Large Language Model as an Excellent Reinforcement Learning Researcher in Financial Single-Agent and Multi-Agent Scenarios

Tianhao Fu,[*], Xinxin Xu,[*] Weichen Xu, Ruilong Ren, Bowen Deng, Xinyu Zhao, Jian Cao, Xixin Cao

Peking University

Abstract

In the quantitative finance area, particularly in order execution, reinforcement learning (RL) has shown great promise due to its ability to interact with market environments based on real data. However, traditional RL methods suffer from slow research speed and rely on static market assumptions, which do not consider the impact of the agent's execution action on the environment. To address these, we propose a Self-Evolutional single-agent/multi-agent Reinforcement Learning (SE-RL) framework. The framework utilizes a Large Language Model to design various RL algorithm modules, such as agent model design, reward function, profiling, communication, and state imagination, by leveraging the LLM generating module output or code. SE-RL could continuously improve the accuracy of LLM-generated RL algorithms through a dual-enhancement kit at both high-level (prompt refinement) and low-level (parameter fine-tuning). Additionally, we use a multi-agent system to simulate dynamic financial markets, accounting for the impact of order executions on market dynamics. To further enhance training in such a dynamic market, we develop a hybrid environment training method that could rebalance each environment's loss weight. Comprehensive experiments on 200 realistic stock datasets demonstrate that our proposed framework outperforms current state-of-the-art baselines. Code is provided in the supplementary material.

Introduction

Optimizing trade execution is one of the most important problems in quantitative finance. Recently, reinforcement learning has been applied to develop execution strategies (Mnih et al. 2015; Sutton 2018). However, two fundamental challenges hinder further progress: (1) Algorithms are evolving at a slow pace. In contrast to rapid speed in computer vision (He et al. 2016), natural language processing (Vaswani 2017), and LLM (Wang et al. 2024b; Brown et al. 2020), the innovation of RL algorithms has exhibited a markedly slower progression. There is still much room to improve each component in RL (Silver et al. 2017). As quantitatively demonstrated in Figure 1a through an analysis of the numbers of the GitHub repository (stars/projects), the official number of articles (SCI articles), and the monthly submissions to arXiv over the past three years, RL emerges as the AI subfield with the lowest growth trajectory. (2) Environmental assumptions are too idealistic. Existing RL-based execution methods rely on static market assumptions that ignore the impact of the order market (Bouchaud, Farmer, and Lillo 2009). This contradicts the dynamic nature of real-world markets. As evidenced by Figure 1b, even medium-sized orders induce measurable market impacts that fundamentally alter trading environments (Kyle 1985).

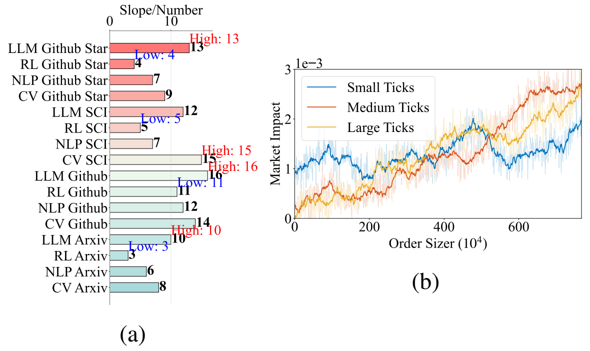


Figure 1: Two important issues of RL-based order execution methods. (a) The low research speed of reinforcement learning algorithms compared to other deep learning areas. We use different observation metrics, such as open source repositories' rate of increase (slope of the number of repositories per month) and academic papers' rate of increase, to analyze different research areas. (b) Analysis of the impact of orders on the market. The stocks of all market capitalization levels have an increasingly significant impact on the market as order volume increases, which cannot be ignored.

Although recent efforts employ LLMs (Ibecheozor, Iwuchukwu, and Akinyele 2025; Cao et al. 2024b) to address auto-research, existing approaches remain fragmented. Some studies employ LLMs for component-level auto-research (e.g., reward function generation, state feature extraction), but neglect holistic improvement of the RL pipeline (Cao et al. 2024a; Wang et al. 2024a). Others investigate complete algorithmic processes, but lack RL-specific adaptations(Silver et al. 2016; Lillicrap 2015).

To accelerate the development of RL algorithms and bridge the gap in the study of RL for trade execution within

[*]These authors contributed equally.

dynamic market environments, we propose three key innovations. First, we introduce the Self-Evolutional Reinforcement Learning (SE-RL) framework, which could continuously generate and improve a single-agent or multi-agent reinforcement learning algorithm. To ensure that LLM research capability could be continually improved, we designed the Dual Level Enhancement Kit (DEK), which performs two-tiered optimization for LLM. At the high level, through the macro-micro mechanism, we generate the intitial prompt and use in-context learning to improve the prompt continually. At the low level, we directly fine-tune the LLM weight, where we use Straight Through Estimator(STE) (Swanson et al. 2024) to skip the non-differentiable part. Second, we develop a dynamic financial market environment based on a multiagent system. In this environment, market makers, informed traders, and noise traders interact through an order book (Glosten and Milgrom 1985) that is updated at each time step. Thus, we can use multi-agent systems to simulate financial markets (Byrd, Hybinette, and Balch 2020). Notably, our execution agent's trades are also incorporated into this process to compute each time step's market state, ensuring that the market impact is accurately modeled in the environment when using the dynamic environment to train an execution agent. Thirdly, we propose a hybrid environment training scheme that takes advantage of the benefits of using real data from a static environment while also considering the impact that agents have on the market environment in a dynamic environment. Our contributions can be summarized as follows:

- We propose SE-RL, the first automated RL research framework that uses an LLM to design the RL algorithm. It integrates the Dual-Level Enhancement Kit to do prompt refinement and weight optimization to continually improve the LLM research capability to evolve the RL algorithm.
- We develop a dynamic financial market simulator based on a multi-agent system, which realistically models market impact through agent-based order book updates.
- We introduce a novel hybrid environment training paradigm that could take advantage of the benefits of using real data from a static environment while also considering the impact that agents have on the market environment in a dynamic environment.
- Comprehensive experiments on 200 realistic stock datasets demonstrate that the proposed SE-RL outperforms current state-of-the-art baselines.

Related Works

Order Execution Strategy

Traditional methods for optimal trade execution in finance are predominantly model-based (Almgren and Chriss 2001; Bertsimas and Lo 1998), relying on stochastic processes and optimal control theory to derive trading trajectories. However, such models often fail to capture the full complexity of real-world market dynamics. In practice, simpler strategies such as Time-Weighted Average Price (TWAP) (Bertsimas and Lo 1998) and the Volume-Weighted Average Price (VWAP) (Kakade et al. 2004) remain popular due to their cost-effectiveness and alignment with market conditions. In contrast, RL has emerged as a robust solution for dynamic markets. Most RL-based approaches focus on short-term execution, often within minutes. Early work applied Q-Learning for order placement (Nevmyvaka, Feng, and Kearns 2006), while subsequent research adapted Deep Q-Networks (DQN) for discrete order volumes. More recent developments include the use of Proximal Policy Optimization (PPO) with raw level-2 market data (Schulman et al. 2017; Wang, He, and Tan 2020; Gu et al. 2021), the introduction of hybrid frameworks for precise limit price selection (Cartea, Jaimungal, and Penalva 2015; Hafsi and Vittori 2024), and the development of cost-efficient RL methods specifically designed for dynamic market conditions (Li et al. 2023; Noor and Fatima 2024).

LLMs for Reinforcement Learning

LLMs have become a transformative tool for advancing RL (Verma et al. 2024). Significant progress has been made in leveraging LLMs to design reward functions using natural language instructions (Hazra et al. 2024; Kwon et al. 2023; Singh, Bhattacharyya, and Namboodiri 2024). For example, reward shaping techniques have been developed by training RL agents to perform intermediate tasks guided by language specifications (Bhambri et al. 2024; Guo et al. 2024; Sun et al. 2024), but these approaches primarily focus on single-agent rather than multi-agent settings (Niño-Mora 2023; Whittle 1988). Beyond reward design, LLMs have been integrated into model-based RL frameworks through two key roles: as simulators for constructing world models (Gao et al. 2024) and as interpreters for policy explanations (Zini and Awad 2022). Their capabilities extend to data processing, where they efficiently handle environment information and task directives (Wu et al. 2024; Keraghel, Morbieu, and Nadif 2024), and even function as decision-making agents (Xu et al. 2024), demonstrating significant improvements in the sample efficiency of offline RL. Furthermore, the extensive pre-trained knowledge and advanced modeling capabilities of LLMs show strong potential for supporting RL agents during complex decision-making processes (Pang et al. 2024).

LLMs for Automatic Research Flow

Recent studies have investigated the potential of LLMs to automate research ideation and scientific paper generation through virtual researcher frameworks (Schmidgall et al. 2025; Ghafarollahi and Buehler 2024; Swanson et al. 2024; Si, Yang, and Hashimoto 2024). A notable example is ResearchAgent (Baek et al. 2024), a system that autonomously generates research concepts, methodologies, and experimental designs through iterative refinement processes guided by multi-agent peer-review simulations. This architecture employs specialized reviewing agents that apply human-aligned evaluation metrics to enhance output quality. Similarly, the AI Scientist framework demonstrates end-to-end automation capabilities, generating novel research ideas, executing experimental code, and compiling complete scientific papers with integrated quality control mechanisms (Lu

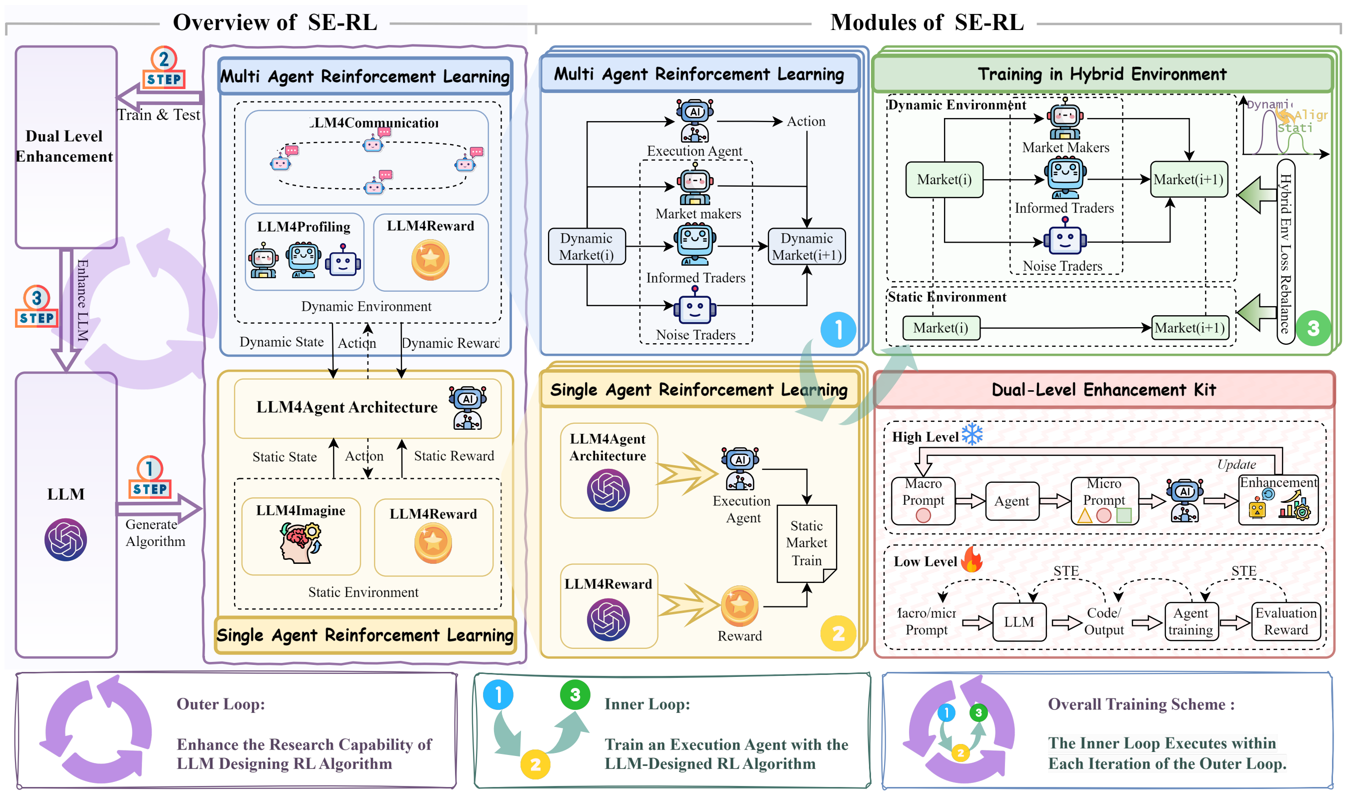


Figure 2: Overview of SE-RL Framework. The framework consists of two core schemes: (1) An outer loop for enhancing the LLM's research capabilities. (2)An inner loop for training an execution agent. In each outer loop iteration, the LLM first designs the key modules of the RL algorithm by generating each module's code or directly using the LLM as each module to do inference. Then we use the generated algorithm to train an agent through an inner loop, which consists of (Train multi-agent system to obtain a dynamic environment, Train execution agent in static environment, Train execution in dynamic environment with hybrid env loss rebalance operation). After that, with the trained execution agent performance as feedback, we use the Dual-Level Enhancement Kit(DEK) to optimize the prompt and LLM weight to further improve the LLM research capability. This workflow loop continues to run until the algorithm proposed by LLM does not show a significant improvement over the previous round.

et al. 2024). Despite these advancements in automated scientific workflows, current implementations remain underexplored in addressing the unique challenges of algorithmic innovation for RL, particularly in dynamic reward shaping and environment modeling.

Method

In this section, we introduce an SE-RL framework for designing various RL algorithms to train an execution agent. An overview of SE-RL is depicted in Figure 2. We'll begin by presenting our complete framework, followed by an explanation of how LLM is used to design RL algorithms. Furthermore, we'll introduce the Dual-Level Enhancement Kit, which continuously improves LLM research capabilities. Finally, a hybrid training method and the entire training scheme will be explained.

SE-RL Framework: An Automated RL Researcher

As illustrated in Figure 2, our methodology is built upon a nested loop framework designed to iteratively enhance an LLM's capability as an automated RL algorithm designer in order execution. The framework's two core components are an outer loop for enhancing the LLM's research capabilities and an inner loop for training an execution agent.

In each outer loop iteration, the LLM first designs the key modules of an RL algorithm. The performance of an execution agent trained with this generated algorithm is then evaluated, and the results are used as feedback to optimize the LLM's design proficiency for the next cycle. This entire feedback and enhancement process is powered by the Dual-Level Enhancement Kit(DEK).

The inner loop, which executes within each outer loop cycle, is dedicated to training the execution agent using the latest LLM-designed algorithm. This training follows a three-stage regimen. First, training in a multi-agent, dynamic market simulation. Second, training in a single-agent, static market setting. Finally, applying a hybrid environment training paradigm to reduce environmental bias and enhance algorithm robustness across regimes.

Note that unless otherwise specified, the agents in this work are small models.

LLM-Powered RL Algorithm Design

Our order execution framework is to train a single agent within a multi-agent market simulation.

The foundation of this framework is a single-agent reinforcement learning paradigm where LLM is innovatively employed to design core modules. As shown in Figure 2, LLM is responsible for generating the Python code for two critical components: (1) the reward function, which steers the agent toward optimal trading strategies, and (2) the agent's network architecture, enabling more sophisticated structures that can adapt to market dynamics. Beyond code generation, LLM also powers an imagination module that infers future states, actions, and rewards from past agent-environment interactions. This faculty for imagination provides the agent with a rich corpus of simulated experiences, markedly accelerating the training and adaptation process. In contrast to these LLM-augmented modules, the agent's action space is human-defined according to the MacMic(Niu, Li, and Li 2024) framework to maintain experimental consistency. For specific details on how LLM designs a single-agent algorithm, refer to the supplementary materials.

The efficacy of any RL agent is fundamentally tied to its training environment. Traditional approaches have often relied on a static market assumption, utilizing historical order books and prices that do not respond to the agent's actions. This methodology inherently neglects market impact, a critical factor that can lead to a severe degradation of performance when an agent is actually deployed in a live, dynamic market. We empirically substantiate this claim in our experiments. To overcome this limitation, we construct our training environment as a dynamic multi-agent system that realistically simulates market responses. This system is composed of three distinct classes of agents: (1) Market makers, tasked with providing liquidity based on historical data. (2) Informed traders, who trade on predictions of future price movements. And (3) noise traders, who react to past price action. To capture a wide range of market behaviors, the noise traders are further differentiated into aggressive, conservative, and balanced profiles. The differences among these types lie in the frequency of each order f, the amount of leverage λ at the time of the order, and the type of retracement r. The specific configurations for these agent profiles can be defined by an LLM, which sets their hyperparameters to ensure diverse and realistic market behavior. For a specific example of how LLM designs a multi-agent algorithm, please refer to the supplementary material.

The simulated environment explicitly connects the single agent to the market. Each time the order book is constructed, it incorporates the orders submitted by our execution agent along with those from the simulator's agents. A new market price is then computed by matching these collected orders, calculated as the midpoint of the highest bid and lowest ask prices at the final matching point. The performance of the simulator's internal agents is then evaluated by rewarding them based on a function of the emergent market price and the real historical price, ensuring the simulation stays grounded in reality.

$$R_i(t) = f(P_{\text{simulate}}(t+1), P_{\text{real}}(t+1)) \tag{1}$$

Such a multi-agent system could create a dynamic and responsive orderbook-based environment for the single agent's training, allowing us to effectively account for the market impact of its order execution strategy.

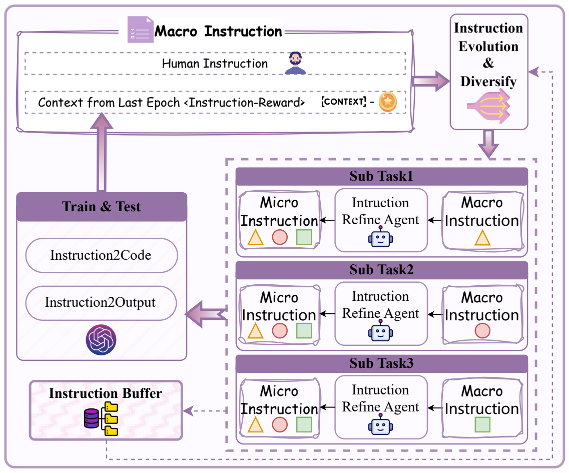


Figure 3: High-level enhancement kit workflow. Each prompt for generating module code consists of two parts: the macro part, which defines the overall task background, and the micro part, which defines the specific module goal and requirements. Each initial human-written prompt is optimized through an LLM-based Instruction Refine Agent. Furthermore, guided by each iteration prompt and corresponding generated algorithm reward pairs, the instruction evolution module could evolve the prompts via a process analogous to in-context learning. Finally, inspired by experience replay, a historical prompt/instruction buffer regularizes the optimization process by anchoring it to the initial human instruction.

Dual-Level Enhancement Kit for LLM Evolution

Our Dual-Level Enhancement Kit is designed to optimize the performance of the LLM as an RL designer, operating on two distinct planes. At the High Level, we focus on iteratively refining prompts to improve the quality of the LLM's generated outputs. At the Low Level, we introduce a method to directly fine-tune the LLM's weights by making its end-to-end contribution to the RL process differentiable.

High-Level Enhancement. Figure 3 illustrates the details of the High-Level Enhancement (HLE) workflow. It uses prompt engineering strategies to guide the LLM's reasoning. The process begins with a manually defined human instruction I_0. The enhancement applies several refinement techniques. First, in a *Macro-Micro Refine* step, a macro instruction establishes the general context, purpose and constraints of the task. Subsequently, micro instructions are generated for each RL module, inheriting the context from the macro instruction to ensure role-specific optimization. To improve the accuracy of the instruction, the instruction refine agent is prompted to rephrase these instructions in its own language.

To guide this evolutionary process, the HLE incorporates a learning loop. Through *In-Context Reward Learning*, the system collects historical prompt-reward pairs, allowing the LLM to infer patterns that correlate with successful outcomes. At the end of a training epoch, the LLM analyzes the agent's returns R_t and automatically adjusts the instructions I_t to generate an improved version I_{t+1} for the next cycle. To ensure the system stability and anchor the evolved prompt to the initial human-written prompt. We set an instruction buffer, which would randomly sample the history prompt and feed it into the Instruction Evolution together with other parts to generate the next epoch refined prompt. Such a setting is inspired by experiment replay(Andrychowicz et al. 2017).

Low-Level Enhancement. The low-level mechanism enables the direct optimization of the LLM's weights by addressing the non-differentiable part in the RL pipeline. The entire workflow can be summarized as *prompt* $\rightarrow$ *LLM* $\rightarrow$ *code/output* $\rightarrow$ *agent training* $\rightarrow$ *evaluation reward*. In this chain, the LLM's output code and the final evaluation metrics are non-differentiable, preventing the backpropagation of gradients to the LLM's parameters. We use STE to address this issue. Notably, such optimization only fine-tunes the weights with a minimal learning rate to ensure the effectiveness of the LLM. For specific algorithms, please refer to the supplementary materials.

Training and Optimization

Algorithm 1 presents the core training scheme of SE-RL, containing two nested loops that jointly address algorithmic evolution and agent training. When training an execution agent in the static and dynamic environment, we employ a rebalancing scheme to balance different environment weights to ensure our model takes advantage of both environment strengths. The final rebalanced losses are formulated as follows:

$$\mathcal{L}_{\text{rebalance}} = \alpha\mathcal{L}_s + \beta\mathcal{L}_d \tag{2}$$

where α and β values will be dynamically adjusted in each round of optimization to ensure that static and dynamic environment losses are of the same order of magnitude.

The whole evolution loop terminates when the LLM-generated algorithm performance does not show a significant improvement over the previous round through the following metrics:

$$\frac{PA_j - PA_{j-1}}{PA_{j-1} - PA_{j-2}} < \epsilon \tag{3}$$

where j denotes the epoch index and PA means the Price Advantage over VWAP, which is our core evaluation metric.

Algorithm 1: SE-RL Training Scheme

```
Input: Base LLM $\mathcal{M}$, Static Env $\mathcal{E}_s$, Dynamic Env $\mathcal{E}_d$
Output: Optimized Policy $\pi^*$
1: Initialize population $\mathcal{P} \leftarrow \emptyset$, performance buffer $\mathcal{B} \leftarrow \emptyset$
2: $P \leftarrow$ BuildTaskPrompt$(\mathcal{P}, \mathcal{B})$, $j \leftarrow 0$
3: {Outer Loop: LLM Research Capability Evolution}
4: while $\neg$ convergence do
5:     {Step1: Design and Generate RL Algorithm}
6:     Generate $A_j \leftarrow \mathcal{M}(\mathcal{P})$, $j \leftarrow j + 1$
7:     {Step2: Algorithm Train (Inner Loop) and Test}
8:     $\mathcal{P}_d \leftarrow$ Train$(\mathcal{M}, \mathcal{E}_d)$
9:     $\mathcal{P}_s \leftarrow$ Train$(\mathcal{M}, \mathcal{E}_s)$
10:    for $i \leftarrow 1$ to $N_{\text{rebalance}}$ do
11:        Sample batch $B_s \sim \mathcal{E}_s$, $B_d \sim \mathcal{E}_d$
12:        $\pi \leftarrow$ MixPolicies$(\mathcal{P}_s, \mathcal{P}_d)$
13:        Update $\pi$ with $\mathcal{L}_{\text{rebalance}} = \alpha\|B_s - \pi(B_s)\| + (1 - \alpha)\|B_d - \pi(B_d)\|$
14:    end for
15:    $\pi_j \leftarrow$ RLTrain$(A_j, \mathcal{E})$
16:    Evaluate $R_j \leftarrow$ Evaluate$(\pi_j, \mathcal{E})$
17:    Update $\mathcal{P} \leftarrow \mathcal{P} \cup \{A_j\}$, $\mathcal{B} \leftarrow \mathcal{B} \cup \{R_j\}$
18:    {Step3: Dual-Level Enhance LLM capability}
19:    $P_{\text{reflect}} \leftarrow$ GenerateReflectionPrompt$(\mathcal{B})$
20:    $\nabla_{\mathcal{M}} \leftarrow$ STE$(\mathcal{L}_{\text{LLM}})$
21:    Update $\mathcal{M}$'s weights using $\nabla_{\mathcal{M}}$
22:    Generate algorithm $A_j \leftarrow \mathcal{M}(P_{\text{reflect}})$
23:    if $j > 2$ and $\frac{R_j - R_{j-1}}{R_{j-1} - R_{j-2}} < \epsilon$ then
24:        break {Performance saturation}
25:    end if
26: end while
27: return $\pi^* \leftarrow \arg\max_{\pi \in \mathcal{P}} \mathbb{E}[R(\pi)]$
```

Experiments

We conduct experiments to investigate the following research questions. **Q1**: Does SE-RL outperform existing algorithms? **Q2**: How does each component of SE-RL affect performance? **Q3**: Does dual enhancement kit training improve performance? **Q4**: Is training in dynamic environments stable? **Q5**: How effective is the agent trained in dynamic environments?

Experiments Setup

Dataset. All our experiments are conducted on historical transaction data from two stock indices following MacMic (Niu, Li, and Li 2024). The first is China A-share CSI100, which comprises 100 component stocks representing the state of the China A-share market. The second is the NASDAQ100, which includes the 100 largest non-financial international companies listed on NASDAQ. We test all the methods in the buying direction and directly use the raw market information as the agent's state the same as MacMic. We also follow the MacMic to set the trading task.

Implementation Details. Although ChatGPT is by far the most effective model, in order to ensure the fine-tunability of the parameters, we have selected open-sourced LLaMA3.1

as the foundational LLM for SE-RL. In particular, for training, we set the ϵ to 0.1, we use bash scripts instead, where human involvement is required to ensure that the entire training process is automated, we use anyloss regression version to do non-differentiable optimization. We only fine-tune the add&normalization layer and positional encoding layer with LORA. For cache replay, we set the α_{bash} to 0.1. All experiments are conducted on 64 NVIDIA H100 GPUs.

Compared Methods and Evaluation Metrics. We compare the performance of our method with other market making methods including traditional methods such as TWAP (Bertsimas and Lo 1998), VWAP (Kakade et al. 2004), AC (Almgren and Chriss 2001) and RL-based methods such as DDQN (Van Hasselt, Guez, and Silver 2016), PPO (Schulman et al. 2017), OPD (Li and Hoi 2014), HALOP (Pan et al. 2022), and MacMic (Niu, Li, and Li 2024). Our evaluation metrics are designed around two core concepts in quantitative finance: the excess return and the risk. Like Price Advantage over VWAP (PA), Win Ratio (WR), Gain-Loss Ratio (GLR) measure excess return, and Averaged Final Inventory (AFI) to measure risk, just like MacMic.

Results and Analysis

Overall Performance. To validate the effectiveness of SE-RL in dynamic markets (Q1), we conducted extensive evaluations across 200 stocks from CSI100 and NASDAQ100. As demonstrated in Tables 1 and 2, our framework demonstrates superior performance across all key metrics. On the CSI100 dataset, SE-RL achieves a PA of 4.25 bps, representing a 28.4% improvement over the strongest baseline MacMic. This performance gain is particularly evident in high-volatility periods where market impact effects are pronounced, with SE-RL maintaining a WR of 0.99 compared to HALOP's 0.65.

The NASDAQ100 results further reinforce these findings, with SE-RL outperforming conventional RL methods by 1.72bps in PA, 0.16 in WR, and 0.19 in GLR, respectively. The comparison results across these 200 stocks demonstrate that the hybrid environment training approach enhances performance stability compared to the various baselines.

Methods	**PA (bps)**$\uparrow$	**PA-std (bps)**$\downarrow$	**WR**$\uparrow$	**GLR**$\uparrow$	**AFI**$\downarrow$
TWAP	-0.12	**3.12**	0.49	0.97	0.00
VWAP	-3.29	4.87	0.39	0.78	0.04
AC	-1.24	4.21	0.45	1.02	0.00
DDQN	-1.21	7.09	0.47	0.99	0.05
PPO	-0.98	7.01	0.52	0.92	0.03
ODP	0.32	6.78	0.53	1.07	0.10
HALOP	2.89	6.12	0.65	1.12	0.07
MacMic	3.31	6.23	0.72	1.37	0.02
Ours	$\mathbf{4.25}^{+0.94}$	$\mathbf{5.12}^{+2.00}$	$\mathbf{0.99}^{+0.27}$	$\mathbf{1.63}^{+0.26}$	$\mathbf{0.00}^{+0.00}$

Table 1: The comparison results of the proposed method and the benchmarks on CSI100.

Component Effectiveness Analysis. To investigate the contribution of each framework component (Q2), we conducted ablation studies across three critical dimensions. As shown in Table 3, the complete SE-RL configuration achieves 4.25bps PA, representing a 129% improvement over the base RL implementation. The DEK contributes 62% of this improvement through its unique combination of high-level prompt engineering and low-level differentiable optimization. When integrating the dynamic market simulator, we observe a 29% increase in winning ratio, confirming the importance of realistic market impact modeling. The hybrid environment training mechanism proves particularly effective in balancing environment-specific biases.

Methods	**PA (bps)**$\uparrow$	**PA-std (bps)**$\downarrow$	**WR**$\uparrow$	**GLR**$\uparrow$	**AFI**$\downarrow$
TWAP	-0.19	4.31	0.49	1.01	0.00
VWAP	-2.89	4.72	0.35	0.79	0.02
AC	-2.31	4.39	0.39	0.96	0.00
DDQN	-1.98	6.89	0.45	0.97	0.03
PPO	-1.20	6.79	0.46	0.96	0.04
ODP	1.03	6.82	0.54	1.05	0.12
HALOP	2.75	5.87	0.63	1.13	0.05
MacMic	3.14	5.69	0.74	1.28	0.01
Ours	$\mathbf{4.86}^{+1.72}$	$\mathbf{3.97}^{-0.34}$	$\mathbf{0.90}^{+0.16}$	$\mathbf{1.47}^{+0.19}$	$\mathbf{0.00}^{+0.00}$

Table 2: The comparison results of the proposed method and the benchmarks on NASDAQ100.

High-level	**Low-Level**	**Dynamic**	**Hybrid**	**PA (bps)**$\uparrow$	**WR**$\uparrow$
✓				$2.96^{+0.70}$	$0.73^{+0.08}$
✓	✓			$3.34^{+1.62}$	$0.75^{+0.02}$
✓	✓	✓		$3.87^{+0.53}$	$0.80^{+0.05}$
✓	✓	✓	✓	$\mathbf{4.25}^{\mathbf{+2.38}}$	$\mathbf{0.99}^{\mathbf{+0.19}}$

Table 3: Ablation studies over different SE-RL components.

Effectiveness of Dual Enhancement Kit on Generated LLM Module Performance. To answer Q3 and assess the impact of the Dual Enhancement Kit (DEK) on the performance of different modules within the SE-RL framework, we compared the optimization results before and after applying the DEK. We use the PA denoted reward in the Figure and multipy it with successful rate to evaluate llm generated module performnace. Results shown in Figure 4, the DEK significantly improved the performance of each module in multiple dimensions, including reward, success rate, agent architecture, communication, state space, profiling, and imagination.

Before applying the DEK, the modules exhibited various degrees of suboptimal performance. This indicated that the initial configurations of these modules, although functional, were not fully optimized for the dynamic and complex nature of financial markets.

After applying the DEK, which includes both high-level prompt refinement and low-level differentiable optimization, we observed notable enhancements in the performance of each module. These findings suggest that the DEK is a critical component in enhancing the overall performance of the SE-RL framework.

Training Effect Analysis in Different Environments. To address Q4, we analyzed the training performance of SE-RL in both a real-world environment using static historical data and in a dynamic simulated environment across various

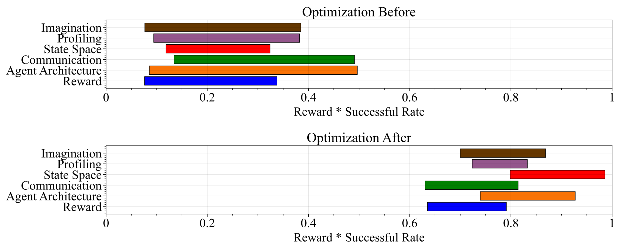
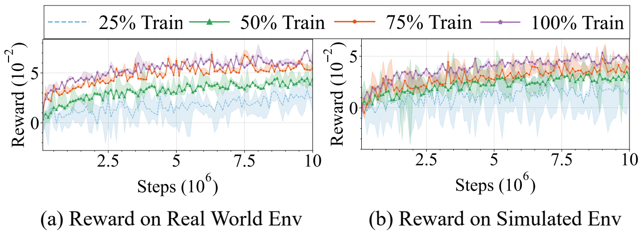


Figure 4: Different LLM algorithm module performance before and after dual enhancement kit training.



(a) Reward on Real World Env (b) Reward on Simulated Env

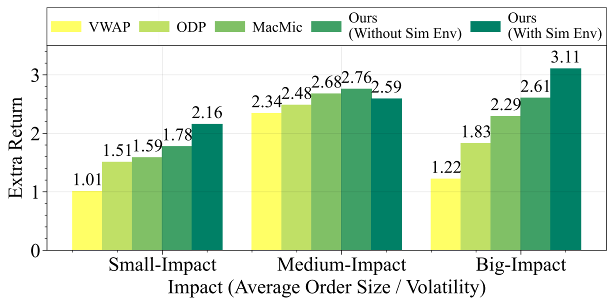
Figure 5: The training reward in different environments. We could observe that even with limited data, our method could also achieve relatively good results.

training data proportions of 25%, 50%, 75%, and 100%. The results are shown in Figure 5a and Figure 5b.

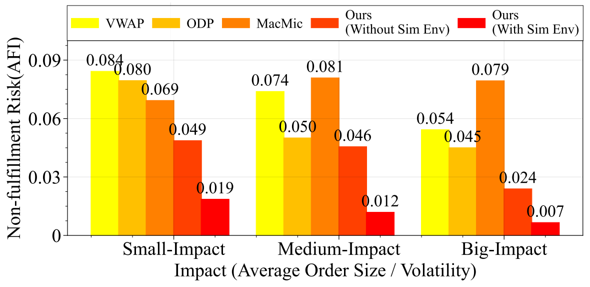
In a real-world environment, the training reward increased as the proportion of training data increased from 25% to 100%. This indicates that SE-RL can effectively learn from historical data, even with limited training samples. However, in a simulated environment, the reward growth was more pronounced, especially between 25% and 50% training data. This suggests that the dynamic nature of the simulated environment, which incorporates market impact, provides a more challenging yet rewarding training scenario.

Comparing the two environments, we find that while the real-world data offers a solid learning foundation, the simulated environment enhances the robustness and adaptability of SE-RL by introducing dynamic market conditions. The higher rewards achieved in the simulated environment highlight the importance of combining static historical data with dynamic simulations to optimize training performance and generalizability under different market conditions.

Performance Under Various Market Impacts. To answer Q5, we categorized experiments into small, medium, and big impact groups based on order size relative to market volatility. As illustrated in Figure 6a, SE-RL demonstrates consistent superiority over all baselines across every group. The advantage of our method is particularly pronounced in the most challenging, high-volatility scenarios. For instance, in the Big-Impact group, SE-RL achieved an extra return of 3.11, substantially outperforming the strongest competitors MacMic at 2.29 and ODP at 1.83. This result underscores the effectiveness of SE-RL's learned execution strategy, which



(a) Extra Return in different impact groups



(b) Non-fulfillment Risk in different impact groups

Figure 6: Performance analysis under various market impacts. (a) shows our method achieves the best extra return in different market impact scenarios. (b) shows that our method could trade with the lowest risk.

adeptly navigates complex market dynamics.

Figure 6b further evaluates our agent's performance by presenting the non-fulfillment risk (AFI) across different impact groups. The results highlight SE-RL's superior capability in risk management. Across all scenarios, SE-RL consistently achieved the lowest AFI, holding its risk to a minimal 0.01 in both the Medium and Big-Impact groups. This stands in stark contrast to baseline methods like VWAP and ODP, whose risks were five times higher at 0.05 in the most volatile scenarios. This superior risk control directly answers Q4: the agent trained in dynamic environments learns a robust policy that not only optimizes returns but also effectively mitigates execution risk.

Conclusion

This paper introduces SE-RL, an automated reinforcement learning framework that integrates LLM-driven algorithmic evolution with a dynamic market simulation. Our key contributions are threefold. (1) A dual-level enhancement mechanism, combining prompt refinement and parameter optimization, to accelerate the development of RL algorithms. (2) A multi-agent market environment that realistically models real-time order impact and agent interactions. And (3) hybrid environment training balancing static/dynamic market biases. Extensive experiments demonstrate SE-RL's superiority. The framework establishes a new paradigm for automated algorithmic research in quantitative finance, showing potential for broader financial applications.

References

Almgren, R.; and Chriss, N. 2001. Optimal execution of portfolio transactions. *Journal of Risk*, 3: 5–40.

Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; and Zaremba, W. 2017. Hindsight experience replay. *Advances in neural information processing systems*, 30.

Baek, J.; Jauhar, S. K.; Cucerzan, S.; and Hwang, S. J. 2024. Researchagent: Iterative research idea generation over scientific literature with large language models. *arXiv preprint arXiv:2404.07738*.

Bertsimas, D.; and Lo, A. W. 1998. Optimal control of execution costs. *Journal of financial markets*, 1(1): 1–50.

Bhambri, S.; Bhattacharjee, A.; Kalwar, D.; Guan, L.; Liu, H.; and Kambhampati, S. 2024. Extracting Heuristics from Large Language Models for Reward Shaping in Reinforcement Learning. *arXiv preprint arXiv:2405.15194*.

Bouchaud, J.-P.; Farmer, J. D.; and Lillo, F. 2009. How markets slowly digest changes in supply and demand. In *Handbook of financial markets: dynamics and evolution*, 57–160. Elsevier.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Byrd, D.; Hybinette, M.; and Balch, T. H. 2020. ABIDES: Towards high-fidelity multi-agent market simulation. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, 11–22.

Cao, M.; Shu, L.; Yu, L.; Zhu, Y.; Wichers, N.; Liu, Y.; and Meng, L. 2024a. Enhancing Reinforcement Learning with Dense Rewards from Language Model Critic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 9119–9138.

Cao, Y.; Zhao, H.; Cheng, Y.; Shu, T.; Chen, Y.; Liu, G.; Liang, G.; Zhao, J.; Yan, J.; and Li, Y. 2024b. Survey on large language model-enhanced reinforcement learning: Concept, taxonomy, and methods. *IEEE Transactions on Neural Networks and Learning Systems*.

Cartea, Á.; Jaimungal, S.; and Penalva, J. 2015. *Algorithmic and high-frequency trading*. Cambridge University Press.

Gao, C.; Lan, X.; Li, N.; Yuan, Y.; Ding, J.; Zhou, Z.; Xu, F.; and Li, Y. 2024. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *Humanities and Social Sciences Communications*, 11(1): 1–24.

Ghafarollahi, A.; and Buehler, M. J. 2024. Sciagents: Automating scientific discovery through multi-agent intelligent graph reasoning. *arXiv preprint arXiv:2409.05556*.

Glosten, L. R.; and Milgrom, P. R. 1985. Bid, ask and transaction prices in a specialist market with heterogeneously informed traders. *Journal of financial economics*, 14(1): 71–100.

Gu, Y.; Cheng, Y.; Chen, C. P.; and Wang, X. 2021. Proximal policy optimization with policy feedback. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(7): 4600–4610.

Guo, Q.; Liu, X.; Hui, J.; Liu, Z.; and Huang, P. 2024. Utilizing Large Language Models for Robot Skill Reward Shaping in Reinforcement Learning. In *International Conference on Intelligent Robotics and Applications*, 3–17. Springer.

Hafsi, Y.; and Vittori, E. 2024. Optimal Execution with Reinforcement Learning. *arXiv preprint arXiv:2411.06389*.

Hazra, R.; Sygkounas, A.; Persson, A.; Loutfi, A.; and Zuidberg Dos Martires, P. 2024. Revolve: Reward evolution with large language models for autonomous driving. *arXiv e-prints*, arXiv–2406.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Ibecheozor, U. H.; Iwuchukwu, E. C.; and Akinyele, A. R. 2025. Deep Learning Application in Sales Automation and Customer Experience Personalization in Small and Medium-Sized Business: A Hybrid Approach Using Transformer-Based Large Language Models and Reinforcement Learning.

Kakade, S. M.; Kearns, M.; Mansour, Y.; and Ortiz, L. E. 2004. Competitive algorithms for VWAP and limit order trading. In *Proceedings of the 5th ACM conference on Electronic commerce*, 189–198.

Keraghel, I.; Morbieu, S.; and Nadif, M. 2024. Beyond words: a comparative analysis of LLM embeddings for effective clustering. In *International Symposium on Intelligent Data Analysis*, 205–216. Springer.

Kwon, M.; Xie, S. M.; Bullard, K.; and Sadigh, D. 2023. Reward design with language models. *arXiv preprint arXiv:2303.00001*.

Kyle, A. S. 1985. Continuous auctions and insider trading. *Econometrica: Journal of the Econometric Society*, 1315–1335.

Li, B.; and Hoi, S. C. 2014. Online portfolio selection: A survey. *ACM Computing Surveys (CSUR)*, 46(3): 1–36.

Li, X.; Wu, P.; Zou, C.; and Li, Q. 2023. Hierarchical Deep Reinforcement Learning for VWAP Strategy Optimization. *IEEE Transactions on Big Data*.

Lillicrap, T. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lu, C.; Lu, C.; Lange, R. T.; Foerster, J.; Clune, J.; and Ha, D. 2024. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

Nevmyvaka, Y.; Feng, Y.; and Kearns, M. 2006. Reinforcement learning for optimized trade execution. In *Proceedings* *of the 23rd international conference on Machine learning*, 673–680.

Niño-Mora, J. 2023. Markovian restless bandits and index policies: A review. *Mathematics*, 11(7): 1639.

Niu, H.; Li, S.; and Li, J. 2024. Macmic: Executing iceberg orders via hierarchical reinforcement learning. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, 6008–6016.

Noor, K.; and Fatima, U. 2024. Meta Learning for Rapid Adaptation in Financial Time Series Forecasting. *IEEE Access*.

Pan, F.; Zhang, T.; Luo, L.; He, J.; and Liu, S. 2022. Learn continuously, act discretely: Hybrid action-space reinforcement learning for optimal execution. *arXiv preprint arXiv:2207.11152*.

Pang, A.; Wang, M.; Pun, M.-O.; Chen, C. S.; and Xiong, X. 2024. iLLM-TSC: Integration reinforcement learning and large language model for traffic signal control policy improvement. *arXiv preprint arXiv:2407.06025*.

Schmidgall, S.; Su, Y.; Wang, Z.; Sun, X.; Wu, J.; Yu, X.; Liu, J.; Liu, Z.; and Barsoum, E. 2025. Agent laboratory: Using llm agents as research assistants. *arXiv preprint arXiv:2501.04227*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Si, C.; Yang, D.; and Hashimoto, T. 2024. Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers. *arXiv preprint arXiv:2409.04109*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, 529(7587): 484–489.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359.

Singh, U.; Bhattacharyya, P.; and Namboodiri, V. P. 2024. LGR2: Language Guided Reward Relabeling for Accelerating Hierarchical Reinforcement Learning. *arXiv preprint arXiv:2406.05881*.

Sun, S.; Liu, R.; Lyu, J.; Yang, J.-W.; Zhang, L.; and Li, X. 2024. A Large Language Model-Driven Reward Design Framework via Dynamic Feedback for Reinforcement Learning. *arXiv preprint arXiv:2410.14660*.

Sutton, R. S. 2018. Reinforcement learning: An introduction. *A Bradford Book*.

Swanson, K.; Wu, W.; Bulaong, N. L.; Pak, J. E.; and Zou, J. 2024. The virtual lab: AI agents design new SARS-CoV-2 nanobodies with experimental validation. *bioRxiv*, 2024–11.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vaswani, A. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Verma, S.; Boehmer, N.; Kong, L.; and Tambe, M. 2024. Balancing Act: Prioritization Strategies for LLM-Designed Restless Bandit Rewards. *arXiv preprint arXiv:2408.12112*.

Wang, B.; Qu, Y.; Jiang, Y.; Shao, J.; Liu, C.; Yang, W.; and Ji, X. 2024a. LLM-empowered state representation for reinforcement learning. *arXiv preprint arXiv:2407.13237*.

Wang, L.; Ma, C.; Feng, X.; Zhang, Z.; Yang, H.; Zhang, J.; Chen, Z.; Tang, J.; Chen, X.; Lin, Y.; et al. 2024b. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6): 186345.

Wang, Y.; He, H.; and Tan, X. 2020. Truly proximal policy optimization. In *Uncertainty in artificial intelligence*, 113–122. PMLR.

Whittle, P. 1988. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A): 287–298.

Wu, S.; Liu, J.; Yin, X.; Cheng, G.; Zhao, X.; Fang, M.; Yi, X.; and Huang, X. 2024. Robust RL with LLM-Driven Data Synthesis and Policy Adaptation for Autonomous Driving. *arXiv preprint arXiv:2410.12568*.

Xu, Q.; Li, Y.; Xia, H.; and Li, W. 2024. Enhancing tool retrieval with iterative feedback from large language models. *arXiv preprint arXiv:2406.17465*.

Zini, J. E.; and Awad, M. 2022. On the explainability of natural language processing deep models. *ACM Computing Surveys*, 55(5): 1–31.

Reproducibility Checklist

1. General Paper Structure

1.1. Includes a conceptual outline and/or pseudocode description of AI methods introduced (yes/partial/no/NA) yes

1.2. Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results (yes/no) yes

1.3. Provides well-marked pedagogical references for less-familiar readers to gain background necessary to replicate the paper (yes/no) yes

2. Theoretical Contributions

2.1. Does this paper make theoretical contributions? (yes/no) no

If yes, please address the following points:

2.2. All assumptions and restrictions are stated clearly and formally (yes/partial/no) no

2.3. All novel claims are stated formally (e.g., in theorem statements) (yes/partial/no) no

2.4. Proofs of all novel claims are included (yes/partial/no) no

2.5. Proof sketches or intuitions are given for complex and/or novel results (yes/partial/no) no

2.6. Appropriate citations to theoretical tools used are given (yes/partial/no) no

2.7. All theoretical claims are demonstrated empirically to hold (yes/partial/no/NA) no

2.8. All experimental code used to eliminate or disprove claims is included (yes/no/NA) no

3. Dataset Usage

3.1. Does this paper rely on one or more datasets? (yes/no) yes

If yes, please address the following points:

3.2. A motivation is given for why the experiments are conducted on the selected datasets (yes/partial/no/NA) yes

3.3. All novel datasets introduced in this paper are included in a data appendix (yes/partial/no/NA) yes

3.4. All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no/NA) partial

3.5. All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations (yes/no/NA) yes

3.6. All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available (yes/partial/no/NA) yes

3.7. All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing (yes/partial/no/NA) yes

4. Computational Experiments

4.1. Does this paper include computational experiments? (yes/no) yes

If yes, please address the following points:

4.2. This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting (yes/partial/no/NA) yes

4.3. Any code required for pre-processing data is included in the appendix (yes/partial/no) yes

4.4. All source code required for conducting and analyzing the experiments is included in a code appendix (yes/partial/no) yes

4.5. All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes (yes/partial/no) yes

4.6. All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from (yes/partial/no) yes

4.7. If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results (yes/partial/no/NA) yes

4.8. This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks (yes/partial/no) yes

4.9. This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics (yes/partial/no) yes

4.10. This paper states the number of algorithm runs used to compute each reported result (yes/no) yes

4.11. Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information (yes/no) yes

4.12. The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank) (yes/partial/no) partial

4.13. This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments (yes/partial/no/NA) partial