 6: Multi-seed, multi-industry comparison of Transformer-based backbones with different front-ends. Results are reported as mean ± std over ten seeds. Best results in each row are highlighted in red.

Industry	Metric	Neural wavelet Transformer	Classic wavelet Transformer	Plain Transformer
Biotechnology	ROI	0.601 ± 0.034	0.323 ± 0.021	0.124 ± 0.066
	Sharpe	1.695 ± 0.050	1.086 ± 0.048	0.511 ± 0.189
Semiconductors	ROI	1.104 ± 0.053	0.416 ± 0.021	0.333 ± 0.021
	Sharpe	2.555 ± 0.081	1.293 ± 0.071	1.134 ± 0.035
Renewable Energy	ROI	0.423 ± 0.074	0.261 ± 0.033	0.169 ± 0.018
	Sharpe	2.775 ± 0.365	1.723 ± 0.161	1.173 ± 0.113
Life Insurance	ROI	0.185 ± 0.004	0.144 ± 0.004	0.120 ± 0.010
	Sharpe	1.472 ± 0.039	1.206 ± 0.027	0.976 ± 0.086
Medical Devices	ROI	0.669 ± 0.049	0.462 ± 0.067	0.289 ± 0.036
	Sharpe	1.979 ± 0.167	1.475 ± 0.178	1.060 ± 0.125
Retail Consumer Goods	ROI	0.659 ± 0.055	0.468 ± 0.143	0.317 ± 0.187
	Sharpe	2.465 ± 0.293	1.850 ± 0.132	1.291 ± 0.185
Overall (avg.)	ROI	0.607 ± 0.045	0.346 ± 0.048	0.225 ± 0.056
	Sharpe	2.157 ± 0.166	1.439 ± 0.103	1.024 ± 0.122

Table 7: LGHI module vs simple concatenation. Results are mean ± std over ten seeds. Best in each row is red.

Industry	Metric	LGHI	Concat
Biotechnology	ROI	0.601 ± 0.034	0.203 ± 0.037
	Sharpe	1.695 ± 0.050	0.640 ± 0.040
Semiconductors	ROI	1.104 ± 0.053	0.374 ± 0.022
	Sharpe	2.555 ± 0.081	0.964 ± 0.060
Renewable Energy	ROI	0.423 ± 0.074	0.150 ± 0.035
	Sharpe	2.775 ± 0.365	1.047 ± 0.165
Life Insurance	ROI	0.185 ± 0.004	0.063 ± 0.012
	Sharpe	1.472 ± 0.039	0.555 ± 0.034
Medical Devices	ROI	0.669 ± 0.049	0.227 ± 0.028
	Sharpe	1.979 ± 0.167	0.747 ± 0.076
Retail Consumer Goods	ROI	0.659 ± 0.055	0.222 ± 0.154
	Sharpe	2.465 ± 0.293	0.930 ± 0.130
Overall (avg.)	ROI	0.607 ± 0.045	0.207 ± 0.048
	Sharpe	2.157 ± 0.166	0.814 ± 0.083

Table 8: Sharpe loss ablation by industry on the test set. Results are reported as mean ± standard deviation over 10 random seeds for ROI, Sharpe ratio and maximum draw-down (MDD, %).

Industry	Soft-label only			Soft-label + Sharpe (ours)		
	ROI	Sharpe	MDD (%)	ROI	Sharpe	MDD (%)
Biotechnology	0.511 ± 0.061	1.187 ± 0.190	17.203 ± 2.962	0.601 ± 0.034	1.695 ± 0.050	10.658 ± 1.932
Semiconductors	0.938 ± 0.113	1.789 ± 0.286	11.413 ± 1.965	1.104 ± 0.053	2.555 ± 0.081	7.071 ± 1.282
Renewable Energy	0.377 ± 0.045	1.943 ± 0.311	10.508 ± 1.809	0.443 ± 0.064	2.775 ± 0.365	6.510 ± 1.180
Life Insurance	0.157 ± 0.019	1.030 ± 0.165	19.810 ± 3.410	0.185 ± 0.004	1.472 ± 0.039	12.273 ± 2.225
Medical Devices	0.569 ± 0.068	1.385 ± 0.222	14.735 ± 2.537	0.669 ± 0.049	1.979 ± 0.167	9.128 ± 1.655
Retail Consumer Goods	0.267 ± 0.032	1.068 ± 0.171	19.109 ± 3.290	0.659 ± 0.055	2.465 ± 0.293	11.838 ± 2.146
Overall (avg.)	0.470 ± 0.056	1.400 ± 0.224	15.463 ± 2.662	0.553 ± 0.035	2.000 ± 0.122	9.580 ± 1.736

uses a 2-layer LSTM with 512 hidden units. We report mean±std over ten random seeds and synchronize stochastic components across models for fair comparison.

Table 9 reports test ROI and Sharpe by industry and on average. The wavelet front-end improves both metrics for every backbone, and LSTM-based models consistently outperform MLP, consistent with prior evidence that recurrent and attention architectures better capture long-range dependencies and regime shifts in financial time series Nelson et al. (2017); Moghar & Hamiche (2020); Bao et al. (2017); Kabir et al. (2025); Mozaffari & Zhang (2024). Overall, WaveLSFormer achieves the best performance with a parameter budget comparable to LSTM variants, reaching 0.607 ± 0.045 ROI and 2.157 ± 0.166 Sharpe, compared with 0.225 ± 0.056 ROI and 1.024 ± 0.122 Sharpe for the plain Transformer. Fig. 9 shows smoother equity growth with smaller draw-downs than the strongest non-wavelet baselines in Renewable Energy and Retail Consumer Goods.

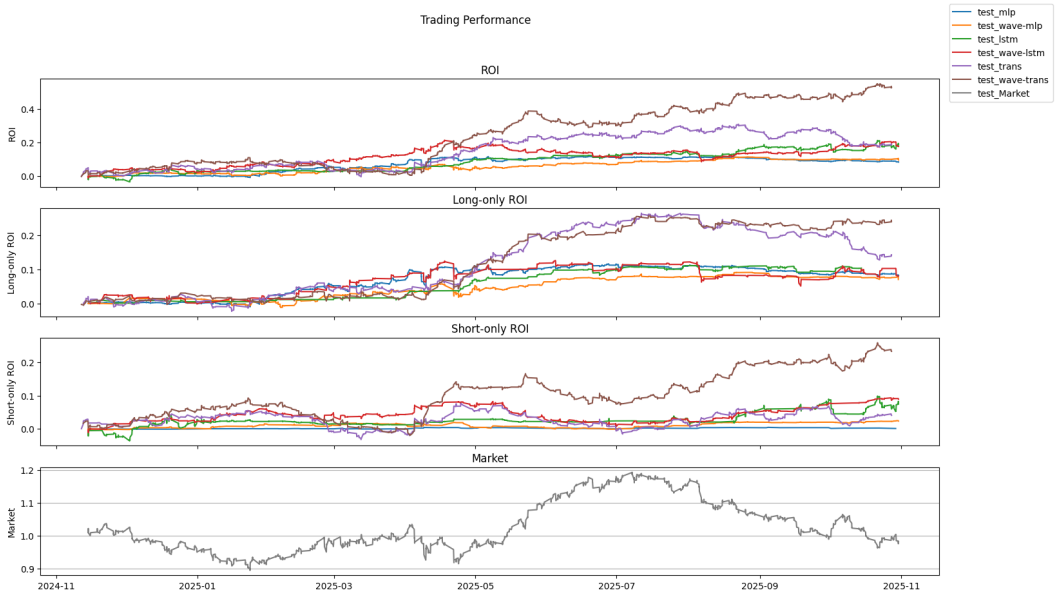
A.10.2 Model Complexity and Overall Performance

Table 1 reports model complexity and trading performance for each backbone with and without the neural wavelet front-end. MLP is a lightweight reference with 8.146M parameters and 3.468G FLOPs, while LSTM and Transformer are substantially more expensive at 12.538M/492.614G and 15.928M/665.921G. To rule out brute-force scaling, we report parameters and FLOPs for paired comparisons within each backbone. The wavelet module introduces only small capacity changes, increasing parameters to 8.151M for MLP, 13.298M for LSTM, and 15.943M for Transformer. Its computational overhead is modest, with LSTM FLOPs increasing by 7.8% and Transformer FLOPs slightly decreasing by 0.9%.

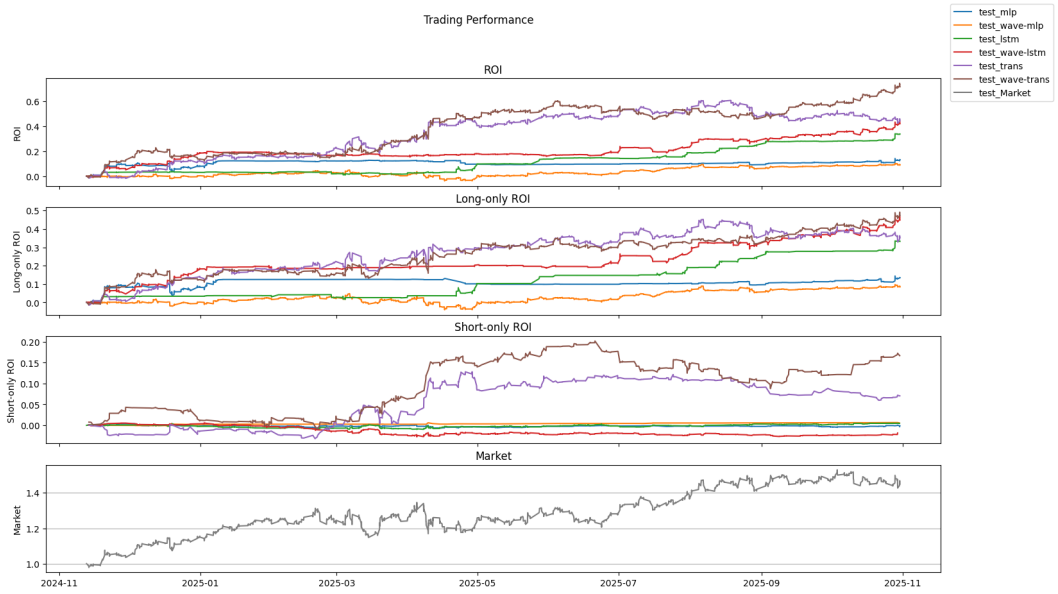
At comparable complexity, the wavelet front-end consistently improves performance. On average, MLP improves from 0.075 ROI and 0.813 Sharpe to 0.165 and 1.079, and LSTM improves from 0.191 ROI and 1.656 Sharpe to 0.317 and 1.879. WaveLSFormer achieves the best overall ROI and Sharpe with virtually the same parameter count and slightly fewer FLOPs than the vanilla Transformer. It also surpasses Wavelet+LSTM, improving ROI from 0.317 to 0.607 and Sharpe from 1.879 to 2.157. These gains without disproportionate scaling suggest that the improvement is driven by the inductive bias of the neural wavelet front-end and its integration with the Transformer backbone, establishing a new Pareto frontier of performance versus complexity.

Table 9: Per-industry comparison of MLP, LSTM and Transformer backbones with and without the wavelet front-end. Results are mean $\pm$ std over ten seeds. The best in each row is red.

Industry	Metric	MLP	Wav.+MLP	LSTM	Wav.+LSTM	Trans.	Wav.+Trans.
Biotechnology	ROI	0.041 ± 0.013	0.092 ± 0.024	0.106 ± 0.016	0.173 ± 0.026	0.124 ± 0.066	0.601 ± 0.034
	Sharpe	0.405 ± 0.157	0.553 ± 0.074	0.823 ± 0.242	0.934 ± 0.102	0.511 ± 0.189	1.695 ± 0.050
Semiconductors	ROI	0.110 ± 0.035	0.246 ± 0.064	0.285 ± 0.043	0.465 ± 0.069	0.333 ± 0.021	1.104 ± 0.053
	Sharpe	0.899 ± 0.349	1.227 ± 0.164	1.826 ± 0.537	2.072 ± 0.227	1.134 ± 0.035	2.555 ± 0.081
Renewable Energy	ROI	0.053 ± 0.020	0.139 ± 0.013	0.159 ± 0.017	0.221 ± 0.022	0.169 ± 0.018	0.423 ± 0.074
	Sharpe	0.851 ± 0.431	1.490 ± 0.054	1.482 ± 0.366	1.656 ± 0.061	1.173 ± 0.113	2.775 ± 0.365
Life Insurance	ROI	0.040 ± 0.013	0.089 ± 0.023	0.103 ± 0.015	0.168 ± 0.025	0.120 ± 0.010	0.185 ± 0.004
	Sharpe	0.774 ± 0.301	1.056 ± 0.141	1.572 ± 0.462	1.783 ± 0.195	0.976 ± 0.086	1.472 ± 0.039
Medical Devices	ROI	0.095 ± 0.030	0.214 ± 0.056	0.248 ± 0.037	0.404 ± 0.060	0.289 ± 0.036	0.669 ± 0.049
	Sharpe	0.840 ± 0.327	0.995 ± 0.133	1.707 ± 0.502	1.937 ± 0.212	1.060 ± 0.125	1.979 ± 0.167
Retail Consumer Goods	ROI	0.110 ± 0.028	0.208 ± 0.089	0.245 ± 0.047	0.471 ± 0.093	0.317 ± 0.187	0.659 ± 0.055
	Sharpe	1.110 ± 0.301	1.154 ± 0.267	2.527 ± 0.862	2.895 ± 0.528	1.291 ± 0.185	2.465 ± 0.293
Overall (avg.)	ROI	0.075 ± 0.023	0.165 ± 0.045	0.191 ± 0.029	0.317 ± 0.049	0.225 ± 0.056	0.607 ± 0.045
	Sharpe	0.813 ± 0.311	1.079 ± 0.139	1.656 ± 0.495	1.879 ± 0.221	1.024 ± 0.122	2.157 ± 0.166



(a) Renewable Energy



(b) Retail Consumer Goods

Figure 9: Equity curves on the test period for (a) Renewable Energy and (b) Retail Consumer Goods. For each sector we compare the non-wavelet MLP and LSTM, the wavelet-enhanced MLP and LSTM, and WaveLSFormer.

Figure 10: Strategy return curve of renewable energy

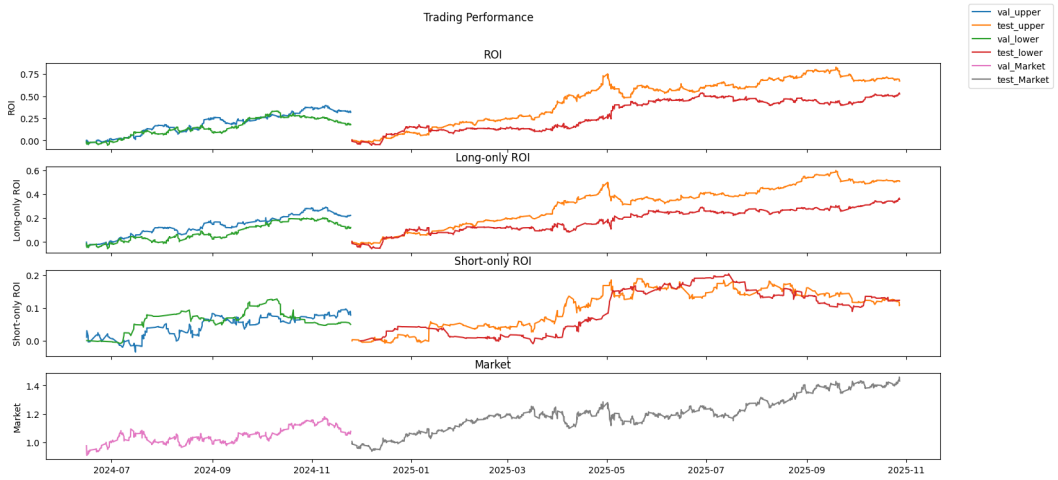


Figure 11: Strategy return curve of retail consumer goods

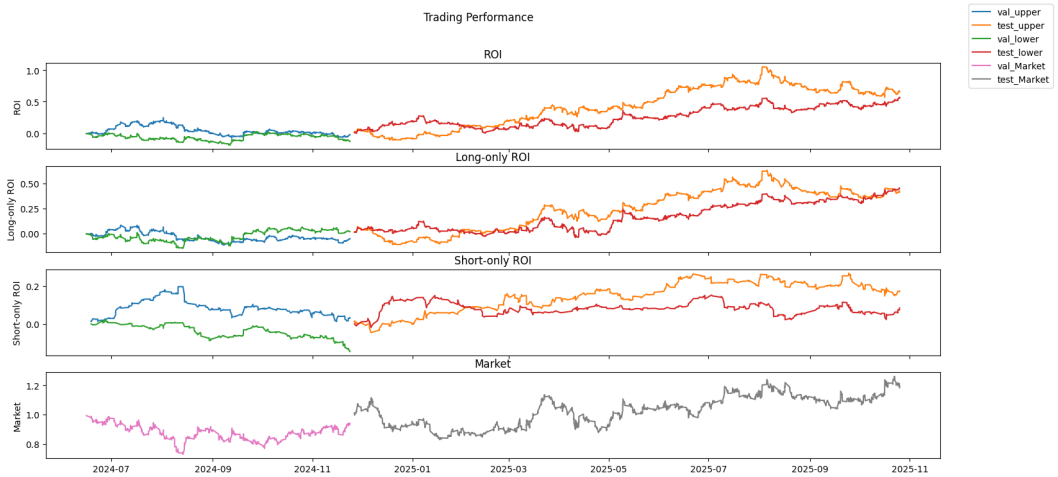


Figure 12: Biotechnology: trading performance curve.

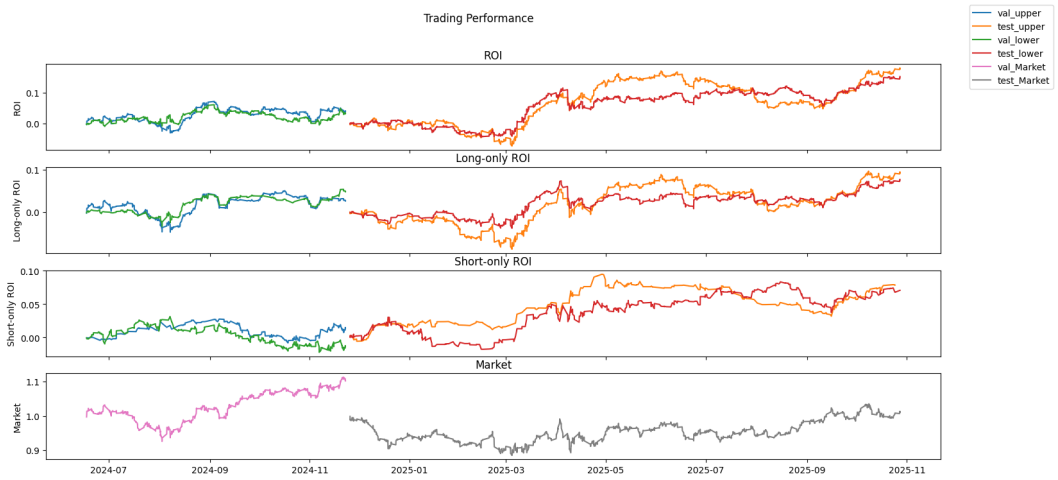


Figure 13: Life Insurance: trading performance curve.

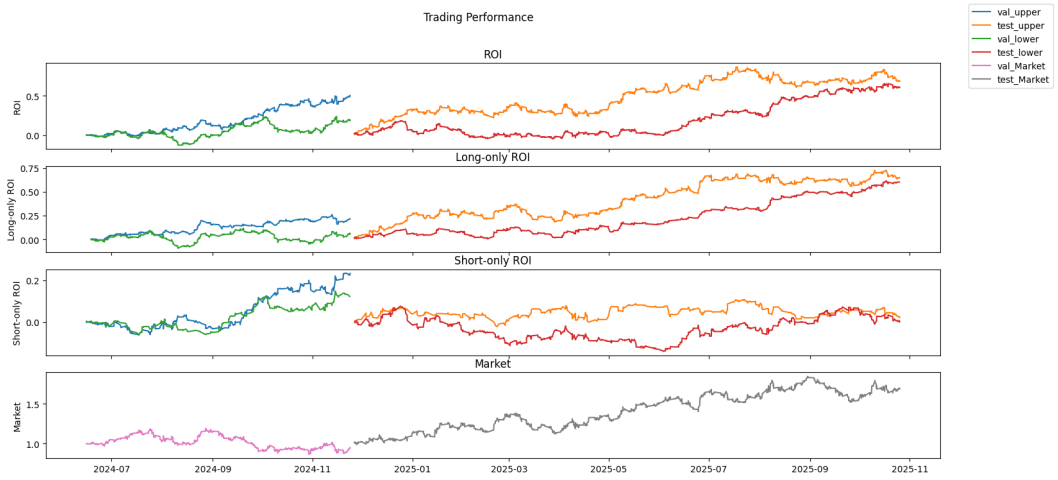


Figure 14: Medical Devices: trading performance curve.

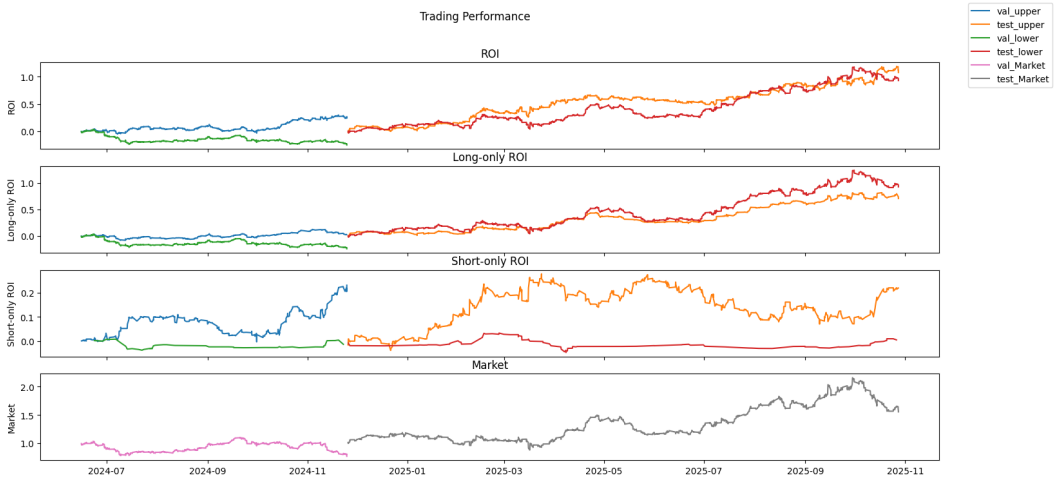


Figure 15: Semiconductor: trading performance curve.