Club GAIA - ESCOM @ FinMMEval 2026 Task 3: Multimodal Financial Decision Making via Compact GRPO Fine-Tuning of Quantized LLMs

FinMMEval at CLEF 2026

Jonatan Bustillos Cruz[1,*], Jesus Baruc Polvo Cuatianquiz [1], Rodrigo Esteban Morales Roldán[1] and David Reyes Ramos[1,*]

[1]*Escuela Superior de Cómputo (ESCOM) - Instituto Politécnico Nacional (IPN), Mexico City, Mexico*

Abstract

In this work we describe a multimodal pipeline that fuses textual news, temporal price sequences, and numerical scalars into a single coherent prompt for a quantized language model. The decision model, based on `Phi-4-mini-instruct` in 4-bit NF4 quantization, is trained via a compact two-stage curriculum that combines supervised fine-tuning (SFT) with Group Relative Policy Optimization (GRPO) under a strict 12 GB VRAM budget. We detail the exact hyperparameters, the asymmetric reward matrix that breaks the Nash equilibrium of inaction, and the action-extraction mechanism that rescues truncated tokens. On the in-distribution cryptocurrency validation split, the deployed agent registers a cumulative return of 90.67%; however, this figure should be interpreted with caution: it was never verified on the held-out out-of-sample test set due to a code-level oversight in the validation engine, and the ground-truth labels used during training are affected by look-ahead bias from global percentile computation. In the corrected official Task 3 leaderboard snapshot, the same agent ranked 12th of 18 on BTC and 18th of 18 on TSLA under the Long/Short cumulative-return metric. Combined with the absence of traditional equity data during training, the reported return likely overstates the model's true generalization ability; we document these limitations transparently. Finally, despite replication constraints from non-deterministic library internals, we reconstruct our training protocol and VRAM management strategy to facilitate study and adaptation on commodity hardware.

Keywords

Financial decision making, Large language models, Multimodal fusion, Group Relative Policy Optimization, Reinforcement learning, FinMMEval, CLEF 2026

1. Introduction

Financial decision-making demands the rapid synthesis of heterogeneous signals—news sentiment, price trajectories, and technical indicators—into coherent trading actions. In the CLEF 2026 FinMMEval Lab [1], Task 3 on Financial Decision Making [2] challenges participants to build agents that process exactly this multimodal mixture and output daily recommendations (BUY, HOLD, SELL) for assets ranging from cryptocurrencies to blue-chip equities, specifically BTC and TSLA.

Large Language Models (LLMs) have shown impressive results in natural-language reasoning, yet their direct application to algorithmic trading [3, 4] is hindered by two obstacles. The first obstacle is regarding raw numerical time series. They lie outside the typical pre-training distribution of text-centric LLMs, so the model must be explicitly taught to interpret price patterns and volatility signals. The second challenge is related to standard supervised fine-tuning (SFT)—the process of adjusting a pre-trained model's weights using labeled instruction-response pairs and standard cross-entropy loss—which can

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany

*Corresponding author.

†These authors contributed equally.

‡Additional author.

joni_343@hotmail.com (J. B. Cruz); barucpolvo@gmail.com (J. B. P. C.); morales.roldan.rodrigoesteban@gmail.com (R. E. M. Roldán); dreyesr2105@gmail.com (D. R. Ramos)

0009-0005-1520-1862 (J. B. Cruz); 0009-0009-1138-1674 (J. B. P. C.); 0009-0005-9461-0647 (R. E. M. Roldán); 0009-0006-0984-5183 (D. R. Ramos)

teach format compliance but cannot shape risk-sensitive behavior; reinforcement learning is required to align the model's outputs with financial reward signals such as Sharpe Ratio or Cumulative Return.

Recent work has demonstrated the power of Group Relative Policy Optimization (GRPO) for financial reasoning [5], but existing systems rely on massive GPU clusters (e.g., 8×H200 nodes) and generate thousands of tokens per decision, placing them far beyond the reach of most academic groups. Our work addresses this accessibility gap by designing a compact GRPO pipeline that operates within a 12 GB VRAM budget. We use `Phi-4-mini-instruct` [6] in 4-bit NF4 quantization via QLoRA [7], feed it a structured multimodal prompt enriched with FinBERT sentiment [8] and Chronos directional forecasts [9] (the latter incorporated only at inference time, not during training), and refine its outputs through a two-stage SFT plus GRPO curriculum.

The main objective of this Working Note is to detail the design, training, and deployment of the pipeline that produced our best model. Section 2 situates our work within the relevant literature on multimodal financial forecasting, memory-augmented trading agents, and parameter-efficient reinforcement learning. Section 3 constitutes the core of the paper: it begins by describing the overall system architecture and the two temporal dataset configurations (Full and Fast mode), then details the five-phase dataset construction pipeline—covering data ingestion, FinBERT sentiment enrichment, scalar merge with base-100 price normalization, volatility-driven label discretization, and temporal split assignment. The section continues with the prompt orchestration module and the Chronos zero-shot forecasting component, followed by the two-stage training curriculum that transitions from a supervised fine-tuning warm-up to Group Relative Policy Optimization under a strict 12 GB VRAM budget, including the active memory logistics, the asymmetric reward matrix engineered to break the Nash equilibrium of inaction, and the Truncated Token Rescue mechanism that prevents gradient collapse. Section 3 closes with the Optuna hyperparameter search protocol, the multi-fold temporal validation framework, and the asynchronous FastAPI serving architecture monitored by a watchdog service. Section 3.9 reports the empirical results from both fast-mode Optuna trials and full validation across eleven candidate models, analyzing the tension between risk-adjusted efficiency and absolute cumulative return. Finally, Section 4 summarizes our findings, acknowledges the key limitations and technical debt identified during development, and outlines directions for future work.

2. Related Work

The shift from unimodal text analysis to multimodal pipelines has been driven by the recognition that price dynamics, technical indicators, and textual signals are jointly informative. Recent datasets such as FinMultiTime [10] provide rich, temporally aligned multimodal streams, and the MultiFinBen benchmark [11] offers a comprehensive evaluation suite spanning multilingual and multimodal financial tasks. Meanwhile, live evaluation frameworks like the Agent Market Arena (AMA) [12] provide standardized, real-time environments in which researchers can rigorously evaluate how well models fuse these heterogeneous inputs under dynamic market conditions. Complementary efforts such as RealFin [13] further probe the limits of LLM financial reasoning under implicit and underspecified user queries, while domain-specific benchmarks including SAHM [14] for Arabic financial and Shari'ah-compliant reasoning and BhashaBench [15] for Indic-language domains demonstrate the growing demand for culturally and linguistically diverse financial evaluation.

Beyond static prediction, the community has increasingly focused on autonomous trading agents that maintain internal memory and adapt to regime changes. Systems such as FINMEM [16] and TradingAgents [17] structure hierarchical memories that allow an agent to reflect on past trades before committing to a new position. More recent multi-agent architectures, including TradingGroup [18] and P1GPT [19], extend this idea by letting specialized agents negotiate and debate individual decisions before a consensus action is emitted.

Training these agents at scale typically requires parameter-efficient supervised fine-tuning (SFT) and reinforcement-learning algorithms that can tolerate sparse, delayed rewards. SFT serves as the primary alignment stage to teach the model formatting, structure, and style through next-token prediction on

curated demonstration datasets. Low-Rank Adaptation (LoRA) [20] reduces the memory footprint of full fine-tuning, and its quantized extension QLoRA [7] further compresses the base weights to 4-bit precision, while Proximal Policy Optimization (PPO) [21] has become the default workhorse for aligning language-model outputs with scalar reward signals. While SFT teaches format compliance and basic style alignment, standard supervised methods struggle to shape risk-sensitive behavior; reinforcement learning is required to optimize for complex, non-differentiable objectives such as financial returns. The theoretical foundations for designing such reward functions draw on Ng et al.'s seminal work on potential-based reward shaping [22], which guarantees policy invariance under specific reward transformations. In the financial domain, FLAG-TRADER [23] and Trading-R1 [5] apply these techniques to trading decisions. Furthermore, the success of reasoning-centric models such as DeepSeekMath [24] and DeepSeek-R1 [25] suggests that chain-of-thought reasoning [26] can further improve decision quality in complex, high-stakes environments. If a model is allowed to "think out loud" without restrictions in a noisy financial environment, its intermediate thoughts tend to become disorganized and accumulate errors, resulting in fragile theses and unreliable decisions [4]. By forcing the model to fill in sequential labels (`<technicals>`, `<news>`, `<conclusion>`), you are providing it with a "reasoning scaffold" that forces its neural network to evaluate the evidence step by step before printing the final decision [4]. This ensures that the reasoning process remains consistent and coherent. In this way, the final action is grounded in a solid logic.

Our work sits at the intersection of these trends: we combine multimodal enrichment (FinBERT plus Chronos in the endpoint), structured chain-of-thought prompting [26], and compact GRPO fine-tuning under severe hardware constraints. Rather than proposing new architectural primitives, we demonstrate how a two-stage training curriculum and targeted reward engineering—specifically designed to break the inaction bias, which in game-theoretic terms constitutes a Nash equilibrium of inaction [27], and overcome length-induced gradient collapse in small models—can produce competitive trading agents on commodity GPUs.

3. Methodology

The proposed methodological framework integrates multimodal signal processing, aggressive memory optimization under tight hardware constraints, and reinforcement learning tailored for financial returns. To make training feasible on a single commodity GPU with a strict 12 GB VRAM limit, we implement a modular architecture using the `Phi-4-mini-instruct` base model quantized to 4-bit NF4 precision via QLoRA. This compression strategy allows the system to process dense news streams, historical price trends, and zero-shot temporal predictions simultaneously without triggering out-of-memory assertions. Consequently, the pipeline is divided into three major operational blocks: a multi-stage data engineering layer, a two-phase training curriculum, and a low-latency serving endpoint.

To balance aggressive hyperparameter exploration with computational feasibility, the experimental methodology was designed around two sequential operational phases. The first phase focuses on a rapid prototyping cycle utilizing a temporally constrained dataset (referred to as *fast* mode) to evaluate hyperparameter candidates efficiently. This fast-mode cycle uses a subset of the validation folds to provide quick feedback loops for our Bayesian or random search samplers while keeping evaluation substantially faster. The planned second phase aimed to execute full-scale model training over the complete historical dataset (Full mode) alongside a rigorous, multi-fold temporal validation framework. However, due to training time and computational constraints, this full-scale training on the complete dataset could not be executed, meaning the final winning model was instead selected from the fast-mode candidates and verified on the Yahoo validation split, establishing full-dataset training as a primary avenue for future work.

At the input level, our data pipeline transforms raw, heterogeneous market feeds into structured, reason-inducing prompts through five sequential stages, designed to accommodate both data configurations. The process begins with data ingestion, collecting news and historical price scalars from Yahoo Finance and the CLEF daily_news repository. Next, the sentiment extraction stage routes the unstruc-

tured textual news summarized for that day through FinBERT, producing discrete sentiment labels and detailed per-class probability distributions. The scalar merge and price normalization stage then applies mathematical equations to generate stationary base-100 price series, followed by a volatility-driven discretization stage that assigns BUY, HOLD, or SELL ground-truth labels based on rolling historical volatility. Finally, the split assignment stage partitions the dataset temporally before the prompt orchestration module integrates these signals with Chronos predictions into structured chain-of-thought XML prompts, ready to be submitted to the fine-tuned quantized model for trading action recommendations in the format `[[[ACTION]]]`.

3.1. Dataset and Feature Engineering

Our dataset engineering pipeline is structured around two distinct temporal configurations: a *Full* mode, which leverages the complete historical dataset to capture long-term market patterns, and a *Fast* mode, which restricts the temporal window to the most recent 12 months to enable rapid experimentation. While our original strategy aimed to train the final model on the Full mode dataset to maximize historical coverage, strict training time and computational constraints required us to use the Fast mode configuration.

Specifically, the Dataset Builder for this Fast mode configuration restricted the Yahoo Finance historical data to the most recent 12 months (360 days), yielding a contiguous window from 2024-02-07 to 2025-02-01, and retained only rows containing news ($\texttt{has\_news} = 1$). This temporal filtering produced 1,082 training rows and 896 validation rows. Yahoo-sourced assets (BTC_YAHOO, ETH_YAHOO, and SOL_YAHOO) were split chronologically into train and validation partitions, while CLEF-sourced assets (BTC_CLEF, ETH_CLEF, TSLA_CLEF, MSFT_CLEF, and BMRN_CLEF) were unconditionally assigned to the out-of-sample test split. This fixed temporal split was motivated by the need to ensure that the model learns from a consistent, high-frequency source while the CLEF dataset provides a rigorous unseen environment. However, several issues affecting this split—including look-ahead bias in label construction, the unused test set, and the crypto-only training bias—are documented in Section 3.10. Table 1 describes the specifications of the primary datasets integrated in our pipeline.

Table 1
Datasets Used in the Study

Dataset	Source	Rows	Usage
Yahoo Finance (Historical)	Yahoo Finance	6,794	Train & Validation
FinMMEval daily_news (CLEF)	Hugging Face / CLEF	1,333	Out-of-sample Test

To formalize the similarities and differences between these configurations, Table 2 provides a detailed quantitative comparison of the dataset divisions in both modes. Structurally, the configurations share identical dataset schema and asset categories. Although the database contains the same out-of-sample test split of 1,333 rows sourced from the CLEF daily_news repository in both modes to maintain schema alignment, the test set was reserved for out-of-sample evaluation but was never loaded during model selection (Section 3.10).

The primary divergence lies in the training and validation footprints. In the Full mode, the training set encompasses 5,787 rows spanning multiple years of historical price and news data. Fast mode reduces this training partition to 1,082 rows, corresponding to an 81.3% reduction. Similarly, the validation set is truncated from 1,007 rows in Full mode to 896 rows in Fast mode (an 11.0% reduction). This targeted compression reduces the combined training and validation footprint by 70.9% (from 6,794 to 1,978 rows), which dramatically decreases training iteration times while preserving the core statistical patterns from the recent market regime.

To analyze how this reduction affects individual assets, Table 3 provides a per-asset breakdown of row counts across both modes. The asset universe remains consistent, containing cryptocurrency pairs sourced from Yahoo Finance (BTC_YAHOO, ETH_YAHOO, and SOL_YAHOO). However, the temporal

Table 2
Dataset comparison: Full vs Fast mode

	Total		Difference	Meaning
Division	Full	Fast mode	(abs, %)	
Train	5,787	1,082	-4,705 (-81.3%)	Reduced to the last 12 months
Val	1,007	896	-111 (-11.0%)	Also truncated to the last 12 months
Test	1,333	444	-889 (-66.7%)	Only one-third evaluated for speed
Train + Val	6,794	1,978	-4,816 (-70.9%)	Effective training/validation footprint
Total (card)	8,127	2,422	-5,705 (-70.2%)	Includes evaluated test subset

truncation affects the assets unevenly depending on their historical coverage. Specifically, BTC_YAHOO experiences the most significant reduction, conserving only 16.8% of its total rows (535 in Fast mode versus 3,178 in Full mode). This is because Bitcoin possesses the longest historical data coverage in our ingested corpus. In contrast, newer assets such as ETH_YAHOO and SOL_YAHOO are less affected, retaining 37.5% (722 rows) and 42.6% (721 rows) of their data, respectively. Despite these shifts in total count, Fast mode maintains a balanced distribution between the validation partitions of Ethereum and Solana (361 rows each), while Bitcoin validation is kept at 174 rows. This allocation ensures that the validation signals across different crypto-assets remain balanced and prevent the model from overfitting to any single dominant asset.

Table 3
Per-asset counts: Full vs Fast mode (other assets remain the same)

Asset	Full			Fast mode			Conserved
	Total	Train	Val	Total	Train	Val	(%)
BTC_YAHOO	3,178	2,967	211	535	361	174	16.8%
ETH_YAHOO	1,923	1,525	398	722	361	361	37.5%
SOL_YAHOO	1,693	1,295	398	721	360	361	42.6%

3.1.1. Enrichment and dataset construction pipeline

The dataset preparation and construction pipeline was executed incrementally, combining historical market feeds with external news repositories to build a unified multimodal dataset. Firstly, the data ingestion and preparation phase acquired the official dataset from the previous year's CLEF competition [28] and consolidated it with the current year's daily news corpus [29]. To expand the training footprint, we integrated historical price data from Yahoo Finance for earlier dates and attached external news summaries sourced from public repositories [30, 31, 32]. These ingested assets were normalized and filtered to retain only rows containing news ($\texttt{has\_news} = 1$) before being stored as structured CSV files. As illustrated in the general diagram in Figure 1, this initial ingestion and filtering stage sets the foundation for a lightweight, four-phase enrichment pipeline that transforms the raw CSVs into the final Parquet dataset consumed by the trainer.

Secondly, the pipeline performs news sentiment enrichment utilizing the domain-specific FinBERT language model [8]. For every row with news summaries, the Sentiment Extractor passes the text through the network to compute a 3-class probability distribution (P_{positive}, P_{negative}, P_{neutral}) alongside a discrete label and absolute confidence score. These results are cached per asset to prevent redundant computation across training runs, as shown in Figure 2. A detailed description of the FinBERT architecture and its softmax formulation is provided in Section 3.2.

Thirdly, the pipeline merges the scalar market features with the cached sentiment outputs by date and applies scale normalization to ensure stationarity. Daily logarithmic returns are extracted and accumulated to reconstruct a normalized closing price series starting from an arbitrary base index of

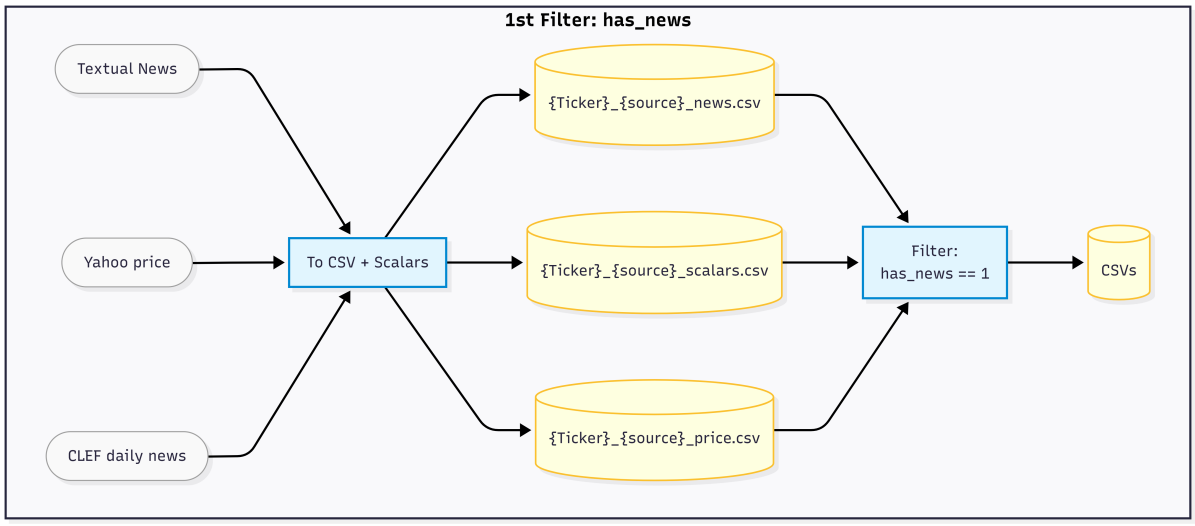

Figure 1: Data enrichment pipeline and the has_news filtering step.

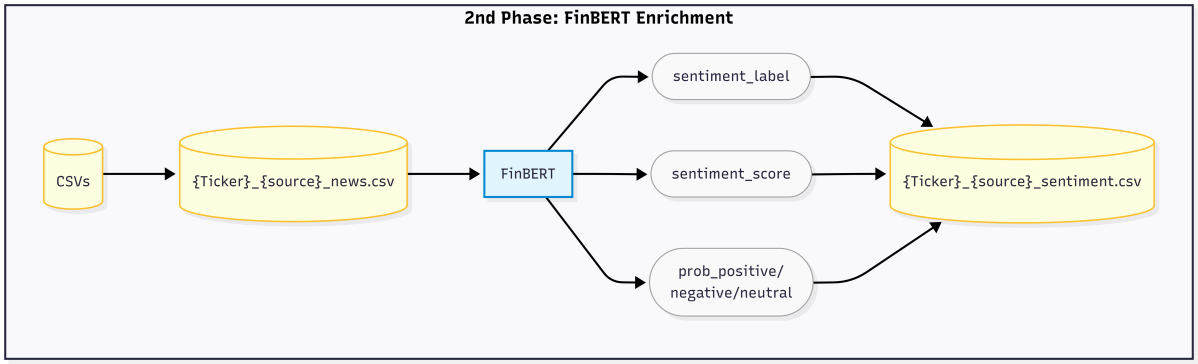

Figure 2: Phase 2 of the dataset construction pipeline: FinBERT sentiment enrichment. Each news-bearing row is processed to extract a sentiment label, confidence score, and per-class probability distribution, which are cached per asset.

100:

$$P_t^{\text{norm}} = 100 \times \exp\left(\sum_{i=1}^{t} r_i\right) \tag{1}$$

where r_i denotes the daily log-return for day i. This normalization strips out absolute scale biases across highly divergent asset classes while preserving market volatility and momentum dynamics [10]. Additionally, a sliding window of the last 10 normalized prices is stored for each row to provide short-term price trajectory context, as depicted in the merge process in Figure 3.

Fourthly, ground-truth target labels are generated through a volatility-driven discretization process to guide the optimization process. The next-day target log-return, representing the compounding growth rate, is defined as:

$$\log\_return_{t+1} = \ln\left(\frac{P_{t+1}}{P_t}\right) \tag{2}$$

To render the target signal scale-independent across different asset profiles, we compute a volatility-normalized Sharpe-like return [5] by dividing the forward return by its rolling standard deviation calculated over the preceding 20 news-bearing rows:

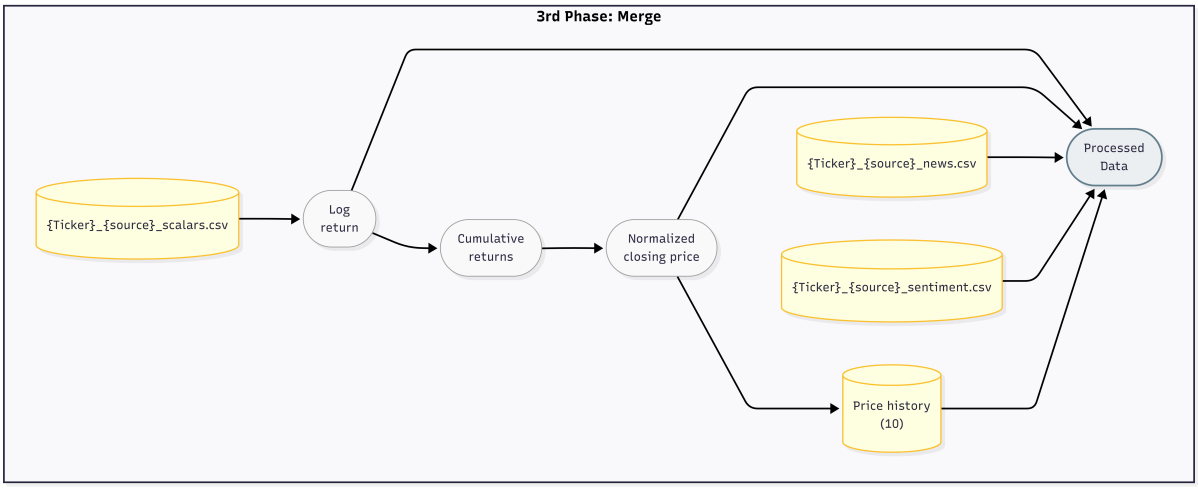

Figure 3: Phase 3 of the dataset construction pipeline: scalar merge and price normalization. The scalar CSV provides log-returns from which cumulative returns and normalized closing prices (base 100) are computed. The resulting processed data fuses news, sentiment, and scalar features by date.

$$S_t = \frac{\texttt{log\_return\_t1}_t}{\sigma_{20,t}} \tag{3}$$

where $\sigma_{20,t}$ is the rolling standard deviation. However, because this standard deviation is computed over non-contiguous news-bearing rows rather than consecutive trading days, it introduces a temporal warping that distorts the true historical volatility, as detailed as a methodological risk in Section 3.10. This normalized signal is then discretized into BUY, HOLD, and SELL action classes using asymmetric percentile thresholds following the methodology of Trading-R1 [5], as visualized in Figure 4.

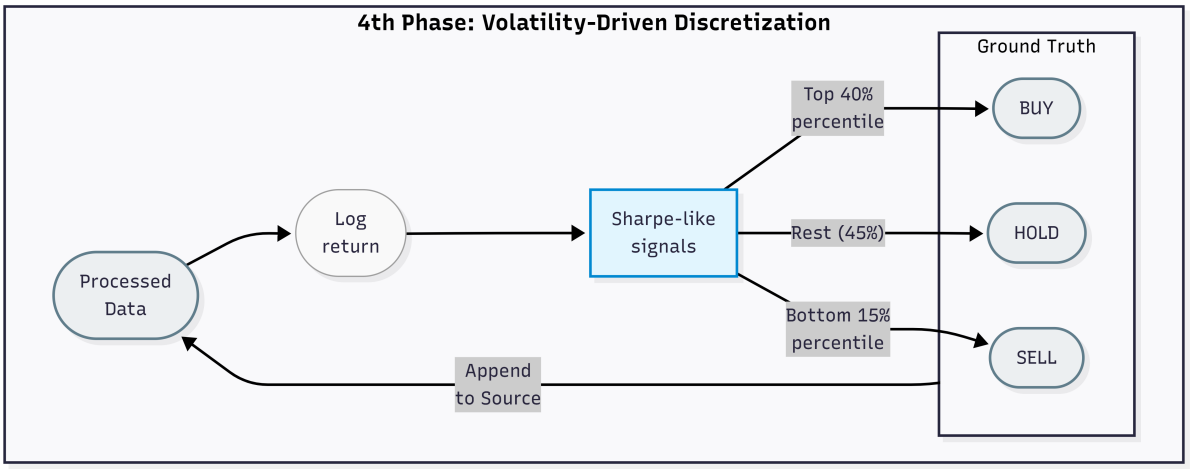

Figure 4: Phase 4 of the dataset construction pipeline: volatility-driven discretization. The log-return is divided by its rolling volatility to produce a Sharpe-like signal, which is then partitioned into BUY, HOLD, and SELL labels using asymmetric percentile thresholds.

Finally, the last phase of the pipeline handles data partitioning using deterministic temporal and source-based split rules, which are theoretically designed to prevent look-ahead contamination. All records originating from the CLEF source are unconditionally assigned to the out-of-sample test split. For Yahoo-sourced assets, rows with dates on or after the validation boundary are assigned to the validation split, while older rows are assigned to the training split. Although this deterministic partitioning, illustrated in Figure 5, aims to enforce chronological separation, a subtle design vulnerability in the order of operations causes global percentiles to contaminate the training split with future information,

Table 4
Key columns of the generated dataset consumed by the model.

Column	Type	Description
`date`	str	Trading date (YYYY-MM-DD)
`asset` / `symbol`	str	Asset identifier
`source`	str	`yahoo` or `clef`
`price_close`	float	Normalized closing price (base 100)
`momentum`	int	Momentum signal $(-1, 0, 1)$
`news`	str	Concatenated news text
`sentiment_label`	str	FinBERT class (positive/negative/neutral)
`sentiment_score`	float	FinBERT confidence
`chronos_direction`	str	Chronos forecast (up/down/neutral)[1]
`chronos_confidence`	float	Chronos confidence
`log_return_t1`	float	Next-day log-return (reward signal)
`ground_truth`	str	BUY / HOLD / SELL label
`split`	str	train / val / test

representing a leakage risk detailed in Section 3.10.

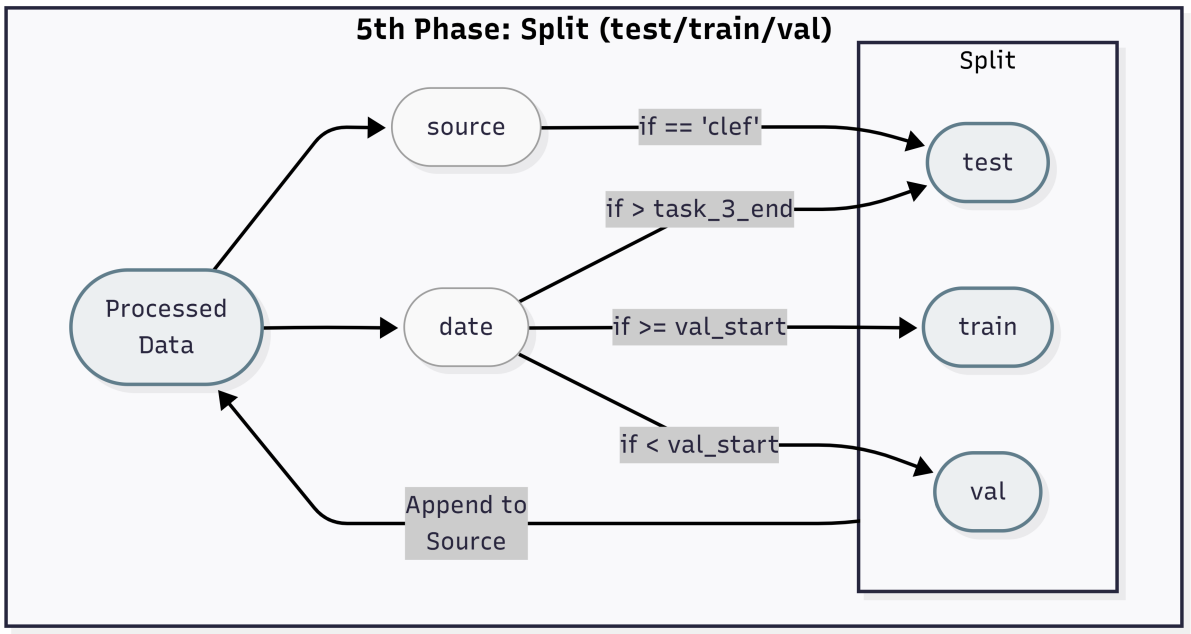

Figure 5: Phase 5 of the dataset construction pipeline: temporal split assignment. CLEF-sourced assets are unconditionally assigned to the test partition. Yahoo-sourced assets are split by date into train and val partitions.

Similarly, the Price Forecaster computes Chronos directional forecasts (up, down, or neutral) with confidence scores; however, these Chronos signals were not injected into the SFT/GRPO training prompts of the winning run and were only incorporated later during endpoint inference (see Section 3.4). The complete set of enriched signals, together with the raw momentum flag, the normalized closing price, and the next-day log-return (used exclusively as a reward signal), form the columns of the final Parquet file consumed by the trainer. Table 4 lists the most relevant fields.

[1]The Chronos columns were present in the dataset schema but were *not* used during SFT or GRPO training; they were incorporated only at inference time (Section 3.10).

3.2. Multimodal Sentiment Extraction

As introduced in Phase 2 of the dataset construction pipeline (Figure 2), the sentiment enrichment is handled by a dedicated Sentiment Extractor that processes unstructured financial news and quantifies them into actionable numerical probabilities. The module is built on *FinBERT* (`ProsusAI/finbert`) [8], a domain-specific language model adapted from BERT-base and specialized on corporate financial text. For every row with available news (`has_news` $= 1$), the module extracts a 3-class probability distribution (P_{positive}, P_{negative}, P_{neutral}) via a softmax layer. The output is cached per asset and contains the `sentiment_label`, the absolute `sentiment_score`, and the per-class probability distribution, allowing the downstream model to weight a weak positive differently from a strong one. A detailed description of the FinBERT architecture and its formulation is provided in Appendix A.1.

3.3. Prompt Orchestration

The *PromptBuilder* is the central component responsible for synthesizing sentiment analysis, technical scalars, and temporal sequences into a highly structured prompt. Crucially, to prevent look-ahead bias, the prompt for day T is generated entirely *before* the market outcome of day T is revealed.

3.3.1. The `grpo_cot_v1` Template

All training and validation examples use a single deterministic template. The template instructs the model to produce a structured response containing four blocks: a technical summary enclosed in `<technicals>` tags, a news summary in `<news>` tags, a synthesis in `<conclusion>` tags, and a final action token of the form `[[[ACTION]]]` where ACTION $\in \{$BUY, HOLD, SELL$\}$. The prompt includes the current date, asset symbol, closing price, momentum signal, FinBERT sentiment label and score, Chronos (if available) forecast direction and confidence, a news summary truncated to 2,000 characters, and a sliding price history covering the last 10 trading days. This design forces the model to reason explicitly over multimodal inputs within a constrained generation window. The complete prompt template is presented in Listing 1.

Listing 1: The `grpo_cot_v1` prompt template as constructed by the *PromptBuilder*. Placeholder variables (`{date}`, `{symbol}`, etc.) are resolved at runtime from the enriched dataset columns.

```
You are a professional financial analyst. Your task is to analyze the provided data and make a trading
    decision.

Format your response using the following structure:
<technicals>
Summarize price action, momentum, and key indicators.
</technicals>
<news>
Summarize sentiment and recent news catalysts.
</news>
<conclusion>
Synthesize why the technicals and news justify the decision.
</conclusion>
[[[ACTION]]]

Where ACTION is exactly one of: BUY, HOLD, SELL.

Data:
Date: {date}
Asset: {symbol}
Current Price: {price}
Momentum: {momentum}
Sentiment: {sentiment_label} (score: {sentiment_score:.3f})
Chronos Forecast: {chronos_direction} (confidence: {chronos_confidence:.3f})
News Summary:
{news_summary}
{sec_section}
Price History ({history_len} days): {price_history}

Analyze carefully and output your final decision in [[[ACTION]]] format.
```

While the current study focuses on structural data fusion, the *PromptBuilder* is engineered with the structural flexibility to optionally receive and process textual SEC filings (such as regulatory forms 10-K and 10-Q). In future iterations, retrieval-augmented generation (RAG) techniques [33] could dynamically select the most relevant filings and historical precedents at inference time, reducing the fixed prompt length while improving context relevance.

3.4. Temporal Encoding: The Chronos Module

To capture the predictive signal embedded in historical price trajectories, the endpoint employs **Chronos** (`amazon/chronos-t5-small`) [9], a zero-shot time-series forecasting model built on the T5 encoder-decoder architecture. Chronos quantizes continuous prices into 256 discrete bins, enabling it to generate complete probabilistic forecasts without asset-specific fine-tuning. The module produces $S = 20$ next-day price samples, from which empirical directional probabilities (P_{up}, P_{down}) are computed and classified into UP, DOWN, or NEUTRAL categories using asymmetric thresholds. Crucially, Chronos was *not* used during SFT or GRPO training—only at inference time—creating a distribution shift that is documented as a risk in Section 3.10. A complete description of the tokenization strategy, forecast extraction, and direction classification is provided in Appendix B.

3.5. Model Optimization and Training Setup

Building on the contribution introduced above, this section details how we operationalize compact GRPO training under a strict 12 GB VRAM budget while preserving structured, interpretable reasoning. Concretely, our optimization design targets two instability modes observed during preliminary runs: (i) zero-variance collapse induced by truncation and length penalties, and (ii) policy collapse toward trivial HOLD behavior, analyzed in detail in Section 3.5.3.

To address these failure modes, we redesigned both reward shaping and action extraction so that gradient flow remains informative even in short completions and low-compute regimes. The following subsections report the exact configuration used to stabilize training and obtain reproducible performance on consumer hardware.

3.5.1. Architecture and Training Configuration

The deployed winner and the best-performing fast-mode candidate use `unsloth/Phi-4-mini-instruct` [6] as the base language model, loaded in 4-bit NF4 quantization via Unsloth following the QLoRA methodology [7]. The broader Optuna search also evaluated alternative base models, but Table 5 summarizes the configuration of the selected Phi-4 mini run only. This setup is specifically designed to preserve GRPO training feasibility under a strict 12 GB VRAM constraint while maintaining structured reasoning quality.

Table 5
Base Model, Quantization, and Adapter Configuration for the Selected Phi-4 Mini Run

Component	Configuration
Base model	`unsloth/Phi-4-mini-instruct`
Quantization	4-bit NF4 (Unsloth)
LoRA target modules	`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`
LoRA rank (r)	4
LoRA scaling (α)	16
LoRA dropout	0.0
Trainable parameters	~1.2M ($<$0.03% of total model parameters)
Approx. VRAM (training)	~4.8 GB (including gradients and optimizer states)
Approx. VRAM (inference)	~2.5 GB
Full-parameter training estimate	$>$48 GB

All experiments were conducted on a single high-end consumer (prosumer) desktop environment. The training system consisted of an AMD Ryzen 9 5900X processor (24 threads, 3.700 GHz base clock), 32 GB of system RAM, and a single NVIDIA GeForce RTX 3080 Ti GPU with 12 GB of VRAM running Manjaro Linux (Kernel 5.15.202-1). The complete software stack and version pinning are detailed in the reproducibility card (Section 3.7). This hardware setup demonstrates that robust reinforcement learning via GRPO can be successfully democratized and executed on high-end consumer hardware rather than requiring enterprise-grade server clusters.

To translate these local hardware capabilities and VRAM constraints into a viable optimization process without sacrificing model reasoning depth, the training protocol is structured as a resource-efficient curriculum. Specifically, our approach adapts and downscales the SFT-plus-GRPO curriculum originally proposed by Trading-R1 [5]—which was designed for 141 GB server-grade deployments—to our 12 GB prosumer setup. Under this downscaled framework, the training pipeline is executed in two strictly sequential phases.

Phase 1: SFT Warm-up. Prior to any reinforcement learning, the model undergoes supervised fine-tuning (SFT)—where the network is optimized via next-token cross-entropy loss to replicate a set of curated formatting examples—for exactly one epoch (approximately 30 minutes) over 1,082 training examples. As shown in Figure 6, the workflow begins by splitting the dataset into training and validation folds, defining the Optuna search space, and sampling hyperparameter values for the trial. The warm-up execution itself is conditional on the configuration. When active, the pipeline applies XML label formatting to target tokens. The purpose of this phase is not to learn trading strategy, but to anchor the structural XML output format used later in GRPO. Each example is formatted with the `grpo_cot_v1` template so the model learns to emit `<technicals>`, `<news>`, `<conclusion>` tags and the final `[[[ACTION]]]` token. We keep the SFT targets ultra-concise because long completions are later truncated by the 128-token GRPO ceiling, which previously caused variance collapse inside the group. After reporting the SFT loss at `step=0`, the system evaluates the trial against Optuna's pruning thresholds. If performance is sub-optimal, the trial is pruned and archived. If the trial passes, the LoRA weights are saved. Finally, to strictly adhere to the 12 GB VRAM budget, a crucial VRAM logistics step unloads the full model from memory, releasing the entire 12 GB footprint before handoff (Connector A in the diagram) to Phase 2. This explicit unloading process invokes active model dismantling, garbage collection, and CUDA cache resetting routines (specifically clearing the CUDA cache, collecting IPC memory, resetting peak memory statistics, and invoking garbage collection). This process is mandatory to purge gradient residues, intermediate tensors, and Unsloth's compiled caches. Without this step, active CUDA allocations accumulate, leading to severe memory fragmentation or device out-of-memory (OOM) assertions. Furthermore, it prevents layer-allocation mismatches and unintended layer-shuffling between the CPU and GPU by the underlying Hugging Face `accelerate` backend when transitioning between optimization trials and phases. This compact warm-up and memory management pipeline are what make the downstream GRPO configuration in Table 6 viable on consumer hardware while preserving a strong zero-shot prior on the output structure.

Phase 2: GRPO Optimization. Following the SFT phase, training proceeds to the reinforcement learning stage shown in Figure 7. This phase begins with a VRAM reloading operation (Connector A), where the base model and the newly saved SFT adapters from Phase 1 are loaded back into GPU memory. Next, the pipeline constructs the GRPO configuration and executes the trainer. Crucially, completions are restricted to a strict 128-token ceiling per output to prevent out-of-memory errors on the 12 GB VRAM budget. To safeguard the training process against known instabilities, the pipeline runs two active callbacks: a collapse detector (to identify zero-variance output formatting failures) and an early stopping callback. Upon completion of training, the trial reports the final reward metrics at `step=1` and is concluded. The full GRPO setup is summarized in Table 6. Each change relative to the larger-server settings in Trading-R1 was deliberate: we use a small group size ($G = 4$) because it is the minimum workable setting for relative-advantage estimation; we cap completions at 128 tokens to

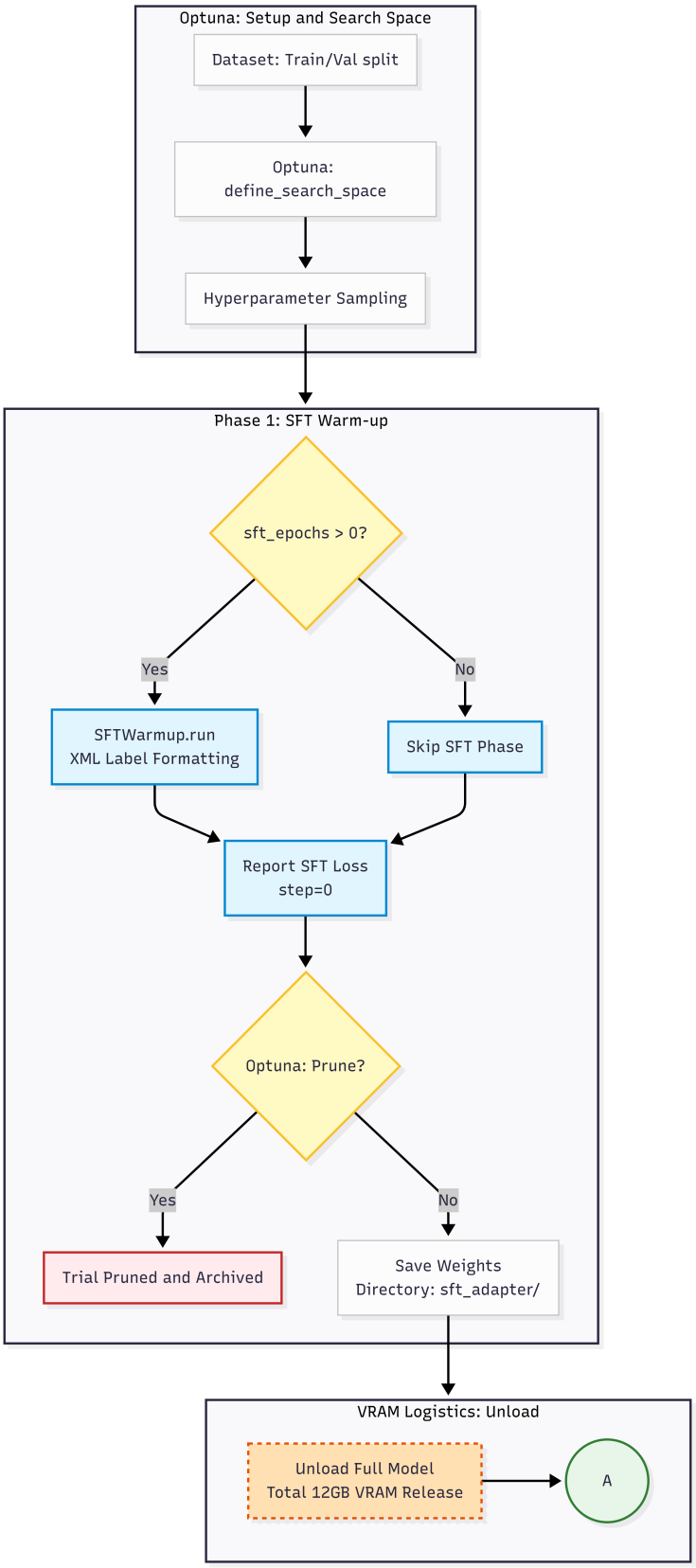

Figure 6: Phase 1 Flowchart: SFT Warm-up and Hyperparameter Optimization Setup. The pipeline defines the Optuna search space, runs conditional SFT warm-up, evaluates pruning, saves the LoRA adapter weights, and unloads the model to release VRAM before starting the RL phase.

reduce memory and latency; we keep the effective batch size at 1 with 32-step gradient accumulation to preserve stability; we set the KL coefficient and clip threshold conservatively to limit divergence; all while using a low learning rate, rank-4 LoRA, and zero dropout to make the optimization tractable on a 12 GB GPU without losing the structured-response behavior learned in Phase 1.

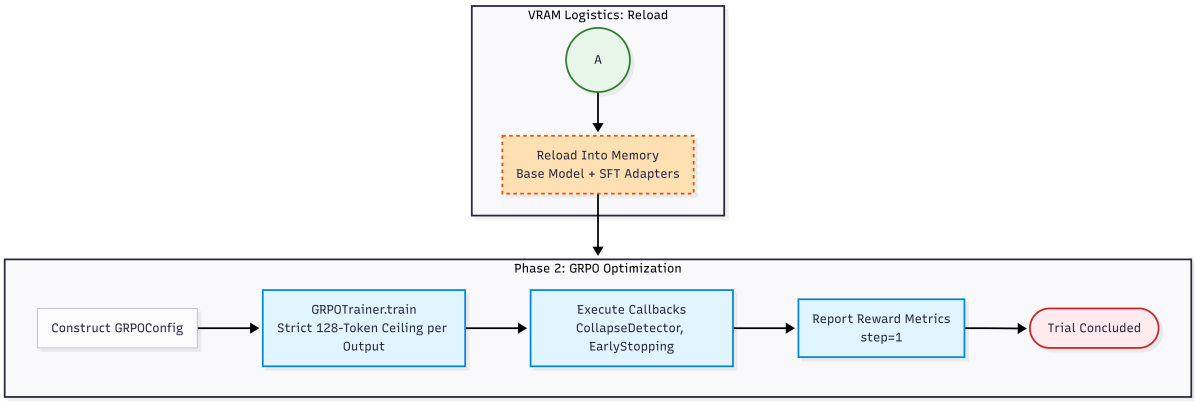

Figure 7: Phase 2 Flowchart: GRPO Optimization. The pipeline reloads the base model and SFT adapters, builds the training configuration, executes the GRPO trainer with a strict 128-token completion limit, employs early stopping and collapse detection callbacks, and saves the final policy parameters upon trial completion.

Table 6
GRPO Training Hyperparameters

Parameter	Value
Base Model	`Phi-4-mini-instruct` (4-bit NF4)
LoRA rank r	4
LoRA α	16
LoRA dropout	0.0
GRPO group size G	4
KL coefficient β	0.051
Clip ϵ	0.104
Max completion length	128 tokens
Temperature	0.573
Learning rate	2.31×10^{-5}
Per-device batch size	1
Gradient accumulation	32
Optimizer max grad norm	0.5
Epochs executed	1 (SFT) + 1 (GRPO)

During preliminary iterations, the model suffered from severe vanishing gradients and underfitting due to a mathematically easy but trivial path: policy collapse to a 100% HOLD strategy. Because cryptocurrency datasets are highly volatile, predicting HOLD minimized the expected losses relative to making directional mistakes (BUY/SELL), creating an equilibrium of inaction whose game-theoretic analysis and mitigation are detailed in Section 3.5.3. In addition, aggressive length penalties combined with model truncations at 128 tokens caused zero-variance gradient collapse, where all members of a GRPO generation group produced identical malformed or truncated responses, resulting in a zero advantage estimate and the complete death of policy gradients. We mitigated these failure modes through two key contributions: first, intensifying the false-HOLD penalty to -2.0 in our asymmetric reward matrix to mathematically break the passive equilibrium; and second, deploying the Truncated Token Rescue mechanism to ensure non-zero gradient variance even when the model's outputs are cut off mid-sentence, which shall be addressed further on.

3.5.2. Optuna Search and Experimental Protocol

The hyperparameter optimization was structured using the Optuna framework, defining a study aimed at maximizing our custom composite validation fitness metric. The study utilized a default pruner (without intermediate pruning to avoid interrupting short SFT-plus-GRPO cycles) and directed optimization toward higher fitness values. To retain high-quality configurations, we implemented an automatic checkpoint nomination and saving heuristic that monitored each trial's progress and serialized the LoRA adapter checkpoints immediately if the validation fitness exceeded a baseline threshold, registering them into a candidate pool designated as the *Hall of Fame* (HoF). Notably, while the competition provides secondary evaluation metrics such as Daily Volatility (DV) and Annualized Volatility (AV), these risk measures were intentionally omitted from the explicit selection heuristic. Because daily and annualized volatilities are directly encapsulated within the denominator of the Sharpe Ratio (SR), optimizing for SR naturally and mathematically penalizes high-volatility behaviors. This encapsulation allowed us to maintain a low-dimensional objective function while robustly controlling for volatility.

To prevent methodological confusion, a clear distinction must be made between the *offline validation fitness* used by the Optuna search wrapper and the *online step-by-step reward* computed during the reinforcement learning training phase. The online GRPO reward is a step-level optimization target computed on each generated completion, comprising the asymmetric action matrix, XML format verification, and a token-length penalty to keep reasoning traces concise (as detailed in Section 3.5.3). Conversely, the offline validation fitness is a post-hoc financial metric calculated only at the end of each training cycle over the validation folds. Specifically, it is defined as:

$$\text{Fitness} = \text{median\_SR} - 10 \times \max(0, \text{MDD}_{\text{worst}} - 0.20) \tag{4}$$

where median_SR is the median Sharpe Ratio across all validation folds, and $\text{MDD}_{\text{worst}}$ is the maximum drawdown represented as a fraction (with 0.20 representing the 20% institutional threshold). This formulation ensures that Optuna penalizes excessive tail risk (each 1% of drawdown beyond 20% reduces the fitness score by 0.1) without mixing formatting constraints or generation length limits into the validation ranking criteria.

The hyperparameter search was coordinated using Optuna with the TPE (Tree-structured Parzen Estimator) sampler and a fixed seed of 42 to ensure reproducibility across runs. Due to compute constraints, the initial experiments were executed in a fast mode with only 3 trials; each trial ran the full pipeline (dataset construction, SFT warm-up, and GRPO optimization) and dumped configuration metadata. This fast-mode configuration prioritized quick, reproducible validations to detect stability issues (e.g., collapse to HOLD) before committing larger computational budgets. Crucially, although TPE is a highly effective Bayesian optimization algorithm, it is constrained by a warm-up phase: during Optuna's initial trials, the sampler selects parameters uniformly at random to build a baseline probability distribution before activating its surrogate Bayesian model. Because our computational budget restricted the search to exactly 3 trials, the optimization never exited this warm-up phase, rendering the search functionally equivalent to a pure Random Search. We acknowledge this limitation with complete transparency, noting that while the winning configuration (Trial 0) achieved exceptional validation results, the discovered parameters represent a localized random success rather than a converged Bayesian optimum. Table 7 presents the details of the three fast-mode trials evaluated during this search.

The hyperparameter search space was structured to balance computational efficiency under strict VRAM limits and policy representation capacity, with the complete configuration detailed in Appendix A.2. The framework was engineered to support five base model architectures: `unsloth/Phi-4-mini-instruct`, `unsloth/Qwen2.5-3B-Instruct`, `unsloth/Qwen3-4B`, `unsloth/Llama-3.2-3B-Instruct`, and `unsloth/Meta-Llama-3.1-8B-Instruct`. However, due to the strict 12 GB VRAM budget, larger models were excluded from active search trials. During the fast-mode search, the sampler primarily evaluated `Phi-4-mini-instruct` and `Qwen2.5-3B-Instruct`; the former performed significantly better, maintaining structural XML compliance and avoiding the text repetition collapse that affected the Qwen base. Consequently, it was chosen as the base model for our final trading agent.

A direct consequence of this rigid seed configuration is the complete determinism of the optimization run: identical hyperparameter suggestions and model outputs are produced across runs. Launching distinct exploratory studies requires explicitly varying the Optuna seed or search-space bounds.

Table 7
Optuna TPE Fast-Mode Trial Comparison

Trial	Experiment ID	Base Model	SR	MDD	Outcome
0	exp_20260506_085649_254046	Phi-4-mini	**2.949**	**0.11%**	Deployed Winner
1	exp_20260506_093420_0bfeea	Qwen2.5-3B	-1.325	4.90%	Format collapse
2	exp_20260506_101537_905679	Phi-4-mini	-1.330	8.18%	HOLD collapse

Beyond the three trials reported in Table 7, the Optuna pipeline also evaluated checkpoints from intermediate epochs. Candidate configurations that exceeded a baseline fitness threshold were automatically saved into a pool of 11 distinct checkpoints; however, all comparisons were performed exclusively on the Yahoo validation split (896 rows), as the CLEF test split was never loaded due to a code-level oversight (Section 3.10). While the automated search isolated configurations with superior risk-adjusted fitness (e.g., a trial yielding a fitness of 1.974 and an MDD of 17.48%), the final deployment choice manually prioritized absolute Cumulative Return because this was the primary evaluation metric in the competition leaderboard, overriding the fitness constraints.

This protocol proved effective for pipeline debugging and for calibrating the asymmetric reward function. Due to limited time, all reported Optuna results correspond to fast-mode trials that executed the full pipeline in a shortened configuration. The misalignment between the fitness metric used for automated search and the Cumulative Return used for final deployment is acknowledged as a limitation in Section 3.10.

3.5.3. Reward Function Design and Engineering Contributions

The core behavioral signal of our reinforcement learning pipeline is provided by an asymmetric 3×3 reward matrix that penalizes directional errors according to their financial severity, scaled by a factor of 1.5799. Table 8 presents the exact values employed in our experiments.

Table 8
Asymmetric Reward Matrix (Values Before Scaling)

Pred. \ True	BUY	HOLD	SELL
BUY	+1.0	-1.0	-2.25
HOLD	$\mathbf{-2.0}$	+1.0	$\mathbf{-2.0}$
SELL	-2.0	-1.0	+1.0

Our key adaptation over the asymmetric matrix proposed by Trading-R1 [5] lies in the calibration of the false-HOLD penalties. In the original formulation, the penalty for incorrectly predicting HOLD during a market rally or crash was set to -1.0 (or at most -1.5). Through empirical autopsy of failed training runs, we computed the expected reward of a trivial HOLD policy on our specific data distribution—encompassing volatile crypto assets such as BTC, ETH, and SOL—and found it to be -0.0945, which dominated the expected rewards of BUY (-0.3856) and SELL (-1.1040). In game-theoretic terms, this configuration constitutes a Nash equilibrium [27] of inaction: the model learned that remaining passive incurred the least damage, collapsing to 100% HOLD predictions. By intensifying the false-HOLD penalty to -2.0, we mathematically destroyed this equilibrium, equalizing the cost of inaction with that of catastrophic directional errors and thereby forcing the policy to explore meaningful trading decisions. This penalty calibration is consistent with the reward-shaping invariance principles established by Ng et al. [22], which guarantee that potential-based reward modifications preserve the optimal policy while accelerating convergence.

To complement this financial utility matrix, we integrate structural guidance and length constraints during reinforcement learning. Drawing on the Decision Placement Reward and Length Penalty concepts introduced by Trading-R1 [5], we implemented a composite format reward adapted to our 128-token generation ceiling. Complete responses with properly closed action tags receive $+0.5$; detectable but truncated actions receive $+0.25$; severely malformed outputs receive -0.5; and missing action tokens receive -1.5 (scaled to ~ -2.37). The full case-by-case specification is provided in Appendix A.3.

Furthermore, the length penalty operates progressively rather than abruptly. Given a soft target of 240 characters (approximately 60 tokens), the penalty is computed as $\max(0, \text{len(response)} - 240) \times 0.0025$. This progressive design prevents the sudden truncation shocks that kill intra-group variance in GRPO when all four group members hit the hard token limit simultaneously.

Alongside these structural formatting guidelines, we develop a robust action extractor capable of rescuing partially generated tokens when the model exhausts its 128-token budget mid-action. For instance, a response terminating in `[[[BU` is heuristically resolved to BUY, while `[[[SEL` resolves to SELL. This rescue mechanism is not merely a post-processing convenience; it is a critical MLOps technique designed to prevent zero-variance gradient death. In standard GRPO, if all group completions are truncated identically before emitting a valid action token, the relative advantage across the group becomes zero, and policy gradients vanish. By extracting semantic signals from incomplete token patterns, we successfully maintain non-zero variance and preserve learning signals throughout training on limited hardware.

Finally, by combining these three distinct optimization objectives, the complete step-level reward function for each generated completion is mathematically formulated as:

$$R = (M \times 1.5799) + F - L, \tag{5}$$

where M is the matrix reward from Table 8, F is the format reward, and L is the length penalty. This composite function balances directional accuracy, structural compliance, and generative conciseness within an ultra-short reasoning window, ensuring that the policy adapts to the market while remaining within the accessible computational boundaries of the environment.

3.6. Inference and Endpoint Pipeline

Following the selection of the optimal candidate model, the system is served as a real-time HTTP service using FastAPI. Figure 8 illustrates the real-time request processing flow, mapping the interactions between the client, endpoint modules, and background workers under a strict 3-minute execution timeout. To manage GPU resources and prevent cold-start latency during trading sessions, the endpoint implements a structured startup and lifespan management protocol alongside a background watchdog service.

Upon startup, the FastAPI application executes a lifespan hook that instantiates the configuration manager and model pool wrapper. If the system is configured in production mode and a valid checkpoint exists, the application triggers model preloading in a separate background thread. the model pool reads the endpoint configuration and loads the quantized model weights into memory. This asynchronous initialization prevents the service startup from blocking, while ensuring the model is fully loaded and ready to serve the first trading request immediately. To prevent memory leaks and idle GPU resource consumption, this same lifespan hook launches a background watchdog service (dubbed *Nightwatch*) that runs continuously on a 30-second loop. The watchdog monitors the loaded model's idle duration and tracks the current UTC time. If the model remains idle for more than 3 hours, or if the system clock approaches midnight UTC (a period of low trading activity), the watchdog unloads all models from the GPU to restore the hardware to a clean, idle state.

When a client sends a trading request containing a JSON payload with the asset's current closing price, news summaries, and scalar metrics, the endpoint routes the payload through the data ingestion module. This module appends the raw JSON payload to a local transaction log to persist every incoming request. To reconstruct the rolling price sequence needed for feature engineering, the handler prioritizes the client-provided price array if available. If this field is absent, the system falls back to loading the

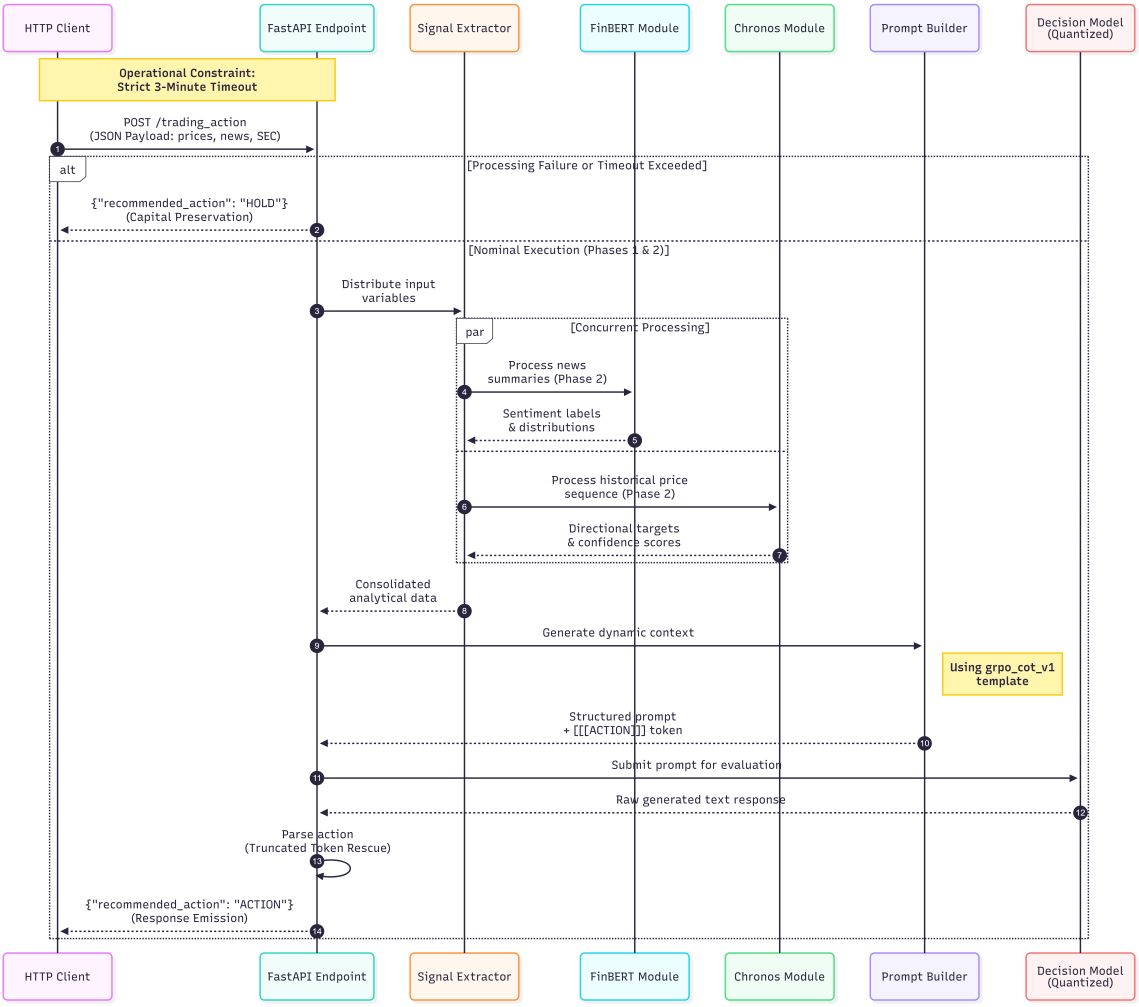

Figure 8: FastAPI Inference and Endpoint Pipeline. The sequence diagram details the concurrent processing of sentiment and trajectory signals on CPU, dynamic prompt orchestration, quantized model inference, token parsing with rescue mechanisms, and global exception fallbacks.

last 10 requests from the local transaction log, reconstructing the sequence on the fly. Following this ingestion step, feature extraction and signal enrichment are executed entirely on the CPU to prevent resource contention with the primary GPU-bound decision model. Specifically, the preprocessor passes the incoming news summaries through a CPU-based FinBERT pipeline to compute sentiment probability distributions, falling back to a keyword-based sentiment heuristic if the module is unavailable or fails. Simultaneously, the reconstructed price sequence is passed to a CPU-based Chronos module to forecast directional target distributions and confidence levels, using a rolling moving-average trend heuristic as a fallback. These outputs are subsequently merged with volatility and momentum scalars into an enriched data dictionary matching the training schema.

Once this data dictionary is compiled, it is ingested by the `PromptBuilder`, which formats the values into a structured text prompt using the `grpo_cot_v1` XML-style template, appending the trailing token sequence `[[[ACTION]]]` to force action generation. The endpoint then queries the model pool to retrieve the inference model wrapper, dynamically reloading the model into GPU memory if it was previously unloaded by the watchdog. The prompt is then submitted for batch evaluation, returning the raw model completion. Next, the raw text output is passed to the parser, which runs our custom action-extraction logic; if the completion is truncated mid-sentence due to the strict 128-token limit, the Truncated Token Rescue mechanism scans the generated text to recover the model's intended action. In accordance with strict capital-preservation principles, the entire request lifecycle is wrapped in a

global exception handler, ensuring that if any failure occurs during ingestion, CPU-based preprocessing, model reloading, or text generation, the system logs the traceback and immediately returns a safe default recommendation of HOLD. Lastly, to facilitate operational monitoring, the endpoint exposes two auxiliary paths: a health endpoint returning operational status and model load metrics, and a timing endpoint returning detailed performance profiles.

3.7. Reproducibility and Replication Environment

Although full deterministic replication of our results is not guaranteed—owing to non-deterministic internals in the Unsloth quantization library that are not fully controllable via seed fixing—we provide a complete specification of seeds, environment, and dataset parameters so that the training methodology and VRAM management strategy can be studied and adapted. The final dataset used for training is packaged as a unified Parquet file with a SHA-256 checksum to guarantee data integrity. The software dependencies are specified in our environment card, including PyTorch (v2.11.0+cu126), Transformers (v4.57.6), PEFT (v0.19.1), TRL (v0.24.0), Unsloth (v2026.3.11), and BitsAndBytes (v0.49.2).

To isolate the performance impact of different hyperparameter configurations, the seed allocation was structured to maintain identical starting conditions across trials. The global model seed (which governs model initialization and LoRA stochasticity) is fixed to 42 across all trials, while the Optuna sampler seed varies across runs so that the TPE sampler selects a different set of hyperparameters for evaluation in each study. The full seed table is provided in Appendix A.5.

However, achieving full determinism across the reinforcement learning pipeline requires addressing multiple sources of stochasticity. To establish a reproducible baseline, a global random seed is initialized at the start of execution, configuring the generators for Python, NumPy, PyTorch, and Hugging Face Transformers. This configuration is consistently maintained in downstream components, such as the parameter-efficient fine-tuning (PEFT) framework and evaluation validators. Despite these precautions, external dependencies can introduce unaligned randomness; specifically, Unsloth's compiled execution cache defaults its internal training seeds to a hardcoded value (3407). To mitigate this replication risk, future iterations of our pipeline will pass the global seed explicitly to all external trainer interfaces and dynamically override the Unsloth seed defaults in runtime memory before fine-tuning, thereby aligning all active components under a single source of truth.

Replicating the experimental pipeline involves three sequential execution steps. First, the dataset builder is run to ingest the granular CSV files, execute the FinBERT cache model, and compile the unified Parquet dataset. Subsequently, the Optuna searcher is executed over three trials with fast mode enabled to conduct rapid SFT-plus-GRPO trials. Finally, validation is performed against the winning LoRA checkpoint to execute a full temporal-fold evaluation on the validation split and export the complete financial metrics.

3.8. Validation Framework

To ensure the statistical rigor of our model assessment and prevent over-optimistic generalization estimates, we established a comprehensive temporal validation framework that operates under the two distinct evaluation regimes introduced in Section 3: a rapid fast-mode protocol for candidate screening during hyperparameter optimization, and a full multi-fold protocol for final model selection. As documented in Section 3.10, the CLEF test split was never loaded, so all reported results correspond exclusively to the Yahoo validation split.

Under the fast-mode regime, the validation process is optimized to provide immediate feedback loops for the Optuna trials. As detailed in Section 3.1, this configuration restricts the temporal scope to the most recent 12 months and evaluates candidate models on a representative subset of 100 validation rows. This reduction in context size cuts the evaluation time of a single model candidate to approximately 32 seconds, enabling rapid detection of formatting failures or policy collapse (such as a 100% HOLD strategy) while preserving the underlying asset and news distributions.

3.8.1. Validation Protocol

To evaluate the candidates, we perform out-of-sample backtesting under the two validation regimes. During training, the fast-mode protocol provides localized feedback on the last temporal fold, whereas the post-training full validation protocol evaluates candidates across the complete validation split of the Fast mode configuration (896 rows). This full validation is executed using three contiguous, non-overlapping temporal folds built in chronological order: Fold 0 (from 2024-02-07 to 2024-06-05, containing 360 rows), Fold 1 (from 2024-06-06 to 2024-10-03, containing 294 rows), and Fold 2 (from 2024-10-04 to 2025-02-01, containing 242 rows). For each individual fold, trading actions are simulated dynamically with weights $w_t \in \{+1, 0, -1\}$, and daily returns are calculated as $r_t = w_t \times (P_t - P_{t-1})/P_{t-1}$. The aggregate fitness score is then computed according to Equation 4, and the definitions of all secondary financial metrics are summarized in Table 9. To maintain statistical isolation, all validation sets contain exclusively YAHOO-sourced assets, specifically BTC_YAHOO, ETH_YAHOO, and SOL_YAHOO. CLEF-sourced assets were reserved as an out-of-sample test set but were never evaluated due to the oversight described in Section 3.10.

3.9. Results

Following the validation framework detailed in Section 3.8, we now present the empirical results from both fast-mode Optuna trials and the comprehensive complete-validation protocol executed against our finalist models.

3.9.1. Training and Fast-Mode Results

Within the fast-mode evaluation, the winning trial (Trial 0, `exp_20260506_085649_254046`) achieved a fitness of 2.949 with an MDD of merely 0.11% over 100 rows. The action distribution at this stage was extremely conservative: 96% HOLD, 4% SELL, and 0% BUY. While this distribution suggested the model had learned to avoid catastrophic errors, it also indicated that the anti-HOLD calibration had not yet fully activated exploratory BUY behavior within the restricted fast-mode window. The two subsequent Optuna trials (using Qwen2.5-3B and a different hyperparameter configuration for Phi-4-mini, respectively) collapsed to repetitive text or 100% HOLD.

3.9.2. Complete Validation Results

The complete validation stage evaluated 11 candidate models on the complete validation split of the Fast mode configuration (comprising 896 rows across the three temporal folds). This full-validation protocol assessed temporal robustness and risk-adjusted performance, prioritizing the fitness metric defined in Equation 4 to penalize drawdowns exceeding the institutional 20% threshold.

To analyze the finalists, each model's performance was evaluated across seven complementary dimensions. Table 9 provides a complete description of each metric used to report validation outcomes. While the fitness metric serves as our primary optimization objective, the remaining six metrics capture different aspects of trading efficacy: risk-adjusted profitability (SR), cumulative return (CR), capital preservation (MDD), directional accuracy (Acc.), trade success rate (WR), and cumulative profitability scaled by drawdown magnitude (PF).

To contextualize these results, Table 11 reports the performance of five naive and zero-shot strategies evaluated on the same complete validation split (1,007 rows) using the identical pipeline and financial metrics.

The baselines reveal that the deployed model's 90.67% Cumulative Return substantially outperforms the Buy & Hold benchmark (28%), confirming that the learned policy captures genuine directional signal rather than passively riding market beta. Furthermore, the best-fitness model (`111504_33de19`) achieves a Sharpe Ratio of 1.97—nearly 40% higher than Buy & Hold (1.41)—while maintaining a drawdown well within institutional bounds (17.5% vs. 42.9%). The random and always-SELL strategies produce negative returns and negative Sharpe Ratios, confirming that the validation period was not

Table 9
Performance Metrics Reference

Metric	Definition	Interpretation
Fitness	Median Sharpe Ratio with penalty multiplier $-10 \times \max(0, \text{MDD} - 0.20)$ for drawdowns exceeding 20%	Primary optimization objective; higher is better; severe penalty for exceeding risk threshold
SR (Sharpe)	Excess return divided by portfolio volatility	Risk-adjusted profitability; higher indicates stable returns relative to fluctuations
MDD	Maximum cumulative loss from peak to trough (%)	Key downside risk measure; values >20% trigger fitness penalties; lower is safer
CR (Cumulative Return)	Total percentage gain or loss across the entire validation period (%)	Absolute profitability over the full period; higher indicates greater total wealth accumulation; can be misleading without considering drawdown
Acc. (Accuracy)	Proportion of BUY/HOLD/SELL decisions aligned with realized price direction	Directional correctness of model predictions; higher indicates better signal quality
WR (Win Rate)	Percentage of closed trades generating profit	Trade success frequency independent of magnitude; higher indicates consistent small wins
PF (Profit Factor)	Cumulative gains divided by cumulative losses	Net profitability; >1.0 profitable, <1.0 net losses, Inf/undefined indicates no losses

Table 10
Complete Validation Results — Selected Finalists

Exp.	Fitness	SR	MDD	CR	Acc.	WR	PF
085649_254046	0.1069	2.9117	48.05%	**90.67%**	38.31%	54.83%	1.513
115709_051aed	-0.4434	1.4039	38.47%	22.74%	38.99%	51.91%	1.249
004508_27f172	-0.0759	**3.0408**	51.17%	89.76%	38.87%	55.81%	1.568
014550_f9ad37	-0.8756	1.4105	42.86%	28.16%	40.23%	50.90%	1.225
113530_8bc775 (ep4)	1.2134	1.2134	**14.15%**	9.44%	40.96%	52.93%	**1.697**
062606_1cd3e0	0.0000	0.0000	0.00%	0.00%	43.32%	0.00%	∞
221904_9bb326	-2.5173	-0.2100	43.07%	-3.18%	43.64%	50.89%	0.928
113530_8bc775 (ep2)	-0.6082	-0.1113	24.97%	-1.12%	42.12%	51.23%	0.970
111504_33de19	**1.9740**	1.9740	17.48%	19.34%	41.85%	**58.14%**	1.687
100749_12f83a	0.0000	0.0000	0.00%	0.00%	43.32%	0.00%	∞
204602_88701e	-1.4462	-0.1954	32.51%	-1.89%	**43.72%**	48.85%	0.912

trivially bullish. Notably, the always-HOLD baseline achieves an accuracy of 43.3%—higher than both trained models—simply because it matches the most frequent ground-truth class in the validation set, yet it produces zero return, illustrating that directional accuracy alone is an insufficient metric without accompanying trade execution. These relative improvements, however, must be interpreted with caution. The zero-shot Phi-4-mini baseline—which receives the identical structured prompt but undergoes no SFT or GRPO training—achieves a Cumulative Return of only 3% and a Sharpe Ratio of 0.57—substantially worse than Buy & Hold in absolute return (3% vs. 28%) and risk-adjusted profitability (SR 0.57 vs. 1.41), though with a lower maximum drawdown (30.1% vs. 42.9%) and comparable accuracy and win rate. The

Table 11
Baseline Strategies on the Complete Validation Split

Strategy	Fitness	SR	MDD	CR	Acc.	WR	PF
Buy & Hold (always BUY)	-0.876	1.41	42.9%	28%	40.2%	50.9%	1.22
Sell & Hold (always SELL)	-5.827	-1.41	64.2%	-33%	16.5%	48.3%	0.82
Always HOLD	0.000	0.00	0.0%	0%	43.3%	0.0%	∞
Random (uniform)	-4.384	-0.87	55.1%	-20%	29.0%	47.3%	0.87
Zero-shot Phi-4-mini	-0.441	0.57	30.1%	3%	40.4%	53.0%	1.21
Deployed model	0.107	**2.91**	48.1%	**90.67%**	38.3%	54.8%	1.51
Best fitness model	**1.974**	1.97	**17.5%**	19.3%	41.9%	**58.1%**	**1.69**

deployed trained model improves upon this zero-shot baseline by an order of magnitude in Cumulative Return (90.67% vs. 3%) and more than fivefold in Sharpe Ratio (2.91 vs. 0.57). While this gap suggests that the SFT-plus-GRPO curriculum contributes measurable value beyond formatting compliance, these numbers are all computed on the same in-distribution validation split and are therefore subject to the same look-ahead bias and lack of out-of-sample generalization documented in Section 3.10 and Section 3.10. Moreov

As summarized in Table 10, the experiments represent a comprehensive suite of 11/11 checkpoints evaluated under the complete validation regime on the Yahoo validation split (896 rows); no evaluation was performed on the held-out CLEF test set (Section 3.10). Notably, trials `062606_1cd3e0` (Phi-4-mini) and `100749_12f83a` (Qwen2.5-3B) both collapsed into a pathological, risk-neutral 100% HOLD strategy, producing metrics identical to the always-HOLD baseline in Table 11. As analyzed in Section 3.5.3, this passive stance is a statistically easy way to minimize short-term drawdown under volatile market conditions, but it fails to capture positive asset returns.

A separate observation from Table 10 also warrants clarification. The coexistence of a high Sharpe Ratio (2.91) with a severe Maximum Drawdown (48.05%) in model `085649_254046` warrants clarification, as these metrics are computed differently across the validation folds. The evaluator produces one *FoldResult* per (fold, asset) pair—yielding 9 independent results (3 folds $\times$ 3 assets)—and the fitness function aggregates them asymmetrically: the reported SR is the *median* across all 9 values, while the MDD is the *worst* (maximum) across the same 9 values, potentially originating from a completely different (fold, asset) pair. Furthermore, the SR is annualized by multiplying the daily Sharpe by $\sqrt{365}$ (the crypto annualization factor), which amplifies a modest daily edge (mean/std ≈ 0.15) into an annualized SR ≈ 2.87. A brief adverse streak of 5–10 consecutive losing days in a single fold can produce a 48% drawdown without significantly affecting the median daily performance across the remaining folds, explaining how both metrics coexist.

In addition, table 10 reveals two co-equal candidate profiles that the pipeline successfully produced. Model `111504_33de19` attained the highest fitness (1.974) with an MDD of only 17.48%, exhibiting superior win rate (58.14%) and accuracy (41.85%). Under classical portfolio theory and institutional risk mandates—where capital preservation and drawdown control are paramount—this would be the preferred deployment candidate, particularly for regulated asset management or fiduciary contexts that enforce hard loss limits. In contrast, model `085649_254046` achieves the highest Cumulative Return (90.67%) at the cost of a 48.05% drawdown—a profile closer to high-conviction discretionary trading, where the investment horizon tolerates significant interim losses in exchange for maximal terminal wealth. The existence of both profiles within the same pipeline demonstrates that the training and reward infrastructure is not locked into a single risk posture; rather, the selection criterion determines the ultimate risk-return trade-off. For the CLEF 2026 competition, model `085649_254046` was selected because Cumulative Return constituted the primary leaderboard metric. We acknowledge, however, that in a production deployment governed by institutional risk constraints, model `111504_33de19` would be the more appropriate choice, and we present both as legitimate outcomes of our framework.

Furthermore, the validation results expose the dangers of relying solely on fast-mode metrics. The

fast-mode MDD of 0.11% was catastrophically optimistic compared to the complete-validation MDD of 48.05%, confirming that evaluation over a single truncated fold produces an illusion of robustness. This finding reinforces the necessity of multi-fold temporal validation in financial machine learning.

3.9.3. Official Task 3 Leaderboard Snapshot

For camera-ready reporting, we freeze the corrected official Task 3 leaderboard snapshot dated 2026-07-05. The ranking metric is Cumulative Return (CR) under the Long/Short setting, averaged across BTC and TSLA. Under this corrected snapshot, BTC ranks 12th out of 18 participants, while TSLA ranks last (18th out of 18). The TSLA reporting window is limited to 2026-05-11 through 2026-06-26, because later TSLA dates were affected by an organizer-side payload issue and were treated uniformly as HOLD/flat for all agents. The supporting dashboards are shown in Figures 9 and 10, and the raw camera-ready reporting snapshot is summarized in Appendix A.4. In both dashboards, the top summary strip is ordered as Agent, Return, Vs Buy & Hold, Sharpe Ratio, and Win Rate.

Table 12
Official corrected Task 3 leaderboard snapshot used for camera-ready reporting. Ranking is based on Long/Short Cumulative Return averaged across BTC and TSLA.

Asset	Coverage used for reporting	Rank / 18	Notes
BTC	2026-05-11 to 2026-07-04	12	Official corrected snapshot; ranking based on Long/Short CR
TSLA	2026-05-11 to 2026-06-26	18	Dates after 2026-06-26 treated uniformly as HOLD/flat due organizer-side payload issue

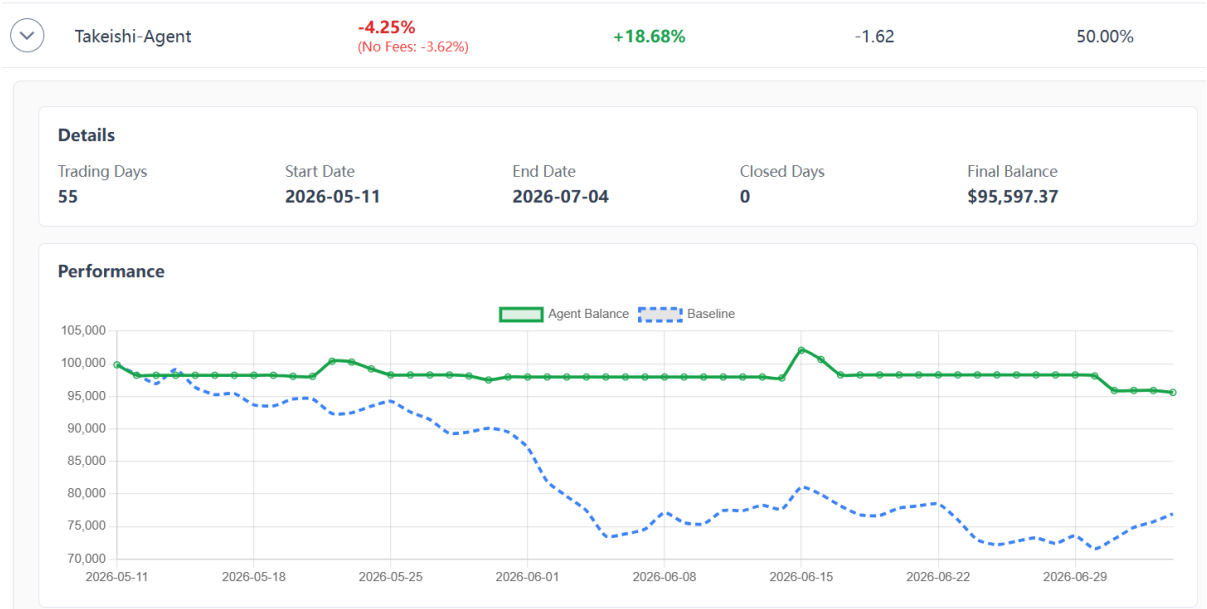

Figure 9: BTC official report window from 2026-05-11 to 2026-07-04. The agent line stays close to the initial capital, shows a brief mid-June spike, and ends at $95,597.37 (-4.25% with fees; -3.62% no fees), while the dashed baseline falls much more sharply. The reported alpha versus Buy & Hold is +18.68%.

3.10. Risks, Limitations, and Technical Debt

In accordance with transparent research practices, we document the core limitations, experimental discrepancies, and outstanding technical debt of the current system. These factors represent key areas

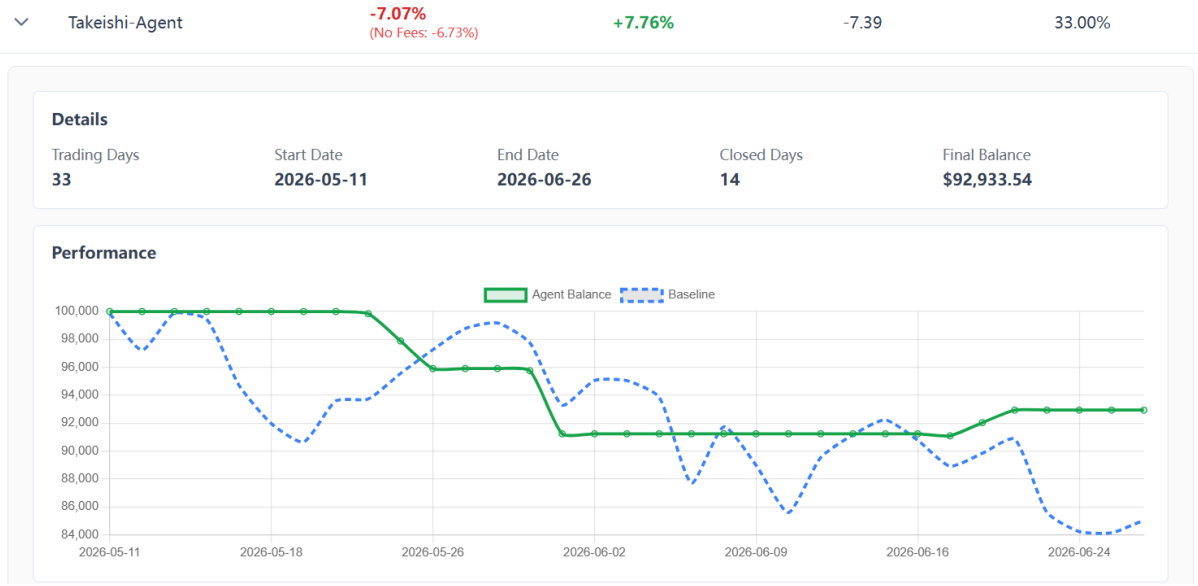

Figure 10: TSLA official report window from 2026-05-11 to 2026-06-26. The agent stays conservative, drops from the initial capital to $92,933.54 (-7.07% with fees; -6.73% no fees), then stabilizes with a slight recovery; the dashed baseline declines more sharply. The reported alpha versus Buy & Hold is +7.76%. Dates after 2026-06-26 were affected by an organizer-side payload issue and were treated uniformly as HOLD/flat.

of caution and serve as the direct foundation for future architectural improvements.

Training-Inference Discrepancy (Chronos Shift) As described in Section 3.1.1, the Chronos directional forecast was omitted during SFT and GRPO training but included in the live inference prompt and post-hoc validation. While this forecast was intended to enrich the model's environment, this discrepancy creates a distribution shift between the text distribution observed during training and that encountered during live execution. In future iterations, SFT and GRPO training must be executed with identical prompt context builders to avoid potential formatting or reasoning degradation. Alternatively, Chronos should be removed from the inference prompt until it can be included during training, to maintain a consistent input distribution.

Training-Serving Skew (Normalized Price Scale-Mismatch) A critical data-engineering risk in production deployment is the training-serving skew arising from price scale differences. The model was trained using exclusively reconstructed normalized closing prices starting from an arbitrary base index of 100 (Equation 1), whereas a live inference endpoint might receive raw, absolute market prices (e.g., BTC priced above $60,000). Because Large Language Models (LLMs) are highly sensitive to the tokenization of numeric magnitudes, presenting raw absolute prices directly to the prompt would represent a severe out-of-distribution format that can degrade reasoning and policy performance. To mitigate this discrepancy, the production inference pipeline must replicate the normalization preprocessing logic in a dedicated input layer—similar to the raw-to-normalized data harmonization layer described in P1GPT [19] and the normalization for stationarity in FinMultiTime [10]. The live endpoint must calculate log-returns from absolute incoming prices and reconstruct the base-100 price sequence before injecting them into the technical indicators block. Crucially, scale-invariant features such as daily volatility, log-returns, and news sentiment are unaffected by this skew.

Lack of Asset Diversity in Training Due to computational constraints, the SFT and GRPO training set was temporally filtered to a 12-month subset consisting exclusively of cryptocurrency assets (BTC, ETH, and SOL). No equity assets (such as TSLA or MSFT) were present in the training corpus. Given that the out-of-sample CLEF test set contains both crypto and highly regulated equity instruments, the

model's policy is exposed to severe out-of-distribution generalization risks when trading traditional equities. Expanding the training corpus to include equities is a necessary precondition for any credible generalization claim.

Execution Constraints and Data Augmentation Both SFT and GRPO phases were limited to a single epoch due to time and resource constraints under a 12 GB VRAM budget. Furthermore, while the configuration scripts and metadata flags allow advanced data augmentation strategies—specifically block shuffling, subset sampling, and news bucketing—these were never actually executed in the training runs that produced the candidate models.

Optuna Trial Limitation and Random Search Equivalence As detailed in Section 3.5.2, the three-trial computational budget confined the Optuna TPE search entirely within its startup phase, rendering the optimization equivalent to a random search. The discovered configuration, while effective, should therefore be considered a localized success rather than a converged Bayesian optimum. Expanding the trial envelope remains a critical direction for future work.

Optimization Strategy and Multi-Objective Alignment Furthermore, we note a conceptual mismatch between the automated optimization target and the final deployment metric. While the compound validation fitness metric was designed to optimize risk-adjusted stability by combining the Sharpe Ratio with severe drawdown penalties, the final deployment decision manually prioritized absolute Cumulative Return to maximize standing in the primary competition ranking. Because the hyperparameter search was constrained to only three initial random trials (effectively behaving as a random search), this misalignment did not result in specialized overfitting to the compound metric, yet it highlights the challenge of multi-objective optimization in financial reinforcement learning. Rather than relying on manual post-hoc overrides that discard risk constraints, future work must study how to formalize multi-objective optimization (such as Pareto-optimal hyperparameter search or multi-objective reward shaping) to simultaneously optimize for risk-adjusted stability and absolute return.

Global Percentile Leakage (Look-Ahead Bias in Ground-Truth Construction) A critical data-integrity risk was identified in the ground-truth labeling pipeline. The BUY and SELL thresholds are derived by computing percentiles over the entire signal column *before* the chronological split is applied. This ordering means that the percentile boundaries used to label each training row are computed with full knowledge of future market regimes, including extreme volatility spikes that occurred months after the row's date. In concrete terms, when the model was learning buy and sell patterns from February 2024, the ground-truth labels assigned to those rows were already contaminated by the price dynamics of late 2024 and early 2025. This constitutes a classic case of look-ahead bias, a well-documented form of data leakage in financial machine learning that inflates apparent model performance by embedding future distributional information into historical labels. While the magnitude of the distortion depends on the stationarity of the signal distribution, the presence of heavy-tailed cryptocurrency returns makes this leakage particularly dangerous, because a single extreme outlier month can shift the global percentile boundaries substantially and relabel otherwise neutral days as actionable signals. The corrective action—computing percentile thresholds exclusively on the training partition after the chronological split—is straightforward and constitutes the highest-priority fix for any continuation of this work.

Time-Warped Volatility Metric (Discontinuous Rolling Windows) A second methodological risk arises from the order in which data cleaning and feature engineering are applied. During dataset construction, rows without associated news are removed from the DataFrame *before* the rolling-window volatility and momentum indicators are computed. Because the news-filtering step eliminates non-contiguous calendar days, the resulting DataFrame no longer represents a continuous time series: consecutive rows may span gaps of several trading days or even weeks. Consequently, a nominal 20-row

rolling window does not correspond to 20 consecutive trading days but may instead cover 40, 50, or more calendar days. This temporal warping corrupts the standard deviation estimate, producing a volatility metric that conflates genuine intra-period variance with artificial jumps introduced by the missing-day gaps. Since this distorted volatility is used both in the denominators of the momentum signal and in the percentile-based ground-truth thresholds, its downstream effects propagate throughout the label space and the reward signal, potentially degrading the quality of the learned policy. The corrective action requires computing rolling-window statistics on calendar-continuous data *before* applying the news filter, then rejoining the features to the filtered dataset.

Transactional Log Contamination (Local Request History) An operational vulnerability in the live endpoint lies in the persistence and reloading of incoming client requests. The data ingestion module appends every POST request to a single local transaction file. If the endpoint is subjected to offline integration testing, sanity checks, or debugging queries, these synthetic requests are recorded in the exact same log alongside official evaluation queries. Since the endpoint reconstructs the recent price history from this transactional log when the payload lacks a client-provided price array, any synthetic test prices can contaminate the sequence. In production environments, testing endpoints and production databases must be strictly isolated to guarantee data integrity.

Pipeline Replication Risk (Unaligned External Stochasticity) Despite setting global seeds, complete deterministic behavior across our reinforcement learning pipeline is not fully guaranteed due to unaligned sources of stochasticity introduced by external library dependencies. While the primary system seed is correctly initialized at startup and propagated to core components, external trainers like Unsloth default their internal seeds to a hardcoded value in compiled code. The presence of multiple, unaligned sources of entropy introduces a risk where identical training runs might produce slightly divergent policy behaviors. To fully eliminate this replication risk in future iterations, the codebase must explicitly pass the system configuration seed to all external trainer interfaces and dynamically override Unsloth's runtime defaults before commencing SFT or GRPO.

Data Provenance Risk (Untraceable Solana News Source) A notable data provenance risk affects the Solana (SOL_YAHOO) subset used during training and validation. While news headlines for Bitcoin and Ethereum were collected from verifiable sources, the specific news headlines and text dataset utilized for Solana were acquired from a legacy repository whose original scraping source and licensing details could not be recovered. Although these Solana news headlines were successfully integrated to train the policy, the inability to trace their exact origin poses a significant reproducibility limitation and highlights outstanding technical debt in our data governance. Future revisions of the pipeline must replace this untraceable dataset with a clean, fully documented, and reproducible news feed.

Unused Out-of-Sample Test Set (Validation Engine Oversight) A significant methodological limitation was identified post-submission: the held-out CLEF test set (comprising 1,333 rows across five out-of-distribution assets: BTC_CLEF, ETH_CLEF, TSLA_CLEF, MSFT_CLEF, and BMRN_CLEF) was correctly constructed and partitioned during dataset building, but was never actually loaded or evaluated during the final model selection process. The root cause is a code-level oversight in the validation engine: the dataset builder returns only the training and validation splits; the test split is never loaded in that execution path. Consequently, all 11 candidate model comparisons—including the selection of the deployed winner—were performed exclusively on the Yahoo validation split (896 rows of in-distribution cryptocurrency data). This means the pipeline lacks any empirical evidence of the model's generalization to unseen asset classes or market regimes beyond the Yahoo-sourced cryptocurrencies used for training and validation. The omission was a human error in code review, not a deliberate design choice. The team became aware of this oversight after the submission deadline, at which point the trained model and production environment were no longer accessible for re-evaluation. We acknowledge this as the

most significant limitation of the current study and prioritize its correction as the highest-priority item for future work.

4. Conclusion

In this Working Note, we documented a multimodal trading pipeline for cryptocurrency and financial asset decision making in the CLEF 2026 FinMMEval Lab [1], specifically targeting Task 3 [2]. Our work demonstrates key technical achievements: we successfully compressed and fine-tuned a quantized `Phi-4-mini-instruct` policy under a 12 GB VRAM budget, enabling rapid hyperparameter exploration and confirming that reloading model checkpoints between curriculum phases prevents formatting collapses. However, these achievements are overshadowed by critical methodological errors—including the unaligned library seeds that prevent deterministic replication (Section 3.10), the untraceable Solana news corpus (Section 3.10), and the validation engine oversight that prevented out-of-sample evaluation (Section 3.10)—which compromise the scientific rigor of the empirical results. Consequently, this study offers a blueprint for resource-efficient training rather than a fully verified, replicable trading agent.

Empirically, our validation results reveal a trade-off between capital growth and risk-preservation constraints. While optimizing our fitness metric yielded risk-adjusted candidates such as `111504_33de19` (fitness 1.974, MDD 17.48%), the final selection favored `085649_254046` (Cumulative Return 90.67%, MDD 48.05%) to align with the competition's primary metric. This divergence highlights a key lesson: optimization metrics must be aligned with competition objectives from the start to avoid post-hoc manual overrides.

To address these shortcomings, future work must follow a strict triage of priorities. **Immediate corrections** (required before any further empirical claims can be made): eliminate the look-ahead bias by computing percentile thresholds exclusively on the training partition after the chronological split (Section 3.10); refactor the validation engine to load and evaluate the CLEF test split, even if results are poor (Section 3.10); and compute rolling-window statistics on calendar-continuous data before applying the news filter (Section 3.10). **Near-term improvements** (required for a competitive re-submission): include Chronos signals during training or remove them from the inference prompt to eliminate the distribution shift (Section 3.10); expand the training dataset to include traditional equities alongside cryptocurrencies (Section 3.10); implement proper price normalization in the FastAPI endpoint (Section 3.10); and execute full-length training runs with an extensive Optuna hyperparameter search (Section 3.10). **Long-term exploration** (once the stable baseline is established): study advanced reward-shaping for the asymmetric matrix, test alternative prompt designs, investigate dual-LLM configurations, and formalize multi-objective optimization to simultaneously optimize for risk-adjusted stability and absolute return (Section 3.10).

Acknowledgments

We extend our sincere gratitude to our school classmate Irving Axel Rodriguez Cuautle for his valuable advice and initial contributions at the beginning of this project; we deeply appreciate his guidance and continuous willingness to support the team before work commitments prevented his further participation. We also express our thanks to the Artificial Intelligence Club (GAIA) at ESCOM-IPN for bringing our team together, fostering our active participation in the lab, and providing essential support in tracking key deadlines and managing our registration throughout the competition.

Declaration on Generative AI

During the preparation of this work, the authors used multiple large language models for drafting assistance, automated review, and formatting support. Specifically, Kimi K2.6 and Gemini 3.5 were employed for writing; Copilot (Auto mode) was used for review; a Notebook-based model was consulted for informational lookups; MiMo V2.5 Pro (via Copilot) assisted in revision and correction of reviewer

feedback; and several Claude variants (Claude Opus 4.6, Claude Sonnet 4.7, and Claude Sonnet 4.6) assisted in investigation, review and writing too. All generated content was subsequently reviewed, fact-checked, and edited by the authors, who take full responsibility for the final text and any remaining errors.

References

[1] Z. Xie, Y. Dai, R. Elbadry, V. Jani, X. Peng, L. Qian, G. Georgiev, D. Dimitrov, F. Zhang, J. Huang, J. Geng, Y. Chen, Y. Yuan, H. Wu, Y. Wang, I. Koychev, V. Stoyanov, M. Song, Y. Chen, S. Liu, P. Nakov, Overview of FinMMEval 2026: Multilingual and multimodal financial evaluation, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction, Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026), Springer Lecture Notes in Computer Science LNCS, Jena, Germany, 2026.

[2] Z. Xie, L. Qian, G. Georgiev, D. Dimitrov, R. Elbadry, F. Zhang, X. Peng, J. Huang, V. Jani, Y. Dai, J. Geng, Y. Chen, Y. Yuan, H. Wu, Y. Wang, I. Koychev, V. Stoyanov, M. Song, Y. Chen, S. Liu, P. Nakov, Overview of the FinMMEval 2026 task 3: Financial decision making, in: CLEF 2026 Working Notes, CEUR Workshop Proceedings, CEUR-WS.org, Jena, Germany, 2026.

[3] H. Yang, X.-Y. Liu, C. D. Wang, Fingpt: Open-source financial large language models, arXiv preprint arXiv:2306.06031 (2023).

[4] B. Zhang, H. Yang, X.-Y. Liu, Instruct-FinGPT: Financial Sentiment Analysis by Instruction Tuning of General-Purpose Large Language Models, arXiv preprint arXiv:2306.12659 (2023).

[5] Y. Xiao, E. Sun, T. Chen, F. Wu, D. Luo, W. Wang, Trading-r1: Financial trading with llm reasoning via reinforcement learning, arXiv preprint arXiv:2509.11420 (2025).

[6] Microsoft, :, A. Abouelenin, A. Ashfaq, A. Atkinson, H. Awadalla, N. Bach, J. Bao, A. Benhaim, M. Cai, V. Chaudhary, C. Chen, D. Chen, D. Chen, J. Chen, W. Chen, Y.-C. Chen, Y. ling Chen, Q. Dai, X. Dai, R. Fan, M. Gao, M. Gao, A. Garg, A. Goswami, J. Hao, A. Hendy, Y. Hu, X. Jin, M. Khademi, D. Kim, Y. J. Kim, G. Lee, J. Li, Y. Li, C. Liang, X. Lin, Z. Lin, M. Liu, Y. Liu, G. Lopez, C. Luo, P. Madan, V. Mazalov, A. Mitra, A. Mousavi, A. Nguyen, J. Pan, D. Perez-Becker, J. Platin, T. Portet, K. Qiu, B. Ren, L. Ren, S. Roy, N. Shang, Y. Shen, S. Singhal, S. Som, X. Song, T. Sych, P. Vaddamanu, S. Wang, Y. Wang, Z. Wang, H. Wu, H. Xu, W. Xu, Y. Yang, Z. Yang, D. Yu, I. Zabir, J. Zhang, L. L. Zhang, Y. Zhang, X. Zhou, Phi-4-mini technical report: Compact yet powerful multimodal language models via mixture-of-loras (2025). URL: https://arxiv.org/abs/2503.01743. arXiv:2503.01743.

[7] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: Efficient finetuning of quantized llms, arXiv preprint arXiv:2305.14314 (2023).

[8] D. Araci, FinBERT: Financial Sentiment Analysis with Pre-trained Language Models, arXiv preprint arXiv:1908.10063 (2019).

[9] A. F. Ansari, L. Stella, C. Turkmen, X. Zhang, P. Mercado, H. Shen, O. Shchur, S. S. Rangapuram, S. Arber, S. Jaleel, M. Bohlke-Schneider, P. Benidis, J. Wang, Y. Xie, T. Januschowski, J. Gasthaus, Chronos: Learning the language of time series, arXiv preprint arXiv:2403.07815 (2024).

[10] W. Xu, D. Xiang, Y. Liu, X. Wang, Y. Ma, L. Zhang, S. Hu, C. Xu, J. Zhang, Finmultitime: A four-modal bilingual dataset for financial time-series analysis, arXiv preprint arXiv:2506.05019 (2025).

[11] X. Peng, L. Qian, Y. Wang, R. Xiang, Y. He, Y. Ren, M. Jiang, V. J. Zhang, Y. Guo, J. Zhao, H. He, Y. Han, Y. Feng, Y. Jiang, Y. Cao, H. Li, Y. Yu, X. Wang, P. Gao, S. Lin, K. Wang, S. Yang, Y. Zhao, Z. Liu, P. Lu, J. Huang, S. Wang, T. Papadopoulos, P. Giannouris, E. Soufleri, N. Chen, Z. Deng, H. Fu, Y. Zhao, M. Lin, M. Qiu, K. E. Smith, A. Cohan, X.-Y. Liu, J. Huang, G. Xiong, A. Lopez-Lira, X. Chen, J. Tsujii, J.-Y. Nie, S. Ananiadou, Q. Xie, MultiFinBen: Benchmarking large language models for multilingual and multimodal financial application, 2025. URL: https://arxiv.org/abs/2506.14028. doi:10.48550/arXiv.2506.14028. arXiv:2506.14028.

[12] L. Qian, X. Peng, Y. Wang, V. J. Zhang, H. He, H. Smith, Y. Han, Y. He, H. Li, Y. Cao, Y. Yu, A. Lopez-

Lira, P. Lu, J.-Y. Nie, G. Xiong, J. Huang, S. Ananiadou, When agents trade: Live multi-market trading benchmark for LLM agents, 2025. URL: https://arxiv.org/abs/2510.11695. doi:10.48550/arXiv.2510.11695. arXiv:2510.11695.

[13] Y. Dai, Y. Lin, Z. Xie, Y. Wang, RealFin: How well do LLMs reason about finance when users leave things unsaid?, 2026. URL: https://arxiv.org/abs/2602.07096. doi:10.48550/arXiv.2602.07096. arXiv:2602.07096.

[14] R. Elbadry, S. Ahmad, A. Heakl, D. Bouch, M. Ahsan, M. AlMahri, M. E. khalil, Y. Wang, S. Lahlou, S. Ananiadou, V. Stoyanov, J. Huang, X. Peng, P. Nakov, Z. Xie, SAHM: A benchmark for arabic financial and shari'ah-compliant reasoning, 2026. URL: https://arxiv.org/abs/2604.19098. doi:10.48550/arXiv.2604.19098. arXiv:2604.19098.

[15] V. Devane, M. Nauman, B. Patel, A. M. Wakchoure, Y. Sant, S. Pawar, V. Thakur, A. Godse, S. Patra, N. Maurya, S. Racha, N. K. Singh, A. Nagpal, P. Sawarkar, K. V. Pundalik, R. Saluja, G. Ramakrishnan, BhashaBench V1: A comprehensive benchmark for the quadrant of indic domains, 2025. URL: https://arxiv.org/abs/2510.25409. arXiv:2510.25409.

[16] Y. Yu, H. Li, Z. Chen, Y. Jiang, Y. Li, D. Zhang, R. Liu, J. W. Suchow, K. Khashanah, Finmem: A performance-enhanced llm trading agent with layered memory and character design, in: Proceedings of the AAAI Symposium Series, volume 3, 2024, pp. 595–597.

[17] Y. Xiao, E. Sun, D. Luo, W. Wang, Tradingagents: Multi-agents llm financial trading framework, arXiv preprint arXiv:2412.20138 (2024).

[18] F. Tian, F. D. Salim, H. Xue, Tradinggroup: A multi-agent trading system with self-reflection and data-synthesis, arXiv preprint arXiv:2508.17565 (2025).

[19] C.-C. Lu, Y.-C. Chou, T.-R. Chen, P1gpt: A multi-agent llm workflow module for multi-modal financial information analysis, arXiv preprint arXiv:2510.23032 (2025).

[20] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: Low-rank adaptation of large language models, in: ICLR, 2022.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[22] A. Y. Ng, D. Harada, S. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, in: ICML, volume 99, 1999, pp. 278–287.

[23] G. Xiong, Z. Deng, K. Wang, Y. Cao, H. Li, Y. Yu, X. Peng, M. Lin, K. E. Smith, X.-Y. Liu, J. Huang, S. Ananiadou, Q. Xie, Flag-trader: Fusion llm-agent with gradient-based reinforcement learning for financial trading, arXiv preprint arXiv:2502.11433 (2025).

[24] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, D. Guo, Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL: https://arxiv.org/abs/2402.03300. arXiv:2402.03300.

[25] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, X. Bi, X. Zhang, X. Yu, Y. Wu, Z. F. Wu, Z. Gou, Z. Shao, Z. Li, Z. Gao, A. Liu, B. Xue, B. Wang, B. Wu, B. Feng, C. Lu, C. Zhao, C. Deng, C. Ruan, D. Dai, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Xu, H. Ding, H. Gao, H. Qu, H. Li, J. Guo, J. Li, J. Chen, J. Yuan, J. Tu, J. Qiu, J. Li, J. L. Cai, J. Ni, J. Liang, J. Chen, K. Dong, K. Hu, K. You, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Zhao, L. Wang, L. Zhang, L. Xu, L. Xia, M. Zhang, M. Zhang, M. Tang, M. Zhou, M. Li, M. Wang, M. Li, N. Tian, P. Huang, P. Zhang, Q. Wang, Q. Chen, Q. Du, R. Ge, R. Zhang, R. Pan, R. Wang, R. J. Chen, R. L. Jin, R. Chen, S. Lu, S. Zhou, S. Chen, S. Ye, S. Wang, S. Yu, S. Zhou, S. Pan, S. S. Li, S. Zhou, S. Wu, T. Yun, T. Pei, T. Sun, T. Wang, W. Zeng, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, W. L. Xiao, W. An, X. Liu, X. Wang, X. Chen, X. Nie, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yang, X. Li, X. Su, X. Lin, X. Q. Li, X. Jin, X. Shen, X. Chen, X. Sun, X. Wang, X. Song, X. Zhou, X. Wang, X. Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. Zhang, Y. Xu, Y. Li, Y. Zhao, Y. Sun, Y. Wang, Y. Yu, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Ou, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Xiong, Y. Luo, Y. You, Y. Liu, Y. Zhou, Y. X. Zhu, Y. Huang, Y. Li, Y. Zheng, Y. Zhu, Y. Ma, Y. Tang, Y. Zha, Y. Yan, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Xie, Z. Zhang, Z. Hao, Z. Ma, Z. Yan, Z. Wu, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Pan, Z. Huang, Z. Xu, Z. Zhang, Z. Zhang, Deepseek-r1

incentivizes reasoning in llms through reinforcement learning, Nature 645 (2025) 633–638. URL: http://dx.doi.org/10.1038/s41586-025-09422-z. doi:10.1038/s41586-025-09422-z.
[26] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: Advances in Neural Information Processing Systems, volume 35, 2022, pp. 24824–24837.
[27] J. Nash, Non-cooperative games, Annals of mathematics (1951) 286–295.
[28] MBZUAI NLP, CLEF_Task3_Trading: Official Dataset for FinMMEval Task 3, https://huggingface.co/datasets/TheFinAI/CLEF_Task3_Trading, 2026.
[29] TheFinAI, daily_news: Daily Financial News Dataset for FinMMEval, https://huggingface.co/datasets/TheFinAI/daily_news, 2025.
[30] E. Schau, bitcoin_news: Historical Bitcoin News Dataset, https://huggingface.co/datasets/edaschau/bitcoin_news, 2025.
[31] T. Majlesi, bitcoin-individual-news-dataset, https://huggingface.co/datasets/tahamajs/bitcoin-individual-news-dataset, 2025.
[32] Y. Kim, Microsoft & Tesla Finance News Articles (2020-2024), https://www.kaggle.com/datasets/journeyyouyeonkim/microsoft-tesla-finance-news-articles2020-2024, 2024.
[33] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. K"uttler, M. Lewis, W.-t. Yih, T. Rockt"aschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, Advances in Neural Information Processing Systems 33 (2020) 9459–9474.
[34] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, DeepAR: Probabilistic forecasting with autoregressive recurrent networks, International Journal of Forecasting 36 (2020) 1181–1191.

A. Supplementary Tables and Reproducibility Details

This appendix contains detailed configuration tables and reproducibility specifications referenced in the main text.

A.1. FinBERT Sentiment Extraction

The sentiment enrichment module is built on *FinBERT* (`ProsusAI/finbert`) [8], a domain-specific language model adapted from the standard BERT-base architecture. FinBERT retains the core architecture of BERT-base, consisting of 12 transformer layers, 12 attention heads, and 110 million parameters. However, unlike general-purpose models, FinBERT was specialized by further pre-training on a large-scale corporate financial corpus (encompassing corporate reports, earnings call transcripts, and analyst notes) and subsequently fine-tuned on the Financial PhraseBank dataset, which contains carefully annotated financial sentiment sentences. For every row with available news (`has_news` $= 1$), the module passes the news text through the model to extract a 3-class probability distribution through a softmax layer:

$$P(C_i|x) = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}} \tag{6}$$

where $C \in \{\text{positive}, \text{negative}, \text{neutral}\}$. The output is cached per asset to prevent redundant computation across training runs. The final structured output contains the `sentiment_label`, the absolute `sentiment_score`, and the per-class probability distribution.

A.2. Optuna Hyperparameter Search Space

This subsection records the hyperparameter ranges and fixed values used to define the Optuna search; the full list is in Table 13, together with the rationale for each choice.

Table 13
Optuna Hyperparameter Search Space (full details)

Parameter	Configuration / Search Range	Reason
Training batch size	Fixed to 1	Preserve stability under low VRAM
Gradient accumulation	Fixed to 32 steps	Maintain effective batch size without exceeding memory
GRPO group size	Fixed to 4	Minimum viable setting for relative-advantage estimation
Prompt template	Fixed to `grpo_cot_v1`	Keep prompt format consistent across trials
LoRA rank	$\{4, 8\}$	Trade off capacity vs. memory footprint
Base model	Five candidates (primarily `Phi-4-mini-instruct` and `Qwen2.5-3B-Instruct`; Qwen3-4B and Llama3.1-8B excluded due to VRAM limits)	Compare compact model families under the same pipeline
Max new tokens	$\{96, 128\}$	Reduce generation cost and truncation risk
SFT epochs	$\{0, 1\}$	Test whether a warm-up pass improves structure
Learning rate	$[1 \times 10^{-5}, 1 \times 10^{-3}]$ (log scale)	Stabilize optimization while allowing exploration
KL coefficient β	$[0.01, 0.1]$ (log scale)	Control policy drift during GRPO
Clip ϵ	$[0.1, 0.3]$	Limit overly aggressive updates
Temperature	$[0.5, 0.9]$	Explore different sampling entropy levels
Reward scaling	$[0.5, 3.0]$	Adjust the magnitude of matrix-based rewards
Buy percentile	$[0.30, 0.50]$	Tune the positive-action threshold
Sell percentile	$[0.10, 0.25]$	Tune the negative-action threshold

A.3. Format Reward Cases

This subsection lists the reward outcomes assigned to each output-format case; the complete mapping is in Table 14, so the main text can refer to the compact reward summary without losing the exact values.

Table 14
Format Reward Cases for Structured Action Output (full details)

Case	Output condition	Reward
(i)	Complete response with properly closed action tags	$+0.5$
(ii)	Detectable action present, but response truncated before closing tags	$+0.25$
(iii)	Severely truncated or malformed response	-0.5
(iv)	No recognizable action token (invalid-format branch selected)	-1.5
(v)	Effective invalid-format reward after applying the global scaling factor	~ -2.37

A.4. Official Task 3 Reporting Snapshot

This appendix records the frozen camera-ready reporting snapshot used for the official Task 3 submission. The ranking metric is Cumulative Return (CR) under the Long/Short setting, averaged across BTC and TSLA. BTC is reported over 2026-05-11 to 2026-07-04, while TSLA is reported over 2026-05-11 to 2026-06-26 because later TSLA dates were affected by an organizer-side payload issue and were treated uniformly as HOLD/flat for all agents.

Table 15
Official reporting snapshot for Task 3 camera-ready evaluation, with the raw metrics used for camera-ready reporting.

Asset	Days	Closed	Final Balance	Return	No Fees	Vs Buy & Hold	Sharpe	Win Rate	Rank
BTC	55	0	$95,597.37	-4.25%	-3.62%	+18.68%	-1.62	50.00%	12/18
TSLA	33	14	$92,933.54	-7.07%	-6.73%	+7.76%	-7.39	33.00%	18/18

A.5. Seed Configurations and Reproducibility

Table 16
Seed Configurations Across Optuna Trials

Trial	Optuna Seed	Global Seed	Outcome / Explanation
0	2	42	Different hyperparameters, identical data split, identical model initialization
1	3	42	Different hyperparameters, identical data split, identical model initialization
2	5	42	Different hyperparameters, identical data split, identical model initialization

To establish a reproducible baseline, a global random seed is initialized at the start of execution, configuring the generators for Python, NumPy, PyTorch, and Hugging Face Transformers. This configuration is consistently maintained in downstream components, such as the parameter-efficient fine-tuning (PEFT) framework and evaluation validators. Despite these precautions, external dependencies can introduce unaligned randomness; specifically, Unsloth's compiled execution cache defaults its internal training seeds to a hardcoded value (3407). To mitigate this replication risk, future iterations of our pipeline will pass the global seed explicitly to all external trainer interfaces and dynamically override the Unsloth seed defaults in runtime memory before fine-tuning, thereby aligning all active components under a single source of truth.

Replicating the experimental pipeline involves three sequential execution steps. First, the dataset builder is run to ingest the granular CSV files, execute the FinBERT cache model, and compile the unified Parquet dataset. Subsequently, the Optuna searcher is executed over three trials with fast mode enabled to conduct rapid SFT-plus-GRPO trials. Finally, validation is performed against the winning LoRA checkpoint to execute a full temporal-fold evaluation on the validation split and export the complete financial metrics.

B. Chronos Module: Technical Details

This appendix provides the full technical description of the Chronos temporal encoding module summarized in Section 3.4.

B.1. Architecture and Execution Paths

Chronos represents a paradigm shift in time-series forecasting by framing numerical forecasting as a classification problem over discrete tokens. Built on the T5 encoder-decoder architecture, Chronos is pre-trained on the Time Series Mix, a massive dataset combining diverse real-world datasets spanning multiple domains. The core innovation of Chronos lies in its tokenization and value binning strategy: continuous real numbers are quantized into 256 discrete bins, mapping numerical prices to discrete vocabulary tokens. This tokenized formulation allows the model to learn generalizable zero-shot cross-domain temporal patterns. Unlike traditional statistical approaches (such as ARIMA or GARCH) which assume rigid linear relationships and must be independently fit for each asset, Chronos zero-shot forecasting generates complete probabilistic distributions that capture non-linear dynamics and structural market uncertainty directly from observed context windows.

The module is loaded via a Base Chronos Pipeline abstraction, which handles device placement automatically. When a CUDA-capable GPU is available, the model is instantiated in `bfloat16` precision to reduce VRAM consumption while preserving numerical stability. In CPU-only environments, standard floating-point inference is applied as a fallback.

The system utilizes two distinct execution paths depending on the phase. In the offline feature generation pipeline, price sequences are processed individually, forecasting a single future step with 20 probabilistic samples to calculate upward and downward probabilities. In the live endpoint preprocessor, Chronos processes the available price history dynamically, using the last 50 prices as context window. The forecast is generated with a short generation length and a single return sequence. Under both paths, temporal isolation is strictly maintained by evaluating only the immediate next-day forecasted change relative to the last observed price.

B.2. Forecast Extraction and Direction Classification

Chronos generates a probabilistic forecast tensor containing S empirical price samples for the target trading horizon. Rather than relying on a single point estimate, we analyze the distribution of these samples to classify market direction and estimate prediction confidence, drawing on established principles of deep probabilistic time-series forecasting [34]. For each sequence, the model draws $S = 20$ samples representing possible next-day prices, denoted as $\{\hat{p}_{t+1}^{(s)}\}_{s=1}^{S}$. We then calculate the empirical probabilities of positive and negative price movements relative to the last observed price p_t:

$$P_{\text{up}} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{I}\left(\hat{p}_{t+1}^{(s)} > p_t\right) \tag{7}$$

$$P_{\text{down}} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{I}\left(\hat{p}_{t+1}^{(s)} < p_t\right) \tag{8}$$

where $\mathbb{I}$ represents the indicator function. The final directional target is classified into three mutually exclusive categories using asymmetric probability thresholds:

$$\text{class} = \begin{cases} \textsc{up} & \text{if } P_{\text{up}} > 0.45 \\ \textsc{down} & \text{if } P_{\text{down}} > 0.40 \\ \textsc{neutral} & \text{otherwise} \end{cases} \tag{9}$$

The corresponding confidence score is assigned as the probability of the selected direction (specifically P_{up} for UP, P_{down} for DOWN, and $1 - P_{\text{up}} - P_{\text{down}}$ for NEUTRAL). These computed metrics are stored as the `chronos_direction` and `chronos_confidence` features, which are cached per asset and injected into the prompt builder.