DIFFUSION-BASED UNCERTAINTY–UTILITY ALIGNMENT FOR PORTFOLIO OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Portfolio optimization requires investment decisions that balance return and risk under uncertainty. Probabilistic forecasting provides an explicit representation of future return uncertainty. However, predictive uncertainty is not inherently aligned with investment utility: portfolio decisions can be sensitive to scenario-sampling variability, while forecasting-only training does not directly account for the investment consequences of the resulting weights. In this paper, we propose diffusion-based uncertainty–utility alignment (DualPO) to align both the exploitation and generation of predictive uncertainty with investment utility. Specifically, DualPO trains an uncertainty-conditioned allocator using differentiable distributional summaries of return scenarios generated by a pretrained diffusion forecaster, aligning portfolio decisions with investment utility. Meanwhile, DualPO backpropagates utility gradients through the allocator to adapt the forecaster, aligning scenario generation with investment utility while retaining denoising supervision. Experimental results on the S&P and CSI real-market benchmarks show that DualPO outperforms state-of-the-art baselines.

1 INTRODUCTION

Portfolio optimization, which aims to determine investment weights that balance return and risk, is a fundamental task in quantitative investment (Zhang et al., 2024; Soun et al., 2026). Since portfolio weights are chosen before future returns are observed, accounting for predictive uncertainty is essential to investment decisions. Existing methods differ in how they determine portfolio weights. Conventional two-stage methods first estimate future returns and risk statistics, and then construct portfolios using an optimization model (Soun et al., 2026). End-to-end methods with direct weight prediction instead train neural networks to map market observations to portfolio weights under investment objectives (Zhang et al., 2024). These architectures learn the portfolio response directly but do not explicitly model a conditional distribution of future returns.

Recently, diffusion models have gained increasing attention in financial time-series prediction and generation (Daiya et al., 2024; Takahashi & Mizuno, 2025; Tanaka et al., 2025). Specifically, conditional diffusion models learn to generate future trajectories by denoising random samples conditioned on historical observations (Rasul et al., 2021; Daiya et al., 2024). The generated scenarios describe the predictive distribution of future returns, providing an explicit representation of uncertainty for investment decisions. This probabilistic forecasting capability has also motivated diffusion-based methods for portfolio optimization (Soun et al., 2026; Zhao et al., 2026).

Existing diffusion-based portfolio methods adopt different strategies for training their forecasters. Two-stage approaches fit return distributions using denoising objectives and then use the predicted moments to construct portfolios (Soun et al., 2026). However, these forecasting objectives do not directly account for the investment utility of the resulting portfolio, leading to misalignment between the scenario generation and investment utility. Decision-focused approaches instead incorporate downstream decision losses into forecaster training while retaining a stochastic optimization layer (Zhao et al., 2026). Nonetheless, they result in averaging the uncertainty in the generated scenarios instead of exploiting its tail characteristics.

In this paper, we propose a novel method called *diffusion-based uncertainty–utility alignment* (DualPO) to align both the exploitation and generation of predictive uncertainty with investment utility. Specifically, DualPO trains an uncertainty-conditioned allocator using differentiable distributional

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

summaries of return scenarios generated by a pretrained diffusion forecaster. The allocator learns a shared mapping to long-only, fully invested portfolio weights across market states and scenario draws, with its decisions evaluated on observed investment outcomes. Meanwhile, DualPO backpropagates utility gradients through the allocator to adapt the forecaster, aligning scenario generation with investment utility while retaining denoising supervision. The two components are optimized jointly, allowing investment utility to guide both the exploitation and generation of predictive uncertainty.

The main contributions of DualPO are summarized as follows:

- DualPO introduces an uncertainty-conditioned allocator to exploit differentiable distributional summaries, learning feasible portfolio weights under an investment objective without a per-instance optimization layer.
- DualPO aligns scenario generation with investment utility by backpropagating realized investment feedback through the allocator to the forecaster while retaining denoising supervision.
- Experimental results on the S&P and CSI real-market benchmarks show that DualPO outperforms state-of-the-art baselines.

2 Related Work

Portfolio optimization. Conventional two-stage methods first estimate return statistics and then construct portfolios by solving an optimization problem. Mean–variance optimization determines portfolio weights from expected returns and covariances (Markowitz, 1952), while Black–Litterman combines equilibrium information with subjective or model-derived views (Black & Litterman, 1992). End-to-end methods with direct weight prediction instead learn a reusable mapping from market observations to portfolio weights under investment objectives, including reinforcement-learning approaches (Wang et al., 2021). These feature-to-weight architectures incorporate risk through their objectives and learned representations, but do not explicitly provide the decision module with a conditional distribution of future returns. Task-based predictive learning also connects probabilistic forecasting to downstream decisions by training predictors under the task objective (Donti et al., 2017).

Diffusion models for financial time series. Diffusion-based probabilistic forecasting models conditional distributions of future sequences through denoising (Rasul et al., 2021). For financial prediction, conditional diffusion models incorporate historical financial indicators and inter-stock relations to model future stock movements (Daiya et al., 2024). Financial data-generation methods instead focus on reproducing statistical properties of market data, including heavy-tailed returns, volatility clustering, and intraday patterns (Takahashi & Mizuno, 2025). Conditional generators further incorporate specified trend and volatility conditions to produce controllable financial trajectories (Tanaka et al., 2025). In portfolio optimization, forecasting-oriented methods generate return scenarios from macroeconomic and asset-specific information and incorporate their estimated moments into a Black–Litterman model (Soun et al., 2026). Utility-guided methods further adapt diffusion forecasters using downstream investment losses while retaining a sample-based stochastic optimization layer to determine portfolio weights (Zhao et al., 2026). DualPO instead trains an uncertainty-conditioned allocator using differentiable distributional summaries and propagates realized investment feedback through the allocator to the forecaster. This aligns both the exploitation and generation of predictive uncertainty with investment utility while retaining denoising supervision.

3 Method

3.1 Problem formulation

At decision date t, the market history $\mathcal{H}_t$ contains observations through the preceding valuation date for N assets. Let $y_{t,t+H} \in \mathbb{R}^N$ denote the cumulative simple returns over the next H trading sessions. We determine long-only, fully invested weights by balancing the conditional return and

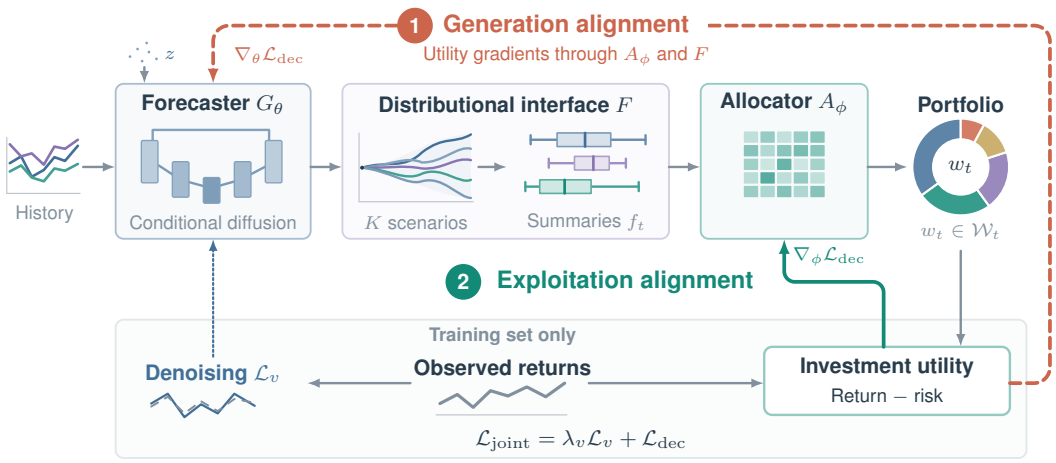


Figure 1: **Overview of DualPO framework.** Gray arrows show the forward computation from market history to portfolio weights. **(1) Generation alignment** adapts the forecaster by propagating utility gradients through the allocator and distributional interface (dashed coral). **(2) Exploitation alignment** learns the allocator's distribution-to-weight response from realized utility (solid green). Denoising supervision is retained to keep forecasting performance (dotted blue). The two alignments are conducted alternatively.

risk of the resulting portfolio:

$$\begin{aligned} \max_{w_t \in \mathbb{R}^N} \quad & \mathbb{E}\Big[w_t^\top y_{t,t+H} - \frac{\gamma}{2}\,\mathrm{Var}\big(w_t^\top y_{t,t+H}\big) \mid \mathcal{H}_t\Big] \\ \text{s.t.} \quad & w_{t,i} \geq 0, \quad i = 1, \ldots, N, \\ & \sum_{i=1}^{N} w_{t,i} = 1, \end{aligned} \tag{1}$$

where $\gamma > 0$ controls risk aversion. The constraints enforce a long-only, fully invested portfolio without leverage. DualPO learns a reusable mapping from the conditional return distribution to a feasible weight vector; the distributional summaries and learned mapping are specified in the subsequent subsections.

3.2 Probabilistic Scenario Generator

DualPO represents predictive uncertainty with a conditional diffusion forecaster. The forecaster consists of a market context encoder and a velocity-prediction denoiser. A contemporaneous market snapshot and recent asset trajectories are constructed causally from $\mathcal{H}_t$; shared projections and temporal residual blocks encode these inputs, while cross-asset attention communicates information across eligible assets. We write the resulting condition as

$$c_t^\theta = E_\theta(\mathcal{H}_t, e_t), \tag{2}$$

and use the same c_t^θ for all scenario draws and denoising steps. The conditioning features use only information available through $t-1$.

v-parameterized diffusion modeling. Let $y_t \in \mathbb{R}^{N \times H}$ denote the future log-return path and let $x_{t,0} = \mathcal{C}_t(y_t)$ be its causally normalized representation. For a linear noise schedule with S diffusion states, define

$$\alpha_s = 1 - \beta_s, \qquad \bar{\alpha}_s = \prod_{j=0}^{s} \alpha_j, \qquad a_s = \sqrt{\bar{\alpha}_s}, \qquad b_s = \sqrt{1 - \bar{\alpha}_s}. \tag{3}$$

For a uniformly sampled diffusion index s, the training state and v-prediction target are

$$x_{t,s} = a_s x_{t,0} + b_s \epsilon, \quad v_{t,s}^\star = a_s \epsilon - b_s x_{t,0}, \tag{4}$$

where $\epsilon \sim \mathcal{N}(0, I)$. The denoiser $v_\theta(x_{t,s}, s, c_t^\theta)$ is pretrained with

$$\ell_{v,t}(\theta) = \frac{1}{NH} \sum_{i=1}^{N} \sum_{h=1}^{H} \left[v_\theta(x_{t,s}, s, c_t^\theta)_{i,h} - v^\star_{t,s,i,h} \right]^2 . \tag{5}$$

Pretraining minimizes the average of $\ell_{v,t}$ over pairs from the training set, and the same denoising term remains active during alignment.

For completeness, the standard v-parameterized DDPM sampler initializes $x_{t,S-1}^{(k)} = z_t^{(k)} \sim \mathcal{N}(0, I)$. Here $k \in \{1, \ldots, K\}$ indexes an independent scenario trajectory in the scenario bank; s indexes a diffusion state, and the DDPM reverse noise is also trajectory-specific. At state s, the denoiser implies a clean-state estimate and an equivalent noise estimate,

$$\widehat{x}_{t,0}^{(k,s)} = a_s x_{t,s}^{(k)} - b_s v_\theta(x_{t,s}^{(k)}, s, c_t^\theta), \tag{6}$$

$$\widehat{\epsilon}_t^{(k,s)} = b_s x_{t,s}^{(k)} + a_s v_\theta(x_{t,s}^{(k)}, s, c_t^\theta). \tag{7}$$

The posterior parameters for the transition from s to $s-1$ are

$$\widetilde{\mu}_t^{(k,s)} = \frac{\sqrt{\bar{\alpha}_{s-1}}\,\beta_s}{1-\bar{\alpha}_s} \widehat{x}_{t,0}^{(k,s)} + \frac{\sqrt{\alpha_s}\,(1-\bar{\alpha}_{s-1})}{1-\bar{\alpha}_s} x_{t,s}^{(k)}, \tag{8}$$

$$\widetilde{\beta}_s = \frac{1-\bar{\alpha}_{s-1}}{1-\bar{\alpha}_s} \beta_s, \tag{9}$$

and the stochastic DDPM update is

$$x_{t,s-1}^{(k)} = \widetilde{\mu}_t^{(k,s)} + \sqrt{\widetilde{\beta}_s}\, \xi_t^{(k,s)}, \qquad \xi_t^{(k,s)} \sim \mathcal{N}(0, I), \tag{10}$$

with the noise term omitted at the final clean state. This full reverse chain uses one denoiser evaluation at every diffusion state and is therefore costly when a scenario bank is differentiated through the sampler.

Differentiable few-step scenario sampling. During utility alignment, we use a selected subgrid $S-1 = s_0 > s_1 > \cdots > s_{J-1} \geq 0$ with $J \ll S$. At each selected state, the same v-prediction identities give

$$\widehat{x}_{t,0}^{(k,s_j)} = a_{s_j} x_{t,s_j}^{(k)} - b_{s_j} v_\theta(x_{t,s_j}^{(k)}, s_j, c_t^\theta), \tag{11}$$

$$\widehat{\epsilon}_t^{(k,s_j)} = b_{s_j} x_{t,s_j}^{(k)} + a_{s_j} v_\theta(x_{t,s_j}^{(k)}, s_j, c_t^\theta). \tag{12}$$

We use the deterministic DDIM transition ($\eta = 0$) between two adjacent selected states:

$$x_{t,s_{j+1}}^{(k)} = a_{s_{j+1}} \widehat{x}_{t,0}^{(k,s_j)} + b_{s_{j+1}} \widehat{\epsilon}_t^{(k,s_j)}, \qquad j = 0, \ldots, J-2. \tag{13}$$

The final clean estimate is

$$\widehat{y}_{t,\theta}^{(k)} = \mathcal{C}_t^{-1}\left(\widehat{x}_{t,0}^{(k,s_{J-1})} \right), \qquad k = 1, \ldots, K. \tag{14}$$

Thus J is the number of denoiser evaluations per scenario. The few-step DDIM sampler preserves a differentiable path from origin noise through the generated distributional features and portfolio weights, while reducing denoiser calls and backpropagation memory from the full schedule to J evaluations per scenario. This makes utility feedback practical for a bank of scenarios; increasing J can improve path fidelity at additional cost. The asset-factorized denoiser models conditional marginal uncertainty rather than an explicit future-shock covariance model.

3.3 Uncertainty-Conditioned Allocator

3.3.1 Distributional Feature Interface

The interface transforms the generated trajectories into differentiable asset-wise features. For asset i and scenario k, let $R_{t,i,\theta}^{(k)}$ denote the terminal simple return obtained by decoding the generated log-return path. The feature vector $d_{t,i}^\theta$ contains nine complementary distributional summaries: $\widehat{\mu}_{t,i}^\theta$

is the mean and represents the location of the predicted opportunity; $\widehat{\sigma}^{\theta}_{t,i}$ is the standard deviation and measures overall scenario spread; $\widehat{d}^{\theta}_{t,i}$ is the downside deviation and isolates adverse dispersion; $\widehat{m}^{\theta}_{t,i}$ is the median absolute deviation and provides a robust spread measure; and $\widehat{q}^{\theta}_{p,t,i}$ contains empirical quantiles for $p \in \{0.10, 0.25, 0.50, 0.75, 0.90\}$, exposing lower-tail, central, and upper-tail locations. The exact statistical definitions are given in Appendix B.2.1.

Collecting these channels gives $d^{\theta}_{t,i} = [\widehat{\mu}^{\theta}_{t,i}, \widehat{\sigma}^{\theta}_{t,i}, \widehat{d}^{\theta}_{t,i}, \widehat{m}^{\theta}_{t,i}, \widehat{q}^{\theta}_{.10,t,i}, \widehat{q}^{\theta}_{.25,t,i}, \widehat{q}^{\theta}_{.50,t,i}, \widehat{q}^{\theta}_{.75,t,i}, \widehat{q}^{\theta}_{.90,t,i}]$. The summaries are permutation-invariant over scenarios and remain differentiable through the generated paths, including the piecewise derivatives of linearly interpolated quantiles. The normalized feature vector and eligibility bit form the allocator input; normalization scales are fitted on the training set and then frozen.

The allocator learns how each asset's predictive distribution should affect its portfolio weight. A shared asset MLP with parameters contained in ϕ embeds $d^{\theta}_{t,i}$, and a four-head self-attention block compares these embeddings across eligible assets. A global descriptor pools the asset-axis mean and population standard deviation of the first four feature channels, communicating the overall opportunity and spread of the investable universe. Combining asset and global embeddings produces a score $o^{\theta,\phi}_{t,i}$.

We obtain feasible portfolio weights through a bounded score map,

$$w^{\theta,\phi}_{t,i} = \frac{e_{t,i} \exp\{\tau_t \tanh(o^{\theta,\phi}_{t,i})\}}{\sum_{j=1}^{N} e_{t,j} \exp\{\tau_t \tanh(o^{\theta,\phi}_{t,j})\}}. \tag{15}$$

The eligibility indicator restricts attention keys and the final allocation. The shared mapping is permutation equivariant and directly outputs long-only, fully invested weights. The bounded temperature is selected to maintain the effective-holdings floor while returning a feasible portfolio .

3.4 Joint Uncertainty–Utility Alignment

Alignment couples the exploitation and generation of predictive uncertainty through one realized investment objective. The allocator is evaluated on the observed future path paired with $\mathcal{H}_t$, whereas the generated paths provide the distributional features used to produce $w^{\theta,\phi}_t = A_\phi(f^{\theta}_t, e_t)$.

Realized investment utility. For a sample pair $(\mathcal{H}_t, y_t)$, let Y_t denote the vector of realized cumulative simple asset returns over the holding horizon. Given $w^{\theta,\phi}_t$, the implementation first computes the realized daily returns $g_{t,1:H}(w^{\theta,\phi}_t)$ of the fixed-share-count buy-and-hold account from the observed path. It then computes their sample variance across the H sessions. The per-pair decision loss is

$$\ell_{\mathrm{dec},t}(\theta, \phi) = -(w^{\theta,\phi}_t)^{\top} Y_t + \lambda_{\mathrm{var}} \operatorname{Var}_{h=1:H}\left[g_{t,h}(w^{\theta,\phi}_t)\right]. \tag{16}$$

Utility-alignment objective. For a sample pair and a fixed scenario bank $z^{(1:K)}_t$, let $f^{\theta}_t = f(\mathcal{S}^{\mathrm{few}}_{\theta}(\mathcal{H}_t, z^{(1:K)}_t), e_t)$ and $w^{\theta,\phi}_t = A_\phi(f^{\theta}_t, e_t)$. The generator substep uses the per-pair loss

$$\ell_{\mathrm{gen},t}(\theta; \phi, z^{(1:K)}_t) = \lambda_v \ell_{v,t}(\theta) + \lambda_{\mathrm{dec}} \ell_{\mathrm{dec},t}(\theta, \phi). \tag{17}$$

The decision term is differentiable with respect to θ through the scenario sampler, distributional features, and allocator, so it supplies utility feedback to the forecaster in addition to the denoising term.

Paired alternating alignment. DualPO separates the two parameter updates and keeps the random noise inputs fixed while generating scenario for this step. For each minibatch, one Gaussian bank $z^{(1:K)}_t$ is drawn and reused by both substeps. With ϕ held fixed, the generator gradient and update are

$$g_\theta = \nabla_\theta \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \ell_{\mathrm{gen},t}(\theta; \phi, z^{(1:K)}_t), \qquad \theta^{+} = \mathrm{AdamW}_\theta(\theta, g_\theta). \tag{18}$$

Algorithm 1 DualPO training and inference

Require: Datasets $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, and $\mathcal{D}_{\text{test}}$; horizon H; scenario count K; diffusion schedule $\{\bar{\alpha}_s\}_{s=0}^{S-1}$

Ensure: Portfolio weights $w_t^{\theta,\phi}$ for each deployment history $\mathcal{H}_t$

Pretraining:

for each minibatch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$ **do**
  $c_t^\theta \leftarrow E_\theta(\mathcal{H}_t, e_t)$ and $x_{t,0} \leftarrow \mathcal{C}_t(y_t)$ for $t \in \mathcal{B}$
  Sample s uniformly and $\epsilon \sim \mathcal{N}(0, I)$; compute $x_{t,s}$, $v_{t,s}^\star$, and $\ell_{v,t}(\theta)$
  Update θ with one AdamW step on $|\mathcal{B}|^{-1} \sum_{t\in\mathcal{B}} \ell_{v,t}(\theta)$
end for
Select $\theta^\star$ using the native validation loss on $\mathcal{D}_{\text{val}}$

Alignment:

Initialize $\theta \leftarrow \theta^\star$ and a fresh allocator ϕ
for each minibatch $\mathcal{B} \subset \mathcal{D}_{\text{train}}$ **do**
  Sample a Gaussian bank $z_t^{(1:K)}$ for each $t \in \mathcal{B}$
  Generate paths with $\mathcal{S}_\theta^{\text{few}}$ and compute $\ell_{\text{gen},t}(\theta; \phi, z_t^{(1:K)})$
  $g_\theta \leftarrow \nabla_\theta |\mathcal{B}|^{-1} \sum_t \ell_{\text{gen},t}(\theta; \phi, z_t^{(1:K)})$; update $\theta^+ \leftarrow \text{AdamW}_\theta(\theta, g_\theta)$
  Regenerate paths with θ^+ and the same $z_t^{(1:K)}$; stop gradients through θ^+
  Compute $\ell_{\text{alloc},t}(\phi; \theta^+, z_t^{(1:K)})$
  $g_\phi \leftarrow \nabla_\phi |\mathcal{B}|^{-1} \sum_t \ell_{\text{alloc},t}(\phi; \theta^+, z_t^{(1:K)})$; update $\phi^+ \leftarrow \text{AdamW}_\phi(\phi, g_\phi)$
  Set $(\theta, \phi) \leftarrow (\theta^+, \phi^+)$
end for

Inference on $\mathcal{D}_{\text{test}}$:

Given $\mathcal{H}_t$, compute c_t^θ, sample $z_t^{(1:K)}$, and apply the selected few-step sampler
Decode and summarize the generated paths; return $w_t^{\theta,\phi} \leftarrow A_\phi(f_t^\theta, e_t)$

The gradient is taken through the allocator computation but only θ is updated. The scenarios are then regenerated with θ^+ and the same noise bank. Holding θ^+ fixed for the allocator substep, DualPO defines

$$\ell_{\text{alloc},t}(\phi; \theta^+, z_t^{(1:K)}) = \lambda_{\text{dec}} \ell_{\text{dec},t}(\theta^+, \phi), \tag{19}$$

and computes

$$g_\phi = \nabla_\phi \frac{1}{|\mathcal{B}|} \sum_{t\in\mathcal{B}} \ell_{\text{alloc},t}(\phi; \theta^+, z_t^{(1:K)}), \qquad \phi^+ = \text{AdamW}_\phi(\phi, g_\phi). \tag{20}$$

The second gradient does not propagate into θ^+. Thus the generator substep aligns scenario generation using both denoising and realized utility, whereas the allocator substep adapts the distribution-to-weight mapping to the updated scenarios using realized utility alone. Reusing the same minibatch and origin noise makes the two updates a paired comparison rather than a change in investment information.

Training recipe and inference. Training has two stages. During pretraining, only the forecaster parameters θ are optimized with $\ell_{v,t}(\theta)$ and the checkpoint with the best native validation velocity loss is retained. During alignment, a fresh allocator is initialized with the selected forecaster. DualPO applies the two substeps in Eqs. 18 and 20 within each minibatch, using separate AdamW states for the forecaster and allocator. The validation set selects checkpoints, and the test set is used only after selection.

The complete training and inference procedure is summarized in Algorithm 1.

4 Experiments

In this section, we evaluate DualPO on real-world financial markets through historical backtesting. Reported results for trained methods are averages over three random seeds. We first compare port-

folio performance with representative classical and diffusion-based baselines. We then ablate the alternating utility-alignment update and the uncertainty interface used by the allocator.

4.1 Experiment Setup

4.1.1 Datasets

We evaluate on two equity markets and two universe sizes per market. Starting from the S&P500 and CSI300 constituents observed on 1 January 2024, we rank securities by market capitalization and retain the top 50 and top 100 assets. The resulting panels are denoted **S&P50**, **S&P100**, **CSI50**, and **CSI100**. All panels use daily adjusted prices and a fixed asset order within each experiment.

The training set spans January 2013–August 2023, the validation set spans September 2023–August 2024, and the test set spans September 2024–February 2025. Forecast horizons crossing a split boundary are removed. All preprocessing statistics are fitted on the training set and subsequently held fixed. The same market calendar, asset order, split, and tradability rules are used by all methods within a panel.

4.1.2 Baselines

We compare DualPO with five portfolio strategies in the same long-only, fully invested backtest. **CRP** is a constant-rebalanced equal-weight portfolio. (Kelly, 1956) **MOM** ranks assets by trailing return and allocates to the highest-ranked assets. (Jegadeesh & Titman, 1993) **MVO** estimates expected returns and covariances from historical observations and solves a long-only mean–variance problem (Markowitz, 1952). **STABLE** generates conditional return scenarios with a diffusion model and inserts the estimated mean and covariance as investor views in a Black–Litterman allocator (Soun et al., 2026). **Diffusion-DFL** trains a diffusion forecaster under a downstream decision loss and maps sampled scenarios to weights with a stochastic optimization layer (Zhao et al., 2026).

4.1.3 Evaluation Metrics

Let r_t denote the daily net portfolio return and V_t the corresponding net asset value. Our primary risk-adjusted metric is the annualized Sharpe ratio (ASR), computed with 252 trading days and a zero risk-free rate. We also report compound annual growth rate (CAGR, %), annualized volatility (Vol, %), and maximum drawdown (MDD, %). We further report PnL curves to show cumulative net asset value after transaction costs.

4.1.4 Implementation Details

The diffusion forecaster uses a 200-state linear noise schedule and generates $K = 50$ future scenarios over a horizon of $H = 20$ trading days. The configuration in main experiment uses deterministic few-step DDIM sampling with three network function evaluations (NFE). The forecaster and allocator embeddings have width 64, and training batches contain four decision dates.

Forecaster pretraining and alignment are run for at most 50 epochs with AdamW and validation-based early stopping with patience 5. The pretraining learning rate is 10^{-4}. During alternating alignment, the forecaster and allocator use separate AdamW optimizers with learning rates 10^{-5} and 10^{-4}, respectively. Weight decay is 10^{-3} and the gradient norm is clipped at 5. The generator loss uses $\lambda_v = 0.01$ and $\lambda_{\mathrm{dec}} = 1$, while the decision loss uses risk-aversion coefficient $\lambda_{\mathrm{var}} = 0.5$. We apply a 10-bps half-ℓ_1 transaction cost to executed portfolio changes. Numerical inputs are standardized using training-set statistics before entering the corresponding modules; the exact transformations are given in Appendix B.1.2. Complete network and optimization specifications are provided in Appendix B.

4.2 Experiment Results

4.2.1 Portfolio Performance

Table 1 reports net performance after transaction costs. DualPO achieves the best ASR and the best CAGR more often than any other method, indicating a stronger ability to generate portfolio return. It

Table 1: Net portfolio performance on the four real-market test sets. CAGR denotes compound annual growth rate, Vol denotes annualized volatility, MDD denotes maximum drawdown, and ASR denotes the annualized Sharpe ratio. CAGR, Vol, and MDD are reported as percentages. Best results are in **bold** and second-best results are underlined.

	CSI-50				*CSI-100*			
Method	ASR ↑	CAGR (%) ↑	Vol (%) ↓	MDD (%) ↓	ASR ↑	CAGR (%) ↑	Vol (%) ↓	MDD (%) ↓
CRP	1.28	32.01	**25.95**	13.01	1.48	39.75	**26.75**	**13.14**
MOM	0.41	6.46	59.98	27.98	0.34	2.98	58.80	26.60
MVO	0.31	1.24	58.17	27.07	-0.23	-22.09	54.23	31.40
STABLE	0.79	23.89	37.55	23.10	1.01	44.60	43.51	22.92
Diffusion-DFL	1.16	**46.58**	43.21	21.37	0.81	26.42	38.96	22.46
DualPO	**1.47**	43.03	26.89	**11.60**	**1.66**	**55.71**	33.44	13.49

	S&P-50				*S&P-100*			
Method	ASR ↑	CAGR (%) ↑	Vol (%) ↓	MDD (%) ↓	ASR ↑	CAGR (%) ↑	Vol (%) ↓	MDD (%) ↓
CRP	1.75	20.25	**10.97**	**4.31**	1.73	19.82	**10.84**	5.54
MOM	-0.07	-7.68	34.16	29.61	-0.07	-6.14	29.57	24.61
MVO	0.05	-8.95	49.11	40.67	0.40	7.97	48.43	39.40
STABLE	1.03	37.24	39.70	21.64	0.53	14.20	40.68	26.29
Diffusion-DFL	0.54	12.70	24.28	16.02	-0.68	-26.95	38.42	29.73
DualPO	**1.82**	**53.37**	29.30	9.25	**1.74**	**20.22**	11.59	**5.50**

also ranks near the front on volatility and maximum drawdown, and in most cases improves on both STABLE and Diffusion-DFL, so this return is balanced by risk control. Diffusion-DFL posts the strongest growth on CSI-50, but that result is accompanied by high volatility and a large drawdown. Surprisingly, STABLE and Diffusion-DFL fail to beat the equal-weight portfolio in many settings: both remain below CRP in Sharpe ratio on every panel, and they often fail to beat it in growth as well.

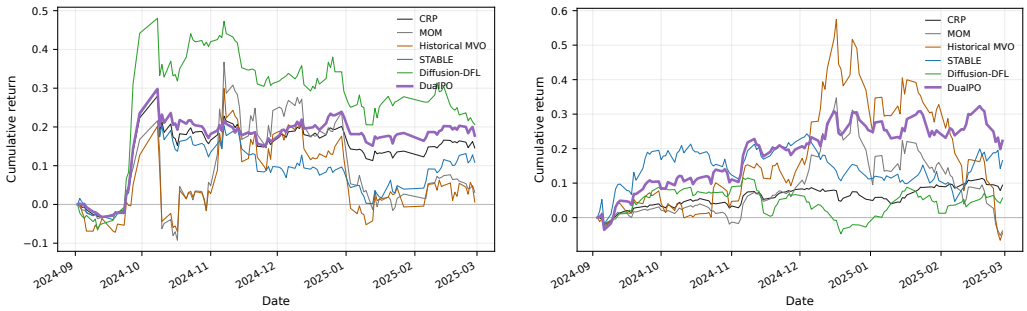
The two markets reach this ranking in different ways. On the CSI panels, DualPO raises growth while staying close to the risk of equal weight: CRP has the lowest volatility and DualPO is next, with a smaller drawdown than CRP on CSI-50 and a comparable one on CSI-100. CSI-100 separates the methods most clearly, since DualPO leads both ASR and CAGR, whereas STABLE's extra growth over CRP is offset by much larger volatility and drawdown. On the S&P panels, CRP is already a low-volatility benchmark, but DualPO still leads ASR and CAGR at both universe sizes. The growth gap is large on S&P-50, where DualPO takes more volatility than CRP while keeping a lower drawdown than both diffusion baselines. On S&P-100, DualPO and CRP are close on every metric, while STABLE is far more volatile and Diffusion-DFL has negative growth.

Figure 2 shows a comparatively stable path for DualPO, consistent with its volatility and drawdown in Table 1. On CSI-50, the curve stays below Diffusion-DFL, but the two finish at nearly the same level, and DualPO avoids the prolonged drawdown that follows Diffusion-DFL's peak. On S&P-50, DualPO rises with small fluctuations, without the extreme swings of MVO and with a steadier upward trend than STABLE and Diffusion-DFL. It finishes with the best overall performance on this panel.

4.2.2 Ablation Study

Table 2 tests two claims on CSI-50 and S&P-50: alternating updates are what produce the return gain, and the uncertainty interface is necessary for that gain. **Alternating** is the complete method in Section 3.4. Within each minibatch, the forecaster is updated with denoising and realized utility, and the allocator is then updated on scenarios from the updated forecaster. **Joint** uses the same losses but updates both modules simultaneously. **Frozen** updates only the allocator and holds the pretrained forecaster fixed.

The alternating schedule is the only one that leads both ASR and CAGR on both panels. The gap relative to Frozen shows that utility feedback has to adapt the forecaster: training the allocator



(a) CSI-50 (b) S&P-50

Figure 2: Cumulative return on the CSI-50 and S&P-50 test sets. Each panel compares CRP, MOM, MVO, STABLE, Diffusion-DFL, and DualPO.

Table 2: DualPO ablations on the CSI-50 and S&P-50 test sets. Best results are in **bold** and second-best results are underlined.

	CSI-50				*S&P-50*			
Variant	ASR ↑	CAGR ↑	Vol ↓	MDD ↓	ASR ↑	CAGR ↑	Vol ↓	MDD ↓
Uncertainty-Utility alignment								
Alternating	**1.47**	**43.03**	6.89	**11.60**	**1.82**	**53.37**	29.30	9.25
Joint	1.39	40.42	29.10	12.71	1.77	47.15	26.64	9.02
Frozen	1.28	34.61	27.04	12.95	1.81	48.68	26.98	9.48
Uncertainty representations								
w/o Distribution Interface	1.26	35.72	27.27	14.06	1.40	24.81	16.83	6.69
w/o Allocator	1.28	34.98	**25.94**	13.00	1.35	20.54	**15.21**	**4.31**

on a frozen denoising model leaves CSI-50 at the equal-weight Sharpe level and does not recover Alternating's growth. The gap relative to Joint shows that the order matters. A simultaneous update uses the same objectives but does not let the allocator train on scenarios from the updated forecaster, and both growth and Sharpe fall behind. On S&P-50 the Sharpe margins among the three schedules are smaller, while the growth advantage of Alternating remains on both markets.

The uncertainty-representation rows then change the interface that maps scenarios to weights. **w/o Distribution Interface** removes the differentiable distributional summaries in Section 3.3.1. **w/o Allocator** discards the learned allocator and uses the mean of the generated scenarios as the weight prediction, normalized by a softmax so that the weights are nonnegative and sum to one. Both changes lower ASR and CAGR relative to Alternating on both panels, with a particularly large loss of growth on S&P-50. On that panel the variants also reduce volatility and drawdown, but the reduction does not offset the loss of growth and Sharpe. The drop shows that the distributional interface and the learned allocator are necessary for the reported gains.

5 Conclusion

We propose diffusion-based uncertainty–utility alignment (DualPO) to align both the exploitation and generation of predictive uncertainty with investment utility. Specifically, DualPO trains an uncertainty-conditioned allocator using differentiable distributional summaries of return scenarios generated by a pretrained diffusion forecaster, aligning portfolio decisions with investment utility. Meanwhile, DualPO backpropagates utility gradients through the allocator to adapt the forecaster, aligning scenario generation with investment utility while retaining denoising supervision. Experimental results on the S&P 500 and CSI 300 benchmarks show that DualPO outperforms state-of-the-art baselines.

REFERENCES

Fischer Black and Robert Litterman. Global portfolio optimization. *Financial Analysts Journal*, 48 (5):28–43, 1992.

Divyanshu Daiya, Monika Yadav, and Harshit Singh Rao. DiffSTOCK: Probabilistic relational stock market predictions using diffusion models. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2024.

Priya Donti, Brandon Amos, and J. Zico Kolter. Task-based end-to-end model learning in stochastic optimization. In *Proceedings of the Neural Information Processing Systems*, 2017.

Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1):65–91, March 1993. doi: 10.1111/j.1540-6261.1993.tb04702.x.

John L. Kelly. A new interpretation of information rate. *Bell System Technical Journal*, 35(4): 917–926, July 1956. doi: 10.1002/j.1538-7305.1956.tb03809.x.

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

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of the International Conference on Machine Learning*, 2021.

Yejun Soun, Hosung Lee, Suyoung Park, and U Kang. Shift-tolerant allocation via Black–Litterman using conditional diffusion estimates. In *Proceedings of the International Conference on Learning Representations*, 2026.

Tomonori Takahashi and Takayuki Mizuno. Generation of synthetic financial time series by diffusion models. *Quantitative Finance*, 25(10):1507–1516, 2025.

Yuki Tanaka, Ryuji Hashimoto, Takehiro Takayanagi, Zhe Piao, Yuri Murayama, and Kiyoshi Izumi. CoFinDiff: Controllable financial diffusion model for time series generation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2025.

Zhicheng Wang, Biwei Huang, Shikui Tu, Kun Zhang, and Lei Xu. DeepTrader: A deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Wentao Zhang, Yilei Zhao, Shuo Sun, Jie Ying, Yonggang Xie, Zitao Song, Xinrun Wang, and Bo An. Reinforcement learning with maskable stock representation for portfolio management in customizable stock pools. In *Proceedings of the ACM Web Conference*, 2024.

Zihao Zhao, Christopher Yeh, Lingkai Kong, and Kai Wang. Diffusion-DFL: Decision-focused diffusion models for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*, 2026.

AI USE STATEMENT

In this work, we used generative AI tools for polish the presentation of text and coding assistance. We have not used generative AI tools for other tasks with required disclosure. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

A DATA AND BACKTEST PROTOCOL

A.1 DATASETS AND ASSET UNIVERSES

The experiments use daily adjusted prices for constituents of the S&P500 and CSI300 indices observed on 1 January 2024. Within each index, securities are ranked by market capitalization on this reference date, and the top 50 and top 100 securities form the four fixed experimental universes:

S&P50, S&P100, CSI50, and CSI100. The index names identify the source universe; the numerical suffix denotes the number of assets retained for the backtest. Asset order and universe membership are fixed within each panel for all methods.

A.2 Backtest settings and missing-value handling

The training set covers January 2013–August 2023, the validation set covers September 2023–August 2024, and the test set covers September 2024–February 2025. Forecast origins whose holding horizon crosses a split boundary are removed. Model selection and preprocessing use the training and validation sets only; the test set is evaluated after the checkpoint and all hyperparameters have been fixed.

Missing-value handling. Missing observations are handled per asset and date rather than by dropping a whole market date. A quotation gap retains its observed-history status and is represented by the corresponding validity and quotation indicators. When a price is carried across a quotation gap to maintain the valuation series, the gap contributes zero valuation return; the next newly observed quotation determines the subsequent valuation change. Historical moments and normalization statistics use only valid observations available at the decision date. An asset without a valid current valuation is not tradable at that rebalance date and is excluded from the portfolio account. No future value is used to fill a missing observation, and missingness never changes the fixed asset order or the split boundaries.

All numerical preprocessing scales are fitted on the training set and then held fixed for validation and test evaluation. The backtest uses a long-only, fully invested portfolio, the common rebalance calendar within each panel, and the transaction-cost convention specified in the main experiment setup.

A.3 Evaluation metrics

Let r_t be the daily net portfolio return, V_t the corresponding net asset value, T the number of evaluation sessions, and Δ the elapsed calendar days. With $\bar{r}$ and s_r denoting the sample mean and sample standard deviation of $r_{1:T}$, the reported metrics are

$$\mathrm{CAGR} = \left(\frac{V_T}{V_0}\right)^{252/\Delta} - 1, \qquad \text{compound annual growth rate,} \tag{21}$$

$$\mathrm{ASR} = \sqrt{252}\,\frac{\bar{r}}{s_r}, \qquad \text{annualized Sharpe ratio with zero risk-free rate,} \tag{22}$$

$$\mathrm{Vol} = \sqrt{252}\,s_r, \qquad \text{annualized volatility,} \tag{23}$$

$$\mathrm{MDD} = \max_{1\le t\le T}\left(1 - \frac{V_t}{\max_{0\le u\le t} V_u}\right), \qquad \text{maximum drawdown,} \tag{24}$$

$$\mathrm{Cost} = 100\,\frac{C_T}{V_0}, \qquad \text{cumulative transaction cost (\%).} \tag{25}$$

Here C_T is the cumulative transaction cost in account-value units. Turnover is the half-ℓ_1 change in executed stock value divided by pre-trade net asset value, averaged over rebalance dates; the initial entry is fee exempt. Effective holdings are reported as the reciprocal Herfindahl index $N_{\mathrm{eff}} = (\sum_i p_{t,i}^2)^{-1}$, where $p_{t,i}$ are the normalized absolute executed holdings. CAGR, ASR, and Vol are the metrics shown in the main performance tables; MDD, Cost, turnover, and effective holdings are diagnostic quantities.

B METHOD IMPLEMENTATION DETAILS

B.1 PROBABILISTIC SCENARIO GENERATOR

B.1.1 MARKET CONTEXT ENCODER

The Market Context Encoder combines a contemporaneous market snapshot with recent asset trajectories. All encoder windows end at the information date $t-1$. The input feature groups and their purposes are listed in Table 3. The return sequence, validity indicators, and quotation indicators are processed causally; no future observation enters the conditioning representation.

Table 3: Causal inputs to the Market Context Encoder. All statistics end at $t-1$. The snapshot contains seven physical values, seven corresponding validity bits, and seven observation-quality values.

Channels	Definition and purpose
Pool return (1)	Equal-weight log return over eligible assets with valid current valuations; contemporaneous pool context.
Momentum (1)	$\log(P_{t-1,i}/P_{t-121,i})$; fixed 120-session trend.
Lag return (1)	Current valuation log return; latest asset movement.
Historical moments (4)	Mean and population standard deviation over valid returns in each of the 120- and 252-session windows.
Validity (7)	One indicator per physical value; distinguishes unavailable inputs from observed zero values.
Window coverage (4)	Valued-day and quoted-day fractions for each window, using 120 or 252 as the denominator.
Quotation staleness (1)	$\log(1+a_{t,i})/\log(121)$, where $a_{t,i}$ is the number of trading sessions since the latest observed quote as of $t-1$. Together with the current-quotation indicator, it distinguishes fresh quotes from carried valuations; $a_{t,i}=0$ when no prior quote exists.
Current quotation (1)	Whether the asset has an observed quote at $t-1$.
Pool quote coverage (1)	Fraction of eligible assets with a current quote.
Trajectory (360)	Three concatenated 120-session sequences: standardized valuation returns, valuation validity, and quote availability.

The trajectory branch projects the 360-channel sequence to width 128. Two residual blocks use Linear–Dropout(0.1)–ReLU–Linear layers followed by two-head cross-asset attention, and a $128 \to 64 \to 16$ projection produces the trajectory embedding. The snapshot branch projects the 21-dimensional snapshot to 16 dimensions. The resulting context is passed to the conditional diffusion denoiser.

B.1.2 NORMALIZATION AND RETURN TRANSFORMS

This subsection specifies the preprocessing used by the current architecture in Section 3. Training-set-fitted snapshot and allocation-feature scales remain frozen during validation and evaluation. Historical-window statistics are recomputed at each decision using only information available through $t-1$.

Snapshot features. Let $p_{t,i,j}$ denote physical feature j and $m_{t,i,j} \in \{0,1\}$ its validity indicator. For each of the seven physical channels, let $\mathcal{P}_j$ contain its valid observations from the training set. The eligible-pool return contributes once per decision date, while each asset-specific channel contributes once per valid asset–date pair. For a nonempty $\mathcal{P}_j$, the scale is

$$d_j = \max\left(\sqrt{\frac{1}{|\mathcal{P}_j|}\sum_{p\in\mathcal{P}_j} p^2}, 10^{-8}\right). \tag{26}$$

If a channel has no valid training-set observations, we set $d_j = 1$. The transformed feature and complete snapshot are

$$\widetilde{p}_{t,i,j} = 3\,\mathrm{asinh}\left(\frac{p_{t,i,j}}{3d_j}\right) \quad \text{when } m_{t,i,j} = 1, \qquad s_{t,i} = [\widetilde{p}_{t,i}; m_{t,i}; q_{t,i}] \in \mathbb{R}^{21}. \tag{27}$$

Invalid physical entries are set to zero. The seven validity bits and seven quality channels $q_{t,i}$ retain their definitions in Table 3; they are not subjected to the physical-feature RMS/asinh transform. Fixed scales fitted on the training set preserve comparability across dates, while asinh compression moderates extreme magnitudes and preserves sign.

Historical trajectories. Let $\rho_{\ell,i}$ be a historical valuation log return and $v_{\ell,i} \in \{0,1\}$ its validity. The window is $\mathcal{T}_t = \{t-120, \ldots, t-1\}$, and $n_{t,i}^{\mathrm{hist}} = \sum_{\ell \in \mathcal{T}_t} v_{\ell,i}$. We compute the valid-day mean and population standard deviation:

$$\bar{\rho}_{t,i} = \frac{\sum_{\ell \in \mathcal{T}_t} v_{\ell,i}\rho_{\ell,i}}{\max(1, n_{t,i}^{\mathrm{hist}})}, \tag{28}$$

$$d_{t,i}^{\mathrm{hist}} = \left[\frac{\sum_{\ell \in \mathcal{T}_t} v_{\ell,i}(\rho_{\ell,i} - \bar{\rho}_{t,i})^2}{\max(1, n_{t,i}^{\mathrm{hist}})}\right]^{1/2}. \tag{29}$$

For a valid return, the standardized value is

$$\widetilde{\rho}_{t,\ell,i} = \frac{\rho_{\ell,i} - \bar{\rho}_{t,i}}{\max(d_{t,i}^{\mathrm{hist}}, 10^{-8})}. \tag{30}$$

Invalid entries are zeroed. Concatenating the 120 standardized returns, their validity indicators, and their quotation indicators produces $u_{t,i} \in \mathbb{R}^{360}$. The two indicator sequences remain unscaled. This window-wise standardization separates trajectory shape from the physical return moments supplied by the snapshot.

Diffusion targets and inverse transformation. The per-decision scale pools valid valuation returns over eligible assets and the same 120-session historical window:

$$\kappa_t = \max\left(\sqrt{\frac{\sum_{(i,\ell)\in\mathcal{O}_t} \rho_{\ell,i}^2}{\max(1, |\mathcal{O}_t|)}}, 10^{-3}\right), \tag{31}$$

where $\mathcal{O}_t$ contains the valid historical asset–date observations available at decision date t under the missing-value rules in Appendix A.2. The diffusion target is

$$x_{t,0,i,h} = 3\,\mathrm{asinh}\left(\frac{r_{t,h,i}}{3\kappa_t}\right). \tag{32}$$

Target validity masks determine which entries contribute to the denoising loss. The trailing pooled scale supplies a common economic scale across assets while adapting causally to market volatility. Following the final reconstruction from the sampler in Eq. 11, we restore log-return units with

$$\widehat{r}_{t,h,i}^{(k)} = 3\kappa_t \sinh\left(\widehat{x}_{t,0,i,h}^{(k)}/3\right). \tag{33}$$

Terminal simple returns and their distributional summaries are computed after this inverse transformation. The historical scale is fixed for all scenarios at a given decision date, and the inverse map remains differentiable with respect to the generated path.

Distributional decision features. Let $a_{t,i} \in \mathbb{R}^9$ concatenate the four return statistics and five quantiles defined in Section 3.3.1. For each channel j, a positive RMS scale is fitted using scenarios from the selected pretrained forecaster over eligible pairs from the training set $\mathcal{I}_{\mathrm{tr}}$:

$$c_j = \max\left(\sqrt{\frac{1}{|\mathcal{I}_{\mathrm{tr}}|}\sum_{(t,i)\in\mathcal{I}_{\mathrm{tr}}} a_{t,i,j}^2}, 10^{-6}\right), \qquad f_{t,i,j} = \frac{a_{t,i,j}}{c_j}. \tag{34}$$

All nine scales remain frozen during portfolio learning and evaluation, so changes in predictive location or dispersion remain visible on a common scale. The allocator is evaluated only on the tradable asset set defined in Appendix A.2.

Table 4: Conditional diffusion generator configuration.

Component	Configuration
Diffusion states	$S = 200$ with a linear β_s schedule from 10^{-4} to 0.1
Prediction target	v-parameterization
Conditioning	Market Context Encoder representation and sinusoidal timestep embedding
Scenario bank	$K = 50$ Gaussian origin states per decision date
Forecast horizon	$H = 20$ trading sessions
Alignment sampler	Differentiable few-step DDIM with $\eta = 0$ and the configured few-step NFE
Denoiser	Shared temporal U-Net with 64-channel input/output width and 128-channel bottleneck
Residual blocks	Kernel-three convolutions with GroupNorm and SiLU

B.1.3 Conditional Diffusion Generator Details

The generator configuration is summarized in Table 4.

B.2 Uncertainty-Conditioned Allocator

The allocator receives the tradable asset set defined by the backtest rules in Appendix A.2. Its distributional interface and the micro–macro embedding construction are detailed below.

B.2.1 Distributional Feature Interface

The generated log-return path for asset i and scenario k is decoded into the terminal simple return

$$R^{(k)}_{t,i,\theta} = \exp\left(\sum_{h=1}^{H} \widehat{y}^{(k)}_{t,h,i,\theta}\right) - 1. \tag{35}$$

Suppressing t, i, θ, write these returns as $R^{(k)}$. The four location and dispersion statistics are

$$\widehat{\mu} = \frac{1}{K}\sum_k R^{(k)}, \qquad \widehat{\sigma} = \left[\frac{1}{K-1}\sum_k (R^{(k)} - \widehat{\mu})^2\right]^{1/2}, \tag{36}$$

$$\widehat{d} = \left[\frac{1}{K}\sum_k \min(R^{(k)} - \widehat{\mu}, 0)^2\right]^{1/2}, \qquad \widehat{m} = Q_{0.5}\Big(\Big\{|R^{(k)} - Q_{0.5}(\{R^{(j)}\}_j)|\Big\}_k\Big). \tag{37}$$

The five empirical quantiles are

$$\widehat{q}_p = Q_p(\{R^{(k)}\}_{k=1}^K), \qquad p \in \{0.10, 0.25, 0.50, 0.75, 0.90\}. \tag{38}$$

All empirical quantiles use linear interpolation between adjacent order statistics. The nine channels are scaled with the frozen training-set statistics in Appendix B.1.2. The basic feature control retains the first four channels and sets the five quantile channels to zero.

B.2.2 Micro and macro embeddings

Let $f_{t,i}$ denote the normalized feature vector for an asset in the tradable set. A shared MLP produces the micro input representation $h^{(0)}_{t,i} \in \mathbb{R}^{32}$ using Linear–LayerNorm–SiLU–Linear–SiLU. A four-head, pre-normalized self-attention block exchanges information across the asset dimension:

$$h_{t,i} = h^{(0)}_{t,i} + \mathrm{MHA}_i\Big(\mathrm{LN}(h^{(0)}_{t,1:N})\Big). \tag{39}$$

The macro input is the concatenation of the cross-sectional mean and population standard deviation of the first four feature channels,

$$g_t = \psi_{\mathrm{macro}}([\bar{f}_t; s^f_t]) \in \mathbb{R}^{32}, \qquad o_{t,i} = \psi_{\mathrm{score}}([h_{t,i}; g_t]), \tag{40}$$

The bounded score map in Eq. 15 converts $o_{t,i}$ into long-only, fully invested weights.