Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Using LLMs for Cryptocurrency Trade Signal Generation

Florian Schneider

florian.schneider@iu.org

IU International University of Applied Sciences

Jonas Delpho

IU International University of Applied Sciences

Visieu Lac

IU International University of Applied Sciences

Research Article

Keywords:

Posted Date: November 20th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11375846/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Using LLMs for Cryptocurrency Trade Signal Generation

Florian Schneider[1*], Jonas Delpho[1] and Visieu Lac[1]

[1]IU International University of Applied Sciences, Erfurt, Germany.

*Corresponding author(s). E-mail(s): florian.schneider@iu.org;
Contributing authors: jonas.delpho@iu-study.org; visieu.lac@iu.org;

Abstract

Large language models (LLMs) can generate structured cryptocurrency trade signals, but valid commands and stable decisions do not necessarily produce profitable trades. We evaluate an implemented signal-generation pipeline through 4,663 model-case-condition evaluations across six experiment families: repeated-call consistency, input representation and model comparisons, setup-selected market cases, input modality, prompt style, and model-specific knowledge-cutoff groups. Models receive historical price and volume tables, indicator-enriched representations, market-structure summaries, or chart images. Signals are validated and evaluated with a common historical backtester incorporating KuCoin contract trading fees and historical funding. Repeated GPT-5.5 calls often converge on stable directions and price levels, yet within-model and cross-model consensus do not consistently predict higher net profit and loss. Representation effects depend on model identity rather than following a universal advantage from richer context. Hybrid text-and-chart input leads the GPT-5.5 modality comparison, while visual-only input lags. Smart Money Concept instructions strongly alter analytical vocabulary and reshape the payoff distribution, but do not establish a profitability advantage. Temporal comparisons show no simple post-cutoff deterioration. The main performance contrasts remain exploratory and statistically uncertain. These findings characterize LLM behavior within a reproducible trading interface while identifying limits from signal selectivity, model provenance, and simplified execution; they do not establish robust live-trading profitability.

Keywords: large language models, cryptocurrency, trade signals, backtesting, multimodal input, prompt design

1 Introduction

Large language models (LLMs) are increasingly used as financial reasoning tools, but their role in trading systems remains difficult to evaluate. Much of the finance-oriented LLM literature studies text analytics, report summarization, sentiment extraction, question answering, or investment explanation. Cryptocurrency trading creates a different task. A model does not merely need to describe a market state. It must either abstain or emit a structured decision that can be parsed, validated, and simulated under explicit execution rules.

This paper studies that operational setting, but its research question is about *trade quality*, not parsing. Producing a structured command containing direction, entry levels, targets, stop loss, amount, and conviction-like metadata is a precondition; our question is how good those trades are once executed—are repeated decisions consistent, are they directionally stable, are they profitable after execution costs—and, crucially, *what makes them better or worse*: the input representation, the input modality, the prompt design, and the model identity. The input to the model is deliberately restricted to information constructed before the decision time: OHLCV candles, computed indicators, generated market-structure summaries, and, in the multimodal experiments, chart images. The output is not accepted as a narrative trading opinion; it is parsed into a machine-readable signal and evaluated through a common historical backtester.

The problem is scientifically interesting because LLM trading-signal generation combines several capabilities that do not necessarily move together. A model may write fluent technical analysis but fail to follow the command syntax. It may produce a valid command but place risk levels inconsistently. It may be directionally stable across repeated calls yet lose money in the backtest. It may benefit from indicators, or it may be distracted by a larger prompt. It may read charts usefully, or it may misinterpret visual structure. These are empirical questions, not properties that can be inferred from general LLM benchmarks or from classical technical-analysis studies alone.

The related literature motivates caution. LLMs can process financial text and some structured numerical inputs, but numerical, tabular, and time-series behavior is prompt-sensitive (Li et al., 2023a; Gruver et al., 2023; Hegselmann et al., 2023; Liu et al., 2024). Cryptocurrency forecasting studies show that OHLCV histories, engineered indicators, and market regimes can matter, but they also show that predictive accuracy and trading performance are not equivalent (Alessandretti et al., 2018; Mallqui and Fernandes, 2019; Sebastião and Godinho, 2021; Zhang et al., 2024b). Visual chart representations can contain useful financial structure, while vision-language models can still fail at chart and graph reasoning (Sezer and Ozbayoglu, 2018; Hung et al., 2021; Zhu et al., 2025b). Finally, backtesting, leakage, model memorization, and transaction costs are major threats to validity in any historical trading study (White, 2000; Sullivan et al., 1999; Bailey et al., 2014; Carlini et al., 2021).

All artifacts behind this study—the per-run model outputs, the computed backtest metrics, the exact prompts, the analysis code, and the notebooks—are released in a public repository.[1] Against that background, our contribution is an empirical study of an LLM-to-trade-command interface on 4663 model–case–condition evaluations.

[1] https://github.com/FlorianSchneiderIU/crypto-llm-results (pinned to the build commit in Appendix A).

The experiments examine repeated-call consistency, representation effects, model differences, scanner-labelled setup conditions, text/image modality, prompt style, validation-feedback behavior, and model-specific temporal robustness. This framing deliberately separates operational validity from trading value: a useful evaluation must track whether a model emits a parseable command, whether the command survives validation, whether repeated calls agree, whether the trade opens, and whether the resulting simplified backtest is profitable.

The experiments address six main empirical themes. First, repeated GPT-5.5 calls on identical market states test whether generated signals are random or stable. Second, the `ohlcv`, `small`, and `large` input levels test how information density changes behavior. The `small` level adds standard technical indicators to the base OHLCV table, while `large` adds generated market-structure and event context. Third, the random-case benchmark compares model labels under common case and representation conditions. Fourth, a setup-conditioned sample tests behavior on scanner-selected trend-pullback, breakout, and mean-reversion cases. Fifth, the modality and prompt-style experiments test text-only, chart-only, hybrid input, and a prompt augmented with Smart Money Concepts (SMC)—a discretionary market-structure and liquidity trading framework derived from the Inner Circle Trader (ICT) methodology—versus a free Baseline prompt. Sixth, the cutoff experiment asks whether the available pre/post-cutoff grouping shows simple post-cutoff degradation.

The strongest findings are descriptive rather than confirmatory. GPT-5.5 repeated calls often converge on a majority direction and stable price zones, but consensus is not the same as profitability; in the random-case benchmark, cross-model agreement under the `large` representation is often associated with worse rather than better mean net PnL. Representation interacts strongly with model identity, so richer information is not universally better. In the controlled modality ablation, hybrid input has the highest mean net PnL for GPT-5.5, textual input is close, and visual-only input is negative on average. The prompt-style experiment supports a strong output-style distinction between the SMC-augmented and Baseline conditions in the expected direction: the SMC-augmented responses contain far more explicit SMC/liquidity terminology, while the free Baseline responses are more indicator-heavy. Both prompt treatments are archived, so this is a clean manipulation; the performance result is nonetheless not a simple dominance claim. The cutoff experiment does not support a simple assertion that post-cutoff cases perform worse.

Accordingly, this paper should be read as a verifiable empirical map of model behavior under a specific signal-generation interface, not as a claim of deployable automated trading superiority. The reported PnL tables are net of KuCoin contract maker/taker trading fees and historical funding cashflows, but they still omit spread, slippage, liquidation mechanics, partial-fill uncertainty, and live execution constraints. The statistical tests are exploratory and the tested contrasts were not pre-registered. The paper's main value is to show which components of the interface behave reliably, which components remain fragile, and which claims require additional controlled experiments.

The rest of the paper is organized as follows. Section 2 positions the work in relation to LLM finance, cryptocurrency forecasting, market-state representation, multimodal chart analysis, and backtesting validity. Section 3 defines the experimental design,

market-state construction, input representations, prompt conditions, validation loop, backtester, baselines, and statistical analysis. Section 4 reports the empirical results and claim-status synthesis. Section 5 summarizes what the current evidence supports and what should be fixed or extended before stronger trading claims are made.

2 Related work

This study sits between four bodies of work: large language models in finance, cryptocurrency forecasting and technical trading, market-state representation, and backtesting validity. Prior work shows that LLMs can support financial text analytics, explanation, factor extraction, and decision support (Li et al., 2023b,a; Yu et al., 2023; Fatouros et al., 2024; Wang et al., 2024). A separate crypto-forecasting literature studies OHLCV histories, engineered indicators, sentiment, deep learning, and trading rules (Zhang et al., 2024b; John et al., 2024; Alessandretti et al., 2018; Mallqui and Fernandes, 2019; Sebastião and Godinho, 2021). The present setting is narrower and more operational: a general-purpose model receives a bounded historical market state and must either abstain or emit a parseable trading command with direction, entries, targets, stop loss, amount, and conviction-like metadata. That command is then validated and backtested. This differs from ordinary financial NLP, where outputs are usually labels or explanations, and from classical forecasting, where the trading rule is typically designed outside the model.

2.1 LLMs, Structured Data, and Financial Reasoning

Finance-oriented LLM studies motivate the experiment but do not settle it. Li et al. (Li et al., 2023a) evaluate LLMs on financial NLP tasks and show that capability is task-specific: fluent financial language does not guarantee robust constrained output, numerical consistency, or valid domain decisions. Work on financial time-series explanation and decision support (Yu et al., 2023; Fatouros et al., 2024; Wang et al., 2024) shows that LLMs can combine heterogeneous financial signals and produce interpretable reasoning. However, those studies usually ask for forecasts, rankings, factors, or explanations, not fully specified executable orders. Our parser-aware design therefore adds an additional layer: the output must survive syntactic parsing, semantic validation, and historical execution simulation.

The feasibility of textual market-state prompts is supported, but also limited, by work on numerical and tabular reasoning. LLMTime shows that numerical sequences can sometimes be serialized and processed by language models (Gruver et al., 2023), while later work warns that zero-shot time-series ability can be fragile under noise and representation changes (Park et al., 2025). Tabular-data studies similarly show that serialization format, table structure, and prompt wording affect model behavior (Hegselmann et al., 2023; Liu et al., 2024). Prompt-calibration work reinforces the same point: small prompt changes can change downstream answers (Zhao et al., 2021). These findings justify treating `ohlcv`, `small`, and `large` not as cosmetic variants, but as experimental factors that may change both validity and trading behavior.

2.2 Cryptocurrency Forecasting and Market Representation

Cryptocurrency markets have been studied with classical machine learning, deep learning, technical indicators, sentiment signals, and hybrid systems (Zhang et al., 2024b; John et al., 2024). Alessandretti et al. (Alessandretti et al., 2018) and Mallqui and Fernandes (Mallqui and Fernandes, 2019) show that OHLCV-derived histories can support predictive modeling in some settings. Sebastiao and Godinho (Sebastião and Godinho, 2021) emphasize a point that is central here: predictive accuracy and trading performance are not equivalent. A directionally correct forecast can still produce a poor trade if stops, targets, leverage, or costs are unfavorable; a less accurate strategy can perform well if payoff asymmetry is favorable.

Technical indicators and engineered features are therefore not merely decorative prompt content. Classical work on technical indicators and financial machine learning shows that feature choice, time window, and representation can materially affect predictions (Kara et al., 2011; Shynkevich et al., 2017; Patel et al., 2015; Qiu et al., 2016). Bitcoin and cryptocurrency studies reach similar conclusions for digital assets (Lee, 2024; Bouteska et al., 2024). In our experiments, the `small` condition externalizes indicator computation, while `large` adds market-structure and event context. Related work suggests two plausible outcomes: added structure may reduce the model's reasoning burden, or it may introduce noise and prompt overload. The effect must be measured rather than assumed.

Visual market representations provide another reason to test modality directly. Chart-image studies show that visual transformations of financial time series can encode useful information for learning systems (Sezer and Ozbayoglu, 2018; Hung et al., 2021; Liu and Si, 2022). At the same time, vision-language models can fail on chart and graph reasoning (Zhu et al., 2025b). Text prompts provide numerical precision and auditability; chart prompts provide spatial structure; hybrid prompts may combine both or create cross-modal conflict. This motivates the controlled comparison of textual, visual, and hybrid input.

2.3 Backtesting, Execution Costs, and Temporal Validity

The backtesting literature motivates caution about every performance claim. Data-snooping and multiple-testing problems are well known (White, 2000; Sullivan et al., 1999; Bailey et al., 2014), and time-series evaluation requires chronological discipline rather than random cross-validation in settings with temporal dependence (Bergmeir and Benítez, 2012; Cerqueira et al., 2020). Leakage can enter through data construction, feature engineering, model selection, and repeated informal testing (Kapoor and Narayanan, 2023). Consequently, this paper treats statistical comparisons as exploratory and reports multiple behavioral quantities: signal status, no-signal rates, metric-backed rows, win rate, mean PnL, account PnL, profit factor, and validation-feedback usage.

Execution modeling is another boundary. Market-microstructure and optimal-execution research shows that realized performance depends on spread, liquidity, impact, fees, and execution timing (Biais et al., 2005; Almgren and Chriss, 2001; Bertsimas and Lo, 1998). Cryptocurrency markets add fragmentation, venue differences, liquidity variation, exchange-specific constraints, and funding payments (Makarov and Schoar,

2020; Dong et al., 2022; Almeida and Gonçalves, 2024). Our reported PnL results include KuCoin contract maker/taker trading fees and historical funding cashflows, but they remain simplified historical simulations. They can show whether generated commands had favorable subsequent price paths under the implemented rules, but they cannot establish live profitability without spread, slippage, liquidation logic, order-book depth, account-tier calibration, and fill assumptions.

LLMs also introduce a temporal-validity concern that classical trading models do not have in the same form. Training data may include historical market episodes, analyses, or news, and language models can memorize or retain stale information (Carlini et al., 2021; Lazaridou et al., 2021; Zhang et al., 2024a; Zhu et al., 2025a; Ushio and Camacho-Collados, 2024). A model-specific pre/post-cutoff split cannot prove the absence of contamination, but it makes the issue visible. This is why the paper reports cutoff-group results and avoids treating pre-cutoff performance as clean evidence of forward generalization.

2.4 Generated Decisions, Confidence, and Explainability

A generated trading command is not just a forecast. It is an operational object. Malformed outputs, invalid risk levels, missing fields, inconsistent targets, and abstentions are meaningful outcomes rather than nuisance errors. Structured-output reliability should therefore be evaluated alongside profitability. A model that produces high PnL only on a small subset of parseable commands may be less operationally useful than a model with moderate PnL and high validity. Conversely, a model that reliably follows syntax but loses money demonstrates instruction following rather than trading skill.

Self-reported confidence or conviction is also not automatically calibrated. Calibration research shows that neural and language-model confidence can be unreliable (Lin et al., 2022; Kumar et al., 2024; Yin et al., 2023). A generated conviction score may reflect narrative fluency, prompt salience, or model certainty about syntax rather than expected profitability. Explainable-AI literature makes a parallel point: explanations improve auditability, but plausible explanations are not evidence of correctness (Guidotti et al., 2018; Barredo Arrieta et al., 2020; Weber et al., 2024). This paper therefore treats rationales and SMC/indicator language as behavioral evidence, while prioritizing parsed commands, validation outcomes, and backtest metrics for empirical claims.

2.5 Implications for This Study

The literature supports the study's design but also narrows its claims. It supports testing raw OHLCV, indicator-enriched, and market-structure-enriched inputs because representation can change model behavior. It supports testing text, chart, and hybrid modalities because visual financial patterns may matter, while general vision-language chart reasoning remains fallible. It supports reporting no-signal and invalid-output rates because generated commands are operational objects. It supports model-specific cutoff analysis because temporal contamination and world drift are plausible. It supports reporting explicit execution assumptions because even execution-cost-adjusted historical simulations are not live-trading evidence.

The research gap is therefore specific: prior work gives adjacent evidence that LLMs can process financial text, numerical sequences, tables, and charts, and that crypto markets can sometimes be modeled from historical features. It does not show whether a general-purpose LLM can convert a bounded crypto market state into a valid, risk-managed, and historically testable trading command. This paper addresses that gap as an exploratory empirical study rather than as a claim that LLMs replace specialized forecasting models or production trading systems.

3 Methodology

This study evaluates whether large language models can convert historical cryptocurrency market states into executable trade signals. The task is not next-candle classification or free-form market commentary. For each market state, a model may either abstain or emit a structured command containing direction, entries, stop loss, targets, position size, and conviction. The retained command is validated, normalized, and evaluated on subsequent market data with the same backtesting engine across experiments.

The evaluation is organized as six experiment families, each isolating one factor; the detailed protocols follow in the subsections below. (1) A *repeated-decision* experiment issues many calls on identical market states to test whether signals are stable or effectively random. (2) A *representation/model benchmark* compares models across the `ohlcv`, `small`, and `large` information levels on a shared 100-case sample. (3) A *setup-conditioned* experiment evaluates scanner-selected trend-pullback, breakout, and mean-reversion cases. (4) A *modality* ablation contrasts textual, visual, and hybrid input under a fixed model and representation. (5) A *prompt-style* experiment compares a Smart-Money-Concept-augmented prompt against a free Baseline prompt. (6) A *temporal-robustness* experiment groups cases by their relation to each model's knowledge cutoff. The remainder of this section details the data handling, interface, validation, and execution model shared by all six.

3.1 Study Design and Unit of Observation

The empirical record contains 4663 reported model–case–condition evaluations across six performance experiment families plus validation-feedback diagnostics: repeated-decision consistency, representation/model benchmarking, setup-conditioned cases, modality ablation, prompt-style comparison, and temporal robustness. For the prompt-style and setup-conditioned families we report GPT-5.5 only: the prompt-style Baseline condition reused a multi-model run batch, and the setup families ran only partial coverage for the other models, so the auxiliary non-GPT-5.5 runs in these two families are excluded from the reported counts to keep the comparisons model-matched. Table 1 summarizes the observed run counts and metric-backed rows.

The basic unit of observation is one model response to one historical market state under one experimental condition. Some responses pass through more than one validation attempt before the final response is retained. Profitability summaries use metric-backed observations: a run enters these tables only when the backtester produced a metric record. Abstentions, invalid signals, and missing outcomes are therefore counted

Table 1: Experiment inventory. Metric rows are observations with backtest metrics; fee metrics are the execution-cost-adjusted reruns used for PnL summaries. The prompt-style and setup-conditioned families report GPT-5.5 only; their auxiliary other-model runs are excluded from the comparison.

Experiment	Attempts	Cases	Models	Details	Metrics	Fee metrics	No signal	Error	Zero PnL
Repeated-decision consistency	763	5	9	large, ohlcv, small	763	763	0	0	309
Representation/model benchmark	2100	100	7	large, ohlcv, small	1900	1900	165	8	224
Setup-conditioned cases	100	100	1	large	98	98	0	0	24
Modality ablation	300	100	1	large	288	288	5	0	72
Prompt-style comparison	200	100	1	large	193	193	1	0	47
Temporal robustness	1200	400	3	large	1126	1126	54	2	228

in coverage tables but are not silently converted into profitable or unprofitable trades. Zero-PnL metric-backed rows are retained because the experimental implementation explicitly treats limit orders that never open as zero-PnL observations when the target was reached before the entry.

The central experimental factors are:

- the market case, defined by symbol, timeframe, target timestamp, and historical prompt window;
- the model label;
- the textual information level: `ohlcv`, `small`, or `large`;
- the input modality: textual, visual, or hybrid;
- the prompt style, reported as the *SMC-augmented* and *Baseline* conditions;
- the market-setup label for scanner-selected cases;
- the model-specific temporal relation between the market timestamp and the model knowledge cutoff.

The main representation/model benchmark evaluates seven language models, all accessed through an institutional multi-provider chat layer: GPT-5.5, GPT-5.4, Claude Opus 4.8, Claude Opus 4.7, Gemini 3.1 Pro, Mistral Medium, and Qwen 3.5 (exact access details and identifiers in Appendix A.5). The repeated-decision archive additionally contains sparse Claude Opus 4.6 and Claude Sonnet 4.6 observations, giving nine model labels in that family's inventory; these auxiliary observations inform the cross-model agreement diagnostic, not the seven-model benchmark. GPT-5.5 serves as the reference model wherever an experiment uses a single-model design.

The design is unbalanced. Some experiments are deliberately single-model ablations, some are broad model comparisons, and some contain reruns in particular cells. Results are therefore reported at the experiment-family level rather than pooled as if all conditions formed one balanced factorial design.

3.2 Market Data and Anti-Lookahead Rule

All analyzed market states use four-hour cryptocurrency candles. For a target timestamp t_{target}, the data-construction procedure downloads a window ending before that timestamp:

$$t_{\text{start}} = t_{\text{target}} - (N_{\text{prompt}} + 200)\Delta t, \tag{1}$$

where N_{prompt} is the number of candles shown to the model and the additional 200 candles provide warm-up history for the longest implemented moving average. The

prompt itself uses the final $N_{\text{prompt}} = 50$ historical candles before the target timestamp. Candles at or after the target timestamp are excluded from model input and are reserved for outcome evaluation. This strict historical split is the main anti-lookahead control.

3.3 Input Representations

The experiments vary how much derived market context is exposed to the model. The `ohlcv` representation contains only open, high, low, close, and volume fields. The `small` representation uses the same price window but adds standard technical indicators, including moving averages, MACD, RSI, ATR, OBV, and related chart panels. The `large` representation further adds market-structure and event annotations such as Fibonacci levels, pivots, order blocks, fair value gaps, candlestick-pattern events, Bollinger events, support/resistance events, momentum and volume events, and ICT-style indicators.

Thus, the detail-level manipulation is representational rather than temporal: all three levels refer to the same historical market window, but they expose different transformations of that window. In result interpretation, this distinction matters because an apparent improvement from `large` can reflect the value of engineered context, not merely the language model's raw ability to infer that context from OHLCV rows.

3.4 Setup-Conditioned Sample

One experiment evaluates market states selected by a setup scanner rather than by random sampling. The archived setup labels divide 100 cases into three approximately balanced classes: trend pullback, breakout, and mean reversion. Each case records the symbol, timeframe, scanner direction, setup-candle time, setup close, and setup identifier. The executed setup-conditioned experiment uses only the `large` representation and is reported for GPT-5.5, which evaluated all 100 cases; Claude Opus 4.8 and Qwen 3.5 ran only a small partial subset and are excluded from the reported comparison.

The setup labels should be interpreted as the scanner's operational definitions of the three trading regimes rather than as externally validated regime ground truth. The scanner is deterministic and is released in full as `scripts/setup_scanner.py` in the reproducibility repository, with its complete rule set (context, location, and trigger conditions for the trend-pullback, breakout, and mean-reversion classes, including the exact thresholds and the stratified case-selection procedure) documented in Appendix A.4; the separate event-detection thresholds that feed the `large` representation are listed there as well (for example, a 1.8× volume-spike multiple over a 20-period average, a Fibonacci swing lookback capped at 200 bars, a minimum order-block size of three candles, and 70/30 RSI thresholds). The selection rules and labels are therefore independently reproducible; the remaining caveat is conceptual, namely that the results compare model behavior conditional on these scanner-defined labels rather than on an externally agreed setup taxonomy.

Table 2: Signal-structure summary after parsing. Allocation completeness is the share of signal rows whose target percentages sum to 100.

Experiment	Signals	Amount mean (%)	Conviction mean	Entries mean	Targets mean	Targets = 100% (%)
Repeated-decision consistency	763	1.83	0.61	2.20	3.79	100.0
Representation/model benchmark	1927	1.96	0.65	2.09	3.24	100.0
Setup-conditioned cases	100	1.81	0.60	2.38	3.55	100.0
Modality ablation	295	1.80	0.60	2.27	3.66	100.0
Prompt-style comparison	199	1.81	0.60	2.44	3.79	100.0
Temporal robustness	1144	1.79	0.59	2.27	3.45	100.0

3.5 Visual and Hybrid Inputs

The modality ablation tests whether chart images provide additional information beyond the textual prompt. It compares three input modes on the same 100 random market cases, all with GPT-5.5 and the `large` representation:

- textual input only;
- visual input only;
- hybrid input combining the textual prompt and chart image.

This is a controlled single-model modality experiment. It should not be read as a general multimodal benchmark across all tested models.

3.6 Prompting and Output Format

All experiments use a structured trading interface. The model is instructed to reason about the market state and then fill a fixed set of structured fields that the parser can read without further interpretation. The analytically relevant fields are:

- a *direction*: long, short, or an explicit abstention when no favorable setup is found;
- an *entry ladder* of one or more entry prices (market or limit, allowing multiple dollar-cost-averaging (DCA) entries);
- a *target ladder* of take-profit levels, each with a percentage allocation, where allocations sum to 100%;
- a *stop-loss* price;
- a *position amount* expressed as a percentage of account equity, tied to conviction;
- a *break-even trigger* (the target after which the stop is moved to entry) and a *conviction* score.

A small number of interface-control switches (for example a moving-stop-loss and a force-open flag) are also part of the format but are not analytically important here. The complete output specification shared by all experiments, including the exact field syntax used internally by the execution bot, is reproduced verbatim in Appendix A.1. Leverage is deliberately not requested from the model: it is recalculated downstream from the parsed entry and stop-loss risk using the deterministic formula $\text{leverage} = \lfloor p_{\text{entry}} / |p_{\text{entry}} - p_{\text{SL}}| \rfloor$ given in Appendix A.3, rather than trusted from the response.

The signal-structure audit in Table 2 checks whether retained signals match the intended interface. Across metric-backed rows, target allocations sum to 100%, the average number of entries is close to two, the average number of targets is close to three

or four, and position amount is consistent with the instructed relationship between account allocation and conviction.

3.7 Smart Money Concept Prompt Extension

The two prompt-style conditions differ in how tightly they constrain the model's reasoning. The *SMC-augmented* condition adds a Smart Money Concept/ICT extension on top of the general trading prompt, narrowing the model toward a specific analytical frame. The extension instructs the model to prioritize liquidity pools, liquidity sweeps, market-structure shifts, displacement, imbalances, order blocks, premium/discount location, institutional order-flow logic, invalidation beyond structural levels, and liquidity-based targets, and to treat conventional indicators as supporting evidence when they conflict with market-structure reasoning. The *Baseline* condition uses the general trading prompt alone and leaves the setup and strategy choice free.

Both treatment texts are archived verbatim: the shared general system prompt in Appendix A.1 and the SMC/ICT extension in Appendix A.2. The SMC-augmented condition is exactly their concatenation, and the Baseline condition is the general prompt without the extension, so the two prompts differ only by the presence of the SMC extension. The prompt-style comparison is therefore a clean, reproducible manipulation of SMC instruction rather than an uncontrolled difference; the response-language analysis in Section 4 serves as a manipulation check that the extension changed the model's reasoning vocabulary as intended. The remaining limits are on the *performance* inference, not on the prompt specification: the model-matched comparison is limited to GPT-5.5 and the statistics are exploratory.

3.8 Validation Feedback

Before a response is backtested, the parsed signal is scored for structural validity by a deterministic validator. The validator evaluates three checks, each returning a pass, warning, or fail status:

1. *Stop-loss logic*: the stop must sit on the correct side of the entries, and its distance must be reasonable (fail below 1% or wrong side, warning below 3% or above 20%, pass in the 3–20% band).
2. *Entry strategy*: a single entry is flagged as a warning; for multiple entries, the maximum spacing gap is checked (pass at most 5%, warning above 5%, stronger warning above 10%).
3. *Target allocation*: take-profit allocations must sum to 100% and the target prices must progress monotonically away from entry in the trade direction (fail above 100%, warning below 100%, pass at exactly 100% with valid progression).

Each status maps to a numeric weight (pass = 1, warning = 0.5, fail = 0), and the overall score is their mean expressed as a percentage. Scores map to qualitative ratings (excellent ≥ 80, good ≥ 60, poor ≥ 40, otherwise dangerous). The validator additionally computes, but does not score, risk diagnostics such as account risk and the allocation-weighted reward-to-risk ratio.

If the overall score is below the target threshold of 75%, the system returns natural-language feedback to the model and requests a revised signal. The feedback is not a numeric gradient: it is the concatenation of the human-readable status messages from each failed or warned check (for example, "Stop loss too tight (0.8%) – high chance of premature exit" or "Target allocation exceeds 100%") together with the generated risk recommendations. The model then resubmits within the same conversation, and the loop repeats up to a configured cap of three attempts; the final attempt is retained. Figure 1 summarizes the loop.

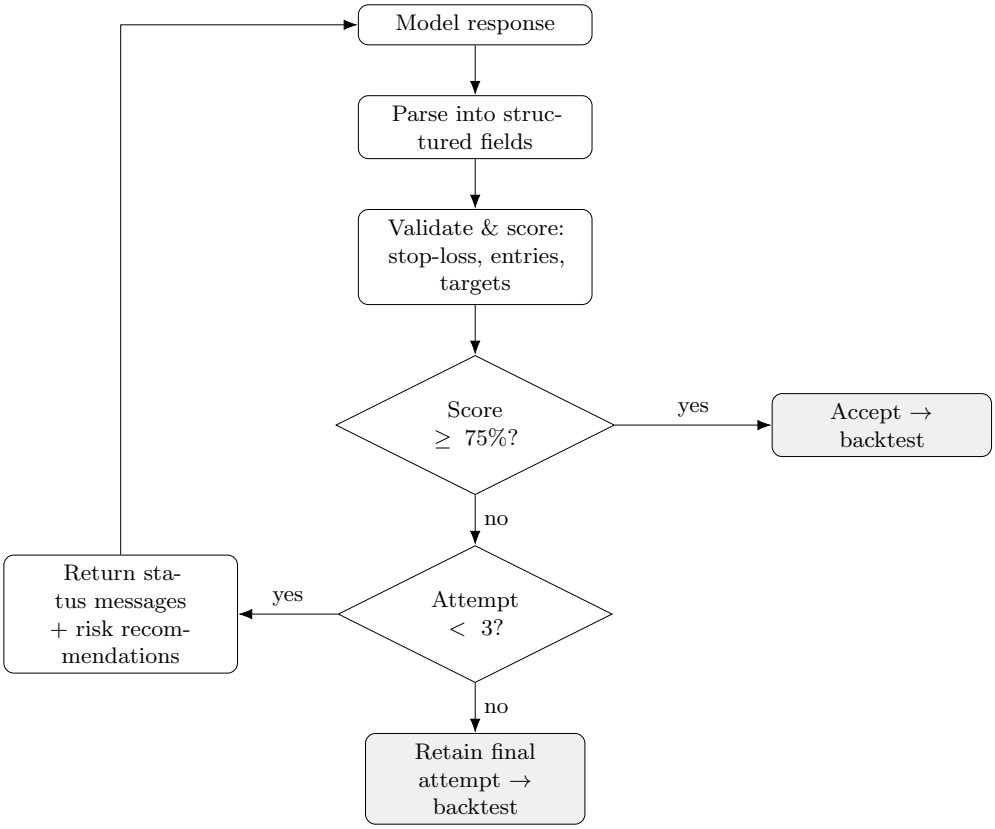


Fig. 1: Validation-feedback loop. A parsed signal is scored on three structural checks; if it falls below the 75% target and the attempt budget is not exhausted, the validator's human-readable messages are returned to the model for revision, up to three attempts.

The feedback mechanism is active in the archived experiments. Table 3 reports how often responses required more than one attempt or stored feedback. Overall, 345 of 4663 histories show evidence of feedback usage, and the maximum observed attempt count is three. All profitability results should therefore be interpreted as final outputs of the implemented model–validator loop, not necessarily as first raw model responses.

Table 3: Validation-feedback usage by experiment family. A history is counted as feedback-used when it contains more than one attempt, an attempt number greater than one, or a stored feedback message.

Experiment	Histories	Turns	Single attempt	Feedback histories	Feedback rate (%)	Feedback messages	Max attempt	Scored
Repeated-decision consistency	763	793	743	20	2.6	30	3	763
Representation/model benchmark	2100	2488	1897	203	9.7	388	3	1935
Setup-conditioned cases	100	108	95	5	5.0	8	3	100
Modality ablation	300	327	286	14	4.7	27	3	295
Prompt-style comparison	200	218	191	9	4.5	18	3	199
Temporal robustness	1200	1379	1106	94	7.8	179	3	1146
All scanned histories	4663	5313	4318	345	7.4	650	3	4438

The validation score is not a profitability score. It measures conformity to the trading-command interface and risk-structure rules. Because there is no matched feedback-disabled experiment, the present results can describe feedback usage and score changes but cannot estimate the causal effect of validation feedback on trading performance.

3.9 Signal Normalization and Backtesting

Parsed signals are normalized before evaluation. Entries and targets are sorted by direction, wrong-side targets are removed, target allocations are normalized when needed, percentage-based levels are converted to prices, and placeholder leverage is replaced by the stop-loss-distance-implied value of Appendix A.3. A model's literal text is therefore not always the exact object evaluated by the backtester; the evaluated object is the normalized trade command accepted by the parser and validator.

The backtester starts with capital 1000, applies the parsed entries, targets, stop loss, recalculated leverage, and position amount, and evaluates the signal on subsequent OHLCV data. It supports market and limit entries, multiple DCA entries, partial take-profit targets, break-even stop movement, stop-loss exits, and skipped trades when the target is reached before entry. When event ordering is ambiguous within a candle, lower-timeframe data are used where available to resolve the sequence.

The original archived metrics were gross of exchange costs. For the present manuscript, every metric-backed run that contributes to a PnL table was rerun through an execution-cost-aware version of the same backtester. The rerun preserves the original signal, timestamp, parser normalization, OHLCV replay rule, and event-ordering logic, but adds KuCoin USDT perpetual trading fees and historical funding cashflows. Contract metadata are read from KuCoin's active-contract endpoint. The tested pairs (BTC/USDT, ETH/USDT, DOGE/USDT, and XRP/USDT) map to the perpetual contract symbols `XBTUSDTM`, `ETHUSDTM`, `DOGEUSDTM`, and `XRPUSDTM`. Maker and taker rates are taken from the corresponding `makerFeeRate` and `takerFeeRate` fields.

For a filled execution with quantity q, execution price p, and liquidity-specific fee rate r, the trading fee is

$$\text{fee} = |qp|r. \tag{2}$$

Because the position size already includes leverage in the implementation, this fee is charged on leveraged notional rather than on unlevered account margin. The archived signals do not contain full exchange order flags, so the rerun uses the following deterministic fill convention: market entries, stop-loss exits, and manual closes are

taker fills; limit entries and take-profit targets are maker fills. Skipped trades receive no trading fee because no position is opened.

Funding is not applied from an assumed 8-hour calendar. For each contract and test interval, the rerun fetches KuCoin's historical funding-rate rows and uses the returned settlement timestamps. A position pays or receives funding only if it is open at a returned settlement timestamp. With side sign $s = 1$ for long and $s = -1$ for short, settlement notional V_t, and funding rate F_t, the signed account cashflow is

$$\text{funding cashflow}_t = -s|V_t|F_t. \tag{3}$$

Positive cashflow increases realized PnL; negative cashflow reduces realized PnL. The implementation uses the simulated position notional and the OHLCV replay price at the settlement-processing candle as the value proxy. This is more realistic than ignoring funding, but it remains an approximation because the archived experiments do not store mark-price snapshots or exchange fill confirmations.

The execution-cost rerun writes a parallel metric record for each PnL-bearing experiment output and keeps the original gross fields for audit. Thus, all main PnL tables and figures in the results section report net values after contract maker/taker trading fees and historical funding cashflows. The backtester still does not model spread, slippage, liquidation mechanics, order-book depth, exchange-specific constraints, latency, stochastic fill probability, or partial fills. A related effect deserves explanation: a few leveraged trades report a net trade PnL slightly below -100% (the worked case study below contains a -115% run). This does not come from a stop placed on the wrong side of the entry. Trade PnL is the leveraged return on the posted per-trade margin, i.e. the adverse price move measured from the average fill multiplied by the position leverage, and the leverage is sized so that the stop is approximately a full-margin (-100%) move. For multi-entry (DCA) signals this sizing is computed from a reference entry that is marginally tighter than the realized average fill, so the integer-rounded leverage can land just above the one-to-one value; the stop then realizes a loss a few percent beyond -100% of that margin, and trading fees and funding add a little more. Single-entry signals do not show this, because the reference price equals the only fill.

Whether such a value is realistic depends on the margin mode. Under *isolated* margin the position would be liquidated once its loss reached the posted margin, so a sub--100% trade PnL is an artifact of omitting liquidation. Under *cross* margin the position is backed by account equity and losses beyond the nominal per-trade margin are absorbed by the account, so a sub--100% trade-level figure is the expected accounting outcome rather than an artifact. Either way the effect is confined to the per-trade scale: account-level PnL multiplies each trade by its small position allocation and stays correspondingly small (the -115% trade above is only -2.1% of the account), so the aggregate results are unaffected. The results are therefore stronger than gross backtests but remain historical simulations rather than live-trading performance estimates.

Table 4: Final three prompt rows of the XRP/USDT case under the `small` representation (selected base and indicator columns shown; the model receives the complete table).

Time	close	volume	sma50	ema50	macd	rsi12	atr14
2025-01-07 00:00:00	2.405010	1924104	2.301058	2.343263	0.020859	52.672179	0.048293
2025-01-07 04:00:00	2.427690	3390848	2.307730	2.346574	0.020865	57.147474	0.047998
2025-01-07 08:00:00	2.421740	4655750	2.314129	2.349521	0.020157	55.641672	0.049015

3.10 Worked Case Study

Having defined the input representations, the structured output format, the validation loop, and the execution-cost-aware backtester, we now make the representation variable concrete on one conducted market state. We use a single XRP/USDT four-hour trend-continuation case dated 2025-01-07, with a target timestamp, 50 prompt candles, and the three text/detail conditions. All timestamps follow the archived construction convention (the case prompt ends at the 2025-01-07 08:00:00 candle boundary); this same convention is applied consistently to prompt construction, the anti-lookahead split, and outcome evaluation, so it does not bias the within-case comparison across representations.

The original chart images for this specific case were not retained with the conducted outputs. The visualizations below were therefore regenerated from the archived case metadata and the original case-construction logic. They illustrate the input levels for a conducted market state; they are not hand-drawn reconstructions and do not introduce manually chosen indicator values.

Indicator-Enriched Prompt Excerpt

The `ohlcv` condition is the base table with columns `Time`, `open`, `high`, `low`, `close`, `volume`, `timestamp`. The `small` condition keeps those base columns and adds `sma25`, `sma50`, `sma200`, `ema5`, `ema12`, `ema26`, `ema50`, `macd`, `macd_signal`, `macd_histogram`, `rsi6`, `rsi12`, `rsi24`, `atr14`. Table 4 shows the final three prompt rows with selected base and added columns; the model receives the complete OHLCV-plus-indicator table.

Market-Structure Prompt Additions

The `large` condition contains the full `small` table plus generated market-structure sections. The compact excerpt below shows representative advanced blocks that are absent from `ohlcv` and `small`.

```
### Latest Fibonacci Retracement:
- Swing High: 2.50888
- Swing Low: 1.99663
- Fib 0% (High): 2.50888
- Fib 23.6%: 2.387989
- Fib 38.2%: 2.3132004999999998
- ... 5 additional lines omitted in print excerpt

### Relevant Pivot Points:
- 2025-01-05 00:00:00 pivot_low level: 2.372520 (strength: N/A)
- 2025-01-05 08:00:00 pivot_low level: 2.325880 (strength: N/A)
```

Table 5: Representative GPT-5.5 runs for the XRP case study. For each detail level, the selected run is the majority-direction run whose net trade PnL is closest to the median net trade PnL within that majority direction. This deterministic rule avoids selecting unusually good or bad examples by hand.

Detail	Run	Majority	Consensus (%)	Signal	Entry 1	SL	Targets	Net trade PnL %	Net account PnL %	Skipped
ohlcv	26_GPT-5.5	LONG	97.7	LONG	2.386	2.297	4	-115.1	-2.14	no
small	12_GPT-5.5	LONG	93.0	LONG	2.367	2.198	4	0.0	0.00	yes
large	3_GPT-5.5	SHORT	69.8	SHORT	2.456	2.548	4	84.0	1.51	no

```
- 2025-01-06 00:00:00 pivot_high level: 2.442250 (strength: N/A)
- ... 3 additional lines omitted in print excerpt

### Order Blocks and FVG:
- bearish_order_block: None to 2024-12-26 08:00:00, level: 2.160100-2.316620
- bullish_order_block: None to 2024-12-28 12:00:00, level: 2.145550-2.194960
- bullish_order_block: None to 2025-01-01 12:00:00, level: 2.090000-2.233450
- ... 3 additional lines omitted in print excerpt

### Candlestick Patterns:
- 2025-01-07 08:00:00 spinning_top bearish (strength: 100)
- 2025-01-06 16:00:00 spinning_top bearish (strength: 100)
- 2025-01-06 08:00:00 doji bullish (strength: 100)
- ... 8 additional lines omitted in print excerpt

### Volume Events:
- 2024-12-02 08:00:00 OBV breakout (bull): 233080569 > 184257766 (gap
48822803 >= floor 45049502)
- 2024-12-02 04:00:00 OBV breakout (bull): 258479298 > 175404458 (gap
83074840 >= floor 43043323)
- 2024-12-02 00:00:00 Volume spike: 34635137 vs avg 17130619 (2.0x)
- ... 8 additional lines omitted in print excerpt

### Momentum Events:
- 2024-12-07 12:00:00 MACD crossed above signal line (0.0364 > 0.0343)
- 2024-12-03 16:00:00 MACD crossed below signal line (0.2729 < 0.2749)
- 2024-12-03 08:00:00 RSI entered overbought territory (77.2 > 70)
- ... 8 additional lines omitted in print excerpt

### Market Structure Events:
- 2024-12-05 08:00:00 Bollinger Band squeeze: width 25.21% vs avg 49.00%
- 2024-12-05 04:00:00 Bollinger Band squeeze: width 25.32% vs avg 50.04%
- 2024-12-05 00:00:00 Bollinger Band squeeze: width 26.18% vs avg 50.90%
- ... 8 additional lines omitted in print excerpt

### Support/Resistance Events:
- 2025-01-07 08:00:00 Strong resistance rejection at 2.45 (1.2% rejection,
close: 2.42)
- None
- None
```

Table 5 and Figure 3 connect the representation change to actual GPT-5.5 outcomes. The selected run for each detail level is chosen deterministically: within each detail level, the majority signal direction across repeated GPT-5.5 runs is identified, and the representative run is the majority-direction observation whose net trade PnL is closest to the median of that majority subset.



(a) Indicator-enriched (`small`) representation. (b) Market-structure-enriched (`large`) representation.

Fig. 2: The same XRP/USDT four-hour market window under two input representations, cropped to their top panels for legibility. Each chart shows only the price (OHLCV) candles, the volume panel, and the On-Balance Volume (OBV) panel. Panel (a), the `small` condition, adds standard technical context such as moving averages and Bollinger Bands to the base OHLCV table; panel (b), the `large` condition, additionally exposes the implemented market-structure overlays, including order blocks, fair-value gaps, and volume/OBV event markers. The remaining lower panels supplied in the same prompt—the Relative Strength Index (RSI) and the moving-average convergence/divergence (MACD)—are omitted here for space. The underlying price history is identical across panels; only the exposed derived context differs.

This example is not used as an aggregate performance claim. It shows why the experiments report consensus, signal status, and PnL separately. In this market state,

Fig. 3: Backtest paths for the representative GPT-5.5 signals of Table 5, replayed through the same execution-cost-aware backtester used for all results. The three panels hold the market state fixed and vary only the input representation; markers show the entries, take-profit targets, and stop loss exactly as the backtester filled them, so the partial fills and early/late exits are the engine's own event log rather than a separate illustrative simulation.

Table 6: Actual GPT-5.5 signals for the three representative case-study runs, with a condensed extract of the model's own reasoning (`## Thought summary`). Entries are the planned (DCA) entry prices, SL the stop loss, and each target is listed as price/allocation. The same market window produces a short in `large` and longs in `ohlcv`/`small`.

Detail	Dir.	Entries	SL	Targets (price/alloc.)	Conv.	Net %	Reasoning (condensed)
`ohlcv`	LONG	2.386, 2.348	2.297	2.445/30%, 2.475/30%, 2.508/25%, 2.56/15%	0.62	-115.1	XRP/USDT on the 4h chart is still structurally bullish after the impulsive breakout from the 2.10 area into the 2.45-2.50 supply zone.
`small`	LONG	2.367, 2.326, 2.266	2.198	2.424/20%, 2.468/30%, 2.508/30%, 2.585/20%	0.64	0.0	XRP/USDT on the 4h chart is still structurally bullish, with price holding above SMA50, SMA200 and the EMA50.
`large`	SHORT	2.456, 2.492	2.548	2.388/35%, 2.367/25%, 2.326/25%, 2.253/15%	0.6	84.0	XRP/USDT is trading around 2.42174 on the 4h chart, inside a short-term consolidation after the impulse from the 1.99663 swing low to the 2.50888 swing high.

GPT-5.5 is mostly long in the `ohlcv` and `small` conditions, while the majority direction flips to short in the `large` condition; the representative outcomes also differ sharply across the three representations.

3.11 Baselines

The original experiments did not include non-LLM trading baselines. To provide a scale reference for the representation/model benchmark, this draft includes deterministic bracket baselines on the same 100-case sample: always long, always short, deterministic random direction by case identifier, trend-following from the prompt window, and mean reversion against that trend. Each baseline uses the same simple bracket specification: 2% stop distance, 50x leverage, 1.8% account allocation, 1R/2R/3R targets, break-even after target 1, and a 24-hour horizon close. Each baseline trade is evaluated through the same fee- and funding-aware backtester as the LLM signals, and the results are reported in Section 4 (Table 20). These baselines are sanity-check scale references rather than optimized trading systems.

3.12 Performance Metrics and Statistical Tests

The main performance summaries report the number of metric-backed observations, win rate, mean net trade PnL percentage, summed net account PnL percentage, profit factor, and zero-PnL count. Win rate is the fraction of metric-backed observations with positive net trade PnL. Profit factor is computed from account-PnL gains and losses after trading fees and funding cashflows and is interpreted descriptively because position sizing and execution assumptions are simplified.

Statistical comparisons are exploratory. To limit the multiple-comparisons burden, three contrasts are designated as primary for this exploratory analysis: hybrid versus visual input (modality), SMC-augmented versus Baseline (prompt style), and post-

versus pre-cutoff performance (temporal robustness). All other contrasts are secondary and descriptive. For each contrast, the paper reports mean trade-PnL differences, bootstrap 95% confidence intervals, Cohen's d, Welch t-tests, and Mann–Whitney U tests. When both conditions contain the same market cases, paired differences, paired bootstrap intervals, Wilcoxon signed-rank tests, and sign tests are also reported where computable. Because the tests are not adjusted for the number of exploratory contrasts, even the primary contrasts should be read as hypothesis-generating rather than confirmatory; a follow-up confirmatory design with family-wise (Holm) or false-discovery-rate (Benjamini–Hochberg) correction and pre-registered sample sizes is needed before any effect is claimed as established.

3.13 Methodological Boundaries

Several limitations constrain the strength of the claims:

- The shared general system prompt, the SMC/ICT extension, the leverage formula, and the `large`-representation event thresholds are archived in Appendix A. The prompt-style conditions therefore differ only by the SMC extension, so the contrast is a clean SMC-presence manipulation; the residual limit is that the model-matched performance comparison is GPT-5.5-only.
- The setup-conditioned cases are selected and labelled by a deterministic setup scanner that is released in full (Appendix A.4; `scripts/setup_scanner.py`), so the selection rules and labels are independently reproducible; the residual caveat is conceptual, in that the trend-pullback, breakout, and mean-reversion labels are the scanner's operational definitions rather than externally validated regime ground truth.
- The validation-feedback loop is active, but no matched feedback-disabled ablation exists.
- The deterministic baselines are simple scale references for one experiment family, not optimized trading systems.
- Backtests include KuCoin contract maker/taker trading fees and historical funding timestamps, but still omit spread, slippage, liquidation mechanics, order-book liquidity, latency, and partial-fill risk.
- Model-specific knowledge-cutoff dates are incomplete for some models and are treated conservatively in the temporal robustness experiment.
- The experiment archive contains reruns and mutable output counts. The manuscript reports observed counts from the current evidence set, but a tagged immutable archive would improve reproducibility.

These boundaries do not invalidate the descriptive comparisons. They define the claim level: the paper evaluates an implemented LLM signal-generation pipeline under controlled historical simulations, not a deployable trading strategy.

4 Results

This section reports descriptive execution-cost-adjusted backtest results for the implemented signal-generation pipeline. Unless stated otherwise, profitability summaries include all metric-backed observations, including zero-PnL skipped trades. PnL values

Table 7: Execution-cost audit for metric-backed backtests. Trading fees use KuCoin contract maker/taker rates and funding uses historical settlement rows; gross fields are retained from the original archived metrics.

Metric rows	Fee rows	Funding rows	Trading fees	Funding paid	Funding received	Net exec. cost	Gross acc. sum (%)	Net acc. sum (%)	Exec. delta (%)
4368	4368	3228	1820.38	329.09	271.30	1861.35	414.73	228.60	-186.13

Table 8: Paired validation-score changes for histories that used feedback. P-values compare final versus first validation score within the same run.

Experiment	Feedback histories	First score	Final score	Delta	Improved (%)	Unchanged (%)	Final ≥75 (%)	Paired t p	Wilcoxon p
Repeated-decision consistency	20	33.4	76.7	43.3	50.0	50.0	50.0	0.0004	0.0020
Representation/model benchmark	203	57.1	64.8	7.7	8.9	91.1	8.9	0.0000	0.0001
Setup-conditioned cases	5	26.7	76.7	50.0	60.0	40.0	40.0	0.0767	0.2500
Modality ablation	14	61.9	69.1	7.1	7.1	92.9	7.1	0.3356	–
Prompt-style comparison	9	66.7	66.7	0.0	0.0	100.0	0.0	–	–
Temporal robustness	94	58.2	67.0	8.9	11.7	88.3	9.6	0.0011	0.0010
All feedback histories	345	56.0	66.5	10.5	12.5	87.5	11.6	0.0000	0.0000

are net of KuCoin contract maker/taker trading fees and historical funding cashflows as described in Section 3. Abstentions, invalid outputs, and execution errors are reported as model behavior but are excluded from profitability means when no backtest metric exists.

The evidence should be read at three levels. First, the original experiment outputs describe model responses, parsed signals, validation histories, and backtest metrics. Second, diagnostic analyses summarize those outputs through consensus tables, response-language markers, baseline comparisons, the execution-cost audit, and exploratory statistical tests. Third, explicit gaps bound the interpretation. The prompt texts, leverage formula, event thresholds, and the full setup scanner are archived in Appendix A, so the main residual gaps are the absence of a feedback-disabled ablation, the GPT-5.5-only fair comparison for the prompt-style experiment, and simplified execution that still excludes spread, slippage, liquidation, partial fills, and order-book effects.

4.1 Validation-Feedback Behavior

Before interpreting PnL, the validation loop must be treated as part of the experimental treatment. Across the scanned evidence set, 345 of 4663 histories show feedback use: the model either received validator feedback, made more than one attempt, or stored an attempt number greater than one.

Among the 345 retried histories, the mean validation score increased from 56.0 to 66.5, a paired gain of +10.5 points. A paired comparison of first-attempt versus final-attempt scores confirms that this shift is not noise: among histories whose score changed, the change was almost always an improvement, giving a Wilcoxon signed-rank $p \approx 2 \times 10^{-9}$ (paired t test $p \approx 3 \times 10^{-11}$; see Table 8). The effect is therefore statistically clear but operationally limited: only 12.5% of feedback histories improved at all, 87.5% were unchanged, and just 11.6% reached a final score at or above the validation target. The loop functions as a structural repair mechanism for some weak outputs, but it is not a guarantee of valid signals. These tests describe validation-score quality only, not trading performance.

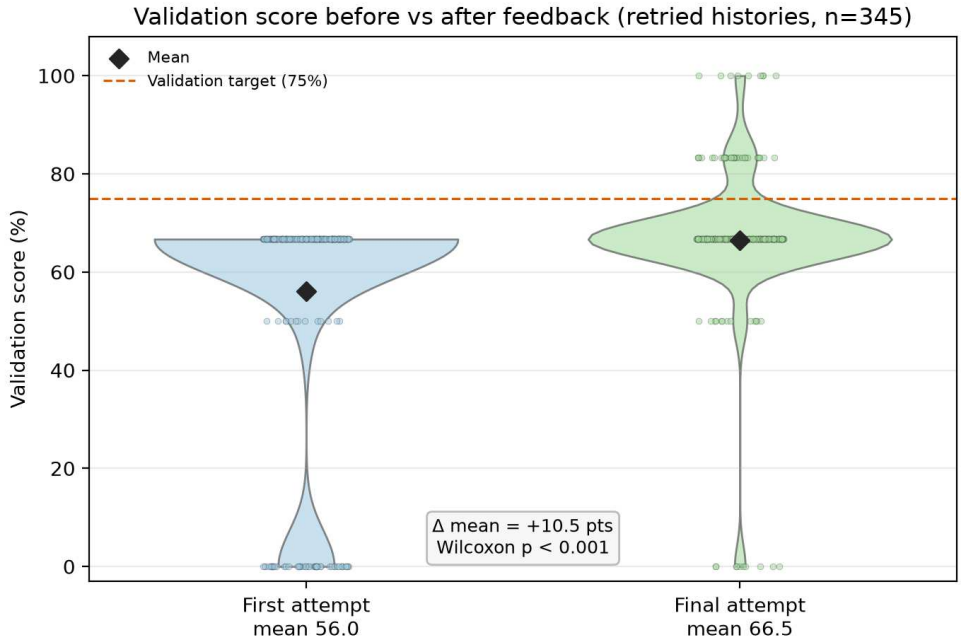


Fig. 4: First-attempt versus final-attempt validation-score distributions for the 345 retried histories. Each condition shows the full distribution (violin), quartiles (box, orange median), and the raw scores (jittered points); black diamonds mark the means. The dashed line marks the 75% validation target. The annotation reports the mean change and the paired Wilcoxon test.

Table 9: Model-level validation-feedback use. Model labels are paper-facing labels, not execution-provider labels.

Model	Histories	Feedback histories	Feedback rate (%)	Feedback messages	Delta if feedback	Final ≥75 if feedback (%)
Mistral Medium	314	142	45.2	283	0.6	0.7
GPT-5.5	1928	91	4.7	171	12.1	12.1
Qwen 3.5	716	54	7.5	107	1.5	1.9
Claude Opus 4.8	715	30	4.2	45	43.3	50.0
Claude Opus 4.7	321	11	3.4	19	24.2	27.3
GPT-5.4	315	8	2.5	14	20.8	25.0
Claude Sonnet 4.6	21	4	19.0	5	66.7	75.0
Gemini 3.1 Pro	316	3	0.9	4	61.1	66.7
Claude Opus 4.6	17	2	11.8	2	83.3	100.0

Feedback use is model-dependent. Mistral Medium accounts for the largest number of feedback histories and has the highest retry rate among the main model set, while GPT-5.5 has a much lower retry rate. Because there is no matched feedback-disabled treatment, the evidence supports documenting the loop as part of the pipeline, but not claiming that it improves trading performance.

4.2 Repeated-Decision Consistency

The repeated-decision experiment asks whether identical market states produce stable trading decisions across repeated GPT-5.5 calls. The reference design contains five market cases, three representation levels, and repeated GPT-5.5 calls per case–representation cell, with sparse observations from additional models. The archived cells contain 40 to 44 GPT-5.5 calls each (slightly above the nominal forty because a few cases were re-run). The current evidence set contains 763 metric-backed observations for this experiment: 628 GPT-5.5 calls used for the reference consensus analysis and 135 auxiliary observations from eight other model labels.

Table 10: Directional consensus for repeated GPT-5.5 calls. Consensus is the share of repeated runs matching the majority direction for each market case and representation.

Test case	Detail	Runs	Majority	Consensus (%)
071_XRP_4h_20260213_1200	large	41	SHORT	100.0
071_XRP_4h_20260213_1200	ohlcv	41	SHORT	80.5
071_XRP_4h_20260213_1200	small	41	SHORT	92.7
09_BTC_4h_20260203_1240	large	43	SHORT	88.4
09_BTC_4h_20260203_1240	ohlcv	43	SHORT	97.7
09_BTC_4h_20260203_1240	small	43	SHORT	93.0
10_XRP_4h_20250107_0809	large	43	SHORT	69.8
10_XRP_4h_20250107_0809	ohlcv	44	LONG	97.7
10_XRP_4h_20250107_0809	small	43	LONG	93.0
72_ETH_4h_20241221_1629	large	41	SHORT	97.6
72_ETH_4h_20241221_1629	ohlcv	41	SHORT	100.0
72_ETH_4h_20241221_1629	small	40	SHORT	97.5
72_XRP_4h_20250108_1800	large	42	SHORT	64.3
72_XRP_4h_20250108_1800	ohlcv	41	LONG	75.6
72_XRP_4h_20250108_1800	small	41	LONG	68.3

The first and motivating result is that repeated calls on an identical market state are highly consistent. Across the repeated GPT-5.5 calls per case–representation cell, the majority direction is reproduced in the large majority of runs: most cells exceed 80% agreement and several are near-deterministic at 97–100%. GPT-5.5 therefore does not answer randomly when queried repeatedly on the same input; it converges on a stable directional thesis. This consistency is the prerequisite that makes the rest of the study meaningful, since an unstable generator could not be characterized at all. Stability is not universal, however: a minority of case–representation cells remain ambiguous, and the worked XRP/USDT case in Section 3.10 shows that changing the representation can flip the majority direction on a fixed market state.

Conditional on the majority direction, entry and stop-loss levels are comparatively stable. This suggests that once GPT-5.5 settles on a directional thesis, it often chooses a consistent trading zone. Instability appears first at the level of directional interpretation.

The sparse cross-model observations indicate that GPT-5.5's majority decision is not always shared by other model families. Mistral Medium has the highest agreement with the GPT-5.5 majority in the current table, while Gemini 3.1 Pro and Qwen 3.5

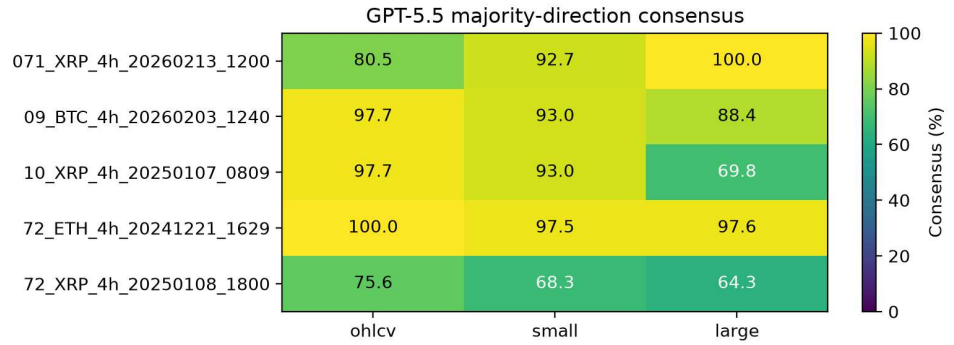


Fig. 5: GPT-5.5 majority-direction consensus by market case and input representation (5 cases, 628 GPT-5.5 calls across 15 case–representation cells). Consensus is high in some cells and ambiguous in others, and changing the representation can flip the majority direction on a fixed market state.

Table 11: Parameter stability for repeated GPT-5.5 calls under the `large` representation. Entry mean is the average of each signal's entry ladder, and only runs matching the majority direction are included.

Test case	n	Dir.	Entry mean	Entry CV (%)	SL mean	SL CV (%)
071_XRP_4h_20260213_1200	41	SHORT	1.4120	0.47	1.4670	0.65
09_BTC_4h_20260203_1240	38	SHORT	80557.0978	1.34	83209.4737	1.81
10_XRP_4h_20250107_0809	30	SHORT	2.4777	0.36	2.5550	0.55
72_ETH_4h_20241221_1629	40	SHORT	3544.9040	1.18	3735.8125	2.19
72_XRP_4h_20250108_1800	27	SHORT	2.4317	0.52	2.5392	0.36

Table 12: Agreement of non-reference models with the GPT-5.5 majority direction. Entry and stop-loss deviations are computed only for same-direction rows.

Model	Comparable rows	Direction agreement (%)	Entry dev. (%)	SL dev. (%)
Mistral Medium	14	78.6	3.37	3.39
Claude Sonnet 4.6	21	71.4	1.64	2.13
Claude Opus 4.6	17	70.6	1.74	2.58
GPT-5.4	15	60.0	1.94	3.12
Claude Opus 4.7	21	42.9	1.92	3.08
Claude Opus 4.8	15	40.0	1.21	2.47
Gemini 3.1 Pro	16	31.2	1.82	1.96
Qwen 3.5	16	31.2	2.01	1.91

agree much less often. Because these non-reference observations are sparse, the table is best interpreted as evidence of heterogeneous model priors rather than as a model ranking.

The second key result is that consensus is not profitability. Some highly consensual cells lose money, while some less consensual cells produce strong positive backtests. Directional agreement is therefore a behavioral diagnostic of model stability, not a validated trading filter.

Table 13: GPT-5.5 consensus and backtest outcome by market case and representation. This is descriptive and does not implement a consensus-filtered trading strategy.

Test case	Detail	Runs	Majority	Consensus (%)	Win (%)	Net trade PnL mean (%)
071_XRP_4h_20260213_1200	large	41	SHORT	100.0	0.0	-107.81
071_XRP_4h_20260213_1200	ohlcv	41	SHORT	80.5	0.0	-81.33
071_XRP_4h_20260213_1200	small	41	SHORT	92.7	0.0	-105.37
09_BTC_4h_20260203_1240	large	43	SHORT	88.4	27.9	-8.97
09_BTC_4h_20260203_1240	ohlcv	43	SHORT	97.7	27.9	9.63
09_BTC_4h_20260203_1240	small	43	SHORT	93.0	23.3	0.68
10_XRP_4h_20250107_0809	large	43	SHORT	69.8	67.4	54.66
10_XRP_4h_20250107_0809	ohlcv	44	LONG	97.7	0.0	-79.15
10_XRP_4h_20250107_0809	small	43	LONG	93.0	46.5	47.90
72_ETH_4h_20241221_1629	large	41	SHORT	97.6	48.8	14.38
72_ETH_4h_20241221_1629	ohlcv	41	SHORT	100.0	4.9	0.61
72_ETH_4h_20241221_1629	small	40	SHORT	97.5	15.0	7.70
72_XRP_4h_20250108_1800	large	42	SHORT	64.3	35.7	25.44
72_XRP_4h_20250108_1800	ohlcv	41	LONG	75.6	85.4	94.41
72_XRP_4h_20250108_1800	small	41	LONG	68.3	85.4	91.16

Table 14: Run-status accounting by model in the representation/model benchmark. Profitability tables condition on metric rows, so no-signal and error rates must be considered alongside PnL.

Model	Attempts	Signal	No signal	Error	Metrics	Zero PnL
Claude Opus 4.7	300	300	0	0	298	60
Claude Opus 4.8	300	300	0	0	296	54
GPT-5.4	300	161	136	3	156	29
GPT-5.5	300	296	3	1	289	50
Gemini 3.1 Pro	300	300	0	0	300	16
Mistral Medium	300	296	1	3	288	8
Qwen 3.5	300	274	25	1	273	7

4.3 Representation and Model Effects

The main random-case benchmark crosses 100 market states with three input representations and seven model labels. It yields 1900 metric-backed observations, 165 abstentions, and 8 errors out of 2100 attempted evaluations. Signal-status behavior is itself an outcome because a selective model and a highly active model are not equivalent trading systems.

The status distribution differs sharply by model. Claude Opus 4.7, Claude Opus 4.8, and Gemini 3.1 Pro emit signals for every attempted evaluation in this benchmark. GPT-5.4 is much more selective, with many abstentions and several errors. GPT-5.5, Mistral Medium, and Qwen 3.5 fall between these extremes. Profitability means are therefore conditional on signal production; trade willingness must be reported separately.

Input representation interacts strongly with model label. Claude Opus 4.8 performs best in `large`; GPT-5.4 is positive in `ohlcv` but negative in richer representations; GPT-5.5 is positive in all three representations, although its `ohlcv` mean net trade PnL is only 0.37%; Gemini 3.1 Pro improves in `large` after poor `ohlcv` performance; and Qwen 3.5 is strong in `ohlcv` but negative in `large`. The result does not support a universal

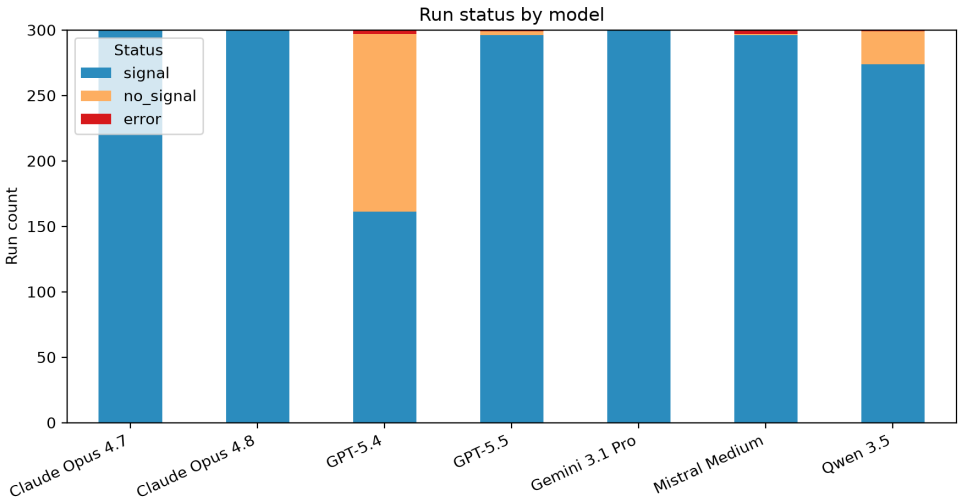


Fig. 6: Signal-status distribution by model in the random-case representation benchmark.

Table 15: Performance by model and input representation. Each row includes zero-PnL metric rows.

Model	Detail	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF	Zero PnL
Claude Opus 4.7	large	99	51.5	7.76	12.19	1.22	16
Claude Opus 4.7	ohlcv	99	46.5	-3.23	-2.94	0.95	23
Claude Opus 4.7	small	100	48.0	12.54	21.58	1.37	21
Claude Opus 4.8	large	99	59.6	14.34	23.23	1.57	19
Claude Opus 4.8	ohlcv	98	55.1	-1.61	-5.20	0.91	15
Claude Opus 4.8	small	99	50.5	-2.57	-3.92	0.93	20
GPT-5.4	large	45	44.4	-6.66	-7.81	0.76	10
GPT-5.4	ohlcv	56	60.7	14.17	16.81	1.54	8
GPT-5.4	small	55	49.1	-16.14	-20.30	0.48	11
GPT-5.5	large	95	55.8	9.19	14.49	1.35	20
GPT-5.5	ohlcv	97	53.6	0.37	2.79	1.05	15
GPT-5.5	small	97	56.7	8.55	15.06	1.28	15
Gemini 3.1 Pro	large	100	61.0	4.98	13.21	1.15	5
Gemini 3.1 Pro	ohlcv	100	47.0	-19.68	-43.71	0.66	4
Gemini 3.1 Pro	small	100	56.0	-3.41	-11.66	0.88	7
Mistral Medium	large	95	53.7	1.29	3.55	1.04	4
Mistral Medium	ohlcv	94	47.9	-4.24	-11.57	0.87	2
Mistral Medium	small	99	42.4	-3.58	-9.76	0.92	2
Qwen 3.5	large	91	50.5	-6.94	-12.61	0.85	5
Qwen 3.5	ohlcv	93	57.0	11.55	19.49	1.23	1
Qwen 3.5	small	89	57.3	2.79	2.27	1.03	1

claim that more engineered context is always better. It supports a model-specific interaction between representation and trade-command behavior.

Pooled across models, `large` has the highest mean net trade PnL, net account PnL sum, win rate, and profit factor. This pooled result is useful as a descriptive aggregate, but it should not replace the model-detail table because pooling hides strong model heterogeneity.

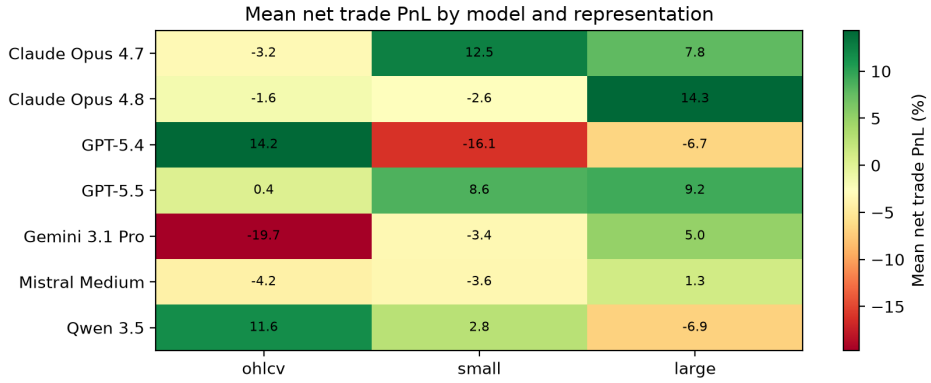


Fig. 7: Mean net trade PnL (%, after fees and funding) by model and input representation in the 100-case random benchmark (1900 metric-backed rows). The best representation differs by model, so no single representation dominates.

Table 16: Pooled performance by input representation across all models. This pooling mixes model families and should be read alongside the model-representation table.

Detail	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF	Zero PnL
ohlcv	637	52.0	-1.48	-24.34	0.95	68
small	639	51.5	0.77	-6.73	0.99	77
large	624	54.6	4.41	46.26	1.11	79

Table 17: GPT-5.5 performance by input representation and model-specific cutoff group. This table is not pooled across models.

Cutoff group	Detail	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)
Post-cutoff	large	80	53.8	7.19	9.41
Post-cutoff	ohlcv	80	56.2	0.74	3.53
Post-cutoff	small	81	58.0	12.02	17.87
Pre-cutoff	large	15	66.7	19.90	5.08
Pre-cutoff	ohlcv	17	41.2	-1.37	-0.74
Pre-cutoff	small	16	50.0	-9.02	-2.81

The GPT-5.5-only cutoff split further weakens any simple representation ranking. In the post-cutoff subset, `small` has the highest mean net trade PnL; in the pre-cutoff subset, `large` is highest. Pre-cutoff cells are small, so the split is not conclusive. It does show that the phrase "`large` is best" would be too broad for the observed evidence.

The enlarged representation benchmark also allows a direct check of agreement as a quality signal. Within a model, agreement across `ohlcv`, `small`, and `large` directions is not consistently beneficial. For GPT-5.5, representation-disagreeing rows have higher mean net trade PnL than representation-agreeing rows; the same pattern appears for several other models, while Qwen 3.5 is an exception.

Cross-model agreement is even less supportive of a simple consensus filter. Under the `large` representation, rows from market cases where the available directional

Table 18: Within-model representation-consensus diagnostic. A market case and model are labelled as agreeing when the `ohlcv`, `small`, and `large` directional signals are all present and identical; profitability rows remain individual representation-level backtests.

Model	Directions	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)
Claude Opus 4.7	Agree	159	49.1	-3.31	-6.68
Claude Opus 4.7	Disagree	139	48.2	16.04	37.51
Claude Opus 4.8	Agree	206	54.9	2.69	6.52
Claude Opus 4.8	Disagree	90	55.6	5.04	7.59
GPT-5.4	Agree	54	51.9	-5.53	-8.07
GPT-5.4	Disagree	24	54.2	-3.25	-2.97
GPT-5.5	Agree	155	51.6	3.22	10.51
GPT-5.5	Disagree	129	59.7	7.65	17.42
Gemini 3.1 Pro	Agree	180	46.7	-17.82	-76.25
Gemini 3.1 Pro	Disagree	120	66.7	11.64	34.09
Mistral Medium	Agree	145	48.3	-4.61	-17.38
Mistral Medium	Disagree	135	48.9	3.32	9.65
Qwen 3.5	Agree	176	54.5	4.55	4.40
Qwen 3.5	Disagree	54	51.9	-6.96	-2.79

Table 19: Cross-model consensus diagnostic for the `large` representation. Consensus is defined among models that emitted a directional signal for the same market case; abstaining models are not imputed.

Model	Model directions	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)
Claude Opus 4.7	Agree	31	41.9	-20.78	-10.95
Claude Opus 4.7	Disagree	68	55.9	20.77	23.15
Claude Opus 4.8	Agree	32	59.4	-4.94	-2.72
Claude Opus 4.8	Disagree	67	59.7	23.55	25.95
GPT-5.4	Agree	15	20.0	-50.81	-15.97
GPT-5.4	Disagree	30	56.7	15.41	8.16
GPT-5.5	Agree	30	43.3	-23.98	-13.47
GPT-5.5	Disagree	65	61.5	24.51	27.96
Gemini 3.1 Pro	Agree	32	53.1	-4.50	-3.82
Gemini 3.1 Pro	Disagree	68	64.7	9.44	17.03
Mistral Medium	Agree	31	45.2	-10.71	-9.06
Mistral Medium	Disagree	64	57.8	7.11	12.61
Qwen 3.5	Agree	31	51.6	-5.71	-5.09
Qwen 3.5	Disagree	60	50.0	-7.58	-7.52

models agree are negative on average for most model labels. For GPT-5.5, the agreed rows are worse than the disagreed rows in the generated table. This does not prove that disagreement is intrinsically good; it shows that naive model agreement is not a sufficient quality criterion in the current interface.

To place these LLM results on an absolute scale, Table 20 reports the five deterministic non-LLM bracket baselines on the same 100 cases, evaluated through the identical fee- and funding-aware backtester. The sample regime is short-favorable: the always-short and mean-reversion baselines are positive (mean net trade PnL 6.32% and 7.26%, profit factor 1.14 and 1.17), whereas the always-long, trend-following, and deterministic-random baselines are negative, with the random baseline weakest. Two comparisons matter. First, the strongest model–representation cells, such as Claude Opus 4.8 under `large`, have higher descriptive mean net trade PnL than every baseline. Second, the pooled `large` LLM result (mean net trade PnL 4.41%) is below

Table 20: Deterministic non-LLM bracket baselines on the 100-case random benchmark, evaluated through the same execution-cost-aware backtester (KuCoin maker/taker fees and historical funding). Each baseline opens a 50x market position with a 2% stop, 1R/2R/3R targets, 1.8% account allocation, break-even after target 1, and a 24-hour horizon close. These are scale references, not optimized strategies.

Baseline	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF	Zero PnL
always long	99	47.5	-5.11	-9.10	0.89	0
always short	99	49.5	6.32	11.27	1.14	0
random	99	44.4	-8.69	-15.48	0.83	0
trend following	100	46.0	-5.92	-10.65	0.88	0
mean reversion	98	51.0	7.26	12.81	1.17	0

Table 21: Setup-conditioned case inventory. Scanner direction and setup identifiers are parsed from the case descriptions; the full scanner rules are documented in Appendix A.4 and released as `scripts/setup_scanner.py`.

Setup	Cases	Metadata	Descriptions	BTC	DOGE	ETH	XRP	Scanner LONG	Scanner SHORT	First date	Last date	Setup IDs
breakout	33	33	33	13	6	7	7	13	20	2025-11-03	2026-04-17	B_LONG, B_SHORT
mean reversion	33	33	33	11	4	11	7	13	20	2025-11-05	2026-04-17	C_LONG, C_SHORT
trend pullback	34	34	34	12	9	3	10	4	30	2025-11-02	2026-04-26	A_LONG, A_SHORT

the mean-reversion bracket by 2.85 percentage points, and many individual model–representation cells underperform a simple always-short rule in this regime. These differences have not been tested as evidence of statistical superiority. The baselines therefore indicate that LLM trade-signal generation is not automatically superior to trivial deterministic strategies; its advantage, where present, is concentrated in specific model–representation combinations rather than in the interface as a whole. Because the baselines apply one fixed bracket to one case sample, they are a scale reference rather than a tuned strategy comparison.

4.4 Setup-Conditioned Market States

The setup-conditioned experiment replaces random sampling with scanner-selected market states. The archived labels divide 100 unique cases into trend-pullback, breakout, and mean-reversion groups. The executed runs use the `large` representation; we report GPT-5.5, which evaluated all 100 cases. Claude Opus 4.8 and Qwen 3.5 ran only a small partial subset (about twenty cases each) and are excluded from the reported comparison to avoid mixing full and partial coverage.

The setup inventory confirms that all 100 selected cases include metadata and scanner descriptions. These records make the labels more informative than arbitrary condition names: they include setup identifiers, scanner direction, setup-candle time, and scanner close. Because the scanner rules and thresholds are now released in full (Appendix A.4), the labels are independently reproducible; they remain the scanner's operational definitions rather than externally validated regime definitions.

Restricted to GPT-5.5, all three setup classes remain positive by mean net trade PnL and profit factor, but the ranking is the opposite of a naive pooled view that mixes in the partial other-model runs: trend pullback has the highest mean net trade PnL and profit factor, mean reversion has the highest win rate, and breakout is the weakest of the three. Because the labels are the scanner's operational definitions and

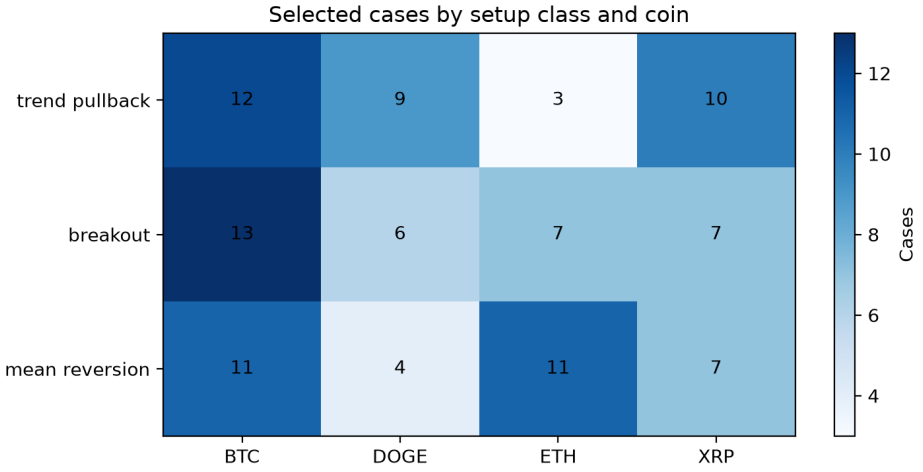


Fig. 8: Selected-case distribution by setup class and coin. Counts are unique market cases, not model runs.

Table 22: GPT-5.5 performance by labelled setup class (`large` representation). The setup labels are scanner-selected conditions; the full scanner rules are documented in Appendix A.4 and released as `scripts/setup_scanner.py`.

Setup	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF	Zero PnL
breakout	32	46.9	8.92	4.42	1.22	7
mean reversion	32	56.2	14.80	8.22	1.68	8
trend pullback	34	50.0	18.31	11.66	1.79	9

Table 23: GPT-5.5 performance by labelled setup and cutoff group. Small cells should be interpreted descriptively.

Cutoff group	Setup	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)
Post-cutoff	breakout	26	38.5	-5.50	-2.69
Post-cutoff	mean reversion	31	58.1	15.28	8.22
Post-cutoff	trend pullback	28	46.4	7.06	3.80
Pre-cutoff	breakout	6	83.3	71.38	7.11
Pre-cutoff	mean reversion	1	0.0	0.00	0.00
Pre-cutoff	trend pullback	6	66.7	70.80	7.86

the per-class samples are small, this is a descriptive summary rather than a controlled ranking of setup types.

Within GPT-5.5, the post-cutoff subset favors mean reversion by mean net trade PnL and profit factor, while breakout is negative. Pre-cutoff cells are very small. The setup-conditioned results are therefore useful exploratory evidence. Although the scanner rules are released, broader conclusions require balanced model coverage and validation beyond the scanner's operational setup definitions.

Table 24: Input-modality comparison for GPT-5.5 under the `large` representation.

Modality	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF	Mean max DD (%)	Zero PnL
Hybrid	95	55.8	9.19	14.49	1.35	-45.32	20
Textual	98	50.0	6.45	12.47	1.22	-57.24	20
Visual	95	38.9	-3.48	-5.56	0.89	-48.70	32

4.5 Input Modality

The modality ablation is one of the cleanest controlled contrasts. It evaluates the same 100 random market cases under textual-only, visual-only, and hybrid input, using GPT-5.5 and the `large` representation throughout. The evidence set contains 300 attempted evaluations, 295 signal responses, 5 abstentions, and 288 metric-backed observations.

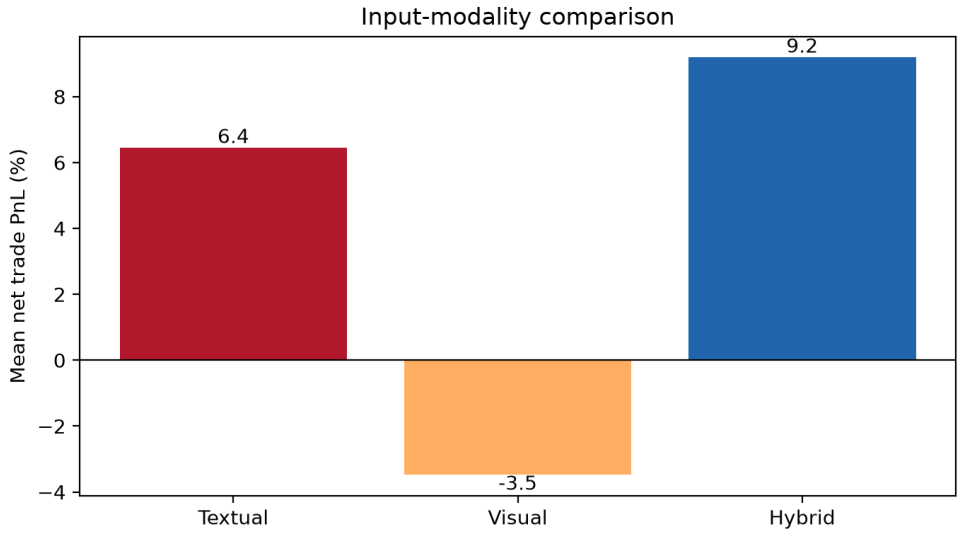


Fig. 9: Mean net trade PnL (%, after fees and funding) by input modality for GPT-5.5 under the `large` representation (100 cases, 288 metric-backed rows). Hybrid text-plus-chart input leads, textual is close, and visual-only is negative; the difference is a primary contrast but is not statistically significant (Section 4.8).

Hybrid input has the highest win rate, mean net trade PnL, net account PnL sum, and profit factor. Textual input is close on mean net PnL and net account PnL, while visual-only input is negative on average and contains the largest number of zero-PnL rows. Because case, model, and representation are controlled, the contrast is informative; because the inference is still one model and exploratory tests have wide intervals, it should be read as evidence that hybrid input is promising rather than as a confirmed multimodal advantage.

Table 25: GPT-5.5 response-language markers by prompt condition. Counts are lexical markers in archived model responses, not reconstructed system-prompt text.

Prompt	Responses	Words mean	SMC terms mean	Indicator terms mean	Any SMC term (%)	Any indicator term (%)
SMC-augmented	100	303.2	14.98	3.60	100.0	96.0
Baseline	100	272.8	0.88	6.78	51.0	100.0

4.6 Prompt Style

The prompt-style experiment compares the SMC-augmented and Baseline conditions on the same 100 random cases under the `large` representation. We report GPT-5.5 for both conditions: the SMC-augmented condition was run only with GPT-5.5, whereas the Baseline batch happened to include other-model runs, which we exclude so the comparison is model-matched (Table 1). The SMC-augmented condition adds the SMC/ICT extension to the general prompt, while the Baseline condition uses the general prompt without it; both texts are archived verbatim (Appendix A), so the conditions differ only by the SMC extension. The contrast is therefore a clean, reproducible SMC-presence manipulation; it is treated as a primary exploratory comparison for the *performance* inference (Section 4.8) because that comparison is GPT-5.5-only and uses a single 100-case sample.

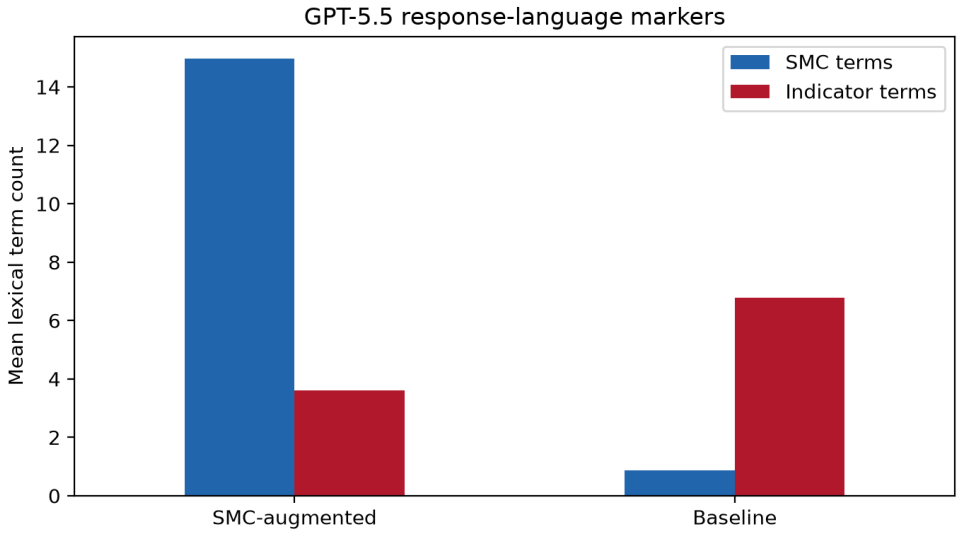


Fig. 10: Response-language markers by prompt condition for GPT-5.5 (100 cases). The SMC-augmented prompt elicits far more SMC/liquidity markers than the free Baseline prompt (14.98 vs 0.88 per response), a manipulation check confirming the treatment took effect. Markers are counted from archived model responses.

The two conditions induce visibly different response styles, in the expected direction, which serves as a manipulation check that the prompt treatment took effect. GPT-5.5

Table 26: Prompt-style comparison for GPT-5.5 post-cutoff metric rows. SMC-augmented and Baseline are the observed prompt-condition labels.

Prompt	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF	PnL/DD	Zero PnL
SMC-augmented	81	46.9	8.84	15.54	1.37	0.004	22
Baseline	80	53.8	7.19	9.41	1.26	0.003	18

Table 27: GPT-5.5 prompt performance split by model-specific cutoff group.

Cutoff group	Prompt	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF
Post-cutoff	SMC-augmented	81	46.9	8.84	15.54	1.37
Post-cutoff	Baseline	80	53.8	7.19	9.41	1.26
Pre-cutoff	SMC-augmented	17	41.2	17.19	4.51	1.41
Pre-cutoff	Baseline	15	66.7	19.90	5.08	1.91

Table 28: Prompt-style comparison for all GPT-5.5 metric rows, before applying the post-cutoff filter.

Prompt	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF	Zero PnL
SMC-augmented	98	45.9	10.29	20.05	1.38	27
Baseline	95	55.8	9.19	14.49	1.35	20

responses under the SMC-augmented condition contain SMC/liquidity terms in every response and average 14.98 SMC markers per response. The free Baseline condition averages only 0.88 SMC markers per response and uses more conventional indicator language. The large gap confirms that the SMC-augmented prompt reshaped the model's analytical vocabulary as intended.

In the post-cutoff GPT-5.5 subset, the SMC-augmented condition has higher mean net trade PnL, net account PnL sum, and profit factor, while the Baseline condition has the higher win rate. The prompt-style difference therefore appears to reshape the payoff distribution rather than simply increase hit rate.

The cutoff split is mixed. The SMC-augmented condition is higher by mean net PnL in post-cutoff rows, while the Baseline condition is higher in the small pre-cutoff cells. The pre-cutoff comparison is too small to support a strong conclusion.

Across all GPT-5.5 metric-backed rows, the SMC-augmented condition remains slightly higher in mean net trade PnL and profit factor, while the Baseline condition remains higher in win rate. This is not a dominance result. It is evidence that prompt style changes reasoning language and payoff distribution.

4.7 Temporal Robustness

The temporal robustness experiment evaluates 400 random cases across BTC, DOGE, ETH, and XRP, split across 2025 and 2026 market periods, with Claude Opus 4.8, GPT-5.5, and Qwen 3.5 under the `large` representation. The goal is to test whether model performance declines after the model-specific knowledge cutoff.

Cutoff grouping is model-specific. GPT-5.5 is assigned a cutoff date of December 1, 2025; Claude Opus 4.8 is assigned January 31, 2026; Qwen 3.5 has an unknown cutoff and is conservatively treated as post-cutoff. Calendar year and cutoff group are therefore not equivalent.

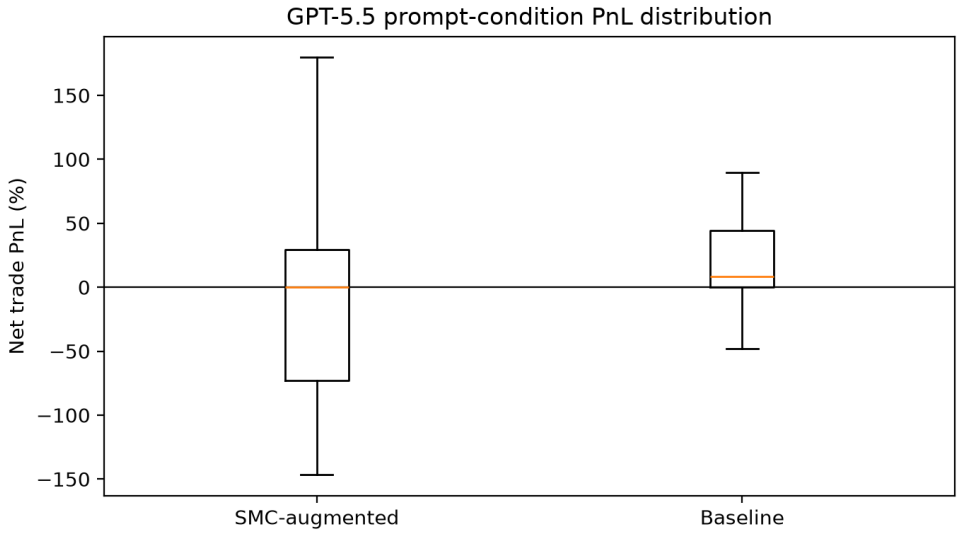


Fig. 11: GPT-5.5 net trade-PnL distribution by prompt condition (SMC-augmented vs free Baseline). The conditions reshape the payoff distribution rather than one dominating; the difference is a primary contrast and is not statistically significant (Section 4.8).

Table 29: Temporal robustness performance by model and model-specific cutoff group. Models with unknown cutoffs are labelled explicitly.

Model	Cutoff group	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF
Claude Opus 4.8	Post-cutoff	155	50.3	7.81	21.61	1.29
Claude Opus 4.8	Pre-cutoff	241	46.5	-0.34	-0.64	1.00
GPT-5.5	Post-cutoff	222	46.8	14.31	57.00	1.56
GPT-5.5	Pre-cutoff	159	43.4	7.75	24.18	1.31
Qwen 3.5	Post-cutoff (unknown cutoff)	349	54.4	10.10	61.77	1.21

Table 30: Temporal robustness performance by calendar year and model. Calendar year is not identical to model-specific cutoff grouping.

Model	Year	n	Win (%)	Net trade PnL mean (%)	Net account PnL sum (%)	PF
Claude Opus 4.8	2025	199	46.7	-1.31	-3.67	0.97
Claude Opus 4.8	2026	197	49.2	7.06	24.65	1.26
GPT-5.5	2025	190	46.3	12.41	44.61	1.51
GPT-5.5	2026	191	44.5	10.73	36.57	1.40
Qwen 3.5	2025	180	55.0	12.99	39.10	1.26
Qwen 3.5	2026	169	53.8	7.02	22.67	1.16

The model-specific cutoff table does not show a simple post-cutoff decay. Claude Opus 4.8 and GPT-5.5 both have higher post-cutoff mean net trade PnL than pre-cutoff mean net trade PnL in the current evidence set. Qwen 3.5 cannot support a verified pre/post comparison because its cutoff is unknown.

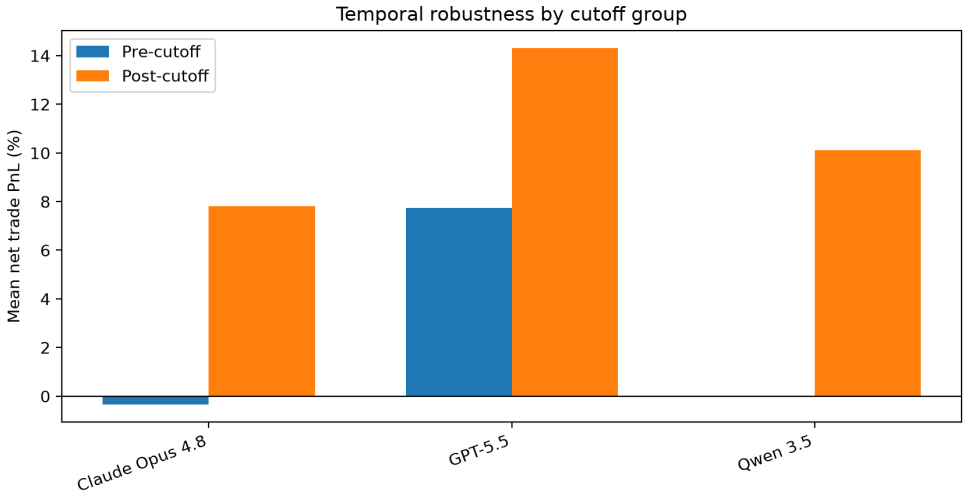


Fig. 12: Mean net trade PnL by model-specific cutoff group. Qwen 3.5 has no verified pre-cutoff group because its cutoff date is unknown.

The calendar-year split adds nuance. GPT-5.5 has higher mean net trade PnL in 2025 than in 2026, even though its model-specific post-cutoff group is higher than its pre-cutoff group. This is possible because GPT-5.5's cutoff is December 1, 2025, so most 2025 cases are already post-cutoff by the configured definition. Claude Opus 4.8 is higher in 2026 than 2025, while Qwen 3.5 is higher in 2025 than 2026. The correct conclusion is limited: these data do not support a simple claim that performance worsens after the configured knowledge cutoff, but they also do not isolate cutoff effects from market regime, volatility, case selection, or signal selectivity.

4.8 Exploratory Statistical Comparisons

The descriptive tables contain several large-looking differences. To quantify their uncertainty, selected contrasts are evaluated with exploratory inferential diagnostics. The tested variable is net trade PnL percentage. Each contrast reports the mean difference, a bootstrap 95% confidence interval, Cohen's d, a Welch t-test p-value, and a Mann–Whitney U p-value. Where both conditions contain the same market cases, paired diagnostics are also reported.

The statistical diagnostics are deliberately conservative. Notably, all three designated primary contrasts (Section 4.8) fail to reach significance: the largest descriptive contrast, Hybrid versus Visual input, has a wide bootstrap confidence interval that includes zero; the SMC-augmented-versus-Baseline comparison is not statistically distinguishable; and the post- versus pre-cutoff comparison is confounded and not distinguishable under these tests. Hybrid versus Textual input is especially uncertain because the 100-case modality sample makes the two means close, and pooled representation comparisons are likewise not distinguishable. The current evidence is best treated as exploratory and hypothesis-generating rather than confirmatory.

Table 31: Exploratory statistical comparisons on net trade PnL percentage. CI is a bootstrap 95 percent interval for Group A minus Group B; p-values are unadjusted.

Comparison	A	B	n A	n B	Mean A	Mean B	Diff	CI low	CI high	d	Welch p	MW p
Modality ablation	Hybrid	Textual	95	98	9.19	6.45	2.75	-23.04	28.30	0.03	0.8366	0.5284
Modality ablation	Hybrid	Visual	95	95	9.19	-3.48	12.68	-10.65	35.75	0.15	0.2886	0.1293
Modality ablation	Visual	Textual	95	98	-3.48	6.45	-9.93	-35.01	15.92	-0.11	0.4495	0.4733
Prompt style, post-cutoff	SMC-aug.	Baseline	81	80	8.84	7.19	1.65	-26.51	29.92	0.02	0.9090	0.4945
Prompt style, all dates	SMC-aug.	Baseline	98	95	10.29	9.19	1.10	-24.64	26.86	0.01	0.9348	0.4157
Temporal robustness, GPT-5.5	Post	Pre	222	159	14.31	7.75	6.55	-11.94	24.73	0.07	0.4971	0.5628
Temporal robustness, Claude Opus 4.8	Post	Pre	155	241	7.81	-0.34	8.15	-10.31	27.46	0.09	0.3980	0.5688
Pooled representation	large	ohlcv	624	637	4.41	-1.48	5.88	-4.95	17.38	0.06	0.2991	0.2521
Pooled representation	small	ohlcv	639	637	0.77	-1.48	2.25	-8.54	13.40	0.02	0.6923	0.6740
Pooled representation	large	small	624	639	4.41	0.77	3.63	-7.52	14.83	0.04	0.5194	0.4671
GPT-5.5 post-cutoff representation	small	large	81	80	12.02	7.19	4.83	-23.36	32.35	0.05	0.7334	0.7230
GPT-5.5 post-cutoff representation	small	ohlcv	81	80	12.02	0.74	11.28	-17.48	39.75	0.12	0.4449	0.5166
OHLCV model contrast	GPT-5.4	GPT-5.5	56	97	14.17	0.37	13.80	-19.91	50.50	0.13	0.4407	0.3477
OHLCV model contrast	Qwen 3.5	GPT-5.5	93	97	11.55	0.37	11.18	-20.89	42.36	0.10	0.4781	0.3280
GPT-5.5 post-cutoff setup	mean reversion	breakout	31	26	15.28	-5.50	20.78	-31.34	72.53	0.20	0.4507	0.1966
GPT-5.5 post-cutoff setup	mean reversion	trend pullback	31	28	15.28	7.06	8.22	-40.17	57.63	0.09	0.7438	0.5354

4.9 Synthesis

Across experiments, the same pattern appears: LLM-generated trade commands are often parseable, behaviorally stable within some market states, and sensitive to representation, modality, prompt style, model identity, and signal selectivity. However, stability is not the same as profitability, and richer representations do not help every model. The prompt-style comparison isolates the presence of the SMC extension, but the exploratory performance estimates do not establish a profitability benefit. Execution-cost-adjusted historical backtests are not live-trading returns.

5 Conclusions and Outlook

This paper evaluated LLMs as generators of structured cryptocurrency trade commands rather than as ordinary price forecasters or sentiment classifiers. The central object of study is the full interface: a bounded historical market state is transformed into either no trade or a parsable signal, then validated and backtested under deterministic rules.

The current evidence supports bounded conclusions. GPT-5.5 repeated calls often converge on stable majority directions and stable entry/stop zones, but consensus is not trading quality: some highly consensual cells lose money, while less consensual cells can perform well. Input representation matters materially, and `ohlcv`, `small`, and `large` do not form a simple ladder where more information always improves results. The worked XRP/USDT case and the random-case model-detail benchmark both show that representation can change model behavior and even dominant direction.

Modality and prompt style also matter, but the claims must remain narrow. Hybrid text-plus-chart input has the highest mean net PnL in the 100-case GPT-5.5 modality ablation, textual input is close, and visual-only input lags. This suggests that chart context can be useful when paired with textual market state, but it does not yet establish a general multimodal advantage across models. The prompt-style experiment shows a strong response-language difference between the SMC-augmented condition and the free Baseline condition, in the expected direction: the SMC-augmented prompt elicits far more explicit market-structure language. Both treatments are now archived verbatim (Appendix A), so the conditions differ only by the SMC extension and the contrast is a clean, reproducible manipulation. The correct conclusion is nonetheless modest: the manipulation reshapes output style and the payoff distribution, but the PnL

Table 32: Claim status for the current methodology and results sections.

Claim	Current evidence	Main caveat
Repeated GPT-5.5 calls can be stable, but consensus is not profitability.	Consensus, parameter-stability, consensus-performance, and cross-model consensus tables show stable majority-direction zones as well as profitable and unprofitable high-consensus cells.	Stability is case- and representation-dependent; no separate consensus-trading strategy is tested.
Input representation changes behavior.	Model-detail tables and the worked XRP/USDT case show different best representations and a direction flip across representations.	Representation effects remain entangled with model-specific interpretation of engineered context.
Hybrid input is promising for GPT-5.5.	The 100-case modality ablation has Hybrid as the best condition by mean net PnL, win rate, net account PnL sum, and profit factor, with Textual close behind and Visual negative on average.	One model only; inferential intervals are wide and do not establish a confirmed multimodal advantage.
Prompt conditions induce different output styles.	Response markers show much more explicit SMC/liquidity language in the SMC-augmented condition; performance tables show higher mean net PnL but lower win rate than the free Baseline condition.	Both prompts are archived (Appendix A), so the manipulation is clean; the residual limit is that the fair comparison is GPT-5.5 only.
Setup-conditioned cases are deterministic scanner outputs.	Metadata and descriptions exist for all 100 selected cases and contain setup IDs, scanner direction, setup-candle time, and scanner close.	The full setup scanner is released (Appendix A.4); the residual limit is conceptual—the labels are the scanner's operational definitions, not externally validated regimes.
Validation feedback is part of the treatment.	The feedback audit finds 345 retried histories and a maximum of three attempts.	No feedback-disabled ablation exists, so performance effect is unknown.
The cutoff experiment does not show simple post-cutoff decay.	GPT-5.5 and Claude Opus 4.8 have higher post-cutoff than pre-cutoff mean net PnL in the model-specific cutoff table.	Cutoff, calendar year, market regime, and case selection are not causally separated.
Execution/value claims remain provisional.	The upgraded backtester uses KuCoin contract maker/taker rates, historical funding rows, and retained gross fields for audit.	Fill liquidity classification, mark-price approximation, spread, slippage, liquidation mechanics, partial fills, and order-book effects remain simplified or omitted.

comparison is GPT-5.5-only and exploratory, not a causal proof that SMC prompting trades better.

The validation-feedback loop is real but limited. Hundreds of histories contain validator-triggered retries, and mean validation score improves among retried histories, but most histories do not improve and only a minority reach the target score. Similarly, the cutoff experiment does not support a simple post-cutoff decay story, but it also does not prove temporal generalization because market regime, sample composition, signal selectivity, and cutoff grouping remain entangled.

The main contribution is therefore an empirical account of LLM behavior inside a trade-signal interface. Conceptually, it shows why LLM trading experiments should separate grammar following, directional stability, risk-level coherence, abstention, representation effects, backtested PnL, and cost sensitivity instead of compressing them into the single question of whether the model "trades well." Methodologically, it demonstrates that operational trading studies need to preserve not only final PnL but

also prompts, responses, parser behavior, validation histories, signal-status outcomes, and execution assumptions.

Several limitations define the boundary of the claims. The shared general system prompt, the SMC/ICT extension, the leverage formula, the `large`-representation event thresholds, and the full setup scanner are now archived (Appendix A); the residual archival gap is limited to the original chart images for some conducted cases. The validation-feedback loop lacks a feedback-disabled counterfactual, and the backtester still omits spread, slippage, liquidation mechanics, order-book depth, latency, and partial-fill risk. Trading fees and funding are modeled from KuCoin contract metadata and historical funding rows, but account-tier fee schedules, mark-price snapshots, and actual exchange fill confirmations are not archived. The statistical tests are exploratory and unadjusted. The paper can claim that the implemented interface produced the reported signals and execution-cost-adjusted historical backtest outcomes under documented assumptions; it cannot yet claim robust live profitability, general superiority over classical systems, calibrated model confidence, or causal effects of prompt style, feedback, or knowledge cutoff.

The next research pass should convert this descriptive evidence into a smaller confirmatory design. Building on the now-archived general prompt, SMC extension, and configuration (Appendix A), the remaining artifacts to version before inference are chart-generation settings, pinned model identifiers, model-call dates, sampling parameters, and original outputs. The prompt-style treatments are already archived as a matched pair that differs only in the SMC extension, so the remaining need for that question is balanced model coverage and a pre-registered, multiple-testing-corrected confirmatory re-run rather than additional prompt documentation. The baseline set should be expanded, execution modeling should add account-tier fee calibration, spread, slippage, liquidation, order-book depth, and fill assumptions, and a matched no-feedback ablation should be run. A final confirmatory experiment should predefine the short list of primary contrasts named in Section 3, increase sparse-cell sample sizes, apply multiple-testing controls, and evaluate promising configurations in forward paper trading before any claim moves from historical simulation to deployable trading behavior.

A complementary path is to stop hand-authoring the prompt and instead optimize it automatically against the validation and backtest metrics defined here. Declarative prompt-optimization frameworks such as DSPy (Khattab et al., 2024) compile and tune a prompt program against a measurable objective rather than relying on trial-and-error templates, and agentic "autoresearch" loops (Karpathy, 2026) iterate prompt or instruction text autonomously, keeping only changes that improve a target metric. Because this study already exposes a deterministic, fee-aware scoring pipeline, the SMC-augmented and Baseline prompts—together with the representation and modality templates—could be treated as optimization targets rather than fixed treatments, turning prompt design from a manual choice into a measured, reproducible search; any such search would still have to be wrapped in the same pre-registered, multiple-testing-aware evaluation to avoid overfitting the historical sample.

Statements and Declarations

Funding

The authors received no funding for this research.

Competing interests

The authors declare that they have no competing interests.

Data availability

The archived model outputs, backtest metrics, prompts, analysis code, and notebooks are available at https://github.com/FlorianSchneiderIU/crypto-llm-results/tree/31bb2f93d6251a8f6e8fba3304e5351e2b816e83. The original thesis analysis archive is available at https://github.com/JonasIU18/krypto-handelssignale-llm-studie/tree/master.

Author contributions

The individual author contributions are reported according to the CRediT Contributor Roles Taxonomy (National Information Standards Organization, 2022; Allen et al., 2014).

Florian Schneider: Conceptualization, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review and editing, Supervision.

Jonas Delpho: Methodology, Software, Formal analysis, Investigation, Validation, Data curation, Visualization, Writing – review and editing.

Visieu Lac: Conceptualization, Supervision, Project administration, Writing – original draft, Writing – review and editing.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the authors used AI-assisted coding and writing tools, namely OpenAI Codex and Anthropic's Claude (via the Claude Code assistant). These tools supported source inspection and cross-checking the manuscript against the experiment code base; reconciling described methods with the implementation (for example, archiving the shared system prompt, leverage formula, and event-detection thresholds, and correcting the prompt-condition labelling); statistical-analysis review, including the paired validation-score test; regenerating figures and tables from the backtester's own archived event logs; implementing and running the deterministic non-LLM baselines through the same execution-cost-aware backtester; preparing manuscript layouts; and general drafting, editing, and LaTeX maintenance. All analyses, code, and computed results were specified, reviewed, and verified by the authors, who edited the AI-assisted material as needed and take full responsibility for the content of the publication. The AI tools are not authors of this work.

A Prompt, Leverage, and Configuration Archive

Line breaks in printed prompt excerpts are adjusted to fit the page width; the released archive preserves the original texts. This appendix reproduces the implementation details that were previously described only in prose, so that the core of the signal-generation interface can be inspected and reused. The material below, together with the per-run model outputs, computed backtest metrics, and analysis code, is released in a public results repository, pinned to the exact commit `31bb2f93d625` used for this manuscript: https://github.com/FlorianSchneiderIU/crypto-llm-results/tree/31bb2f93d6251a8f6e8fba3304e5351e2b816e83. The aim is to raise the reproducibility level of the study, while still marking honestly which treatment details were not archived.

A.1 General System Prompt

Every experiment shares the same general system prompt. It is reproduced here verbatim from the released prompt archive, without modification. This is the common base of the SMC-augmented and Baseline prompt conditions and of all representation, modality, setup, and temporal experiments.

```
You are an expert in Crypto trading. Your task is to create a message in the
below defined <format>. To do so, use plain text only (no markdown or
internal
numbering). You will work through each part of the information that you get
and
give a summary of your findings. This will be a multi-step approach.

# Steps:
1. Start with a reasoning step (## Reasoning). The user will not see this;
it is
   just for you to come up with a plan. Write down all your thoughts about
the
   input data and what you can conclude from that.
2. Follow up with a new section (## Thought summary). Summarize the
important
   findings concisely, in technical trader language. Explain what you saw on
   which chart/data, whether it is bullish/bearish for the corresponding
   timeframe, and how it affects your decision. Include your entries,
targets
   and stop loss.
3. Give the signal section (## Signal) in the provided output format.

# Additional constraints
- Choose the amount as 3% * conviction (the less sure you are, the smaller
the
  margin).
- *CRITICAL* Do *NOT* calculate the leverage. It will be calculated by the
user
  afterwards. Always set leverage = -1 in the output.
- Never add any additional context to the signal. The signal must always be
  exactly in the provided format.
- The market is very manipulative right now. Expect a lot of SL hunt and
wicking
  near resistance/support zones before the major move. Therefore, keep a
less
  tight stop loss.
- Choose safe entry points close to key levels (supply or demand, fibonacci,
  channels, and so on). It is better to not enter into a trade than to enter
far
  away from the stop loss.
- *CRITICAL* You can choose the strategy (mean reversion, trend following,
```

```
etc.)
  freely. Try to find a setup that maximizes Winrate * R. If you cannot find
a
  favorable setup, refuse to give a signal.
- *CRITICAL* You can choose the number of entries and targets freely.

# Message Format
/place MT
Coin: #<COIN>/USDT
Leverage: -1
Direction: <LONG/SHORT>
Entry: 1) <M or float value> 2) <DCA entry> 3) ...
Target: 1) <Level 1>/<Percentage 1> 2) <Level 2>/<Percentage 2> 3) ...
Trigger-BE-Target: <after reaching this target, we activate breakeven.
Normally 1>
Moving-Stop-Loss: 0
SL: <STOP LOSS float>
Amount: <PERCENTAGE VALUE of the account equity, <= 3%>
Confirmation: 0
Force Open: 0
Conviction: <0.1 to 1.0, where 1.0 is very sure>
```

The prompt is reproduced in condensed form for readability; the four worked few-shot examples that accompany it are omitted here but are present unchanged in the released prompt archive.

A.2 Smart Money Concept Extension and the Prompt-Style Conditions

The *SMC-augmented* condition (the narrow, "enger" prompt) appends the Smart Money Concept/ICT extension reproduced verbatim below to the general system prompt of Appendix A.1. The extension instructs the model to interpret price primarily through liquidity, market structure, displacement, imbalance, premium/discount, and institutional order flow, and to treat conventional indicators as supporting evidence rather than the main decision engine. The *Baseline* condition (the wide, "weiter" prompt) uses the general system prompt *without* this extension and leaves the setup and strategy choice free; it is the control condition.

Because both the general system prompt (Appendix A.1) and the SMC/ICT extension (below) are archived verbatim, the two prompt-style treatments are now fully specified: the SMC-augmented condition is their concatenation, and the Baseline condition is the general prompt alone. The treatment difference is therefore exactly the presence of this extension, so the prompt-style contrast is a clean and reproducible SMC-presence manipulation. The response-language counts in Section 4 (far more SMC/liquidity terminology under the SMC-augmented condition) confirm that the manipulation took effect. Two caveats remain for the *performance* comparison: the model-matched fair comparison is limited to GPT-5.5, and the analysis uses a single 100-case sample with exploratory, unadjusted statistics, so the prompt-style effect on PnL is hypothesis-generating rather than confirmatory. The extension is reproduced exactly as supplied, including its informal Markdown-style emphasis.

```
# Smart Money Concept / ICT Trading Framework
- _CRITICAL_ Base your market interpretation primarily on Smart Money
Concepts. Do
  not evaluate price action only as isolated indicator signals. Always think
```

```
in terms
  of liquidity, market structure, displacement, imbalance, premium/discount,
and
  institutional order flow.
- Smart Money Concepts assume that price often moves from one liquidity pool
to
  another. Therefore, identify where obvious liquidity is located: previous
swing
  highs, previous swing lows, equal highs, equal lows, session highs/lows,
range
  highs/lows, and clean support/resistance levels where retail traders place
stops.
- Look for liquidity sweeps before entries. A strong setup often forms after
price has
  raided liquidity above a prior high or below a prior low and then shows a
reaction in
  the opposite direction. Treat such sweeps as potential stop hunts, not
automatically
  as breakouts.
- Distinguish between a real breakout and a liquidity grab. A real breakout
should show
  displacement, strong close, volume participation, and continuation
structure. A
  liquidity grab often shows a wick through a key level followed by
rejection back into
  the prior range.
- Evaluate market structure before choosing direction. For longs, prefer
bullish market
  structure such as higher highs, higher lows, bullish break of structure,
or bullish
  change of character after a liquidity sweep. For shorts, prefer bearish
market
  structure such as lower lows, lower highs, bearish break of structure, or
bearish
  change of character after a liquidity sweep.
- Use Fair Value Gaps and imbalances as important context. A strong
displacement candle
  can leave an imbalance that price may later retrace into. Entries near a
relevant FVG,
  order block, mitigation zone, or demand/supply area are preferred over
random entries
  in the middle of nowhere.
- Use order blocks conceptually as the last opposing candle before
displacement. A
  bullish order block can act as a demand area after bullish displacement. A
bearish
  order block can act as a supply area after bearish displacement. Do not
blindly trade
  every order block; require confluence with structure, liquidity, and
risk-reward.
- Consider premium and discount zones. For long trades, prefer entries in
discount
  relative to the relevant swing range. For short trades, prefer entries in
premium
  relative to the relevant swing range. Avoid longing high in premium unless
a strong
  breakout continuation is clearly confirmed. Avoid shorting low in discount
unless
  bearish continuation is clearly confirmed.
- Prefer setups where the entry, stop loss, and targets align with Smart
Money logic.
  The stop loss should usually sit beyond the invalidation point, such as
beyond the
  swept low/high, beyond the order block, or beyond the relevant structural
swing. Do
  not place the stop loss directly at obvious liquidity where it is likely
to be hunted.
- Targets should preferably be located at opposing liquidity pools, prior
```

```
swing
  highs/lows, range extremes, unfilled imbalances, or major
support/resistance levels.
  Do not choose arbitrary round targets if better liquidity-based targets
are visible.
- A valid signal should ideally combine several SMC factors: liquidity
sweep, market
  structure shift, displacement, retracement into imbalance/order block, and
a clear
  liquidity target. If this confluence is missing, reduce conviction or
refuse a signal.
- If indicators and Smart Money structure disagree, prioritize market
structure and
  liquidity logic. Indicators such as RSI, EMA, MACD, Bollinger Bands, ATR,
and volume
  are supporting evidence, not the main decision engine.
- Always explain in the Thought summary how the Smart Money interpretation
affects the
  trade idea: where liquidity was taken, what structure is bullish/bearish,
where the
  invalidation lies, and which liquidity pool or imbalance is used as
target.
```

A.3 Leverage Recalculation

The model is instructed to output placeholder leverage -1; the evaluated leverage is recomputed downstream from the parsed entry and stop-loss risk and is never trusted from the response. With average entry price p_{entry} and stop-loss price p_{SL}, the validator computes the fractional stop distance $r = |p_{\text{entry}} - p_{\text{SL}}|/p_{\text{entry}}$ and sets

$$\text{leverage} = \left\lfloor \frac{1}{r} \right\rfloor = \left\lfloor \frac{p_{\text{entry}}}{|p_{\text{entry}} - p_{\text{SL}}|} \right\rfloor . \tag{4}$$

This ties leverage deterministically to stop-loss distance, so a tighter stop yields higher leverage. The leverage is obtained by integer truncation of this ratio. Because the resulting position size already embeds this leverage, the trading-fee charge of Section 3 is applied to leveraged notional.

A.4 Event Thresholds and the Setup Scanner

The annotations added in the `large` representation are produced by deterministic event detectors whose thresholds are fixed in the event-detection implementation, and are listed here:

- Fibonacci retracement levels and events use a swing-high/low lookback of min(chart bars, 200) bars.
- Pivot points use a 20-bar lookback.
- Order blocks and fair-value-gap blocks use a minimum block size of 3 candles.
- Volume-spike events fire when bar volume $\geq 1.8\times$ the 20-period volume moving average.
- Momentum events use conventional RSI overbought/oversold thresholds (70/30).

These parameters define the *event annotations* exposed to the model.

The separate *setup scanner* that selects the setup-conditioned cases and assigns the trend-pullback, breakout, and mean-reversion labels (`setup_A`/`setup_B`/`setup_C`) is released in full as `scripts/setup_scanner.py` in the reproducibility repository, so its complete rule set is now independently reproducible rather than only recoverable from outputs. The scanner reads KuCoin OHLCV bars and, on each closed candle, evaluates fixed long/short rules of the form *context* (trend or regime) → *location* (where price sits) → *trigger* (confirmation on the current bar). The indicators are EMA(20/50/200), RSI(14), ATR(14), Bollinger Bands$(20, 2\sigma)$ with relative width (upper − lower)/middle, the 20-period volume average, the prior 20-bar high/low (excluding the current bar), and the 3-bar EMA20 slope. A shared noise filter requires ATR(14)/close ≥ 0.005. The three rule families are:

- **A — Trend pullback (long).** EMA50 > EMA200, close > EMA50, a rising EMA20 slope, a wick that tests EMA20 (low $\leq 1.005\,$EMA20) while the bar closes back above it, a bullish candle, and $40 \leq$ RSI ≤ 65. The short rule mirrors this (EMA50 < EMA200, close < EMA50, falling slope, high $\geq 0.995\,$EMA20, $35 \leq$ RSI ≤ 60).
- **B — Breakout (long).** Close above the prior 20-bar high, volume $> 1.5\times$ its 20-period average, close > EMA50, candle range $\geq 0.8\,$ATR, close in the top 40% of the bar (close − low $\geq 0.6 \times$ range), and not over-extended (close $\leq 1.06\,$EMA20). The short rule mirrors this against the prior 20-bar low.
- **C — Mean reversion (long).** A range regime ($0.97\,$EMA200 $\leq$ EMA50 $\leq 1.03\,$EMA200), Bollinger width ≥ 0.04, the *previous* close below the lower band, RSI < 35, and a reversal trigger (current close above the previous close, a bullish candle, close $\geq 0.90\,$EMA20). The short rule mirrors this at the upper band with RSI > 65.

To build the experiment, matches are computed with a 300-bar indicator warmup, the last (possibly unfinished) candle of each fetch is dropped to avoid look-ahead, and cases are drawn by stratified random sampling across the three categories (for example 7/7/6 for twenty cases). Each selected match becomes a test case at a cutoff of the setup candle plus one timeframe, so the setup candle itself is included. The script also reports direction-corrected forward returns over several horizons as a first-pass edge check, which is explicitly *not* a full backtest.

A.5 Model Access and Identifiers

The main multi-provider comparisons were run through an institutional chat access layer, which exposes models by display label rather than by pinned API snapshot. The main benchmark labels are GPT-5.5, GPT-5.4, Claude Opus 4.8, Claude Opus 4.7, Gemini 3.1 Pro, Mistral Medium, and Qwen 3.5. The repeated-decision archive also includes sparse Claude Opus 4.6 and Claude Sonnet 4.6 observations through the same access layer. Locally hosted and direct-API models (for example via OpenAI, Groq, and LMStudio) were also available but are not part of the reported model set. Because the access layer abstracts exact provider snapshots, the manuscript reports display labels and model-specific knowledge-cutoff dates (Section 4) rather than pinned version strings; this is a known limitation for exact replication.

References

Alessandretti L, ElBahrawy A, Aiello LM, et al (2018) Anticipating cryptocurrency prices using machine learning. Complexity 2018:8983590. https://doi.org/10.1155/2018/8983590, URL https://arxiv.org/abs/1805.08550

Allen L, Scott J, Brand A, et al (2014) Publishing: Credit where credit is due. Nature 508(7496):312–313. https://doi.org/10.1038/508312a

Almeida J, Gonçalves TC (2024) Cryptocurrency market microstructure: A systematic literature review. Annals of Operations Research 332:1035–1068. https://doi.org/10.1007/s10479-023-05627-5

Almgren R, Chriss N (2001) Optimal execution of portfolio transactions. Journal of Risk 3:5–39

Bailey DH, Borwein JM, López de Prado M, et al (2014) The probability of backtest overfitting. Journal of Computational Finance 20(4)

Barredo Arrieta A, Díaz-Rodríguez N, Del Ser J, et al (2020) Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. Information Fusion 58:82–115. https://doi.org/10.1016/j.inffus.2019.12.012

Bergmeir C, Benítez JM (2012) On the use of cross-validation for time series predictor evaluation. Information Sciences 191:192–213. https://doi.org/10.1016/j.ins.2011.12.028

Bertsimas D, Lo AW (1998) Optimal control of execution costs. Journal of Financial Markets 1(1):1–50. https://doi.org/10.1016/S1386-4181(97)00012-8

Biais B, Glosten L, Spatt C (2005) Market microstructure: A survey of microfoundations, empirical results, and policy implications. Journal of Financial Markets 8(2):217–264. https://doi.org/10.1016/j.finmar.2004.11.001

Bouteska A, Abedin MZ, Hajek P, et al (2024) Cryptocurrency price forecasting – a comparative analysis of ensemble learning and deep learning methods. International Review of Financial Analysis 92:103055. https://doi.org/10.1016/j.irfa.2023.103055

Carlini N, Tramèr F, Wallace E, et al (2021) Extracting training data from large language models. In: 30th USENIX Security Symposium (USENIX Security 21). USENIX Association, pp 2633–2650, URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting

Cerqueira V, Torgo L, Mozetič I (2020) Evaluating time series forecasting models: An empirical study on performance estimation methods. Machine Learning 109:1997–2028. https://doi.org/10.1007/s10994-020-05910-7

Dong B, Jiang L, Liu J, et al (2022) Liquidity in the cryptocurrency market and commonalities across anomalies. International Review of Financial Analysis 81:102097

Fatouros G, Metaxas K, Soldatos J, et al (2024) Can large language models beat wall street? unveiling the potential of ai in stock selection. URL https://arxiv.org/abs/2401.03737, arXiv:2401.03737

Gruver N, Finzi M, Qiu S, et al (2023) Large language models are zero-shot time series forecasters. In: Advances in Neural Information Processing Systems, pp 19622–19635, https://doi.org/10.52202/075280-0861, URL https://arxiv.org/abs/2310.07820

Guidotti R, Monreale A, Ruggieri S, et al (2018) A survey of methods for explaining black box models. ACM Computing Surveys 51(5):93:1–93:42. https://doi.org/10.1145/3236009

Hegselmann S, Buendia A, Lang H, et al (2023) Tabllm: Few-shot classification of tabular data with large language models. In: Proceedings of The 26th International Conference on Artificial Intelligence and Statistics, Proceedings of Machine Learning Research, vol 206. PMLR, pp 5549–5581, URL https://proceedings.mlr.press/v206/hegselmann23a.html

Hung CC, Chen YY, Lin CH, et al (2021) Dpp: Deep predictor for price movement from candlestick charts. PLOS ONE 16(6):e0252404. https://doi.org/10.1371/journal.pone.0252404

John DL, Binnewies S, Stantic B (2024) Cryptocurrency price prediction algorithms: A survey and future directions. Forecasting 6(3):637–671. https://doi.org/10.3390/forecast6030034

Kapoor S, Narayanan A (2023) Leakage and the reproducibility crisis in machine-learning-based science. Patterns 4(9):100804. https://doi.org/10.1016/j.patter.2023.100804

Kara Y, Boyacioglu MA, Baykan ÖK (2011) Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. Expert Systems with Applications 38(5):5311–5319. https://doi.org/10.1016/j.eswa.2010.10.027

Karpathy A (2026) autoresearch: AI agents running research automatically. GitHub repository, URL https://github.com/karpathy/autoresearch

Khattab O, Singhvi A, Maheshwari P, et al (2024) DSPy: Compiling declarative language model calls into self-improving pipelines. In: International Conference on Learning Representations (ICLR), URL https://arxiv.org/abs/2310.03714, arXiv:2310.03714

Kumar A, Morabito R, Umbet S, et al (2024) Confidence under the hood: An investigation into the confidence-probability alignment in large language models. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Bangkok, Thailand, pp 315–334, https://doi.org/10.18653/v1/2024.acl-long.20, URL https://aclanthology.org/2024.acl-long.20/

Lazaridou A, Kuncoro A, Gribovskaya E, et al (2021) Mind the gap: Assessing temporal generalization in neural language models. In: Advances in Neural Information Processing Systems, pp 29348–29363, URL https://arxiv.org/abs/2102.01951

Lee MC (2024) Bitcoin trend prediction with attention-based deep learning models and technical indicators. Systems 12(11):498. https://doi.org/10.3390/systems12110498

Li X, Chan S, Zhu X, et al (2023a) Are chatgpt and gpt-4 general-purpose solvers for financial text analytics? a study on several typical tasks. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track. Association for Computational Linguistics, Singapore, pp 408–422, https://doi.org/10.18653/v1/2023.emnlp-industry.39, URL https://aclanthology.org/2023.emnlp-industry.39/

Li Y, Wang S, Ding H, et al (2023b) Large language models in finance: A survey. URL https://arxiv.org/abs/2311.10723, arXiv:2311.10723

Lin S, Hilton J, Evans O (2022) Teaching models to express their uncertainty in words. Transactions on Machine Learning Research URL https://arxiv.org/abs/2205.14334

Liu L, Si YW (2022) 1d convolutional neural networks for chart pattern classification in financial time series. The Journal of Supercomputing 78:14191–14214

Liu T, Wang F, Chen M (2024) Rethinking tabular data understanding with large language models. In: Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). Association for Computational Linguistics, Mexico City, Mexico, pp 450–482, https://doi.org/10.18653/v1/2024.naacl-long.26, URL https://aclanthology.org/2024.naacl-long.26/

Makarov I, Schoar A (2020) Trading and arbitrage in cryptocurrency markets. Journal of Financial Economics 135(2):293–319. https://doi.org/10.1016/j.jfineco.2019.07.001

Mallqui DCA, Fernandes RAS (2019) Predicting the direction, maximum, minimum and closing prices of daily bitcoin exchange rate using machine learning techniques. Applied Soft Computing 75:596–606. https://doi.org/10.1016/j.asoc.2018.11.038

National Information Standards Organization (2022) ANSI/NISO Z39.104-2022, CRediT, Contributor Roles Taxonomy. https://doi.org/10.3789/ansi.niso.z39.104-2022, URL https://www.niso.org/publications/z39104-2022-credit

Park J, Lee H, Lee D, et al (2025) Revisiting llms as zero-shot time-series forecasters: Small noise can break large models. URL https://arxiv.org/abs/2506.00457, arXiv:2506.00457

Patel J, Shah S, Thakkar P, et al (2015) Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Systems with Applications 42(1):259–268. https://doi.org/10.1016/j.eswa.2014.07.040

Qiu M, Song Y, Akagi F (2016) Predicting the direction of stock market index movement using an optimized artificial neural network model. PLOS ONE 11(5):e0155133. https://doi.org/10.1371/journal.pone.0155133

Sebastião H, Godinho P (2021) Forecasting and trading cryptocurrencies with machine learning under changing market conditions. Financial Innovation 7(1):3. https://doi.org/10.1186/s40854-020-00217-x

Sezer OB, Ozbayoglu AM (2018) Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. Applied Soft Computing 70:525–538. https://doi.org/10.1016/j.asoc.2018.04.024

Shynkevich Y, McGinnity TM, Coleman SA, et al (2017) Forecasting price movements using technical indicators: Investigating the impact of varying input window length. Neurocomputing 264:71–88. https://doi.org/10.1016/j.neucom.2016.11.095

Sullivan R, Timmermann A, White H (1999) Data-snooping, technical trading rule performance, and the bootstrap. The Journal of Finance 54(5):1647–1691. https://doi.org/10.1111/0022-1082.00163

Ushio A, Camacho-Collados J (2024) A systematic analysis on the temporal generalization of language models in social media. In: Proceedings of the 14th Workshop on Computational Approaches to Subjectivity, Sentiment, and Social Media Analysis. Association for Computational Linguistics, Bangkok, Thailand, pp 52–62, https://doi.org/10.18653/v1/2024.wassa-1.5, URL https://aclanthology.org/2024.wassa-1.5/

Wang M, Izumi K, Sakaji H (2024) Llmfactor: Extracting profitable factors through prompts for explainable stock movement prediction. URL https://arxiv.org/abs/2406.10811, arXiv:2406.10811

Weber P, Carl KV, Hinz O (2024) Applications of explainable artificial intelligence in finance: A systematic review of finance, information systems, and computer science literature. Management Review Quarterly 74(2):867–907. https://doi.org/10.1007/s11301-023-00320-0

White H (2000) A reality check for data snooping. Econometrica 68(5):1097–1126. https://doi.org/10.1111/1468-0262.00152

Yin Z, Sun Q, Guo Q, et al (2023) Do large language models know what they don't know? In: Findings of the Association for Computational Linguistics: ACL 2023. Association for Computational Linguistics, Toronto, Canada, pp 8653–8665, https://doi.org/10.18653/v1/2023.findings-acl.551, URL https://aclanthology.org/2023.findings-acl.551/

Yu X, Chen Z, Ling Y, et al (2023) Temporal data meets llm – explainable financial time series forecasting. URL https://arxiv.org/abs/2306.11025, arXiv:2306.11025

Zhang H, Lin Y, Wan X (2024a) Pacost: Paired confidence significance testing for benchmark contamination detection in large language models. In: Findings of the Association for Computational Linguistics: EMNLP 2024. Association for Computational Linguistics, Miami, Florida, USA, pp 1794–1809, https://doi.org/10.18653/v1/2024.findings-emnlp.97, URL https://aclanthology.org/2024.findings-emnlp.97/

Zhang J, Cai K, Wen J (2024b) A survey of deep learning applications in cryptocurrency. iScience 27(1):108509. https://doi.org/10.1016/j.isci.2023.108509

Zhao Z, Wallace E, Feng S, et al (2021) Calibrate before use: Improving few-shot performance of language models. In: Proceedings of the 38th International Conference on Machine Learning, Proceedings of Machine Learning Research, vol 139. PMLR, pp 12697–12706, URL https://proceedings.mlr.press/v139/zhao21c.html

Zhu C, Chen N, Gao Y, et al (2025a) Is your llm outdated? a deep look at temporal generalization. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). Association for Computational Linguistics, Albuquerque, New Mexico, pp 7433–7457, https://doi.org/10.18653/v1/2025.naacl-long.381, URL https://aclanthology.org/2025.naacl-long.381/

Zhu Y, Bai X, Chen K, et al (2025b) Benchmarking and improving large vision-language models for fundamental visual graph understanding and reasoning. In: Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Vienna, Austria, pp 30678–30701, https://doi.org/10.18653/v1/2025.acl-long.1482, URL https://aclanthology.org/2025.acl-long.1482/