

AutoQuant: An Auditable Expert-System Framework for Execution-Constrained Auto-Tuning in Cryptocurrency Perpetual Futures

Kaihong Deng

*School of Modern Information Industry, Artificial Intelligence Program, Guangzhou
College of Commerce, Guangzhou, 510555, China*

Abstract

Backtests of cryptocurrency perpetual futures are fragile when they ignore microstructure frictions and reuse evaluation windows during parameter search. Using four liquid perpetual contracts (BTC/USDT, ETH/USDT, SOL/USDT, and AVAX/USDT), we study whether an execution-constrained configuration-selection pipeline can reduce performance overestimation and expose parameter fragility.

We propose AutoQuant, an execution-centric and *alpha-agnostic* framework that can be viewed as an *auditable expert system* for strategy configuration selection. AutoQuant encodes strict execution semantics, funding alignment, and trading-cost/constraint profiles as a knowledge base; applies Bayesian optimization plus a two-stage double screening protocol across windows and cost scenarios; and exports deterministic, machine-readable artifacts and accounting-invariant audits for traceability. AutoQuant does not generate new trading rules; it selects and audits configurations within a pre-specified signal family under strict execution and cost semantics.

Empirically, fee-only “standard” backtests and zero-cost upper bounds materially inflate annualized performance relative to a fully costed configuration that includes funding and slippage. Double screening does not guarantee higher returns, but it tends to select configurations with smaller drawdowns under identical strict semantics. A CSCV/PBO diagnostic [1] indicates substantial residual overfitting risk, supporting our framing of AutoQuant as

Email address: 202312100202@xs.gcc.edu.cn (Kaihong Deng)

auditable validation infrastructure rather than a proof of persistent alpha. Reported returns are computed for small-account simulations under linear trading costs and exclude market impact and capacity constraints.

Keywords: expert systems, decision support, auditability, fintech infrastructure, cryptocurrency derivatives, perpetual futures, backtesting realism, transaction costs, Bayesian optimization, robustness screening, risk management

1. Introduction

Motivation and background.

The proliferation of cryptocurrency derivatives has fundamentally reshaped the digital asset landscape, with perpetual futures contracts emerging as a dominant instrument for both retail and institutional participants. Traded 24/7 on global exchanges, instruments like the BTC/USDT perpetual exhibit high volatility, complex non-linear dynamics, and periodic liquidity crises. The market's microstructure introduces unique challenges for quantitative strategy development, most notably through its intricate cost structure. Unlike traditional equity markets, a strategy's profitability is co-determined by a mix of bilateral trading fees, periodic funding payments that anchor the contract price to the spot index, execution slippage, and stringent constraints on leverage and notional position sizes. The confluence of these factors makes the journey from a research backtest to live trading exceptionally fragile. From an expert-systems and decision-support perspective, the reliability of backtests in such markets directly affects how organizations build, validate, and govern algorithmic trading and risk control components that may be embedded in digital-asset products, automated trading platforms, and DeFi-style protocols.

Within this high-stakes environment, the processes of backtesting and parameter tuning are prone to significant biases. Common methodological pitfalls frequently lead to a substantial overestimation of performance. These include **look-ahead bias**, which arises from the improper use of future information—such as referencing K-line data from beyond the current timestamp or incorporating future funding rates into past performance calculations—and **unrealistic cost assumptions**, where critical factors like funding payments, slippage, and leverage constraints are either ignored or oversimplified. Furthermore, many strategies rely on manual, *ad hoc* parameter tuning, yielding configurations that appear profitable in-sample but fail to

generalize to out-of-sample data or the unforgiving conditions of live markets. This persistent gap between research and reality motivates the development of an implementation-oriented framework designed explicitly to bridge the divide between promising backtests and deployable strategies for cryptocurrency perpetuals.

This paper is organized around three empirical questions in the STRICT4H setting across liquid perpetual contracts: (i) how large can performance inflation become when costs and constraints are removed in a frictionless backtest; (ii) how sensitive are tuned configurations to a validation-style window that is held out from the Stage I search but reused for screening; and (iii) how does cost stress affect conclusions about apparent profitability and robustness. AutoQuant is designed to answer these questions in an auditable way by enforcing strict execution semantics, explicit cost modeling, and a screening protocol that makes window reuse and multiple-testing risk visible rather than implicit. Empirically, we anchor the cost-inflation experiment on BTC/USDT and provide direct one-stage versus two-stage replications on ETH/USDT, SOL/USDT, and AVAX/USDT under the same execution semantics.

Our findings are qualitatively consistent with limits-to-arbitrage arguments [2] in our BTC/USDT perpetual case study. Frictionless backtests can produce abundant high-Sharpe momentum signals, but funding and slippage materially compress these apparent opportunities. We therefore interpret the performance overestimation documented here as case-specific evidence about the role of realistic frictions in backtest validity, not as a structural test of limits to arbitrage. At a high level, our central claim is that in high-friction perpetual markets, execution-constrained and double-screened configuration selection is a prerequisite for auditable backtest evidence; naive one-stage tuning under simplified costs can yield systematically overstated and non-replicable results.

From an expert-systems and decision-support perspective, we view AutoQuant as auditable validation infrastructure for execution-constrained configuration selection in cryptocurrency perpetual futures. The system encodes strict execution semantics, funding alignment, and cost policies as inspectable constraints, and exports deterministic artifacts and accounting invariants to support traceable backtest-to-live consistency.

In this study we ask: within a four-asset BTC/USDT, ETH/USDT, SOL/USDT, and AVAX/USDT STRICT4H case study, to what extent can an execution-centric Bayesian configuration-selection pipeline reduce backtest

overestimation and parameter fragility relative to naive one-stage tuning under simplified costs? We answer in two ways. First, we quantify return inflation across a deliberately optimistic zero-cost upper bound, an intermediate fee-only “standard” setup, and a fully costed configuration that includes funding and slippage. Second, we compare the stability of one-stage versus two-stage selection across multiple windows. Blind hold-outs (post-ETF for BTC; post-2024 for ETH/SOL/AVAX) are treated as auxiliary stress tests and are never used for selection. The remainder of the paper answers this question within this narrowly defined setting and does not claim that the reported magnitudes extend unchanged to other assets, horizons, or strategy families.

Limitations of existing practice.

While numerous open-source frameworks such as Backtrader [3], Zipline [4], Freqtrade [5], and FinRL [6] have democratized algorithmic trading research, their design philosophy often prioritizes ease of strategy prototyping over the enforcement of strict realism. In many equity or low-friction settings, careful users can configure these tools in a way that is broadly consistent with realistic execution and cost assumptions. Our concern is more specific: in high-friction cryptocurrency perpetual markets, and especially when default or highly simplified configurations are used, it is easy to obtain backtests that look attractive on paper but do not survive realistic cost and execution modeling. From a production-oriented perspective, and as documented in their official repositories and user guides, these tools typically leave critical choices about execution semantics and cost realism to the end user. First, they seldom enforce no-look-ahead invariants at the architectural level, leaving the onus on the practitioner to avoid subtle forms of forward-looking information leakage. Second, their cost models are often only partially specified, commonly simplifying costs to a fixed transaction fee while providing at best rudimentary or optional support for the detailed modeling of funding dynamics and leverage constraints. Third, the process of parameter search is frequently decoupled from cost realism and risk management, leading to optimization over an idealized environment that does not, by default, reflect true trading conditions. We do not view the frameworks themselves as flawed; rather, we emphasize the risk that, in the absence of strict guardrails, common usage patterns in high-friction markets can generate a substantial illusion of profitability.

To illustrate the implementation burden, Appendix Appendix A includes a minimal third-party semantics replay in Backtrader: we export the directional signal series from our engine, replay it under cheat-on-close execution, and disable costs and funding to isolate raw-return semantics. Even this narrow

check requires bespoke data plumbing and careful timestamp alignment. Extending a generic framework to a full STRICT4H-style experiment would further require (i) a separate funding-rate time series aligned under a strict $t+1$ convention, (ii) consistent fee, slippage, leverage, and notional-cap constraints, (iii) deterministic export of per-bar return and cost components, and (iv) full-chain accounting invariants that reconcile offline and live-style replays.

A more fundamental limitation is that the majority of prior academic and practitioner work is inherently **alpha-centric**. The primary focus is typically on proposing novel predictive signals or factors, with the validation pipeline treated as a secondary concern. Less attention has been paid to the systematic, automated, and robust generation and validation of the *parameters* that govern these alpha signals. The pipeline itself—the machinery that transforms a raw strategy idea into a vetted, tradable candidate—remains an under-explored and often artisanal domain.

Contributions and positioning.

This paper addresses these gaps by proposing AutoQuant, an execution-centric, *alpha-agnostic* framework that shifts the focus from discovering new predictive signals to the robustness and auditability of the parameterization and validation process in cryptocurrency perpetual futures. In other words, AutoQuant is a configuration-selection system (not a strategy generator): it treats the signal family as fixed and makes the *selection process* auditable. From an expert-systems and decision-support perspective, AutoQuant is positioned as infrastructure for realistic, auditable quantitative research in high-friction digital asset markets: it aims to bridge the gap between research backtests and deployable trading or risk-control systems under realistic cost and execution constraints, rather than to introduce a new alpha. At the architectural level the framework is signal-modular, treating the core signal generation logic as a configurable black box and concentrating on a systematic methodology for finding and verifying durable parameter configurations. In this manuscript, however, all empirical evidence is deliberately restricted to a single STRICT4H momentum-and-gating family, with BTC/USDT as the primary audited case study and parallel replications on ETH/USDT, SOL/USDT, and AVAX/USDT. We treat these as evidence for the auditability and pipeline-level portability of the validation protocol across several liquid perpetual contracts rather than as a claim of universal multi-asset, multi-family validation. Our contributions are:

- An auditable, rule-based decision-support framing for auto-

Table 1: **AutoQuant versus common trading research frameworks (feature matrix).** Entries indicate whether a feature is *native*, *configurable*, or absent (\times). This classification reflects the capabilities described in official documentation and common usage patterns, and is not intended as an exhaustive audit of all possible extensions.

Framework	Funding	T+1 semantics	Scenario-grid stress	Bayesian tuning	Full-chain invariants	Guard overlay	Perp-specific constraints
AutoQuant (this paper)	native	native	native	native	native	native	native
Backtrader [3]	configurable	configurable	\times	configurable	\times	configurable	configurable
Zipline [4]	\times	configurable	\times	configurable	\times	\times	\times
Freqltrade [5]	\times	configurable	\times	configurable	\times	configurable	\times
FinRL [6]	\times	configurable	\times	configurable	\times	configurable	\times

tuning: we make execution semantics, funding alignment, and cost/constraint profiles explicit as a knowledge base and screening policies (an expert-system-like decomposition) rather than as implicit backtest settings.

- **An inference pipeline for configuration selection:** Bayesian search plus a two-stage double-screening protocol that re-evaluates a fixed candidate pool across windows and a small cost-mis-specification grid under strict $t+1$ semantics.
- **Traceable artifacts and consistency checks:** deterministic machine-readable exports and full-chain accounting invariants that reconcile offline backtests with live-style replays, plus a guard overlay concept for post-deployment risk control.
- **Case-study evidence on perpetual futures:** BTC/USDT as an audited anchor with ETH/SOL/AVAX replications quantifying cost-driven performance inflation, window fragility, and residual overfitting risk under identical strict semantics.

Transferable audit principle. We operationalize auditability via (i) strict $t+1$ execution semantics, (ii) no-look-ahead funding alignment, (iii) explicit cost accounting and stress scenarios, (iv) explicit window semantics (including validation reuse), and (v) deterministic artifacts and invariants for end-to-end replay.

Scope and interpretation (to prevent misreading). Throughout the paper we adopt a conservative interpretation of empirical results and make three scope commitments explicit:

- **No alpha claim:** the manuscript’s contribution is an execution-centric

validation and auditing stack, not a new predictive signal or a claim of persistent abnormal returns.

- **No capacity claim:** reported backtest returns are produced under a linear cost model with explicit leverage and notional caps, but *without* non-linear market impact and liquidity constraints; extreme compounded returns (CAGR > 1.0 in decimal units) can occur in small-account simulations and should be read alongside drawdowns and robustness diagnostics, not as scalable profitability.
- **Limited empirical scope:** core evidence is restricted to a single STRICT4H strategy family on four liquid perpetual contracts (BTC/USDT, ETH/USDT, SOL/USDT, AVAX/USDT); we treat cross-asset results as portability checks for the pipeline rather than broad market validation.

Paper organization.

The paper proceeds as follows. Section 2 presents the core methodology of our execution-centric auto-tuning and double-screening framework. Section 3 reviews related literature. Section 4 describes the market setting and the data used in our study. Section 5 details system architecture and implementation. Section 6 reports the empirical results. Section 7 discusses implications and limitations, and Section 8 concludes.

Reproducibility pointer. All core evidence in this manuscript is generated from deterministic machine-readable artifacts produced by the pipeline and plotted via `scripts/paper_assets/make_figures.py`. Appendix Appendix A lists minimal scripts and commands to regenerate the reported tables and figures, subject to the data-access constraints described there.

Timestamp convention. All date ranges in this manuscript are expressed in UTC and refer to the *bar-open timestamp* used to index the OHLCV series (e.g., a 4-hour bar starting at 2019-09-08 16:00:00+00:00 is indexed by that start time). Under STRICT4H, signals are computed at bar close and, by design, take effect from the subsequent bar ($t+1$ execution).

2. Methodology: Execution-Centric Auto-Tuning and Double Screening

This section details the core methodology of our framework, which is designed to systematically find and validate robust parameter configurations

for quantitative strategies in a signal-modular, execution-centric manner. The process is divided into two primary stages: a broad parameter search using Bayesian optimization, followed by a rigorous double-screening validation protocol, and is complemented by a post-deployment risk management layer. **Expert-system decomposition.** For readers from the expert-systems and applied AI community, AutoQuant can be viewed as a rule-based, knowledge-driven decision-support system (an expert-system-like decomposition) for strategy configuration selection. The knowledge base encodes execution semantics, funding alignment, trading costs, and feasibility constraints as non-negotiable rules, together with explicit screening policies (e.g., scenario-grid thresholds). The inference engine combines Bayesian search and policy-driven screening to recommend a small set of stable configurations. The explanation interface exports deterministic artifacts and accounting-invariant audits to support traceability and backtest-to-live consistency.

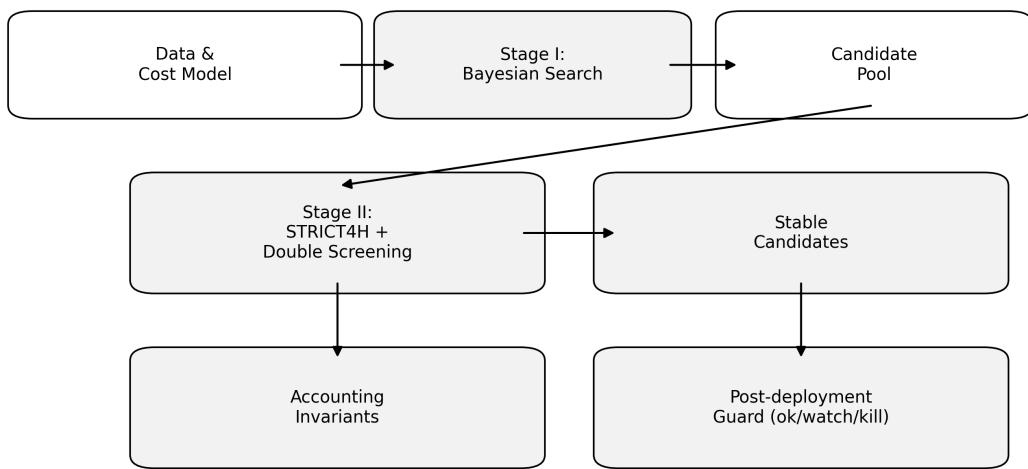
We use the term “expert system” in the ESWA sense of explicit, auditable rules and policy-driven recommendations, and we do not claim novelty in symbolic reasoning or knowledge representation beyond this disciplined, traceable decomposition.

Concrete examples of knowledge-base rules include: (i) a strict $t + 1$ execution delay (signals computed at bar close take effect only from the next bar); (ii) no-look-ahead funding alignment (funding values are carried forward only after their timestamp and are never backfilled from future updates); (iii) leverage and notional caps enforced by construction; and (iv) an explicit cost-scenario grid and stable-candidate thresholds that are applied uniformly across candidates.

Figure 1 provides a high-level overview of the two-stage workflow and its audit-oriented interfaces (window splits, scenario grids, and stable-candidate selection).

Table 2: **ESWA-aligned decision-support mapping of AutoQuant components.** The table makes the system decomposition explicit: a knowledge base that codifies execution and cost rules, an inference engine that performs search and screening, and an explanation/governance interface that exports auditable artifacts for traceability and consistency checks.

ESWA component	AutoQuant realization	Decision-support / audit output
Knowledge base	STRICT4H execution semantics; funding alignment; cost/constraint profiles; screening policies	Declarative rules that prevent look-ahead and enforce feasibility; explicit cost-scenario grid and window mantics.
Inference engine	Stage I Bayesian optimization (TPE) + Stage II policy-driven double-screening	Stable configuration recommendation with robustness diagnostics across windows and cost-mis-specification scenarios.
Explanation interface	Deterministic machine-readable exports; ac-counting invariants; full-chain audits	Traceable artifacts for table/figure generation and backtest-to-live consistency checks.
Governance overlay	Live guard triggers + invariant monitoring	Run-time supervision signals and reconciliation deltas to detect drift and implementation inconsistencies.



STRICT4H enforces T+1 execution and no-look-ahead funding decisions.

Figure 1: **AutoQuant methodology flowchart.** Stage I Bayesian search produces candidates that undergo Stage II double-screening; stable sets are supervised by a live guard overlay. STRICT4H enforces $t+1$ execution and no-look-ahead funding alignment.

2.1. Algorithmic Summary (Pseudo-code)

To make the end-to-end procedure directly reproducible from the text, we summarize the pipeline as a minimal interface and pseudo-code. Inputs are (i) a time-aligned 4-hour OHLCV series, (ii) an 8-hour funding-rate series aligned to the STRICT4H grid, (iii) a fixed cost profile and a predefined cost-scenario grid, (iv) a window split (Training/Validation/Long-horizon/Blind hold-out), and (v) a Stage I search budget N_{opt} . Outputs are the selected configuration, the backtest logs, and the exported machine-readable artifacts described in Appendix Appendix A.

```
# Stage I: Bayesian search on Training (objective: annualized net return)
history = []
for i in 1..N_opt:
    theta_i ~ TPE(history)
    r_net = STRICT4H_backtest(
        data=Training, theta=theta_i, costs=baseline_costs
    )
    score_i = annualized_return(r_net)
    history.append((theta_i, score_i))
Theta_pool = top-K candidates by score_i

# Stage II: double-screening (no re-optimization; Validation reused for selection)
for theta in Theta_pool:
    for scenario in cost_grid:
        r_net = STRICT4H_backtest(
            data=Training+Validation, theta=theta, costs=scenario
        )
        compute robustness diagnostics (monthly returns, drawdown, turnover)
select theta_star via ex-ante filters + aggregated scenario metrics

# Stress tests (Blind hold-out only; not used for selection)
evaluate theta_star on Long-horizon and Blind hold-out windows
export tabular summaries
```

In our implementation, the “export tabular summaries” step is backed by deterministic machine-readable outputs (e.g., configuration json, per-bar ledger csv, robustness summaries, and invariant-audit files) that allow independent regeneration of the manuscript tables/figures and backtest-to-replay reconciliation, as summarized in Appendix Appendix A.

2.2. The Signal-Modular Parameter Space

Our framework deliberately treats the underlying alpha-generation logic as a black-box mapping

$$f : \mathcal{I}_t \times \Theta \rightarrow S_t,$$

where \mathcal{I}_t denotes the information set available at time t , Θ the hyperparameter space, and $S_t \in \{-1, 0, 1\}$ the target position or trading signal. In the STRICT4H case study, the core momentum signal is derived from the normalized divergence between fast and slow Exponential Moving Averages (EMAs) of the 4-hour close, compressed into $[0, 1]$ via a smooth tanh transform and combined with auxiliary mean-reversion and breakout channels. The details of this construction are implementation-specific; in this section we focus on the structure and constraints of Θ rather than on the functional form of the alpha engine itself.

While the AutoQuant architecture is signal-agnostic—meaning it can wrap any signal-generating function $f(\cdot)$ —the specific hyperparameter search space Θ is optimized for trend-following and mean-reversion families on cryptocurrency perpetual futures. These structures reflect the dominant algorithmic flows we observe in this market. Extending the framework to high-frequency market-making or arbitrage strategies would require redefining Θ and the associated cost-model primitives, although the validation pipeline in Sections 2.4 and 5 would remain unchanged. Importantly, AutoQuant does not attempt to invent f automatically in this manuscript; the novelty is to enforce strict semantics and provide an auditable selection and screening pipeline for $\theta \in \Theta$.

Momentum score:

$$m_t(\theta_{\text{momentum}}) = \frac{1}{2} \left[1 + \tanh \left(\frac{\text{EMA}_{\text{fast},t} - \text{EMA}_{\text{slow},t}}{\sigma(\theta_{\text{momentum}}) \text{EMA}_{\text{slow},t}} \right) \right]. \quad (1)$$

Here $\theta_{\text{momentum}} \geq 0$ is a volatility-scaling hyperparameter and $\sigma(\theta_{\text{momentum}})$ is a smooth, monotonically increasing scaling function that maps θ_{momentum} to the same units as the EMA spread, ensuring that $m_t(\theta_{\text{momentum}})$ remains in $[0, 1]$.

Composite score:

$$C_t = w_{\text{mom}} \cdot m_t + (1 - w_{\text{mom}}) \cdot a_t. \quad (2)$$

Here $w_{\text{mom}} \in [0, 1]$ is a fixed weight and $A_t = \{a_{t,1}, \dots, a_{t,K}\}$ denotes the collection of enabled mean-reversion (anomaly) sub-scores at time t , each

mapped by a monotone squashing function to a unit-free range in $[0, 1]$ (e.g., based on Bollinger-band deviations and/or tail-quantile exceedances). We aggregate these sub-scores into a single scalar anomaly score $a_t := \max_k a_{t,k}$, aligned with our implementation.

Signal generation:

$$\text{Signal}_t = \begin{cases} +1, & C_t \geq \tau_t^{\text{long}}, \\ -1, & C_t \leq 1 - \tau_t^{\text{short}}, \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Signal & Risk Parameters. Instead of enumerating implementation-specific variable names, we define the hyperparameter space Θ in terms of a small number of interpretable blocks. The momentum and mean-reversion block controls the EMA lookback lengths, volatility-scaling factors, and Bollinger-band widths and tail quantiles that determine how quickly the strategy reacts to trends or reversals. The risk-control block specifies minimum holding times, cooldown periods after exits, ATR-based stop-loss and take-profit multipliers, and several exposure-smoothing and maximum-exposure constraints. The Bayesian optimization stage explores Θ within fixed bounds chosen to respect realistic leverage, exposure, and cost constraints. Bounds are set *ex ante* based on exchange-rule feasibility and common practitioner ranges, and are not tuned on a per-window basis.

Funding & Cost Parameters. To mitigate adverse funding costs, we introduce a soft funding-bias mechanism that raises entry thresholds in periods of expensive carry. Let fr_t denote the 8-hour funding rate (in decimal units) observed from an external funding series and aligned to the 4-hour grid via carry-forward, and let τ_{base} be a baseline long-entry threshold. For long positions we define an effective threshold

$$\tau_{\text{long}}(t) = \tau_{\text{base}} + \kappa \cdot \max(|fr_t| - \vartheta_{\text{bias}}/10000, 0), \quad (4)$$

where $\vartheta_{\text{bias}} \geq 0$ is an activation threshold (expressed in basis points; we divide by 10,000 to match fr_t 's decimal units) and $\kappa \geq 0$ controls how quickly the threshold increases as funding moves deeper into the tails (in threshold units per 1.0 funding-rate decimal). An analogous expression applies to the short side, with the sign of fr_t reversed in the adverse-direction case. When $\kappa = 0$ the funding-bias channel is effectively disabled, and the backtest reduces to a pure price-based strategy with realistic but funding-agnostic trading costs.

The optimization process in Stage I is guided by a well-defined objective function. The goal is to maximize a single scalar objective dominated by the cost-adjusted annualized return. While other performance metrics such as the Sharpe ratio, maximum drawdown, and win rate are computed concurrently and can be combined into a fixed-weight composite score for diagnostic purposes, the Bayesian optimization process itself is not a multi-objective one. The trade-offs between risk and return are primarily addressed through the subsequent robustness screening in Stage II, rather than by treating metric weights as tunable hyperparameters.

Formally, Stage I solves

$$\max_{\theta \in \Theta} J(\theta), \quad J(\theta) := r_{\text{ann}}(\theta), \quad (5)$$

where $r_{\text{ann}}(\theta)$ is the annualized net return computed from the STRICT4H net-return series under the Stage I window and the baseline cost profile (defined below in Eq. (6)).

Performance metrics.. We define the primary performance metric, annualized return r_{ann} , based on the geometric compounding of the 4-hour strategy returns over the entire backtest period. Let R_{tot} denote the total compounded return of the strategy over N 4-hour bars and let AF be the annualization factor (set to 2190, i.e., 6 periods per day \times 365 days per year). We compute

$$r_{\text{ann}} = (1 + R_{\text{tot}})^{AF/N} - 1. \quad (6)$$

Unless otherwise stated, we report annualized return as CAGR in decimal units (e.g., $0.10 = 10\%$). For readability, we also report CAGR in percentage units in our empirical tables and figures. Maximum drawdown (MaxDD) is reported as the peak-to-trough equity decline over the window, expressed as a fraction in $[0, 1]$ (e.g., $0.30 = 30\%$).

2.3. Stage I: Bayesian Auto-Tuning under Realistic Costs

The first stage of our framework employs Bayesian optimization to efficiently search the vast parameter space. We utilize the Tree-structured Parzen Estimator (TPE) algorithm [7], as implemented in the Optuna framework [8], to guide the search. The process begins by defining a wide range for each parameter, allowing the optimizer to explore diverse regions of the configuration space. In each trial, the optimizer suggests a new parameter set, a backtest is executed, and the resulting scalar objective function value is

reported back. The TPE algorithm uses this history to build a probabilistic model, progressively focusing the search on more promising regions that are likely to yield high cost-adjusted performance.

Why TPE (and why not grid / evolutionary / GP-BO)? We select TPE because our search space Θ contains mixed continuous/discrete parameters, conditional blocks (many knobs are enabled/disabled), and hard feasibility constraints. In this setting, the TPE family is a practical and widely used sampler for conditional search spaces [7, 8] and performs well in compute-limited regimes where only a modest number of strict backtest evaluations is feasible. By contrast, grid search scales poorly with dimension and discretization choices; random search is a strong baseline for hyperparameter optimization under limited budgets [9]; evolutionary methods can be effective but often require larger evaluation budgets and more careful constraint handling [10–12]; and Gaussian-process Bayesian optimization is less convenient for high-dimensional conditional spaces. Importantly, AutoQuant is sampler-agnostic: any black-box optimizer that proposes θ and consumes scalar scores can be substituted without changing the strict execution/cost semantics or the Stage II screening policies.

In the BTC/USDT STRICT4H case study reported in this manuscript, we fix the Stage I search budget *ex ante* at $N_{\text{opt}} = 40$ TPE trials on the training window. This budget is deliberately modest and is not intended to exhaustively explore the strategy space; it is used to generate a small but diverse candidate pool whose robustness is then tested in Stage II.

We use annualized net return as the Stage I scalar objective (see the pseudo-code in Section 2.1) to keep the optimization problem simple and to avoid embedding additional risk preferences into the sampler. Risk and fragility controls (drawdown and turnover caps, worst-month floors, and scenario-aggregated stability criteria) are enforced in Stage II as explicit, auditable policy constraints (see Section 2.4). Alternative objectives (e.g., Sharpe-based or multi-objective formulations) are compatible with the pipeline but are not the focus of this manuscript.

A key design choice in our case study is that Stage I and Stage II evaluations share the same STRICT4H execution and funding-alignment semantics; Stage I differs only in its role (Training-only objective under a baseline cost profile) while Stage II re-evaluates a fixed candidate pool under additional windows and cost-mis-specification scenarios.

A critical feature of this stage is that the backtest engine already incor-

porates a realistic trading-cost model (Section 4.4). In our implementation this component is named `RealisticTradingCosts`. This ensures that the optimization is grounded in a realistic assessment of market friction from the outset, preventing the discovery of spurious strategies that are only profitable in a frictionless environment. However, despite this realism, Stage I is conducted on a single, primary data window and cost assumption. It is therefore still susceptible to overfitting, necessitating a subsequent, more rigorous validation stage. The output of Stage I is a large pool of high-performing candidate parameter sets, ranked by their objective function scores.

Budget note. For the manuscript’s direct comparison tables, we use a higher Stage I budget (120 trials) to avoid underpowered “best-of-40” comparisons. We retain $N_{\text{opt}} = 40$ as a compute-limited baseline for the deflated Sharpe ratio (DSR) diagnostic and for illustrating how the pipeline behaves under modest budgets; the replication package includes both budgets and their corresponding best-so-far curves.

Optimizer baseline. As a minimal baseline under the same STRICT training window and objective, we compare the best-of- N curve of TPE against random sampling [9] in Table 5.

Table 3: **Minimal STRICT4H hyperparameter summary (Θ) with default bounds and fallbacks.** Ranges correspond to the default Stage I Optuna search space (Appendix Appendix A replication code). “Default” denotes the fallback value used when a parameter is omitted; many risk knobs default to 0 (disabled) and are enabled only when explicitly set by the tuned configuration.

Block	Parameter	Range (default)	Constraint / interpretation
Momentum	<code>ema_fast</code>	6–32 (12)	EMA span in 4h bars; must be shorter than <code>ema_slow</code> .
Momentum	<code>ema_slow</code>	20–96 (26)	EMA span in 4h bars; <code>ema_fast < ema_slow</code> .
Momentum	<code>ema_threshold</code>	0.0005–0.0060 (0.003)	Entry threshold on EMA spread (decimal units).
Mean-rev	<code>bb_period</code>	10–30 (20)	Bollinger lookback in 4h bars.
Mean-rev	<code>bb_dev</code>	1.0–2.5 (2.0)	Band width in standard deviations.
Risk	<code>min_hold_bars</code>	1–6 (1)	Minimum holding duration in 4h bars.
Risk	<code>cooldown_hours</code>	0–8 (0)	Cooldown after exits to reduce rapid flip-flopping.
Risk	<code>atr_period</code>	0–30 (0)	ATR period in 4h bars; 0 disables ATR-based exits.
Risk	<code>atr_k_sl</code> , <code>atr_k_tp</code>	0–3 / 0–5 (0)	ATR stop-loss / take-profit multipliers; 0 disables.
Risk	<code>max_exposure_abs</code>	0–5 (0)	Account-level exposure cap (in notional equity multiples); 0 disables.
Funding	<code>funding_bias_thr</code> <code>_bps</code>	0–10 (0)	Soft threshold ϑ_{bias} (bps) before raising entry thresholds under adverse funding.
Funding	<code>funding_bias_k_t</code> <code>hr_per_bps</code>	0–0.005 (0)	Slope κ : threshold increase per 1.0 funding-rate decimal above ϑ_{bias} (converted to decimal).

Table 4: **Stage I optimization best-so-far curve versus trial budget (BTC/USDT training).** The score is the Stage I scalar objective (annualized net return under strict execution and the baseline cost profile) and is reported in decimal units. Values are taken from the machine-readable replication artifacts.

Budget (trials)	Best score so far	Best trial ID so far
10	0.0174	7
20	0.0362	14
40	0.1172	38
60	0.1334	53
80	0.1508	67
100	0.1590	85
120	0.1590	85

Table 5: **Optimizer baseline under identical STRICT training window.** Best-of- N objective (annual). Random values are mean \pm sd over seeds.

Budget N	TPE best	Random best (mean \pm sd)
10	0.0174	-1.0000 \pm 0.0000
20	0.0362	-0.7832 \pm 0.4848
40	0.1172	-0.7832 \pm 0.4848
60	0.1334	-0.7832 \pm 0.4848
80	0.1508	-0.6638 \pm 0.4915
100	0.1590	-0.4600 \pm 0.5163
120	0.1590	-0.3580 \pm 0.6763

Compute budget disclosure. The dominant cost driver is the number of rigorous STRICT4H backtest evaluations. Stage I runs $N_{\text{opt}} \in \{40, 120\}$ TPE trials on the Training window (one STRICT4H backtest per trial). Stage II re-evaluates a fixed candidate pool across a small cost-scenario grid and window segments; in our reproducible BTC snapshot, the screening applies 9 cost scenarios to the top 40 candidates over the combined Training+Validation window (7,262 4-hour bars), totaling 360 strict backtest runs, plus a small number of long-horizon and blind hold-out stress-test evaluations. All experiments are CPU-only (no GPU required). On a representative workstation (Windows 11; AMD Ryzen 9 7945HX, 16C/32T), end-to-end runtime is typically hours to under one day per asset under these budgets.

2.4. Stage II: Rigorous 4h Backtest and Double Screening

The candidate parameter sets from Stage I are then passed to the second stage for a rigorous “double-screening” validation. This stage utilizes our STRICT4H backtest engine, which is architected to eliminate common sources of backtest inflation and ensure consistency with live execution. The key features of this engine include:

Strict T+1 Execution: A signal generated at the close of 4-hour bar t , using all information available up to that point, only affects the strategy’s position and profit-and-loss (PnL) calculation starting from the subsequent bar, $t+1$. This strictly prohibits intra-bar execution and prevents look-ahead bias.

No-Look-Ahead Funding Model: Funding costs are calculated and applied on a bar-by-bar basis. Funding rates are treated as an external time series and are aligned to the 4-hour grid via carry-forward. The backtest uses only the funding values available up to each bar timestamp (no backfilling from future funding updates), scaled by the bar’s duration (4 hours). The engine is architecturally prevented from peeking into the future path of funding rates.

Funding-gate ablation. To isolate whether funding-aware alignment rules matter beyond simply charging funding as a cost, we keep STRICT execution and all costs fixed and disable only the funding-related gating/bias rules. Table 6 reports the resulting change in performance for a representative tuned configuration.

To make the execution semantics explicit, let $S_t \in \{-1, 0, 1\}$ be the directional signal computed at the close of bar t using only \mathcal{I}_t . STRICT4H enforces a strict $t+1$ delay by applying an *executed* signed nominal exposure π_t during bar t that is computed from the previous bar’s signal S_{t-1} after

Table 6: **Funding gate ablation (trial 109).** Costs and STRICT execution are held fixed; only funding-related gating/bias rules are disabled.

Variant	Train monthly geom	Train MaxDD	Val monthly geom	Val MaxDD
Full	0.089	0.331	0.004	0.365
NoFundingGates	0.067	0.331	0.004	0.365

exposure caps/smoothing and other fixed (ex ante) risk controls. Given the close-to-close market return r_t^{mkt} in Eq. (9), the raw strategy return is $r_t^{\text{raw}} = \pi_t r_t^{\text{mkt}}$. Transaction costs are applied on exposure changes at bar boundaries, $\Delta\pi_t = \pi_t - \pi_{t-1}$, with fees and slippage proportional to $|\Delta\pi_t|$. Funding costs are accrued on the notional exposure π_t using the aligned 8-hour funding-rate series (scaled to the bar length). We therefore define the per-bar net return as

$$r_t^{\text{net}} = r_t^{\text{raw}} - C_{\text{fee},t} - C_{\text{slip},t} - C_{\text{fund},t}. \quad (7)$$

All reported performance metrics in Section 6 are computed from the resulting $\{r_t^{\text{net}}\}$ series unless otherwise stated.

The double-screening protocol consists of two parallel validation streams applied to a fixed pool of Stage I candidates; there is no further re-optimization in this stage:

Held-Out Window Robustness: To test for temporal stability, the performance of each candidate parameter set is re-evaluated across multiple, overlapping rolling windows of fixed length (2000 bars in our case study, stepped forward by 500 bars) that are held out from the Stage I optimization but may be reused within Stage II. This process is distinct from a traditional walk-forward optimization, as we do not re-train on each window; rather, we apply the same fixed parameter set to different time periods to assess its consistency.

Cost Scenario Robustness: The same parameter sets are re-evaluated under a bounded set of cost-mis-specification profiles. In the current implementation, this stream has two components. First, for stable-candidate filtering we use a discrete sensitivity grid that varies taker fees and funding multipliers ($\text{taker_bps} \in \{3, 4, 6\}$ and $\text{fund_mult} \in \{0.5, 1.0, 1.5\}$), producing scenario-aggregated monthly and drawdown summaries for each candidate. This grid intentionally includes both optimistic and pessimistic perturbations around a conservative baseline to make cost sensitivity explicit. Second, as an

Table 7: **Core STRICT4H symbols, units, and accounting bases.** Unless stated otherwise, returns and costs are in decimal units; bps inputs are converted by dividing by 10,000.

Symbol	Units / range	Meaning
S_t	$\{-1, 0, 1\}$	Directional signal computed at the close of bar t .
π_t	unitless, bounded	Executed signed nominal exposure held during bar t , applied with a strict $t+1$ delay from S_{t-1} after fixed caps/smoothing and other risk controls.
r_t^{mkt}	decimal	Close-to-close market return of the underlying over bar t .
$C_{\text{fee},t}, C_{\text{slip},t}$	decimal	Turnover-based trading costs applied at bar boundaries, proportional to $ \Delta\pi_t $ (bps inputs apply to notional turnover).
$C_{\text{fund},t}$	decimal	Funding cost accrued on notional exposure π_t using the aligned 8-hour funding-rate series (scaled to 4-hour bars).

additional stress test on selected configurations, we report pessimistic-cost scaling scenarios that multiply transaction-related costs (fees and slippage) by fixed factors (e.g., $2\times$ and $3\times$), together with a funding-off stress. These stress tests probe the sensitivity of conclusions to increased market friction while keeping the execution semantics fixed.

The selection of “stable candidates” from this stage is a multi-step process. First, we apply a set of absolute robustness thresholds to filter out underperforming configurations. In the BTC STRICT4H experiments, these hard-coded rules include minimum average monthly returns, a floor for the worst-performing month, a ceiling on average maximum drawdown, and limits on position-switching frequency. Second, the surviving candidates are ranked based on their average monthly return (descending) and average maximum drawdown (ascending) on the training-plus-validation regime. Finally, we select the top- K configurations from this ranked list to form the final pool of stable BTC/USDT STRICT4H candidates.

To make these rules auditable, Table 8 reports the exact thresholds used in our stable-candidate filter. In the replication code, these rules are implemented as an ex-ante filter over scenario-aggregated performance summaries (followed

by a simple lexicographic ranking by mean performance and drawdown), and they are not tuned post hoc on a per-experiment basis.

Metric choice note. We rank stable candidates using monthly geometric mean, drawdown, and switching frequency because these are directly interpretable and robust to extreme compounding in high-volatility regimes. Other stability metrics (e.g., Sharpe-, Sortino-, or Calmar-style summaries) can be substituted in the same policy layer; a systematic comparison of screening metrics is left as future work.

Stable-candidate statistics and aggregation. For a candidate parameter set and a cost scenario s from the predefined grid, we compute a net return series $\{r_t^{\text{net}}\}$ and resample it into calendar-month compounded returns $R_{m,s} = \prod_{t \in m} (1 + r_{t,s}^{\text{net}}) - 1$. We then compute a scenario-level monthly geometric mean $g_s = \exp(\frac{1}{M} \sum_m \log(1 + \max(R_{m,s}, -0.999999))) - 1$, and report $\text{monthly_true} = g_s$. The stable-candidate filter aggregates these scenario-level statistics across the scenario grid: $\text{mean_monthly_true} = \mathbb{E}_s[g_s]$, $\text{min_monthly_true} = \min_s g_s$, and $\text{maxDD_mean} = \mathbb{E}_s[\text{maxDD}_s]$. The position-switch density is computed per scenario as $\text{switch_density}_s = \#\{\Delta\pi_t \neq 0\}/N$, and we report $\text{switch_density_mean} = \mathbb{E}_s[\text{switch_density}_s]$. These definitions match the exported robust-summary files referenced in Appendix A.

2.5. Live Guard and Accounting Invariants

The final component of our methodology is a post-deployment layer designed to promote consistency and help manage risk in a live trading environment. This layer is explicitly separated from the Stage I and II optimization and backtesting processes and acts as a post-deployment overlay.

First, we define a set of full-chain accounting invariants. These are mathematical identities that must hold, within a small tolerance, across the entire lifecycle of a strategy instance. For any given parameter set, key metrics such as cumulative exposure, total fees paid, total slippage incurred, net funding costs, and total PnL must reconcile between the offline backtest, a paper trading simulation, and the logs from a paper-trading or execution run produced by the same code path. These invariants serve as a powerful tool for auditing system integrity and detecting divergences in accounting.

Concretely, for each bar t we record the raw strategy return r_t^{raw} and cost components $C_{\text{fee},t}$, $C_{\text{slip},t}$, and $C_{\text{fund},t}$, and define the net return

$$r_t^{\text{net}} = r_t^{\text{raw}} - C_{\text{fee},t} - C_{\text{slip},t} - C_{\text{fund},t}. \quad (8)$$

Table 8: Stable-candidate selection rules used in the BTC/USDT STRICT4H case study. Statistics are aggregated across the predefined cost-scenario grid described in Section 2.4. After filtering, we select the top $K = 5$ candidates by sorting `mean_monthly_true` descending and `maxDD_mean` ascending.

Statistic	Meaning	Threshold	Role
<code>mean_monthly_true</code>	mean monthly geometric return (across scenarios)	≥ 0.005	minimum performance
<code>min_monthly_true</code>	worst-case scenario-level monthly geometric return	≥ 0.0	downside floor
<code>maxDD_mean</code>	mean maximum drawdown (across scenarios)	≤ 0.30	drawdown cap
<code>switch_density_mean</code>	mean position-switch density (across scenarios)	≤ 0.12	turnover/fragility cap

Full-chain audits verify that these components reconcile (within a small tolerance) across offline backtests, live-style replay, and execution logs produced by the same code path.

In the replication snapshot used for this manuscript, we validate these invariants on representative configurations (two BTC STRICT4H baselines and one ETH STRICT4H aggressive line) and observe exact agreement at the bar level for the exported signal and exposure series (i.e., `diff_signal` and `diff_exposure` are identically zero in the full-chain audits). We report these audits as internal consistency checks; they do not constitute evidence of long-horizon live performance. Table 9 summarizes representative full-chain audit results for BTC/USDT and ETH/USDT from the replication package.

Table 9: **Full-chain audit snapshot (offline backtest vs live-style replay).** We report maximum absolute differences in the exported signal and exposure series, together with accounting invariant diffs (all as backtest-minus-live). Values are generated from the STRICT4H fullchain logs described in Appendix A and are intended as system-consistency evidence, not as a live performance claim.

Asset	Bars	$\max \Delta S $	$\max \Delta \pi $	<code>trades_diff</code>	<code>fees_diff</code>	<code>slip_diff</code>	<code>fund_diff</code>
BTC/USDT	8000	0	0	0	0	0	0
ETH/USDT	8299	0	0	0	0	0	0

Second, we conceptualize a live guard mechanism responsible for real-time risk supervision. This guard continuously monitors a dashboard of rolling performance metrics, such as 30-day and 90-day annualized returns, peak-to-

trough drawdown from recent highs, maximum daily loss, and trade frequency. Based on predefined rules, the guard can issue one of three decisions: ok, watch, or kill. A “kill” decision triggers an automatic flattening of the strategy’s position and, depending on the configured mode, can either temporarily or persistently disable further trading until an explicit manual resume is issued. This mechanism acts as a final safety net that is designed to help contain strategy risk when market conditions deviate significantly from those observed in historical data.

From an empirical standpoint, the current evidence for the live guard is deliberately limited. We verify that full-chain accounting invariants hold to within tight tolerances when replaying STRICT4H logs from backtests and from a small number of paper-trading or limited live runs, and we support the design with trade-level guard simulations on historical data. However, we do not report a broad, multi-year live track record for the guard itself. Throughout this paper we therefore treat the live guard as a post-deployment risk-control pattern with preliminary, configuration-specific evidence rather than as a fully validated production risk-management system.

2.6. Multiple-Testing Adjustment via Deflated Sharpe Ratio

While the bulk of our robustness analysis relies on walk-forward evaluation and trading-cost stress scenarios, we also report a simple multiple-testing adjustment for our flagship STRICT4H configuration. Following [13], we compute a normal-approximation deflated Sharpe ratio (DSR-style) using the annualized Sharpe estimated from the long 4-hour BTC/USDT series (2019-09-08–2025-10-14) and an estimate of the effective number of trials generated by the AutoQuant pipeline. Concretely, we approximate the total number of effective trials as $N_{\text{total}} \in [N_{\text{opt}} \times N_{\text{windows}}, N_{\text{opt}} \times N_{\text{windows}} \times N_{\text{scen}}]$, where N_{opt} is the number of Stage I configurations and N_{windows} is the number of rolling evaluation windows used in Stage II. The upper bound additionally counts the N_{scen} cost-scenario grid evaluations used in stable-candidate screening as distinct selection degrees of freedom. In our reproducible case study we use $N_{\text{opt}} = 40$ and a 6000-bar walk-forward grid with 2000-bar windows stepped by 500 bars ($N_{\text{windows}} = 9$) and a 9-scenario grid ($N_{\text{scen}} = 3 \times 3 = 9$), yielding $N_{\text{total}} \in [360, 3240]$. In the replication materials we provide helper code for recomputing the same statistic under alternative trial-count assumptions. The resulting DSR-style statistic is interpreted *only* as an internal sanity check, as the calculation relies on stylized assumptions about return independence, approximate normality, and a closed-form approximation. In the current

reproducible snapshot, the long-window Sharpe is close to zero, and the DSR-style diagnostic does not suggest that Sharpe alone provides strong evidence against a zero-true-Sharpe null after accounting for multiple trials. We therefore place greater weight on the structural safeguards provided by our walk-forward and cost-stress experiments.

Trial budget and overfitting diagnostics.. To address the concern that $N_{\text{opt}} = 40$ may be insufficient, we report a best-so-far objective curve versus trial budget for the BTC Stage I study in the replication package. In this reproducible snapshot, the best score improves materially beyond 40 trials and plateaus only around 100–120 trials, implying that a 40-trial budget should be viewed as a compute-limited setting rather than as a converged optimum. We therefore interpret our empirical results as evidence about sensitivity and auditability, not as a claim of global optimality within Θ .

We also add an explicit multiple-testing diagnostic based on combinatorial symmetric cross-validation (CSCV) and the probability of backtest overfitting (PBO) [1], computed on the Stage I candidate pool. The resulting PBO estimate in our BTC snapshot is materially above zero (PBO = 0.586 under a standard 8-segment CSCV design with 70 combinatorial splits over the top-40 Stage I candidates), indicating that data-snooping risk remains substantial even under strict execution and realistic cost modeling. These diagnostics are used defensively and are not interpreted as proof of persistent alpha.

Implementation details for the DSR calculation, together with exact formulas and numerical results for the BTC/USDT anchor case study, are documented in the accompanying replication code and summarized in Appendix A. Practically, the DSR-style statistic serves as a consistency check for the walk-forward and cost-stress evidence rather than as standalone proof of economic significance, and its numerical values should be interpreted in light of the underlying distributional and trial-count assumptions.

3. Related Literature

AutoQuant sits at the intersection of (i) cryptocurrency perpetual microstructure and its funding- and slippage-driven frictions, (ii) hyperparameter search and overfitting control in quantitative trading, and (iii) production-grade risk management and backtest-to-live consistency. We therefore draw from three related literature streams: realistic backtesting under market frictions, Bayesian optimization for strategy parameterization, and risk overlays

and accounting invariants for auditable deployment in production decision-support systems. This framing is consistent with digital finance and fintech research, which emphasizes that measurement, execution, and control layers are integral to operational decision support and model governance [14, 15].

3.1. Expert Systems, Decision Support, and Auditable Pipelines

AutoQuant is positioned as an *expert-system-like*, rule-based decision-support pipeline: it codifies non-negotiable semantics and constraints as a knowledge base, uses an inference procedure (Bayesian search plus policy-driven screening) to select robust configurations, and exports auditable artifacts for traceability (Section 2.1 and Table 2). This emphasis is aligned with decision-support objectives in operational fintech contexts, where governance depends on explicit measurement, execution, and control layers rather than on opaque “best backtest” reports [14, 15]. In this sense, AutoQuant is closer to an execution-aware, audit-oriented AutoML workflow [16] than to a strategy-generation system: the signal family is fixed, and the contribution lies in making configuration selection reproducible and inspectable under strict $t+1$ semantics and cost realism.

3.2. Backtesting and Execution Biases in Cryptocurrency Markets

Realistic backtesting is difficult even in traditional assets [17], and the problem is amplified in cryptocurrency perpetual futures. The 24/7 trading cycle, coupled with frequent and extreme volatility, makes data alignment and timestamp integrity critical [18, 19]. Early work in this area highlighted the prevalence of look-ahead bias, where information unavailable at the time of a decision is improperly included in a backtest, leading to systematically inflated performance metrics. Recent work also documents substantial microstructure frictions, arbitrage constraints, and liquidity fragmentation across venues in cryptocurrency markets [20], reinforcing the need for conservative execution assumptions in backtests.

More specific to perpetual contracts, the modeling of funding payments represents a significant source of potential error. Both academic studies and practitioner reports emphasize that funding payments and contract design features can be first-order for both profitability and risk, and that naive backtests that omit or misalign funding can be materially misleading [21–23]. Recent work also studies market-quality implications of perpetual-futures design [24] and documents strong linkages between spot and derivatives markets, including price discovery effects in Bitcoin futures [25, 26]. Empirical

work exploring modeling and forecasting of perpetual-futures prices for trading applications further underscores that the data-generation and settlement mechanics of the contract family matter for inference [27]. Our work builds upon this literature by architecting a backtest engine that programmatically enforces T+1 execution and no-look-ahead funding calculations, moving beyond identification of the problem to providing an architectural solution.

3.3. Parameter Tuning and Bayesian Optimization in Finance

Most quantitative strategies are highly sensitive to parameter choices. Manual tuning and naive grid search are inefficient and can promote overfitting [28]. Bayesian optimization and related methods can navigate large search spaces more efficiently [29].

Random search provides a competitive and widely used baseline for hyperparameter optimization under limited evaluation budgets [9], and evolutionary computation methods (e.g., genetic algorithms, differential evolution, and CMA-ES) provide a complementary family of black-box optimizers with different trade-offs in constraint handling and sample efficiency [10–12].

In finance, Bayesian optimization has been applied to find optimal parameters for trading rules and portfolio allocation models. However, a common limitation of these applications is the decoupling of the optimization process from a realistic cost and risk model. The search is often conducted in a simplified environment, with the resulting parameters later proving fragile when subjected to real-world frictions. This concern is closely related to classic data-snooping and multiple-testing problems: repeated search over strategy variants or hyperparameters can induce selection bias even when each backtest is implemented correctly [30–33]. Our use of DSR/PBO-style diagnostics is therefore defensive and complements (rather than replaces) strict execution and cost semantics [1, 13]. Our framework contributes by integrating Bayesian optimization with a realistic cost model from the first stage and by keeping strict execution semantics fixed during Stage II screening. By framing the problem in a signal-modular and execution-centric manner, we emphasize the reusability of the pipeline itself, consistent with broader AutoML principles [16].

3.4. Risk Management, Guards, and Full-Chain Consistency

A robust quantitative trading system requires more than a profitable signal; it necessitates layers of risk management and validation that ensure consistency from research to live deployment. The concept of risk-based overlays or

“guards” that monitor strategy performance and can intervene to reduce or halt trading is a cornerstone of institutional risk management practice. These systems often track metrics like portfolio drawdown, volatility, or value-at-risk to make dynamic decisions. Recent work on decentralized finance likewise emphasizes protocol-level risk controls and auditable accounting mechanisms as first-class design objectives [34], reinforcing the view that infrastructure and monitoring are integral components of robust decision support and model governance.

A parallel concern is the “full-chain consistency” between the backtest environment and the live execution engine. Divergences in data handling, cost calculation, or order logic can invalidate backtest results and lead to unexpected live performance [17]. Our work synthesizes these two concepts by proposing a framework that not only includes a post-deployment live guard but also enforces a set of accounting invariants across the entire backtest-to-live chain. This provides a formal, auditable link between theoretical research and practical implementation, a critical but often overlooked aspect of financial data science.

4. Data and Market Setting

This section outlines the financial instruments, data sources, and market assumptions that form the foundation of our empirical analysis. We place a strong emphasis on data integrity and the realistic modeling of trading costs, as these are critical prerequisites for robust backtesting.

4.1. Instruments and Data Sources

The primary instrument for this study is the BTC/USDT perpetual futures contract, and we report parallel replications on ETH/USDT (large-cap) and SOL/USDT and AVAX/USDT (mid-cap) perpetual contracts using the same STRICT4H family and evaluation scripts. We utilize a comprehensive dataset of 4-hour (4h) OHLCV bars sourced from major centralized exchanges, with prices and volumes constructed from exchange-level perpetual futures markets and cleaned using the procedures described below. In the anonymized replication package this is implemented using BTC/USDT perpetual data from Binance, Bybit, and OKX, as detailed in Appendix Appendix A. The dataset is partitioned into two segments: a Core Sample (2019-09-08–2021-12-31, $N = 5,069$ 4h returns) used for descriptive statistics and for our core

cost-accounting experiments, and an Extended Long-Horizon Sample (2019-09-08–2025-10-14, $N = 13,368$ 4h returns) used for long-horizon diagnostics and robustness stress tests. Crucially, our dataset includes a synchronized time series of funding rates extracted at 8-hour intervals, which we align to the 4-hour time axis as a step function (carried forward between funding timestamps under a carry-forward alignment). Where realized funding is unavailable for a portion of the historical sample, we fall back to a conservative constant funding-rate assumption from the same trading-cost profile.

For ETH/USDT, SOL/USDT, and AVAX/USDT, we use 4-hour OHLCV series and corresponding 8-hour funding-rate series drawn from major exchanges. Data availability differs by asset in our reproducible snapshot: the ETH series spans 2021-12-31–2025-11-24, while the SOL/AVAX series spans 2021-01-01–2025-12-16. Our evaluation windows for the replications begin on 2021-12-31 for all three assets (Section 4), and we evaluate them under the same strict execution semantics and a conservative cost family (asset-specific fee/slippage baselines plus the cost-sensitivity grid described in Section 2.4).

Table 10: Data summary for BTC/USDT 4-hour returns.

Sample	# Returns (4h)	Mean Return	Std Dev	Min Return	Max Return
Core					
(2019-09-08–2021-12-31)	5,069	0.00043	0.01615	-0.21046	0.14734
Long					
(2019-09-08–2025-10-14)	13,368	0.00026	0.01309	-0.21046	0.14734

In this manuscript, the label *core* refers specifically to the BTC/USDT 4-hour segment used for the naive-versus-rigorous cost and constraint stress test in Section 6.3 (2019-09-08–2021-12-31). Separately, the BTC/USDT windowed experiments (baseline window diagnostics and the one-stage versus two-stage comparison) use the explicit training/validation splits that extend through 2023-01-01, and long-horizon diagnostics extend through 2025-10-14. Table 10 reports descriptive statistics for the naive-versus-rigorous core segment and the long-horizon segment.

Throughout the paper we use the following terminology for BTC/USDT 4-hour windows (and analogous windows for the ETH/SOL/AVAX replications):

- *Training window* (2019-09-08–2021-01-01): used exclusively for Stage I Bayesian optimization.

- *Validation window* (2021-01-01–2023-01-01): used within the Stage II double-screening protocol for robustness checks and selection, but not for Stage I candidate generation.
- *Long-horizon robustness window* (2019-09-08–2025-10-14): the extended series used for long-run diagnostics (including the deflated Sharpe calculation) and cost-stress tests, not for parameter selection.
- *Blind post-ETF window* (2024-01-01–2025-10-14): a hold-out segment within the extended series that is never used for tuning or selection and serves solely for post-hoc robustness assessment.

For the ETH/USDT, SOL/USDT, and AVAX/USDT replications, we use an analogous scheme with training (2021-12-31–2023-01-01), validation (2023-01-01–2024-01-01), and a post-2024 hold-out starting 2024-01-01 (ending 2025-11-24 for ETH and 2025-12-16 for SOL/AVAX).

All Stage II thresholds and filters (Table 8) are fixed *ex ante* prior to any evaluation on the long-horizon or blind hold-out windows, and we do not modify hyperparameters, thresholds, or selection rules after observing hold-out outcomes.

Table 11 summarizes how these windows are reused (or held out) across Stage I optimization, Stage II screening, and post-hoc robustness checks.

When we refer to the *core sample*, we mean the 2019-09-08–2021-12-31 BTC/USDT 4-hour series described above, which is used for the naive-versus-rigorous stress test in Section 6.3. For the windowed results (baseline windows and one-stage versus two-stage comparisons), we always use the explicit window names in the itemized definitions above (Training, Validation, Long-horizon, and the blind hold-out window: Post-ETF for BTC / Post-2024 for non-BTC) to avoid ambiguity about date ranges and about which windows are held out from Stage I.

Returns are computed as simple 4-hour arithmetic close-to-close returns,

$$r_t^{\text{mkt}} = \frac{P_t^{\text{close}}}{P_{t-1}^{\text{close}}} - 1 \quad (9)$$

where P_t^{close} denotes the 4-hour close price. Under the STRICT4H execution semantics (Section 2.4), the position applied to r_t^{mkt} is lagged by one bar, so that the raw strategy return at bar t depends only on information available by the close of bar $t - 1$.

Table 11: Mapping between evaluation stages and BTC/USDT 4-hour windows in the AutoQuant pipeline.

Stage / component	Primary role	Windows used	Key settings
Stage I (Bayesian search)	Parameter generation	Training	TPE; $N_{\text{opt}} = 40$
Stage II (double-screening)	Robustness screening	Training + validation	$K = 5$; WFE win=2000, step=500
Long-horizon diagnostics	DSR and cost stress	Long-horizon robustness	cost stress $2\times, 3\times$
Blind evaluation	Post-ETF robustness	Blind post-ETF	post-hoc only

The unified codebase supports multiple time frequencies. For the results reported in this manuscript, however, both Stage I tuning and Stage II screening are reproduced from the standardized 4-hour (4h) OHLCV (Open, High, Low, Close, Volume) series together with the aligned 8-hour funding series under STRICT4H semantics; no auxiliary 1m/15m inputs are required to regenerate the reported Stage I/II tables and figures. Separate high-frequency guard experiments are described in Appendix Appendix A and are not part of the main empirical results.

4.2. Data Cleaning and Validation

Our policy for handling OHLCV data gaps is conservative and designed to prevent the introduction of artificial price data. The system does not perform interpolation or filling (such as forward-fill) of missing OHLCV bars. Funding-rate timestamps are treated as a separate, external series: when available, they are aligned to the 4-hour grid using carry-forward between funding updates with no backfilling from future updates, and when unavailable we fall back to a conservative constant assumption from the cost profile. We do not impute missing funding observations with zeros; missing values are handled by carry-forward and the conservative fallback under the same carry-forward semantics. Where realized funding coverage is partial in historical samples, we treat this fallback as a conservative approximation and quantify sensitivity via the funding-multiplier cost scenarios used in Stage II. Instead,

a continuity check routine logs a warning for any detected gaps and raises a data integrity exception if the number of missing bars exceeds a predefined, minimal tolerance. For our core backtesting datasets, this tolerance is typically set to zero, meaning any gap in the OHLCV stream is treated as a fatal error, forcing a rejection of the data source and halting the pipeline until the data is manually corrected or replaced. This strict approach ensures that our backtests are run only on clean, contiguous, and reliable historical data.

4.3. Data Construction Protocol (4-hour OHLCV)

To make the BTC/USDT 4-hour series auditable, we adopt a simple and explicit construction protocol: (i) use exchange-provided perpetual-futures candle fields (open, high, low, close, volume) as the primary price/volume source (trade-price candles, not mark/index); (ii) represent each bar by its *bar-open* timestamp in UTC; (iii) enforce basic OHLCV sanity constraints (e.g., $\text{high} \geq \max(\text{open}, \text{close}, \text{low})$, $\text{low} \leq \min(\text{open}, \text{close}, \text{high})$, non-negative volume); (iv) sort, de-duplicate by timestamp, and run a strict continuity check at the target frequency; and (v) construct the STRICT4H funding series separately and join it as an external input with an explicit fallback when realized funding is unavailable.

In the unified codebase used for this manuscript, the data-fetch and merge logic is implemented in lightweight scripts that editors and reviewers can execute directly. For example, `tools/fetch_full_history.py` fetches Binance USD-M perpetual 1-minute candles and deterministically resamples them into 15-minute and 1-hour OHLCV; `tools/gen_4h_from_1h.py` aggregates 1-hour bars into 4-hour OHLCV; and the continuity/validation checks are enforced via `quant/data_validators.py`. For incremental updates (with provider fallback), `tools/fetch_klines_incremental.py` refreshes a target OHLCV file by paging K-lines from public endpoints and de-duplicating timestamps. For a one-command build/validate/report workflow, the repository snapshot also includes `tools/build_and_pack_4h.py`, which orchestrates CI-style validators and report generation for the 4-hour dataset.

For the funding input, the repository snapshot includes `tools/generate_synthetic_funding_from_ohlc.py`, which generates a deterministic 8-hour funding-rate baseline aligned to the OHLCV time range (used as a conservative fallback); when exchange-backed funding is available, we backfill it via `tools/funding_backfill_from_binance.py` and align it to the STRICT4H grid using the carry-forward semantics described in

Section 2.4. Figure 2 previews the resulting baseline performance across the three evaluation windows.

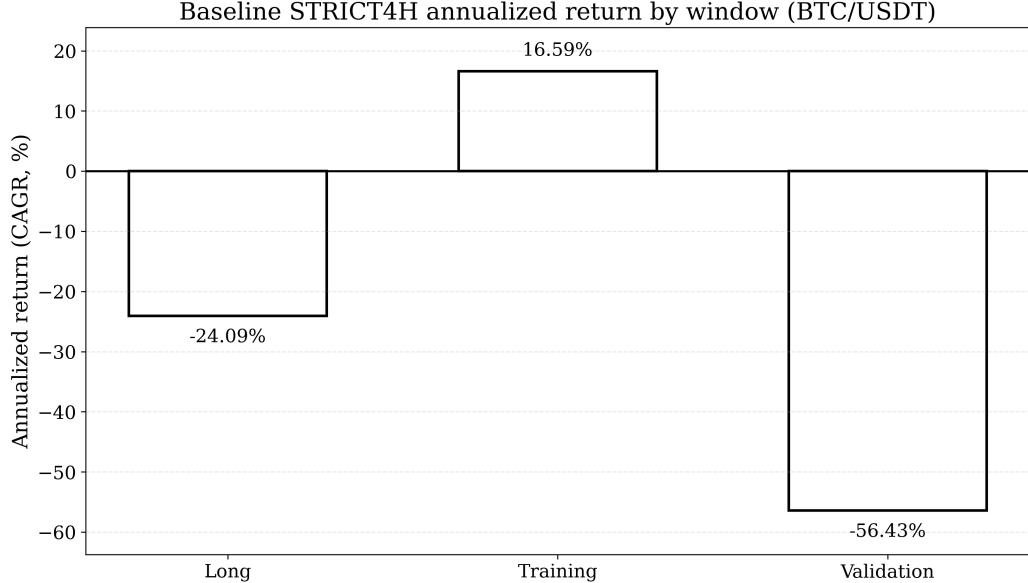


Figure 2: Baseline STRICT4H annualized return by window (BTC/USDT).

4.4. Trading Cost Model and Constraints

A cornerstone of our framework is a realistic trading-cost model, which aims to provide a comprehensive and conservative estimate of market frictions. The model is applied uniformly across both backtesting and simulation stages to ensure consistency. In our implementation this component is named `RealisticTradingCosts`. It incorporates the following components:

Trading Fees: Fees are applied when the position changes. In the implementation, the fee cost on a bar is proportional to absolute turnover, $C_{\text{fee},t} = |\Delta \text{pos}_t| \cdot f_{\text{eff}}$, where f_{eff} is an effective fee rate. While the model supports a maker/taker mixture, our STRICT4H case study uses pure taker execution (maker ratio = 0), so f_{eff} reduces to the taker fee.

Slippage: Slippage is parameterized as a per-trade basis-point cost, again applied to absolute turnover. In the implementation, $C_{\text{slip},t} = |\Delta \text{pos}_t| \cdot s$, with $s = \text{base_slippage_bps}/10,000 \times \text{slippage_multiplier}$. Pessimistic scenarios scale the fee and slippage terms via higher `taker_fee` and `base_slippage_bps`.

Funding Payments: Funding is charged on the signed nominal exposure using an 8-hour funding rate series, scaled to the bar length. If fr_t^{8h} denotes the realized 8-hour funding rate aligned to bar t under a strict $t+1$ convention (the rate observed at a funding timestamp is applied only from the subsequent 4-hour bar onward), the funding term is $C_{\text{fund},t} = \text{pos}_t \cdot \text{fr}_t^{8h} \cdot (\Delta h / 8)$, where Δh is the bar duration in hours. Importantly, all funding-dependent decision rules (e.g., funding gates, biasing, and entry blocking) use only funding information available at the bar close and never read future funding rates.

In our BTC/USDT STRICT4H experiments, the baseline cost profile assumes an initial equity of 10,000 USDT, a maximum leverage of $5\times$, a notional cap of 50,000 USDT, a taker fee of 4 bps per trade, and a base slippage of 2 bps per trade. The cost profile also includes a fallback funding rate of 1 bps per 8-hour period, but in the core experiments funding costs are computed from the realized 8-hour funding-rate time series described above (the fallback is used only if the series is unavailable). For ETH/USDT STRICT4H aggressive we maintain the same equity, leverage, and notional-cap assumptions but slightly increase taker fees, funding rates, and base slippage relative to BTC. These parameter choices are calibrated to be broadly consistent with fee schedules and realized funding ranges on major centralized derivatives exchanges over the 2019–2025 period, while remaining deliberately conservative relative to the most favorable taker tiers. In particular, we do not model maker rebates or aggressive fee discounts; our reported results therefore err on the side of slightly overstating rather than understating effective trading frictions for small- to mid-sized positions.

Table 12 summarizes the baseline cost profile and the stress scenarios used in the BTC/USDT STRICT4H case study.

Table 12: Cost-profile parameters used in the BTC/USDT STRICT4H experiments and stress scenarios. Fees and slippage are specified in basis points (bps). Funding uses the realized `funding_rate_8h` series when available, with a conservative fallback when the realized series is unavailable.

Scenario	Equity	Lev.	Cap	Fee	Slip.	Funding
Baseline	10,000	5	50,000	4	2	real. + fb
Funding-off	10,000	5	50,000	4	2	off
Pess. ($2\times$)	10,000	5	50,000	8	4	real. + fb
Pess. ($3\times$)	10,000	5	50,000	12	6	real. + fb

Leverage and Notional Caps: The backtest engine strictly enforces both

maximum leverage and absolute notional position caps. If a trade signal would imply a resulting position that exceeds either of these limits, the engine does not reject the trade. Instead, it caps the trade size, automatically reducing the number of units to be traded so that the final position adheres precisely to the most restrictive of the configured limits. This ensures that the backtest never simulates a position that would be impermissible in a live trading environment.

5. System Architecture and Implementation

This section summarizes implementation choices that are most relevant to the auditability of our empirical results under strict execution and cost modeling. The full AutoQuant codebase contains additional production components (deployment utilities, exchange adapters, and monitoring dashboards). In this manuscript we focus on the design choices that bear directly on research validity: (i) reuse of the same STRICT4H signal and cost engine across offline evaluation and live-style replay; (ii) explicit, reconcilable accounting of returns and costs; and (iii) a post-deployment guard that is intentionally separated from the optimization objective.

5.1. Three Evaluation Stages: Data, Optimization, and Out-of-Sample Validation

The workflow is organized into three stages. (i) *Data and environment validation* performs the integrity checks in Section 4.2 and fixes a cost profile and constraint set that remain unchanged during optimization and screening. (ii) *Optimization and screening* implements the two-stage methodology described in Sections 2.1–2.4: Stage I generates candidates on a single training window under the realistic cost model, while Stage II screens candidates across held-out evaluation windows and cost stress scenarios using the STRICT4H engine. (iii) *Live-style replay and risk control* applies the same STRICT4H logic on a rolling window of recent data to produce a target exposure and optionally overlays a guard for post-deployment risk supervision. In this paper, Stage (iii) is treated as an auditability pattern (consistency of signals and cash-flow accounting) rather than as evidence of long-horizon live performance.

5.2. The Live Engine and Executor

The live engine is implemented as a thin wrapper around the same core STRICT4H strategy and cost logic used offline. On each update cycle it

recomputes the target exposure from a bounded sliding window of recent 4-hour data and applies the same strict T+1 semantics described in Section 2.4 (signals are produced at bar close and only take effect from the next bar). The executor maps the target exposure to exchange orders while enforcing the same leverage and notional caps used in the backtest engine and logging all fills and cash-flow components (fees, slippage, funding, and PnL) for subsequent reconciliation.

5.3. Guard and Monitoring Infrastructure

A live guard provides a post-deployment risk overlay that monitors a dashboard of rolling performance and risk metrics computed from execution logs and can issue an *ok/watch/kill* decision. A “kill” decision triggers position flattening and can optionally disable further trading until manual intervention. Because the guard is triggered by realized PnL and risk states rather than by predictive signals, we treat it as a safety mechanism that is conceptually separated from the Stage I/II optimization objectives. In this paper we report only limited offline and paper-trading evidence and therefore interpret the guard component as an auditability pattern rather than as a claim of live performance.

6. Empirical Evaluation

This section presents a series of empirical experiments designed to evaluate the effectiveness of our proposed execution-centric auto-tuning and validation framework. Using a multi-year dataset of BTC/USDT perpetual futures, we aim to quantify the impact of common backtesting pitfalls and demonstrate how our systematic, multi-stage approach produces more robust and realistic performance assessments.

To align with ESWA-style component validation, we structure the empirical section as a set of ablation-style diagnostics around the pipeline: (i) one-stage tuning versus two-stage double-screening isolates the incremental effect of policy-driven screening; (ii) a cost-ablation ladder (fee-only versus fully costed versus zero-cost upper bound) isolates the contribution of each friction channel; and (iii) an execution-delay sanity check ($t+1$ versus a deliberately naive $t+0$ convention) isolates the role of strict execution semantics.

6.1. Experimental Setup

Our experiments are conducted on 4-hour OHLCV datasets for BTC/USDT, ETH/USDT, SOL/USDT, and AVAX/USDT perpetual contracts, built from

the reproducible core 4h dataset described in Section 4. This dataset covers the period from 2019-09-08 to 2021-12-31 and contains 5,070 4h bars (5,069 non-overlapping 4h returns) after validation. For the primary comparative experiment in Section 6.2, we supplement this core dataset with the extended 4-hour BTC/USDT series described in Section 4 and define three distinct, non-overlapping (and for BTC deliberately non-contiguous) windows: an in-sample training window from 2019-09-08 to 2021-01-01 used exclusively for Stage I Bayesian optimization; a validation and selection window from 2021-01-01 to 2023-01-01 used within the Stage II double-screening protocol; and a blind test window from 2024-01-01 to 2025-10-14 that is never used for parameter tuning or selection and serves solely as a post-ETF hold-out period for assessing robustness. For BTC, we intentionally leave the 2023-01-01–2024-01-01 interval unused to separate the validation-era selection window from the post-ETF hold-out and to avoid hindsight-driven recutting of the regime boundary. All backtests are executed using the system architecture described in Section 5, with costs modeled according to the realistic trading-cost profile (Section 4.4). For ETH/USDT we mirror the study design with a training window (2021-12-31–2023-01-01), validation window (2023-01-01–2024-01-01), and post-2024 hold-out (2024-01-01–2025-11-24), using the same STRICT4H configuration family and evaluation scripts. For SOL/USDT and AVAX/USDT, we use the same training and validation window definitions as ETH and extend the post-2024 hold-out through 2025-12-16.

Table 13: **Configuration naming used in the manuscript.** We distinguish *baseline* (a frozen reference from Stage I), *one-stage* (top-1 on training), and *two-stage* (selected via stable-candidate screening). “Naive” and “standard” are diagnostic stress-test variants applied to a separate stress-test configuration in Section 6.3.

Name in text	Definition	Used for selection?	Where referenced
Baseline STRICT4H	Frozen reference built from top-ranked Stage I parameters; evaluated across windows	No	Table 14, Fig. 2
One-stage (top-1)	Best Stage I trial by training-window objective	Yes (Stage I only)	Section 6.2
Two-stage (selected)	Stable candidate selected by ex-ante filter and scenario-grid aggregation	Yes (Stage II)	Section 6.2
Standard (fee-only)	Diagnostic backtest with taker fees only; same strict execution	No	Section 6.3
Naive (zero-cost)	Upper bound with fees/slippage/funding removed (and optional cap relaxation)	No	Section 6.3

To provide a point of comparison, we define a Baseline Strategy configuration. This is not a toy example but a realistic, production-oriented baseline built from the top-ranked Stage I parameter sets for our BTC/USDT STRICT4H family, and then frozen for all subsequent window and stress-test evaluations. It serves as a fixed reference configuration for auditing and ablation, using a mix of standard indicator lookbacks (e.g., for trend and mean-reversion signals) and conservative risk management rules, and is held fixed throughout the experiments. Its performance over three windows is summarized in Table 14. For clarity, we distinguish three roles: (i) the training window (2019-09-08–2021-01-01) serves as the primary optimization window for Stage I Bayesian search; (ii) the validation window (2021-01-01–2023-01-01) serves as a validation-style window that is held out from Stage I but explicitly reused within Stage II double-screening; and (iii) the long-horizon window (2019-09-08–2025-10-14) is a stress window that is used only for robustness

checks (including the DSR calculation), not for parameter selection.

Table 14: Baseline STRICT4H performance by window (BTC/USDT; annualized return reported as CAGR, %; window definitions follow Section 4). Values are taken from the replication outputs described in Appendix Appendix A.

Window	Ann. (CAGR, %)	MaxDD	Sharpe	Trades
Long (2019-09-08–2025-10-14)	-24.09%	0.9536	-0.0738	4,814
Training (2019-09-08–2021-01-01)	16.59%	0.6041	0.5233	1,027
Validation (2021-01-01–2023-01-01)	-56.43%	0.9174	-0.5913	1,549

In keeping with our emphasis on interpretable and economically meaningful performance metrics, Table 14 reports only normalized statistics—annualized return, maximum drawdown, Sharpe ratio, and total trade count—taken directly from the STRICT4H machine-readable outputs in our replication package. The same outputs also contain raw total return and final equity metrics, which we treat as internal accounting checks and deliberately omit from the main text, as they are highly sensitive to sample length and to the arbitrary choice of initial capital and are not used as inputs to our statistical inference or robustness conclusions.

The performance of each experiment is evaluated using the set of metrics defined in Section 2.2 (with market returns defined in Eq. (9)). The Sharpe ratio, a key metric, is annualized from the 4-hour returns using the annualization implied by the 4-hour frequency (2190 4h bars per year) and assumes a constant annual risk-free rate of 3%. Because 4-hour crypto returns are autocorrelated and heavy-tailed, we treat Sharpe as a descriptive statistic rather than as a parametric test. Our analysis focuses not just on the absolute values of these metrics, but on their sensitivity across different time periods and cost assumptions.

Reading guide for compounded return magnitudes. Depending on the experiment, we report annualized return either as CAGR in % (for readability) or as CAGR in decimal units (e.g., 0.10 = 10%). In high-volatility regimes, compounded CAGR values can exceed 1.0 (i.e., exceed 100% per year) even under strict execution and explicit leverage caps. We therefore interpret large CAGRs as *diagnostic* signals of sensitivity to execution, window choice, and costs, and we emphasize drawdown, stability across windows, and cost-stress outcomes. None of the reported return magnitudes should be interpreted as statements about institutional-scale deployability, since non-linear market

impact is not modeled.

6.2. One-Stage Tuning vs. Two-Stage Double Screening

The first experiment is designed to test the core hypothesis of our paper: that a simple, one-stage parameter optimization is insufficient for identifying robust strategies. We compare two distinct approaches:

One-Stage Tuning: We run the Stage I Bayesian optimization exclusively on the training window (2019-09-08–2021-01-01) to find the top-performing parameter set based solely on its in-sample, cost-adjusted annualized return.

Two-Stage Framework: We take the pool of high-performing candidates from Stage I and subject them to the full Stage II double-screening protocol, including out-of-sample rolling window analysis and cost-mis-specification scenarios. The final selected parameter set is the one that ranks highest according to the stable-candidate criteria described in Section 2.4 and Table 8.

To make this comparison concrete, we compute performance metrics on the training window (2019-09-08–2021-01-01) and on the validation window (2021-01-01–2023-01-01) that is out-of-sample with respect to Stage I, but still used within the Stage II screening protocol. By construction, the simple one-stage tuning procedure tends to identify parameter sets that look attractive on the training window but degrade sharply when evaluated on the validation window. In our audited BTC/USDT STRICT4H case study, the baseline configuration—which is deliberately frozen after Stage I and used as a reference point—illustrates this fragility. As shown in Table 14, it attains 16.6% annualized return with a maximum drawdown of about 60% on the training window, but then degrades sharply on the validation window under the same cost and execution model. These numbers are taken directly from the STRICT4H walk-forward artifacts generated by our backtest engine using the BTC/USDT datasets described in Section 4. This comparison illustrates why Stage II screening is required: under any reasonable stability criteria, a configuration whose validation-window return collapses and whose drawdown expands materially should be flagged as fragile and treated as unsuitable for deployment without further controls. We therefore interpret the validation-window results as a validation-style robustness check rather than as a fully untouched, final out-of-sample test, and we treat the Stage II protocol as a mechanism for making such degradation visible and auditable rather than as a guarantee of profitability.

Interpretation relative to buy-and-hold. Table 15 includes buy-and-hold as a simple external yardstick for the underlying (no trades, no trading

Table 15: **Direct comparison: one-stage versus two-stage selection under identical **STRICT4H** execution and rigorous costs (BTC/USDT).** One-stage selects the single best Stage I trial on the training window. Two-stage selects a stable candidate via the ex-ante filter and cost-scenario aggregation described in Section 2.4 and Table 8. Validation is held out from Stage I but reused within Stage II screening; Post-ETF is a blind hold-out used only for robustness. To reduce the visual dominance of extreme compounded annualized returns in high-volatility regimes, we report the monthly geometric mean of net returns (decimal units; e.g., 0.10 = 10% per month) alongside Sharpe, drawdown, and trade counts. Annualized return figures remain available in the replication outputs described in Appendix Appendix A. Market impact and capacity constraints are not modeled. Buy-and-hold is included as a simple external yardstick (no trades) computed on the same OHLCV series.

Method	Window	Monthly geom (dec.)	Sharpe	MaxDD	Trades
One-stage (top-1)	Training	0.222	3.164	0.283	548
Two-stage (selected)	Training	0.145	2.388	0.265	485
Buy-and-hold (underlying)	Training	0.067	1.530	0.563	—
One-stage (top-1)	Validation	0.051	1.238	0.318	423
Two-stage (selected)	Validation	0.038	1.070	0.231	348
Buy-and-hold (underlying)	Validation	-0.024	0.005	0.771	—
One-stage (top-1)	Post-ETF	0.023	0.845	0.220	259
Two-stage (selected)	Post-ETF	0.016	0.664	0.196	223
Buy-and-hold (underlying)	Post-ETF	0.048	1.405	0.307	—

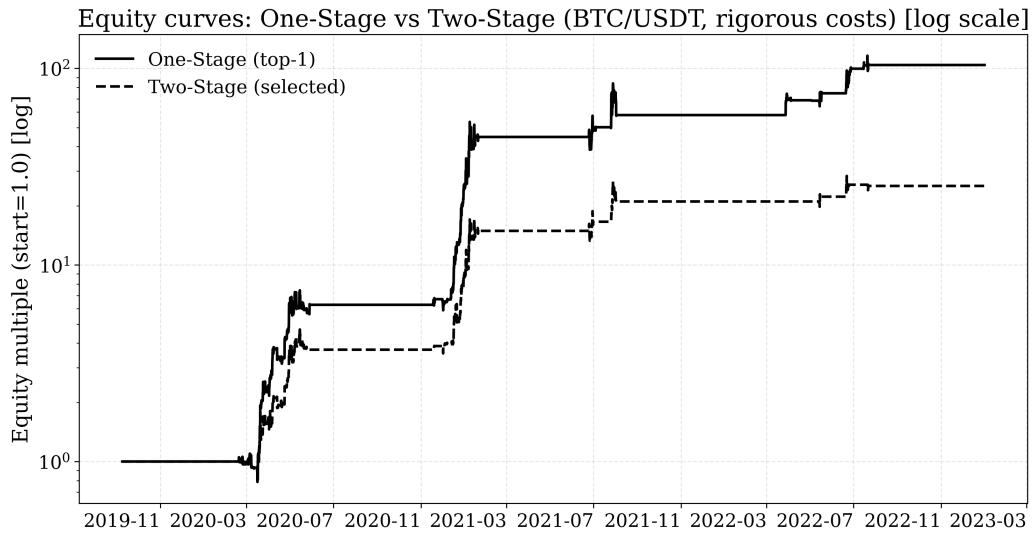


Figure 3: **Equity curves for one-stage (top-1) versus two-stage (selected) under rigorous costs on BTC/USDT.** Curves start at 1.0 and are plotted over the training+validation period used in Table 15. The y-axis is log-scaled to avoid visual dominance by extreme compounding. The dotted vertical line marks the start of the validation window. The goal is diagnostic: the two-stage selection tends to reduce drawdown severity and extreme compounding relative to the top-1 in-sample winner, but it does not guarantee higher returns.

costs). In the BTC post-ETF window, buy-and-hold outperforms both tuned STRICT4H configurations on the reported risk-adjusted summaries. We interpret this not as a contradiction but as a stress-test outcome that reinforces our framing: the paper’s contribution is an auditable evaluation and selection protocol under explicit execution and cost semantics, not a claim that a fixed strategy family dominates a passive benchmark across regimes. **Block-bootstrap check (validation window).** As a minimal inferential supplement that is robust to time dependence, we compute a moving block bootstrap [35] on the monthly return series for the validation segment, reporting a 95% interval for the difference in monthly geometric mean returns between the two-stage and one-stage procedures (two-stage minus one-stage).

Table 16: **Moving block bootstrap CI for monthly geometric mean differences (BTC/USDT validation).** Monthly geometric means are in decimal units; Δ is reported as two-stage minus one-stage.

Window	Δ monthly geom (dec.)	95% CI (dec.)	Block length (months)
Validation	-0.016	[-0.040, 0.008]	3

Threshold sensitivity.. Because stable-candidate selection relies on ex-ante thresholds (Table 8), we report a sensitivity scan in the replication package. Across reasonable variations of the monthly return floor, drawdown cap, and switch-density cap, the selected trial identity and its validation performance can change, highlighting that these thresholds function as *risk-control design choices* rather than as universal constants. We therefore treat the stable-candidate filter as a transparent, auditable policy rather than as an optimized meta-learner. In the $3 \times 3 \times 3$ grid (27 policy variants), the selected configuration toggles among three trial IDs (52, 87, 108; each selected 9 times), with the number of passing candidates ranging from 12 to 31. Across these policy variants, validation-window CAGR spans 0.332–1.058 (decimal), Sharpe spans 0.844–1.667, and MaxDD spans 0.209–0.386. Table 17 shows representative settings.

As an additional stress-test exercise, we keep selected configurations fixed and extend them into the 2021-12-31–2025-10-14 BTC/USDT 4-hour period (covering both the pre-ETF 2022–2023 segment and the post-ETF segment after 2024), including a blind post-ETF hold-out window (2024-01-01–2025-10-14). We treat these extensions as post-hoc robustness checks rather than

Table 17: **Representative threshold-sensitivity settings for stable-candidate selection (BTC/USDT).** Thresholds are ex-ante design choices; we report the resulting selected trial ID and its validation performance under rigorous costs. Annual return is CAGR in decimal units; MaxDD is a fraction in $[0, 1]$.

Monthly floor	MaxDD cap	Switch cap	Selected	Val			
				CAGR (dec.)	Sharpe	Val MaxDD	Val trades
0.003	0.25	0.10	87	0.332	0.844	0.386	438
0.003	0.30	0.10	108	0.572	1.070	0.231	348
0.003	0.35	0.10	52	1.058	1.667	0.209	467

as headline results, and we do not use them for parameter selection or for the core claims of the paper. In particular, post-ETF performance can be highly regime-dependent; we therefore emphasize the methodology (strict execution semantics, cost modeling, and multi-window screening) and report post-ETF evaluations as auxiliary stress tests in the replication package.

6.3. Naive Backtest vs. Rigorous Backtest

Our second experiment aims to quantify the degree of performance overestimation that results from extreme simplifications to the cost and constraint model. For this purpose, we construct a deliberately optimistic “Naive” configuration that holds the **STRICT4H** signal logic and T+1 execution semantics fixed but removes market frictions and relaxes capacity constraints. Concretely, relative to the rigorous configuration, the naive configuration (i) sets fees, slippage, and funding costs to zero; and (ii) relaxes leverage and notional caps so that the same signal path can be evaluated in a near-frictionless environment. This should be interpreted as an upper bound on friction- and constraint-related performance inflation in our setting, rather than as a representation of typical industry backtesting practice.

We next isolate the impact of cost modeling itself. For this purpose, we take a **separate STRICT4H configuration** used specifically for this stress test (it is not the baseline used in Table 14) and evaluate it under a ladder of increasingly optimistic assumptions on the BTC/USDT core sample (2019-09-08–2022-01-01, UTC; i.e., through 2021-12-31 under bar-open indexing). The *rigorous* variant uses the full realistic trading-cost profile (fees, slippage, and funding) under strict $t+1$ execution. The *standard* variant keeps strict execution but includes fees only. The *naive* variant removes fees, slippage, and funding to form a zero-cost upper bound; we additionally test a cap-relaxed version (leverage and notional caps relaxed), which does not change

the reported metrics in this snapshot. Because cost-aware channels (e.g., funding-based gates) depend on the cost and funding inputs, the realized trade sequence (and thus trade count) can differ across variants even when hyperparameters are fixed.

Table 18: **Cost ablation ladder for the naive-versus-rigorous stress test (BTC/USDT core sample).** Fixed parameters; the realized trade sequence (and thus trade count) may differ across variants. Annual return is reported as CAGR in decimal units (e.g., 0.10 = 10%); values may exceed 1.0 under compounding and should be interpreted as a diagnostic of cost sensitivity rather than as a scalability claim. Market impact is not modeled. Values are taken from the replication outputs described in Appendix Appendix A.

Variant	Ann. (CAGR, dec.)	Sharpe	MaxDD	Trades
Rigorous (fees+slippage+funding)	2.726	2.049	0.265	636
Standard (fee-only)	4.308	2.506	0.262	636
Naive (zero-cost, upper bound)	5.225	2.711	0.260	535

Table 18 quantifies how optimistic cost simplifications inflate reported performance for this fixed stress-test configuration. Even a fee-only “standard” backtest can overstate performance relative to a fully costed run that includes slippage and funding, and removing all costs provides an additional optimistic upper bound. We treat these results as diagnostic baselines rather than as evidence of scalable profitability, and we do not interpret them as an AutoQuant-versus-standard uplift. The baseline STRICT4H figures in Table 14 correspond to a separate, production-oriented parameter set and a different experiment.

6.4. Robustness and Sensitivity Checks

The final set of experiments evaluates the robustness of the parameter sets identified by our full two-stage framework under various alternative conditions, using data from the core 4-hour horizon and, where applicable, the extended sample. First, to address the single-asset concern and to validate that the workflow is not BTC-specific, we repeat the one-stage versus two-stage comparison on ETH/USDT using the same STRICT4H family and identical execution and cost semantics. Table 19 reports the direct comparison. Similar to BTC, the two-stage selection does not dominate the one-stage top-1 on annualized return but tends to select more conservative configurations in terms of drawdown. We treat this as evidence that the audit-oriented screening mechanism is portable at the pipeline level across

two liquid perpetual contracts, while emphasizing that it does not establish universal profitability or transferability beyond these two assets and this single strategy family.

Table 19: Direct comparison: one-stage versus two-stage selection under identical STRICT4H execution and rigorous costs (ETH/USDT). Windows mirror the ETH study design: training (2021-12-31–2023-01-01), validation (2023-01-01–2024-01-01), and a post-2024 hold-out (2024-01-01–2025-11-24) as a blind robustness segment. We report the monthly geometric mean of net returns (decimal units; e.g., 0.10 = 10% per month) alongside Sharpe, drawdown, and trade counts; annualized return figures remain available in the replication outputs described in Appendix Appendix A. Market impact and capacity constraints are not modeled. Values are taken from the replication outputs described in Appendix Appendix A.

Method	Window	Monthly geom (dec.)	Sharpe	MaxDD	Trades
One-stage (top-1)	Training	0.233	3.009	0.317	2112
Two-stage (selected)	Training	0.173	2.607	0.276	346
One-stage (top-1)	Validation	0.085	2.157	0.186	2012
Two-stage (selected)	Validation	0.046	1.368	0.201	457
One-stage (top-1)	Post-2024	0.151	2.590	0.381	3880
Two-stage (selected)	Post-2024	0.096	2.009	0.368	852

Second, to probe higher-friction mid-cap contracts, we repeat the same protocol on SOL/USDT and AVAX/USDT using more conservative cost profiles (higher fee and slippage assumptions) but the same strict execution semantics. Table 20 reports the direct comparison. The results illustrate the intended role of double-screening as a *fragility filter*: the one-stage top-1 can exhibit extreme compounding and large drawdowns, while the two-stage selection tends to select more stable, lower-drawdown configurations under more conservative cost assumptions. As with BTC and ETH, these results are diagnostic and do not constitute a claim of scalable alpha.

Third, to test portability beyond a single contract, we evaluate an ETH-trained STRICT4H aggressive configuration on BTC/USDT over the overlapping window (2021-12-31–2025-11-24, UTC). Table 21 reports the key summary metrics under consistent rigorous trading-cost profiles, and Figure 4 visualizes annual return and maximum drawdown for the in-sample (ETH) and cross-asset (BTC) evaluations.

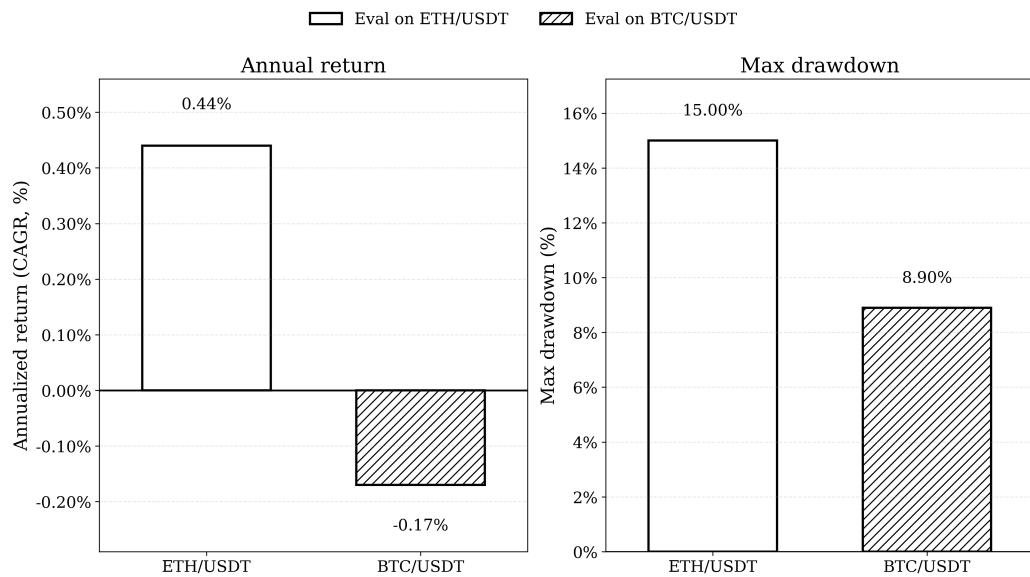


Figure 4: Cross-asset robustness of the ETH-trained STRICT4H aggressive configuration evaluated on ETH/USDT (in-sample) and BTC/USDT (cross-asset) over the overlapping data window (2021-12-31–2025-11-24, UTC; replication outputs in Appendix Appendix A). Annual return is reported as CAGR in % (percentage points), and maximum drawdown is reported as a fraction in $[0, 1]$ in Table 21 but displayed in % for readability in this figure.

Table 20: Direct comparison on mid-cap contracts under stricter cost profiles (SOL/USDT** and **AVAX/USDT**).** To avoid the visual dominance of extreme compounded CAGR values in small-sample, high-volatility regimes, we report the monthly geometric mean of net returns (decimal units; e.g., 0.10 = 10% per month) alongside Sharpe, drawdown, and trade counts. Annualized return figures remain available in the replication outputs described in Appendix Appendix A. Market impact and capacity constraints are not modeled.

Asset	Method	Window	Monthly geom (dec.)	Sharpe	MaxDD	Trades
SOL/USDT	One-stage (top-1)	Training	0.141	2.625	0.319	1567
SOL/USDT	Two-stage (selected)	Training	0.124	3.013	0.152	1605
SOL/USDT	One-stage (top-1)	Validation	0.227	3.398	0.239	1596
SOL/USDT	Two-stage (selected)	Validation	0.362	4.208	0.242	1653
SOL/USDT	One-stage (top-1)	Post-2024	0.066	1.747	0.243	3295
SOL/USDT	Two-stage (selected)	Post-2024	0.068	2.301	0.195	3450
AVAX/USDT	One-stage (top-1)	Training	0.479	3.239	0.795	2063
AVAX/USDT	Two-stage (selected)	Training	0.091	4.856	0.055	1739
AVAX/USDT	One-stage (top-1)	Validation	0.572	3.851	0.621	1908
AVAX/USDT	Two-stage (selected)	Validation	0.087	4.185	0.096	1118
AVAX/USDT	One-stage (top-1)	Post-2024	0.797	4.407	0.607	4158
AVAX/USDT	Two-stage (selected)	Post-2024	0.094	3.994	0.142	3663

Table 21: ETH STRICT4H cross-asset performance (4h aggressive line) under consistent realistic trading-cost profiles for ETH and BTC over the overlapping data window (2021-12-31–2025-11-24, UTC; annual return is CAGR in %). Values are in percentage points (e.g., 0.44% corresponds to 0.0044 in decimal units). Values are taken from the replication outputs described in Appendix Appendix A.

Strategy	Train	Eval	Ann. (%)	MaxDD	Trades
ETH STRICT4H aggressive v1	ETH	ETH	0.44%	0.150	3,100
ETH STRICT4H aggressive v1	ETH	BTC	-0.17%	0.089	3,403

Fourth, we conduct a robustness analysis with respect to drawdown-aware risk budgeting. We examine how the stable candidate’s annualized return and Sharpe ratio change when exposure is scaled down in deeper drawdown buckets under our baseline cost profile for the 2019-09-08–2025-10-14 BTC/USDT long window. This analysis illustrates the strategy’s tolerance to extended drawdowns and clarifies that its edge does not rely on maintaining full exposure during the worst episodes. Over the 2019-09-08–2025-10-14 long window, the drawdown-bucket risk-budgeting variants improve annualized return by roughly 24–33 percentage points and reduce maximum drawdown by about 14–26 percentage points relative to the STRICT4H baseline (Table 22, Figure 5). These results support the view that simple, state-dependent exposure scaling can materially improve risk/reward in a high-drawdown regime for this BTC/USDT long-window analysis.

Table 22: Drawdown-bucket risk-budgeting variants vs the STRICT4H baseline on the 2019-09-08–2025-10-14 long window (annualized return reported as CAGR, %). Values are taken from the replication outputs described in Appendix Appendix A.

Config	Ann. (%)	MaxDD	Sharpe	Δ Ann. (pp)	Δ MaxDD (pp)	Δ Sharpe
v1 (0–20–35)	4.06%	0.760	0.247	28.15	-19.36	0.321
v2 (0–18–33)	0.37%	0.810	0.194	24.46	-14.32	0.268
v3 (0–22–38)	8.46%	0.695	0.329	32.55	-25.83	0.403

The shorthand labels v1–v3 correspond to the three drawdown-bucket risk-budgeting policy variants summarized in Table 22.

Here the differences are reported relative to the STRICT4H baseline over the same 2019-09-08–2025-10-14 long window. Annualized return and maximum drawdown differences are expressed in percentage points (pp), while Sharpe differences are expressed in raw Sharpe units.

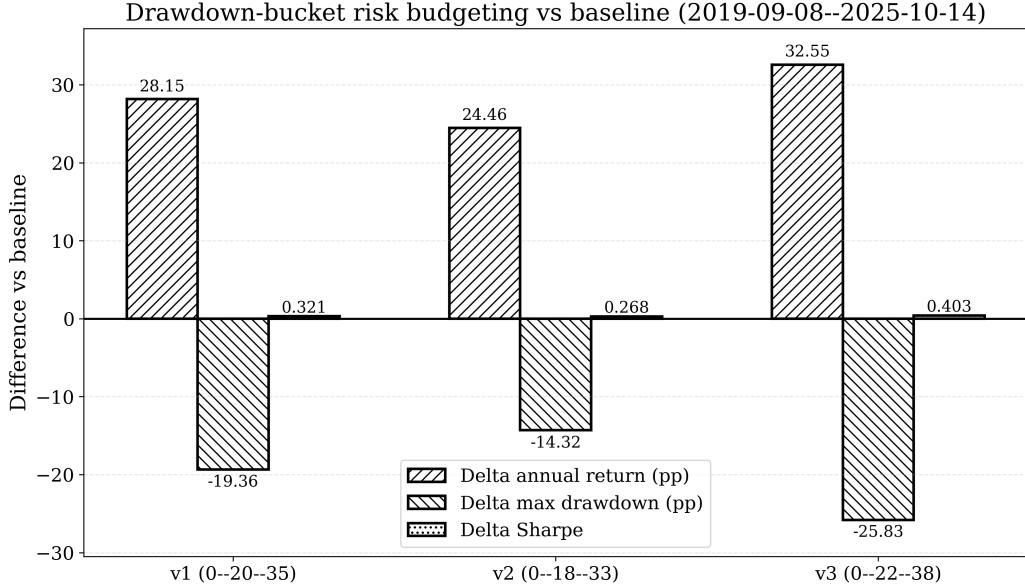


Figure 5: STRICT4H baseline vs drawdown-bucket variants over the 2019-09-08–2025-10-14 BTC/USDT 4h long window: differences in annualized return (pp), maximum drawdown (pp), and Sharpe ratio (raw) versus the STRICT4H baseline.

Fifth, we assess the impact of cost mis-specification more directly on an ETH STRICT4H aggressive configuration using the cost-sensitivity grid in the ETH full-chain replication output (Appendix Appendix A). In this grid we perturb the baseline cost assumptions (fee and funding multipliers around the baseline profile) while holding the signal and STRICT4H execution semantics fixed. Across plausible cost scenarios, annualized returns and Sharpe can compress materially relative to the baseline, underscoring that moderate underestimation of trading frictions can distort conclusions about apparent profitability even when execution semantics are held constant.

Finally, we briefly discuss the conceptual impact of the Live Guard. While the guard’s actions do not alter the backtest results, its logged “watch” and “kill” triggers from paper-trading deployments and historical guard simulations help identify periods where the strategy’s live performance begins to diverge from expectations set by the STRICT4H backtests. These experiments provide qualitative, configuration-specific insights into how the guard may function as a final layer of risk control in practice, but they stop short of establishing a long-horizon, production-level track record.

7. Discussion and Limitations

The framework presented in this paper offers a systematic approach to mitigating common pitfalls in quantitative strategy development for cryptocurrency perpetuals. By prioritizing methodological rigor and end-to-end consistency, it shifts the focus from the search for a “perfect alpha” to the process of identifying more auditable and less fragile trading-system configurations. However, this approach entails a series of trade-offs and is subject to several important limitations. At a conceptual level, our results support the view that, in high-friction cryptocurrency perpetual markets, realistic execution constraints and a double-screened auto-tuning pipeline are prerequisites for producing auditable backtests, whereas naive one-stage tuning under simplified costs can generate a substantial illusion of profitability.

At a higher level, our results can be interpreted as a practical protocol for realistic auto-tuning of perpetual-futures strategies in high-friction markets. In a typical deployment, a practitioner would (i) validate data integrity and cost profiles; (ii) run a Bayesian search under realistic costs on a primary window; (iii) subject the resulting candidates to evaluation-window and cost-stress screening on multiple windows that are held out from Stage I but reused within Stage II; (iv) interpret the surviving configurations in light of multiple-testing diagnostics such as the deflated Sharpe ratio; and (v) enforce full-chain accounting invariants and live guard triggers during paper-trading and production deployment. While our empirical analysis concentrates on BTC/USDT, ETH/USDT, SOL/USDT, and AVAX/USDT, the protocol itself is deliberately formulated at a higher level and could in principle be adapted to other derivative markets with similar frictional structures.

Scope note: AutoQuant should be interpreted as an execution-aware configuration-selection and auditing system, not as a generative strategy-discovery engine. The inference step searches within a pre-specified signal family and can be viewed as a disciplined curve-fitting process; its value lies in making the fitting assumptions explicit (execution, costs, and window semantics) and in exporting auditable artifacts and invariants for governance. Research on grammar-based or evolutionary discovery of new trading rules is a separate direction and is left to future work.

7.1. Scope of the profitability illusion

The profitability illusions documented in this manuscript arise from three concrete mechanisms within our BTC/USDT STRICT4H case study. First,

naive cost modeling can substantially overstate achievable performance: Table 18, based on a BTC/USDT STRICT4H stress-test configuration on the core sample, shows that progressively removing friction channels inflates annualized performance. In particular, the annualized return increases from 2.726 (fully costed, fees+slippage+funding) to 4.308 under fees-only, and to 5.225 under a zero-cost upper bound (all CAGR in decimal units). These magnitudes are diagnostic and are not intended to be read as realistic, capital-scalable outcomes for the production baseline in Table 14. Second, treating a single evaluation window as permanently “out-of-sample” obscures its role in parameter selection. The comparisons in Table 14, based on STRICT4H walk-forward runs over the training window (2019-09-08–2021-01-01) and the validation window (2021-01-01–2023-01-01), reveal that the validation segment simultaneously serves as a validation-style robustness check in Stage II and as the source of performance numbers often reported as out-of-sample, so we explicitly interpret it as a validation window rather than as a fully untouched test set. Third, repeated parameter search across many configurations and windows introduces multiple-testing risk, which we address only partially through the DSR-style sanity check discussed in Section 2.6 and through conservative interpretation of the empirical results.

Beyond methodological correctness, these mechanisms have practical consequences: they can encourage overconfident capital allocation, under-provisioned risk buffers, and brittle deployment of quantitative components inside production decision-support systems. In high-friction markets where limits to arbitrage are first-order, an explicit audit trail that makes execution semantics, funding alignment, and window-reuse rules observable is therefore not only a research convenience but also a governance and risk-management requirement for auditable, rule-based quantitative tooling.

As a result, the performance overestimation discussed in this paper should be understood in a narrow, data-driven sense: it refers to the extent to which BTC/USDT STRICT4H backtests can be inflated by optimistic cost assumptions, by reusing a validation window as if it were strictly out-of-sample, and by implicit multiple testing within our specific AutoQuant pipeline. We do not claim to have exhaustively characterized all sources of backtest illusion across asset classes, strategy families, or modeling choices. Within this scope, the BTC/USDT STRICT4H configuration provides the primary, fully audited evidence base, and we also report parallel one-stage versus two-stage comparisons on ETH/USDT, SOL/USDT, and AVAX/USDT under identical strict execution semantics (Tables 15, 19, and 20). Using the same

accounting and execution assumptions on assets with distinct volatility and funding dynamics, these replications support pipeline-level portability across four liquid perpetual contracts, but they do not constitute a full multi-asset validation of the framework and do not imply that similar parameterizations will transfer to less liquid assets.

A primary trade-off exists between the reusability of this execution-centric framework and the potential for bespoke optimization. By treating the signal engine as a black box, our system can be readily applied to a wide variety of underlying strategy logics. The cost of this generality is that it may not achieve the same level of finely-tuned performance as a pipeline designed monolithically around a single, specific alpha. Furthermore, the robustness of the multi-stage validation process comes at a significant computational cost. The extensive Bayesian search and the numerous backtests required for the double-screening protocol are resource-intensive, demanding substantial computational power and time. This complexity also extends to maintenance; ensuring the integrity of the full-chain invariants and the modular pipeline requires disciplined engineering and careful logging infrastructure.

The framework’s effectiveness is also fundamentally dependent on the quality of the underlying data. While our data validation module (Module A) enforces strict continuity and integrity checks, it cannot correct for latent flaws in the historical data provided by exchanges. The “garbage in, garbage out” principle remains a central challenge, and the system’s output is only as reliable as its inputs.

Finally, it is crucial to acknowledge the problem of model risk. Our framework is designed to identify parameter sets that have been robust to historical market conditions, but it does not, and cannot, eliminate the risk of a fundamental structural break in the market. The Live Guard mechanism serves as a reactive, last-line-of-defense against such events, but its detection capabilities are performance- and accounting-driven. At present, the guard does not perform explicit statistical tests for real-time structural break detection (e.g., by monitoring changes in market volatility or return distributions). It detects regime changes only indirectly, via their impact on performance metrics such as drawdown, rolling returns, or prolonged losing streaks. A parameter set that was historically profitable may therefore still fail if the underlying market dynamics change in a way that is not immediately captured by these performance-based triggers.

An important statistical caveat concerns multiple-testing bias and formal performance validation. The AutoQuant pipeline generates and evaluates

many candidate parameter sets, which raises the risk that some apparently strong backtest results are artifacts of repeated trial-and-error. As a complementary diagnostic (not a substitute for our walk-forward and cost-stress evidence), we report a deflated Sharpe-ratio sanity check in the spirit of [13]. We treat each Stage II configuration–window combination as one trial and compute a simple closed-form DSR-style statistic from the observed long-window Sharpe and an effective trial count (on the order of a few hundred to a few thousand in our logs, around 360–3240 depending on whether the cost-scenario grid is counted). In the current reproducible snapshot, the long-window Sharpe is close to zero, and the DSR-style diagnostic does not provide strong evidence against a zero-true-Sharpe null after accounting for multiple trials. We therefore interpret the DSR-style statistic only as a consistency check and place greater weight on the structural safeguards provided by our multi-window screening and cost-stress experiments. Complementing this DSR-style check, we report a CSCV/PBO-based diagnostic [1] computed on the Stage I candidate pool in our replication package. This diagnostic estimates the probability that an apparent in-sample winner fails to rank well out-of-sample across combinatorial train/test splits of the monthly return series. In our reproducible BTC snapshot, the PBO estimate is materially above zero, reinforcing that overfitting risk remains substantial and that the manuscript’s primary contribution is auditability and bias control rather than statistical proof of alpha. The underlying numeric summary is exported as part of the replication artifacts described in Appendix A.

Our empirical analysis also distinguishes between pre- and post-ETF periods through explicit windowing. The core BTC/USDT 4-hour sample (2019-09-08–2021-12-31) anchors our descriptive statistics and the naive-versus-rigorous cost inflation experiment. The extended long series (2019-09-08–2025-10-14) is used to define the training and validation windows in Table 14, to compute long-horizon diagnostics such as the deflated Sharpe-ratio statistic, and to run post-ETF hold-out experiments in which we extend a small number of pre-selected parameter sets to the 2024–2025 blind window without any additional re-optimization. As a result, the performance levels in our main tables should be interpreted as pertaining primarily to the training/validation-era experiments, while the long-horizon and post-ETF windows serve as additional stress tests rather than as a fully re-tuned, post-ETF validation.

7.2. Limitations

Our study is subject to several specific limitations that point toward avenues for future research:

Limited Asset Coverage: The empirical evaluation in this paper focuses on four liquid perpetual contracts (BTC/USDT, ETH/USDT, SOL/USDT, and AVAX/USDT) within a single STRICT4H strategy family. While these are significant instruments, the framework’s performance and the specific parameter sets identified may not generalize directly to other, less liquid crypto assets with different volatility and cost profiles. Contract lifecycle and survivorship: We restrict attention to large, continuously traded perpetuals on major venues in our reproducible snapshot. We do not systematically study delistings, symbol migrations, or exchange-specific contract re-specifications, which can introduce survivorship and data-availability biases in broader cross-sectional studies; we therefore do not claim portability of the reported parameterizations to newly listed, discontinued, or thinly traded contracts. While the long-horizon diagnostics span multiple market regimes, we do not separately isolate black-swan event windows (e.g., LUNA/FTX-style episodes) as dedicated stress tests in the main text and leave such event-conditioned analyses to future work.

External Benchmarks Under Identical Semantics: While we contrast one-stage and two-stage selection, include buy-and-hold as a simple yardstick, and provide cost- and window-based stress tests, we do not report a full end-to-end benchmark against third-party backtesting or AutoML toolkits under identical strict execution and funding-alignment semantics. Many off-the-shelf frameworks can be configured to approximate realistic costs, but subtle differences in timestamp alignment, funding availability conventions, and execution semantics can dominate results in perpetual markets. As a minimal external audit, we include a third-party semantics replay that reproduces our strict close-to-close, $t + 1$ raw-return accounting in Backtrader (Appendix Appendix A). A fair, full external benchmark of end-to-end pipelines would still require re-implementing STRICT4H-style invariants and the same data-validation protocol inside each baseline, which we leave to future work.

Optimizer baselines: Although we justify TPE as a practical sampler for conditional mixed search spaces, our optimizer evidence is intentionally limited. We report a minimal baseline comparison of TPE versus random sampling under a matched strict-backtest evaluation budget (Table 5), but we do not provide a broader head-to-head study against evolutionary methods

or other Bayesian samplers. We therefore frame the contribution as the execution- and auditability-preserving pipeline (strict semantics, cost realism, and policy-driven screening) rather than as a claim that TPE is uniquely optimal for this setting.

Data-Construction Disclosure: Section 4.2 documents our core OHLCV construction protocol (UTC bar-open indexing, sanity filtering, de-duplication, and continuity validation) and points to the scripts used to rebuild long-window 4-hour candles and funding baselines. Nevertheless, several exchange-specific data-construction choices can still matter in perpetual markets (e.g., whether a venue’s OHLCV reflects last-trade, mark, or index pricing; cross-venue aggregation rules; and handling of contract-specification changes). These details are implemented in the replication code and configuration templates and remain a natural target for a dedicated data-pipeline audit in future work.

Linear Cost Model and Position Scaling The cost profile we use in the STRICT4H experiments assumes an initial capital of 10,000 USDT, a maximum leverage of $5\times$, and a notional cap of 50,000 USDT, consistent with the baseline cost assumptions described in Section 4.4. The reported annualized returns and Sharpe ratios should therefore be interpreted as describing the behaviour of small- to mid-sized positions in a frictional but non-impact world. In particular, we do not model non-linear market impact or exchange-specific liquidity constraints, so we do not claim that the same percentage returns would be attainable at institutional asset levels (for example, 10 million USD or more). Consistent with this view, we treat the very large final-equity figures that arise in long synthetic compounding runs in our simulation logs and machine-readable exports as accounting artifacts, and we deliberately omit them from the main tables so as not to suggest unrealistic capital scalability. More broadly, our cost model is linear in turnover and does not capture non-linear impact (e.g., the empirical square-root law of market impact; see [36–38]); incorporating impact-aware execution models is an important direction for future work, especially for less liquid assets or larger AUM settings. All data and code required to reproduce the reported results are organized within the unified codebase and the anonymized replication materials described in Appendix Appendix A, which we provide to editors, reviewers, and qualified researchers on reasonable request, subject to the data-access constraints discussed above.

Alpha Logic Constraints: While the framework is signal-modular at the architectural level, integrating highly complex, non-parametric machine learn-

ing models (e.g., deep neural networks) would require additional engineering. While the backtest engine can readily consume signals from a pre-trained model at inference time, the current auto-tuning pipeline is not designed to manage the separate training, validation, and GPU resource management loops required by such models, and the empirical evidence in this paper is limited to a STRICT4H momentum-style family on BTC/USDT, ETH/USDT, SOL/USDT, and AVAX/USDT. Relatedly, automated *candidate generation* (e.g., factor-library composition, template expansion, or genetic programming / symbolic regression that proposes new functional forms) is outside the scope of this manuscript. Such generators can be integrated as upstream modules, but their outputs should be evaluated under the same strict $t+1$ semantics, cost modeling, and multi-window screening used here.

8. Conclusion and Future Work

In this paper, we introduced a systematic framework for the auto-tuning and multi-stage validation of quantitative strategies for cryptocurrency perpetual futures. We demonstrated, with BTC/USDT as an audited anchor case study and with replications on ETH/USDT, SOL/USDT, and AVAX/USDT, that common backtesting practices which neglect realistic costs or are prone to look-ahead bias can lead to a material overestimation of strategy performance. Our primary contribution is a structured, two-stage selection pipeline (Stage I search + Stage II screening) complemented by a post-deployment layer of monitoring and risk management. Concretely, the pipeline combines Bayesian optimization under realistic cost models, a rigorous “double-screening” protocol using held-out and cost-stress scenarios, and a live-style audit/guard overlay.

Our empirical results on multi-year data across the four contracts indicate that, in this specific high-friction setting, enforcing realistic execution and costs can materially change candidate rankings and reveal degradation on a held-out validation window, as evidenced by the comparisons in Table 14 and Section 6.2. The ETH results in Tables 19 and 20 provide a four-asset replication (as a pipeline-level portability check) of the one-stage versus two-stage comparison under identical strict execution semantics (with more conservative cost profiles for mid-caps), but they should not be interpreted as a broad multi-asset validation beyond these four assets. By enforcing full-chain accounting invariants and employing a live guard mechanism, the framework aims to provide a practical and auditable pathway from research

to live execution within this asset class. We view this as a design objective supported by preliminary offline and paper-trading evidence rather than as a guarantee that live performance will match backtests; realized results remain subject to model risk, structural breaks, and implementation constraints.

From an expert-systems and decision-support perspective, the practical implication is that quantitative evidence in 24/7 derivatives markets should be delivered as an *auditable artifact*: a complete specification of timestamp conventions, execution semantics, cost and funding alignment, and window-reuse rules, together with deterministic scripts that regenerate the reported tables and figures. To support this auditability goal, our Appendix Appendix A includes both internal full-chain consistency checks and minimal external semantics replays.

Managerial implications. For practitioners building algorithmic trading or risk-control pipelines in 24/7 derivatives markets, our evidence supports a set of concrete governance practices: (i) treat execution semantics and funding alignment as non-negotiable policies rather than as optional backtest settings; (ii) pre-register window semantics and selection rules to avoid hidden reuse of validation segments; (iii) maintain deterministic, per-run artifacts (signals, exposures, and cost decompositions) to enable independent audits; and (iv) deploy lightweight guard overlays that monitor invariant violations and reconcile offline and live-style replays.

Finally, we stress that all numeric results reported in the tables and figures can be exactly reproduced from the unified codebase and the anonymized replication materials described in Appendix Appendix A. These materials bundle cleaned 4-hour OHLCV and funding-rate series where redistribution is permitted, together with scripts that re-download and rebuild the cleaned series when redistribution is restricted; they also include walk-forward result tables, configuration files, and execution scripts with step-by-step instructions, allowing qualified researchers to reproduce the backtests and validation procedures reported here, subject to the data-access constraints discussed above.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability

Due to exchange licensing and storage constraints, we do not redistribute raw tick-level data. The replication materials described in Appendix Appendix A provide scripts and configuration templates that allow qualified researchers to re-download data from original sources (subject to exchange terms) and to regenerate the cleaned 4-hour OHLCV and funding-rate series used in our experiments.

Appendix A. Reproducibility, Code, Data, and AI Tools

We place a strong emphasis on reproducibility and transparency. All experiments in this paper were conducted using a single, unified codebase that implements the backtest engine, Bayesian optimization pipeline, double-screening protocol, and live guard logic described above. For this manuscript we maintain an anonymized replication package, built from this codebase, that provides the core 4-hour datasets (and/or scripts to regenerate them), configuration files, and scripts needed to reproduce the reported backtests, validation procedures, and tables. This package is shared with editors, reviewers, and qualified researchers on reasonable request, subject to exchange data-licensing and storage constraints. The numeric results shown in the main text are generated directly from simulation outputs to ensure consistency between the manuscript and the underlying experiment logs.

For the core tables and figures in this manuscript, the replication package includes deterministic scripts that regenerate the machine-readable outputs used to produce the tables and figures. The replication package contains a brief README that lists the exact commands and expected output filenames for each table and figure, together with brief notes on input data and runtime. For transparency, the replication package also includes a small script that regenerates the figures included in this L^AT_EX submission.

Example commands (Windows, local venv).

```
.\.venv\Scripts\Activate.ps1  
py scripts\paper_assets\make_figures.py
```

```

py tools\gen_4h_from_1h.py
py tools\generate_synthetic_funding_from_ohlcv.py

```

This repository snapshot includes derived machine-readable artifacts under `reports/paper/` that are used to generate the tables and figures. End-to-end regeneration scripts for all tables (and optional third-party semantics replays) are provided in the companion replication package described above; because full-chain logs can be large, this snapshot includes the derived artifacts used in the manuscript.

Table A.23: Minimal audit artifacts exported by AutoQuant. Each run exports machine-readable files that allow independent regeneration of tables/figures and reconciliation of net returns into raw-return and cost components. Exact filenames and schemas are documented in the replication README.

Artifact (format)	Contents (examples)	Audit purpose
Configuration (json)	θ , cost profile, window spec, seeds	Reproducible run specification and provenance
Per-bar ledger (csv)	timestamp, signal S_t , exposure π_t , r_t^{mkt} , r_t^{raw} , $C_{\text{fee},t}$, $C_{\text{slip},t}$, $C_{\text{fund},t}$, r_t^{net}	Full reconstruction of metrics and cost decomposition
Robust summary (csv)	window/scenario aggregates (monthly geom, MaxDD, turnover)	Stage II screening inputs and sensitivity reporting
Invariant audit (json/csv)	max $ \Delta S $, max $ \Delta \pi $, component diffs	Backtest-to-replay consistency checks

Execution-semantics sanity check. As an additional minimal audit that can be reproduced from the repository snapshot, we report how performance changes when the same 4-hour strategy parameters are evaluated under strict $t+1$ execution versus a deliberately naive same-bar ($t+0$) execution convention. This is a semantic sensitivity check only; it is not presented as a realistic trading assumption and is not used for parameter selection. Values in Table A.24 are generated from the replication artifacts described above.

Batch semantics uplift (Stage I top- N). To avoid over-interpreting a single parameter setting, we also report a batch summary over the top- N Stage I trials: we re-evaluate each trial on the same hold-out window under STRICT $t+1$ versus NAIVE $t+0$, keeping all other settings fixed. This batch audit quantifies whether $t+0$ execution *systematically* inflates performance in our snapshot. For BTC/USDT on 2022-01-01–2023-01-01 using the top $N=30$ Stage I trials, the median uplift (NAIVE minus STRICT) in annualized

Table A.24: **Execution-semantics sanity check (BTC/USDT 4h; 2022-01-01–2023-01-01).** Fixed parameters; only the execution delay differs: strict $t+1$ versus same-bar $t+0$. Annual return is CAGR in decimal units; monthly geometric mean is in decimal units; MaxDD is a fraction in $[0, 1]$.

Semantics	Ann. (CAGR, dec.)	Monthly geom (dec.)	Sharpe	MaxDD	Trades
STRICT ($t+1$)	-0.221		-0.021	-2.413	0.221
NAIVE ($t+0$)	-0.264		-0.025	-2.998	0.264

return is negative and the fraction of trials with positive uplift is below 50%, indicating that naive same-bar execution does not act as a universal performance booster in this setting. The batch outputs are generated by a deterministic helper script included in the companion replication package.

Table A.25: **Batch execution-semantics uplift (BTC/USDT 4h; 2022-01-01–2023-01-01; top $N=30$ Stage I trials).** Uplift is NAIVE($t+0$) minus STRICT($t+1$) in annualized return (CAGR, decimal units).

N	Median uplift	25th pct	75th pct	% uplift > 0
30	-0.018	-0.361	0.033	0.367

Third-party semantics replay (Backtrader). As a minimal external audit of our strict close-to-close $t+1$ execution accounting (raw returns only; no costs), we replay the same directional signal series in the Backtrader engine under cheat-on-close execution and compare the resulting per-bar return series to the vectorized STRICT4H raw-return construction. On BTC/USDT 4h over 2022-01-01–2023-01-01, the maximum absolute per-bar difference is on the order of 10^{-6} and the correlation is effectively 1.0, supporting the claim that the paper’s strict raw-return semantics are not an artifact of a single bespoke implementation. Outputs are generated by a deterministic helper script included in the companion replication package.

Table A.26: **Backtrader raw-return replay check (BTC/USDT 4h; 2022-01-01–2023-01-01).** Raw returns only; costs and funding disabled for this semantics-only audit.

Bars	Max $ \Delta $	RMSE	Corr.
2190	2.75×10^{-6}	2.94×10^{-7}	1.000

The historical BTC/USDT, ETH/USDT, SOL/USDT, and AVAX/USDT perpetual data used in this study were obtained from major centralized exchanges (primarily Binance, Bybit, and OKX). Due to exchange licensing and storage constraints, we do not redistribute raw tick-level data. The replication materials include scripts and configuration templates that allow qualified researchers to re-download data from original sources (subject to exchange terms) and to regenerate the cleaned 4-hour OHLCV and funding-rate series used in our experiments. For example, `tools/fetch_klines_incremental.py` rebuilds 4-hour OHLCV; `tools/funding_backfill_from_binance.py` backfills exchange funding; and `tools/generate_synthetic_funding_from_ohlcv.py` provides a deterministic funding baseline aligned to the STRICT4H grid. Processed summary statistics and anonymized walk-forward tables can be shared with editors and reviewers on reasonable request during peer review.

High-frequency guard experiments. As part of our broader research agenda we conducted a series of high-frequency ATR-guard experiments on 1m–60m resampled BTC/USDT data; these auxiliary experiments are not part of the main empirical results reported in this paper. These experiments applied ATR-based stop and take-profit rules on intraday bars on top of the STRICT4H baseline. Empirically, none of the tested configurations delivered a clear improvement in risk-adjusted performance once realistic fees and slippage were taken into account; most settings either left performance essentially unchanged or degraded it by introducing noise-induced turnover. We therefore interpret these results as context-specific evidence that, in the high-friction cost regime considered here, a 4-hour decision horizon is more suitable than naive 1m–15m overlays, while recognizing that more sophisticated high-frequency designs remain an open avenue for future work.

Declaration of generative AI and AI-assisted technologies

During the preparation of this manuscript, we used OpenAI Codex CLI (model: GPT-5.2) in a limited, assistive capacity for English copy-editing and tightening of non-technical prose, and for drafting internal checklists for reproducibility and table/figure traceability. After using this tool, we reviewed and edited the content as needed and take full responsibility for the content of the published article. All experimental results, numerical findings, and conclusions reported in this manuscript were generated and verified solely by the authors using the described codebase and exported artifacts. This AI

tool had no role in producing data, selecting configurations, or computing metrics.

References

- [1] D. H. Bailey, J. M. Borwein, M. López de Prado, Q. J. Zhu, The probability of backtest overfitting, *The Journal of Computational Finance* 20 (4) (2017) 39–69, accessed 2025-12-24. doi:10.21314/jcf.2016.322. URL <https://doi.org/10.21314/jcf.2016.322>
- [2] A. Shleifer, R. W. Vishny, The limits of arbitrage, *The Journal of Finance* 52 (1) (1997) 35–55, accessed 2025-12-24. doi:10.1111/j.1540-6261.1997.tb03807.x. URL <https://doi.org/10.1111/j.1540-6261.1997.tb03807.x>
- [3] Backtrader contributors, backtrader: Python backtesting library for trading strategies, <https://github.com/momentum/backtrader>, computer software. Accessed 2025-12-24 (2025).
- [4] Zipline contributors, Zipline: A pythonic algorithmic trading library, <https://github.com/zipline-live/zipline-reloaded>, computer software. Accessed 2025-12-24 (2025).
- [5] Freqtrade contributors, Freqtrade: Free, open source crypto trading bot, <https://github.com/freqtrade/freqtrade>, computer software. Accessed 2025-12-24 (2025).
- [6] AI4Finance-Foundation, Finrl: Deep reinforcement learning framework for quantitative finance, <https://github.com/AI4Finance-Foundation/FinRL>, computer software. Accessed 2025-12-24 (2025).
- [7] J. Bergstra, R. Bardenet, Y. Bengio, B. K'egl, Algorithms for hyper-parameter optimization, in: *Advances in Neural Information Processing Systems*, Vol. 24, 2011, pp. 2546–2554, accessed 2025-12-24. URL https://papers.nips.cc/paper_files/paper/2011/hash/86e8f7ab32cf12577bc2619bc635690-Abstract.html
- [8] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: *Proceedings*

- of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2623–2631, accessed 2025-12-24. doi:10.1145/3292500.3330701.
URL <https://doi.org/10.1145/3292500.3330701>
- [9] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, Journal of Machine Learning Research 13 (2012) 281–305, accessed 2025-12-24.
URL <https://jmlr.org/papers/v13/bergstra12a.html>
- [10] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
- [11] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359, accessed 2025-12-24. doi:10.1023/A:1008202821328.
URL <https://doi.org/10.1023/A:1008202821328>
- [12] N. Hansen, The cma evolution strategy: A comparing review, in: Towards a New Evolutionary Computation, Springer, 2006, pp. 75–102, accessed 2025-12-24. doi:10.1007/3-540-32494-1_4.
URL https://doi.org/10.1007/3-540-32494-1_4
- [13] D. H. Bailey, M. López de Prado, The deflated sharpe ratio: Correcting for selection bias, backtest overfitting, and non-normality, The Journal of Portfolio Management 40 (5) (2014) 94–107, accessed 2025-12-24. doi:10.3905/jpm.2014.40.5.094.
URL <https://doi.org/10.3905/jpm.2014.40.5.094>
- [14] T. Philippon, The fintech opportunity, Tech. Rep. w22476, National Bureau of Economic Research (2016). doi:10.3386/w22476.
- [15] P. Gomber, J.-A. Koch, M. Siering, Digital finance and fintech: current research and future research directions, Journal of Business Economics 87 (5) (2017) 537–580. doi:10.1007/s11573-017-0852-x.
- [16] F. Hutter, L. Kotthoff, J. Vanschoren, Automated Machine Learning: Methods, Systems, Challenges, Springer, Cham, 2019, accessed 2025-12-24. doi:10.1007/978-3-030-05318-5.
URL <https://doi.org/10.1007/978-3-030-05318-5>

- [17] M. López de Prado, Advances in Financial Machine Learning, John Wiley & Sons, Hoboken, NJ, 2018, accessed 2025-12-24.
URL <https://openlibrary.org/books/OL33645013M>
- [18] Y. Liu, A. Tsyvinski, Risks and returns of cryptocurrency, *Review of Financial Studies* 34 (6) (2020) 2689–2727. doi:10.1093/rfs/hhaa113.
- [19] Y. Liu, A. Tsyvinski, X. Wu, Common risk factors in cryptocurrency, *The Journal of Finance* 77 (2) (2022) 1133–1177. doi:10.1111/jofi.13119.
- [20] I. Makarov, A. Schoar, Trading and arbitrage in cryptocurrency markets, *Journal of Financial Economics* 135 (2) (2020) 293–319, accessed 2025-12-24. doi:10.1016/j.jfineco.2019.07.001.
URL <https://doi.org/10.1016/j.jfineco.2019.07.001>
- [21] D. Ackerer, J. Hugonnier, U. Jermann, Perpetual futures pricing, *Tech. Rep. w32936*, National Bureau of Economic Research (2024). doi:10.3386/w32936.
- [22] U. J. Jermann, Pricing perpetual futures, *SSRN Electronic Journal* (2023). doi:10.2139/ssrn.4573912.
- [23] W. Gornall, J. M. Rinaldi, Y. Xiao, Funding payments crisis-proofed bitcoin's perpetual futures, *SSRN Electronic Journal* (2025). doi:10.2139/ssrn.5036933.
- [24] Q. Ruan, Perpetual futures contracts and cryptocurrency market quality, *SSRN Electronic Journal* (2025). doi:10.2139/ssrn.5323703.
- [25] J.-C. Hung, H.-C. Liu, J. J. Yang, Trading activity and price discovery in bitcoin futures markets, *Journal of Empirical Finance* 62 (2021) 107–120. doi:10.1016/j.jempfin.2021.03.001.
- [26] A. Shynkevich, Impact of bitcoin futures on the informational efficiency of bitcoin spot market, *Journal of Futures Markets* 41 (1) (2020) 115–134. doi:10.1002/fut.22164.
- [27] A. Malik, A comparison of machine learning and econometric models for pricing perpetual bitcoin futures and their application to algorithmic trading, *Expert Systems* 40 (10) (2023). doi:10.1111/exsy.13414.

- [28] M. López de Prado, Seven sins of quantitative investing, *The Journal of Portfolio Management* 45 (1) (2019) 12–25.
- [29] J. Snoek, H. Larochelle, R. P. Adams, Practical bayesian optimization of machine learning algorithms, in: F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems*, Vol. 25, Curran Associates, Inc., 2012, pp. 2951–2959, accessed 2025-12-24.
URL https://papers.nips.cc/paper_files/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html
- [30] R. Sullivan, A. Timmermann, H. White, Data-snooping, technical trading rule performance, and the bootstrap, *The Journal of Finance* 54 (5) (1999) 1647–1691. doi:10.1111/0022-1082.00163.
- [31] H. White, A reality check for data snooping, *Econometrica* 68 (5) (2000) 1097–1126. doi:10.1111/1468-0262.00152.
- [32] P. R. Hansen, A test for superior predictive ability, *Journal of Business & Economic Statistics* 23 (4) (2005) 365–380. doi:10.1198/073500105000000063.
- [33] C. R. Harvey, Y. Liu, H. Zhu, ... and the cross-section of expected returns, *Review of Financial Studies* 29 (1) (2015) 5–68. doi:10.1093/rfs/hhv059.
- [34] F. Schär, Decentralized finance: On blockchain- and smart contract-based financial markets, *Federal Reserve Bank of St. Louis Review* 103 (2) (2021) 153–174, accessed 2025-12-24. doi:10.20955/r.103.153-74.
URL <https://doi.org/10.20955/r.103.153-74>
- [35] H. R. Künsch, The jackknife and the bootstrap for general stationary observations, *The Annals of Statistics* 17 (3) (1989) 1217–1241, accessed 2025-12-24. doi:10.1214/aos/1176347265.
URL <https://doi.org/10.1214/aos/1176347265>
- [36] R. Almgren, N. Chriss, Optimal execution of portfolio transactions, *The Journal of Risk* 3 (2) (2001) 5–39. doi:10.21314/jor.2001.041.
- [37] B. T
'oth, Y. Lemp

'eri
'ere, C. Deremble, J. de Lataillade, J. Kockelkoren, J.-P. Bouchaud,
Anomalous price impact and the critical nature of liquidity in financial
markets, *Physical Review X* 1 (2) (2011). doi:10.1103/PhysRevX.1.0
21006.

- [38] J. Gatheral, No-dynamic-arbitrage and market impact, *Quantitative Finance* 10 (7) (2010) 749–759. doi:10.1080/14697680903373692.