The Losing Winner: An LLM Agent that Predicts the Market but Loses Money

Youwon Jang [1]* **Joochan Kim** [2]* **Byoung-Tak Zhang** [1]†
[1] Seoul National University
[2] Korea Institute of Science and Technology
{sharifa, btzhang}@snu.ac.kr, joochan.k@kist.re.kr

Abstract

Recent advancements in Large Language Models (LLMs) have spurred significant interest in their application to autonomous financial trading. This paper investigates the efficacy of fine-tuning a small-scale LLM, Qwen2.5-3B-Instruct, for Bitcoin (BTC) trading by framing the task as a market state prediction problem. Using daily price data, volume, and technical indicators, we train the model with Reinforcement Learning with Verifiable Reward (RLVR) to classify the next day's market into one of three states: bullish, consolidation, or bearish. Our experiments reveal a critical paradox: the fine-tuned agent achieves significantly higher classification accuracy compared to a zero-shot baseline, demonstrating a clear ability to learn the defined task. However, when deployed in a simulated trading environment, this improved predictive power results in a lower cumulative return than the baseline. We attribute this divergence to a classic case of objective mismatch and reward hacking. The agent optimizes for the simple, discrete reward of a correct classification, a proxy objective that fails to capture the complexities of profitable trading, such as risk management and the magnitude of price movements. This work serves as a cautionary case study, highlighting the inherent challenges of designing effective reward functions for LLM-based financial agents and underscoring the importance of aligning proxy tasks with the ultimate goal of profit maximization.

1 Introduction

The advent of generative AI, particularly Large Language Models (LLMs), has opened new frontiers for automating complex decision-making processes [Chang et al., 2024]. The financial domain, with its vast amount of noisy and sequential data, stands as a prime candidate for LLM-driven innovation, especially in the realm of algorithmic trading [Wang et al., 2024, Li et al., 2024, Yuksel, 2025]. The vision is to create autonomous agents that can comprehend market dynamics, process diverse data streams, and execute profitable trades. A popular approach to developing such agents is to simplify the complex task of trading into a more tractable supervised or reinforcement learning problem [Kabbani and Duman, 2022, Srivastava et al., 2025], such as predicting the direction of price movement or classifying the overall market sentiment.

In this study, we explore this paradigm by fine-tuning a contemporary small-scale LLM, Qwen2.5-3B-Instruct [Yang et al., 2024], to act as a trading agent in the volatile Bitcoin (BTC) market. We formulate the trading decision process as a classification task: predicting whether the market on the following day will be bullish, in consolidation, or bearish. The agent is trained using a Reinforcement Learning (RL) framework on historical daily price data, trading volume, and a set of technical indicators. The

*Equal contribution
†Corresponding author

reward function is predominantly designed to incentivize correct classification, allocating 90% of the reward to prediction accuracy.

Our initial hypothesis is that a higher accuracy in predicting the market state directly translates into a more effective trading strategy and, consequently, higher cumulative returns. The results, however, present a stark contradiction. While our RL-trained agent demonstrates a marked improvement in classification accuracy over its zero-shot counterpart, its back-tested trading performance is unexpectedly inferior, leading to a net decrease in cumulative returns.

This counter-intuitive outcome exposes a fundamental challenge in applying AI to real-world problems: the gap between a proxy objective and the true goal. We argue that our agent, while successful at the market state classification, falls victim to 'reward hacking' [Eisenstein et al., 2024, Skalse et al., 2025]. It *learns* to maximize its reward by becoming an excellent classifier, but this proxy task *is* an inadequate representation of the ultimate objective: maximizing profit. The simple, binary reward for a correct prediction does not account for the nuanced realities of trading, including the magnitude of market moves, the timing of entry and exit, transaction fees, and risk management. A correct bullish prediction followed by a marginal price increase, for instance, can easily result in a net loss after accounting for trading costs.

The primary contribution of this paper is to provide a clear, empirical case study of objective mismatch in the context of LLM-based financial agents. We demonstrate that naively framing trading as a classification problem can be misleading and counterproductive. Our findings serve as a cautionary tale, emphasizing that for generative AI to succeed in finance, a more sophisticated approach to reward engineering and agent alignment is imperative. This work highlights the critical need to design objectives that holistically encapsulate the multifaceted nature of profitability, rather than relying on simplified, intermediate proxy tasks.

2 Methods

Our LLM agent undergoes a two-stage fine-tuning process to effectively perform the complex task of market prediction. The first stage, Supervised Fine-tuning (SFT), equips the model with the fundamental ability to understand the given market data and generate a prediction. The second stage, Reinforcement Learning Fine-Tuning (RFT), uses the SFT-trained model as an initial policy and enables the agent to autonomously improve and optimize its prediction strategy by maximizing a specific reward function. Detailed training setup can be found in the Appendix A.

2.1 Supervised Fine-tuning (SFT)

SFT serves as an initial phase of task-specific adaptation, where the LLM agent learns the nuances of financial market prediction. The primary goal of this stage is to teach the model how to comprehend the context of structured market data prompts and to generate responses in the predefined format of 'Bullish', 'Consolidation', or 'Bearish'.

Mechanism The SFT process follows the standard paradigm of language model training. Each sample in the training dataset consists of a pair (s_t, a_t^*), where s_t is the prompt containing market data at time t, and a_t^* is the ground-truth label representing the actual market state on the next day. The model is trained to maximize the probability of generating a_t^* given s_t. This is achieved by minimizing the standard cross-entropy loss.

Objective Through SFT, we obtain an initial policy, π_{SFT}, which can make reasonable predictions based on the given data. This policy serves as a strong baseline and provides a robust starting point for the subsequent reinforcement learning phase.

$$\pi_{\text{SFT}} = \arg\max_{\pi} \; \mathbb{E}_{(s_t, a_t^*)\sim\mathcal{D}}\big[\log \pi(a_t^* \mid s_t)\big]$$

2.2 Reinforcement Fine-Tuning (RFT)

While SFT trains the model to imitate the 'correct answers' from a static dataset, RFT allows the agent to learn actively from the outcomes (rewards) of its own predictions (actions) and to refine its policy. The goal of this stage is to further improve the agent's policy, leveraging outcome-based rewards.

RL Framework We optimize the agent's policy by adopting the Reinforcement Learning with Verifiable Reward (RLVR) [Lambert et al., 2025] framework, utilizing Guided Reward Policy Optimization (GRPO) [DeepSeek-AI, 2025] as the specific optimization algorithm. To perform this fine-tuning step efficiently, we employ Low-Rank Adaptation (LoRA) [Hu et al., 2022]. In this process, the model receives a **state** (s_t), which is a natural language prompt containing historical market data. Given this state, the policy model, initialized from the SFT-trained policy π_{SFT}, samples an **action** (a_t) in the form of a market prediction. This action is then evaluated by a verifier against ground-truth data to calculate a verifiable **reward** (R_t). This objective and trustworthy reward signal allows GRPO to efficiently and stably refine the LLM's policy.

The optimization objective is to find a policy π_{RFT} that maximizes the expected reward while regularizing the policy to prevent it from deviating too far from the initial π_{SFT}, as shown below:

$$\pi_{\text{RFT}} = \arg\max_{\pi} \ \mathbb{E}_{s_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi(\cdot|s_t)} [R_t(a_t)] - \beta \, \mathbb{D}_{\text{KL}} \big(\pi(\cdot|s_t) \, || \, \pi_{\text{SFT}}(\cdot|s_t) \big) \right]$$

where $\mathbb{D}_{\text{KL}}$ denotes the Kullback-Leibler divergence, and β is the coefficient controlling the strength of the KL penalty. The R_t is a weighted sum of two components: prediction accuracy (R_{acc}) and format accuracy (R_{format}). Both components provide a binary reward of $+1$ for a correct outcome and 0 for an incorrect one. In our experiment, we set the format weight w_{format} to 0.1.

$$R_t = w_{\text{format}} \cdot R_{\text{format}} + (1 - w_{\text{format}}) \cdot R_{\text{acc}}$$

3 Experiments

3.1 Implementation Details

Our experimental framework is built upon a custom data preprocessing pipeline that converts raw time-series market data into a format suitable for instruction-tuned LLMs. Additional details can be found in Appendix A.

Data Source and Feature Engineering We construct our dataset using daily BTCUSDT closing prices and volume data sourced from Binance [Binance, 2025]. To enrich the input features, we incorporate the daily Fear & Greed Index as a market sentiment indicator. For each trading day, we engineer five key technical indicators using the TA-Lib library [Benediktsson et al., 2024]: SMA (20), EMA (20), RSI (14), MACD, and Bollinger Bands (20-day, 2 standard deviations).

Target Variable and Data Structuring The prediction task is framed as a three-class classification problem. We label the next day's market state based on the percentage change from the previous day's closing price: **Bullish** for an increase greater than 1%, **Bearish** for a decrease greater than 1%, and **Consolidation** for price changes within a ±1% range.

Dataset Splitting To partition our data, we follow the setting of prior work [Wang et al., 2024]. To ensure a robust evaluation and prevent data leakage, we employ a strict temporal splitting strategy. The data is partitioned into a general training set, a validation set, and two distinct test sets designed to evaluate performance under specific market conditions: a bullish period and a bearish period.

Trading Strategy The market state predictions generated by the LLM agent are translated into discrete trading actions according to a simple, rule-based policy. This all-or-nothing strategy is designed to directly test the utility of the agent's classification accuracy. i) **Bullish Prediction:** The agent allocates 100% of its capital to purchase the asset (a full long position). If already invested, the position is held. ii) **Consolidation Prediction:** The agent maintains its current position without making any new trades (i.e., holds the asset or cash). iii) **Bearish Prediction:** The agent liquidates 100% of its asset holdings into cash. If already in cash, no action is taken.

3.2 Experiment Results

In Table 1, we evaluate our fine-tuned trading agents against a Buy & Hold baseline across three distinct market periods, focusing on prediction accuracy, cumulative return, and the Sharpe ratio.

Table 1: Performance comparison of different trading strategies across various market periods, evaluated by predictive accuracy (Acc), cumulative return (Cum. Return), and Sharpe ratio.

Strategy	**Validation Set**			**Bullish Set**			**Bearish Set**		
	Acc	Cum. Return	Sharpe	Acc	Cum. Return	Sharpe	Acc	Cum. Return	Sharpe
Buy & Hold	-	+15.4%	2.03	-	+78.3%	8.91	-	-17.5%	-5.02
Qwen2.5-3B-Instruct	25.4%	+10.1%	2.30	8.3%	+6.2%	3.49	38.5%	-0.4%	-2.65
+ SFT	44.1%	+8.3%	2.03	27.1%	+50.9%	8.05	23.1%	-10.2%	-2.76
+ RFT	64.4%	+0.0%	0.0	66.7%	+0.0%	0.0	75.0%	+0.0%	0.0
+ SFT & RFT	45.8%	+15.4%	2.03	41.7%	+78.3%	8.91	44.2%	-14.8%	-3.09

The results empirically validate our central thesis: improving an agent's predictive accuracy on a proxy task does not guarantee enhanced profitability and can, paradoxically, lead to worse financial outcomes. This illustrates a critical case of *objective mismatch*, where the agent becomes a 'winner' at classification but a 'loser' in the true task of trading.

The SFT-only agent demonstrates a clear improvement in task understanding over naive baselines, with its accuracy rising substantially. While this enhancement translates into strong returns and a solid Sharpe ratio in the bull market, it suffers severe losses in the Bearish Set. This provides an early indication that higher accuracy does not necessarily equate to better risk management.

The most striking evidence of this paradox comes from the RFT-only agent. It achieves the highest classification accuracy across all test sets—reaching an impressive 75.0% in the Bearish Set—yet its cumulative return and Sharpe ratio remain consistently at zero. This outcome is a clear example of *reward hacking*: to maximize its accuracy-based reward, the agent adopts an ultra-conservative strategy of taking no risks and earning no returns. It effectively wins the prediction game while completely failing at the financial one.

The combined SFT & RFT agent further highlights the danger of this misaligned objective. Although SFT initialization enables it to take active positions, the subsequent RFT phase drives it to over-optimize for classification accuracy. Consequently, while it appears balanced by matching Buy & Hold performance in the Bullish Set, this parity merely suggests a replicated strategy rather than a sophisticated one. The fundamental flaw becomes evident in the Bearish Set: despite achieving higher accuracy than the SFT-only agent, it incurs even greater losses and a worse Sharpe ratio. The agent mastered classification but ignored the magnitude of risk, resulting in confident yet financially catastrophic decisions.

These findings reaffirm that in complex domains like finance, success hinges not on excelling at simplified proxy tasks but on aligning with the true, multifaceted objective of risk-adjusted profit generation.

4 Conclusion

This paper presents a core paradox in applying LLMs to finance: the fine-tuned agent's predictive accuracy increased while its trading returns decreased. The agent, optimized via Reinforcement Learning, successfully learned its given proxy objective of market state classification, but this led to 'reward hacking' by ignoring crucial factors like the magnitude of returns and risk management. 'Losing Winner' thus serves as a critical cautionary tale against naively framing complex financial tasks as simple classification problems. This study underscores that the future success of generative AI in finance depends not on perfecting proxy metrics, but on designing sophisticated reward functions that align agents with the true, holistic goal of risk-adjusted profit maximization.

5 Limitations and Future Works

This study's limitations highlight two critical directions for future work. First, there is a need for sophisticated reward engineering that moves beyond simple accuracy metrics to directly optimize for risk-adjusted returns, thus aligning the agent's objective with true profitability. Second, as our findings are based on a small-scale LLM, it is essential to investigate whether this "Losing Winner" paradox persists in larger, state-of-the-art models with more advanced reasoning capabilities. Addressing these challenges of objective alignment and model generalizability will be paramount for developing practical AI trading agents.

Acknowledgements

This work was partly supported by the IITP (RS-2021-II212068-AIHub/10%, RS-2021-II211343-GSAI/10%, RS-2022-II220951-LBA/15%, RS-2022-II220953-PICA/15%), NRF (RS-2024-00353991-SPARC/15%, RS-2023-00274280-HEI/15%), KEIT (RS-2024-00423940/10%), and Gwangju Metropolitan City (Artificial intelligence industrial convergence cluster development project/10%) grant funded by the Korean government.

References

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45, 2024.

Qian Wang, Yuchen Gao, Zhenheng Tang, Bingqiao Luo, Nuo Chen, and Bingsheng He. Exploring llm cryptocurrency trading through fact-subjectivity aware reasoning. *arXiv preprint arXiv:2410.12464*, 2024.

Yuan Li, Bingqiao Luo, Qian Wang, Nuo Chen, Xu Liu, and Bingsheng He. Cryptotrade: A reflective llm-based agent to guide zero-shot cryptocurrency trading. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1094–1106, 2024.

Kamer Ali Yuksel. Alphaportfolio: Discovery of portfolio optimization and allocation methods using llms. *Available at SSRN*, 2025.

Taylan Kabbani and Ekrem Duman. Deep reinforcement learning approach for trading automation in the stock market. *IEEE Access*, 10:93564–93574, 2022.

Uditansh Srivastava, Shivam Aryan, and Shaurya Singh. A risk-aware reinforcement learning reward for financial trading, 2025. URL `https://arxiv.org/abs/2506.04358`.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Jacob Eisenstein, Chirag Nagpal, Alekh Agarwal, Ahmad Beirami, Alex D'Amour, DJ Dvijotham, Adam Fisch, Katherine Heller, Stephen Pfohl, Deepak Ramachandran, Peter Shaw, and Jonathan Berant. Helping or herding? reward model ensembles mitigate but do not eliminate reward hacking, 2024. URL `https://arxiv.org/abs/2312.09244`.

Joar Skalse, Nikolaus H. R. Howe, Dmitrii Krasheninnikov, and David Krueger. Defining and characterizing reward hacking, 2025. URL `https://arxiv.org/abs/2209.13085`.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training, 2025. URL `https://arxiv.org/abs/2411.15124`.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL `https://arxiv.org/abs/2501.12948`.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Binance. Binance public data repository. `https://github.com/binance/binance-public-data`, 2025. Accessed: 2025-08-30.

John Benediktsson et al. TA-Lib Python: A python wrapper for the technical analysis library. `https://github.com/ta-lib/ta-lib-python`, 2024. Accessed: 2025-08-30.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024.

Zewei Zhou, Tianhui Cai, Yun Zhao, Seth Z.and Zhang, Zhiyu Huang, Bolei Zhou, and Jiaqi Ma. Autovla: A vision-language-action model for end-to-end autonomous driving with adaptive reasoning and reinforcement fine-tuning. *arXiv preprint arXiv:2506.13757*, 2025.

Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*, 2018.

Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. *arXiv preprint arXiv:1911.10107*, 2019.

Antonio Briola, Jeremy Turiel, Riccardo Marcaccioli, Alvaro Cauderan, and Tomaso Aste. Deep reinforcement learning for active high frequency trading. *arXiv preprint arXiv:2101.07107*, 2021.

A Detailed Experimental Setup

A.1 Dataset Specification and Date Ranges

The dataset was partitioned according to the following chronological periods to prevent any future information from leaking into the training or validation sets [Wang et al., 2024].

- **Training Set:** General market data from February 1, 2018 to July 31, 2025. This set excluded the periods designated for validation and testing.
- **Validation Set:** November 16, 2023 – January 15, 2024.
- **Test Bullish Set:** January 24, 2024 – March 13, 2024. This period was selected for its distinct upward market trend.
- **Test Bearish Set:** May 21, 2024 – July 13, 2024. This period was selected for its distinct downward market trend.

Table 2 provides a detailed summary of our dataset partitioning, including the distribution of samples across the three market state classes. The training set, comprising 2,482 samples, is used to fine-tune the LLM agent. The ±1% threshold for the Consolidation class was empirically chosen as it yields a well-balanced class distribution across our historical dataset: **Bullish** (32.5%), **Consolidation** (38.3%), and **Bearish** (29.2%). This balance indicates that the ±1% range effectively captures periods of low volatility or market indecision, distinguishing them from more significant directional movements.

Table 2: Summary of dataset splits and class distribution. Each row represents a different data partition, showing the total number of samples and the breakdown across the three target classes: Bullish, Consolidation, and Bearish.

Split	Total Samples	Bullish	Consolidation	Bearish
Train	2,482	806	945	731
Validation	60	21	24	15
Test Bullish	49	21	24	4
Test Bearish	53	12	20	21
Total	2,644	860	1,013	771

To provide the model with a quantitative basis for its predictions, we engineered a set of five widely-used technical indicators from the historical price and volume data. The feature set includes a 20-day Simple Moving Average (SMA), a 20-day Exponential Moving Average (EMA), a 14-day Relative Strength Index (RSI), the Moving Average Convergence Divergence (MACD), and Bollinger Bands (20-day, 2 standard deviations).

A.2 Computational Resources

All fine-tuning experiments were conducted on a single server equipped with six NVIDIA A6000 GPUs, each featuring 48 GB of VRAM. The complete fine-tuning process for the 3B-parameter model on our training dataset required approximately 12 hours of execution time. During training, the memory utilization on each GPU was consistently high, approaching the full 48 GB capacity. We note that this hardware configuration represents a practical boundary for the model scale in our experiments. Attempts to fine-tune models with 4 billion or more parameters consistently resulted in out-of-memory (OOM) errors, preventing the training process from completing.

A.3 Prompts

The following are examples of system prompts used to assign an expert persona to the model:

```
You are a Bitcoin market analysis expert with expertise in technical
analysis. You are also recognized as the world's top market timing analyst.
Based on the given market data and technical indicators, you must provide
accurate and logical price direction predictions.
```

The user prompt is meticulously structured to transform the time-series prediction task into a comprehensive, self-contained natural language instruction for the LLM. Each prompt is designed not only to provide data but also to define the task and constraints explicitly within the context. The structure includes several key components: 1) task directive, 2) structured data presentation (10-day lookback of market information), 3) in-prompt class definition, 4) output format.

```
Please analyze the following Bitcoin market data and technical indicators to
predict the future price direction:

Market Data and Technical Indicators for the Past 10 Days:
10 days ago: Closing Price $10,437.60, Volume 68,113.82, Fear & Greed Index:
54 (Neutral), SMA(20): $9139.11, EMA(20): $9979.45, RSI(14): 52.12, MACD:
13.6733, BB Upper: $11645.83, BB Lower: $6632.39
(...)
1 days ago: Closing Price $11,039.00, Volume 23,910.71, Fear & Greed Index:
47 (Neutral), SMA(20): $10086.84, EMA(20): $10214.65, RSI(14): 56.52, MACD:
151.0302, BB Upper: $11773.72, BB Lower: $8399.96

Based on the historical market data, technical indicators, and fear & greed
index above, please predict whether the Bitcoin price will be bullish,
bearish, or consolidate today.
**Price Movement Classification Criteria:**
- **Bullish**: Price increase > 1% from previous day's closing price
- **Bearish**: Price decrease > 1% from previous day's closing price
- **Consolidation**: Price change within ±1% from previous day's closing price
Please analyze the market data and technical indicators when you make
a decision.
Format your answer as:

### final answer: [bullish/bearish/consolidation]
```

To elaborate on the data presentation, the 10-day lookback mentioned above consists of the most recent market data formatted directly into the main text of the prompt. In addition to this, a longer-term context is provided as auxiliary input, which contains a preceding 30-day window of historical data. This supplementary data stream includes the daily closing price, volume, and the Fear & Greed Index (the technical indicators are not included).

A.4 Technical Stack and Dependencies

The data processing and analysis are conducted using the following primary Python libraries:

- **Technical Analysis:** `TA-Lib` [Benediktsson et al., 2024][3]
- **Data Sources:** `Binance-public-data` [Binance, 2025][4], `Fear and Greed Index`[5]
- **Train Framework:** `Volcano Engine Reinforcement Learning for LLMs` [Sheng et al., 2024]

A.5 Training Setup

We mainly followed the training setup from Zhou et al. [2025]. In detail, we used $lr = 10^{-5}$, $\beta = 10^{-3}$, with 5 epochs for SFT. For RFT, we used $lr = 3 \times 10^{-5}$, $\beta = 4 \times 10^{-2}$, with 5 epochs. We also set the temperature to 1.0, top-p to 1.0, and top-k to 0.0 for generation parameters, while we set the LoRA rank and alpha to 8. During the evaluation stage, we set the temperature to 0.0.

B Related Works

Large Language Models as Financial Trading Agents The recent success of LLMs has introduced a new paradigm for creating autonomous financial agents. Unlike specialized models, LLMs can process and reason over a mixture of quantitative data and qualitative text, making them well-suited for financial tasks. Several studies have demonstrated the potential of LLMs in finance, framing trading as a prediction or classification problem based on market news, sentiment analysis, and numerical indicators [Wang et al., 2024, Li et al., 2024]. For instance, Yuksel [2025] used an LLM to generate and refine financial metrics for portfolio optimization. A common methodology, which we also adopt, is to utilize or fine-tune a pre-trained LLM on historical market data to predict future market states. These works primarily focus on the capabilities of LLMs to achieve profitable outcomes.

Reinforcement Learning for Financial Markets To enable agents to learn from the consequences of their actions, Reinforcement Learning (RL) has become a popular training paradigm in finance [Huang, 2018, Zhang et al., 2019, Briola et al., 2021, Kabbani and Duman, 2022]. RL allows an agent to develop a policy by interacting with an environment and receiving rewards, moving beyond the static imitation of a dataset. However, the effectiveness of RL is critically dependent on the design of the reward function. A poorly specified reward can lead to unintended agent behavior, a phenomenon known as 'reward hacking' [Skalse et al., 2025]. Our work directly contributes to this critical area of research. While many studies focus on developing RL agents that succeed, our paper presents a cautionary case study of failure due to objective mismatch.

[3]https://github.com/ta-lib/ta-lib-python

[4]https://data.binance.vision/?prefix=data/spot/monthly/klines/BTCUSDT/1d/

[5]https://alternative.me/crypto/fear-and-greed-index/