EarnHFT: Efficient Hierarchical Reinforcement Learning for High Frequency Trading

Shayan Kebriti

June 2025

Introduction

High Frequency Trading (HFT)

- Uses algorithms to make trading decisions in short time scales (e.g., second-level)

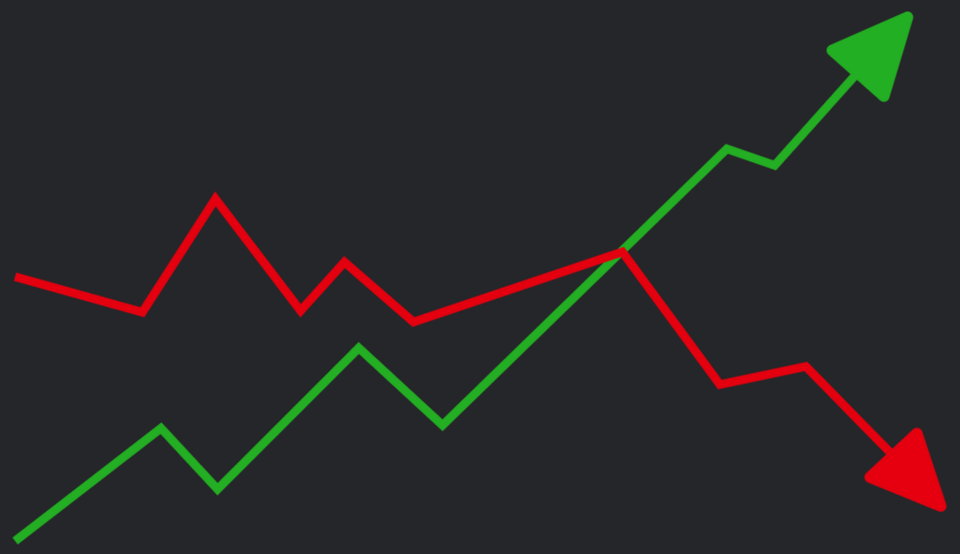
- Widely used in Cryptocurrency (e.g. Bitcoin)
    - <u>High potential</u> for profitability due to <u>high volatility</u>
    - Lower Risk
    - Run 24/7, No overnight risk

- Helps the discovery of the true price of an asset

Chellenges

- An extremely large time horizon
  - Atari games → ~ 6,000 steps
  - HFT → ~ 1 million steps (agents need to be evaluated in dozens of days)
    - Large time horizons need more data to converge

- Dramatic market changes
  - Agents trained on historical data fail in maintaining performance over long periods



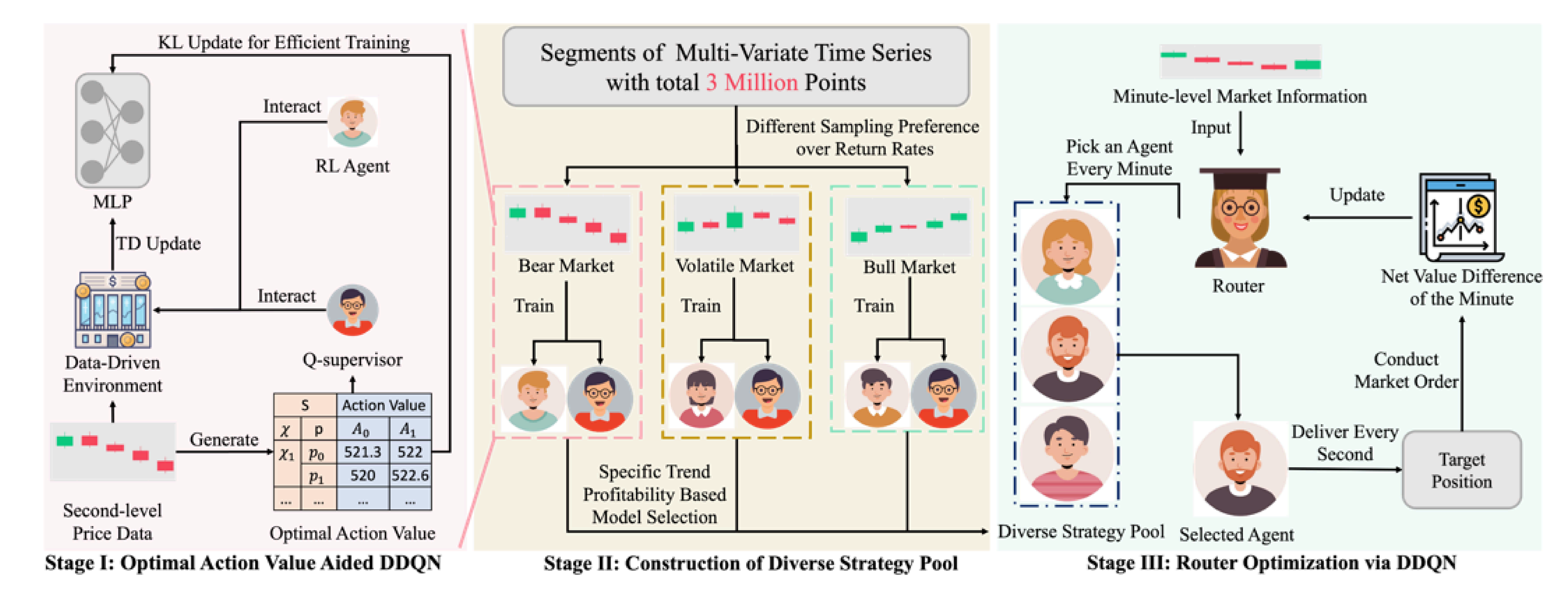
Proposed Method

Hierarchical MDP Formulation

- Low-level MDP (second-level operation)
  - Action: Choose a target position
  - State: Second-level market features
  - Reward: Net value change from the last second
- High-level MDP (minute-level operation):
  - Action: Choose a low-level policy (agent)
  - State: Minute-level market characteristics
  - Reward: Total profit from the chosen agent over the past minute
- Goal
  - Learn a set of diverse low-level policies for different market trends
  - Train a high-level policy to <u>dynamically select</u> the right agent based on current market conditions

Overview of EarnHFT

- Three Stages for Training
- Two-level Inference



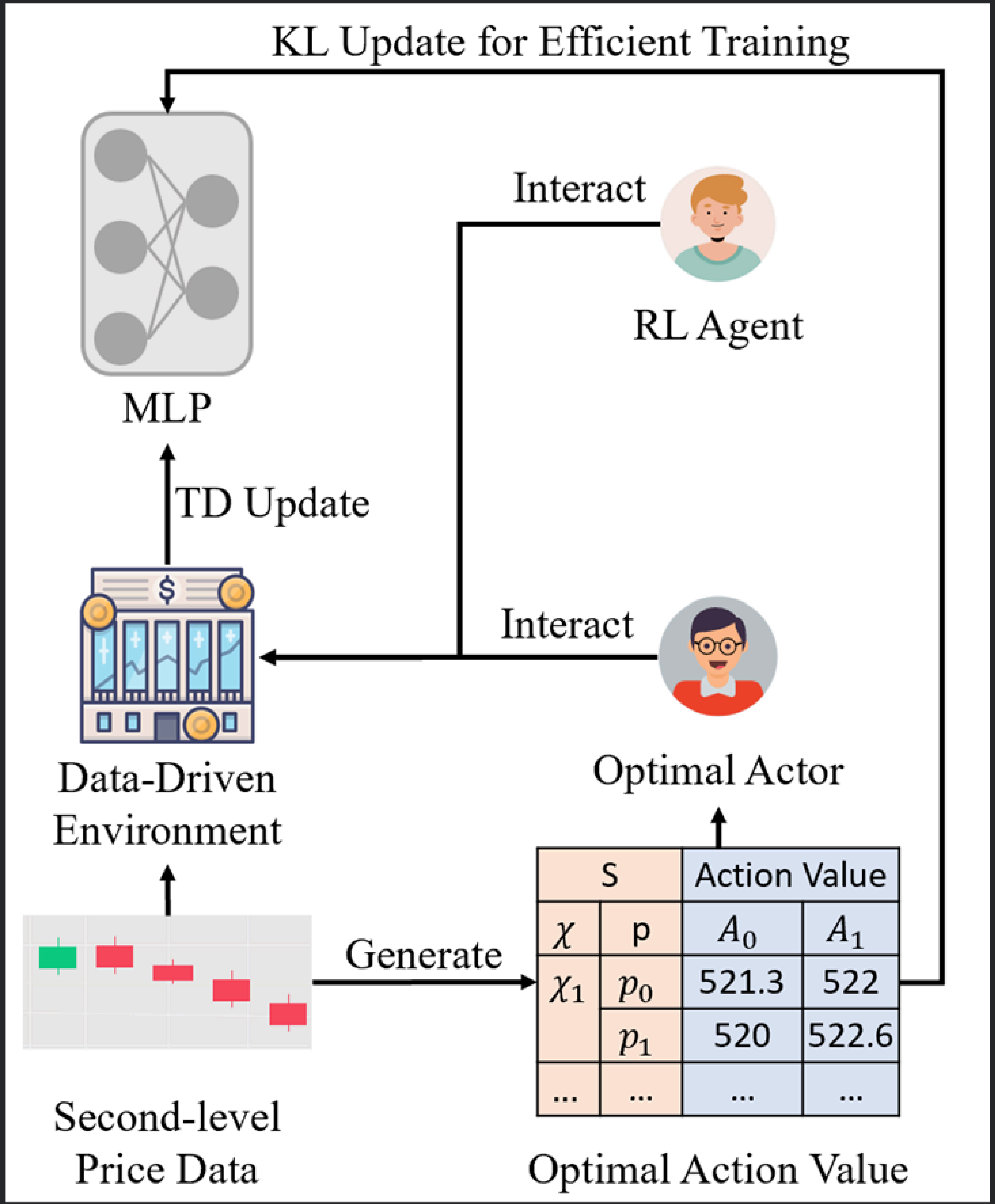
Stage 1: Efficient RL with Q-Teacher

- Computing Q* using Dynamic Programming
- Using DDQN for Q
- Overall loss:

$$L(\theta_i) = L_{td} + \alpha KL(Q_t(\chi, p, \cdot; \theta_i) || Q^*(\chi, p, \cdot))$$

$$L_{td} = (r + \gamma \max Q_t(\chi', a, \cdot; \theta_i') - Q_t(\chi, p, a; \theta_i))^2$$

- The *Optimal Actor* helps overcome the drift due to exploration of the *RL Agent*



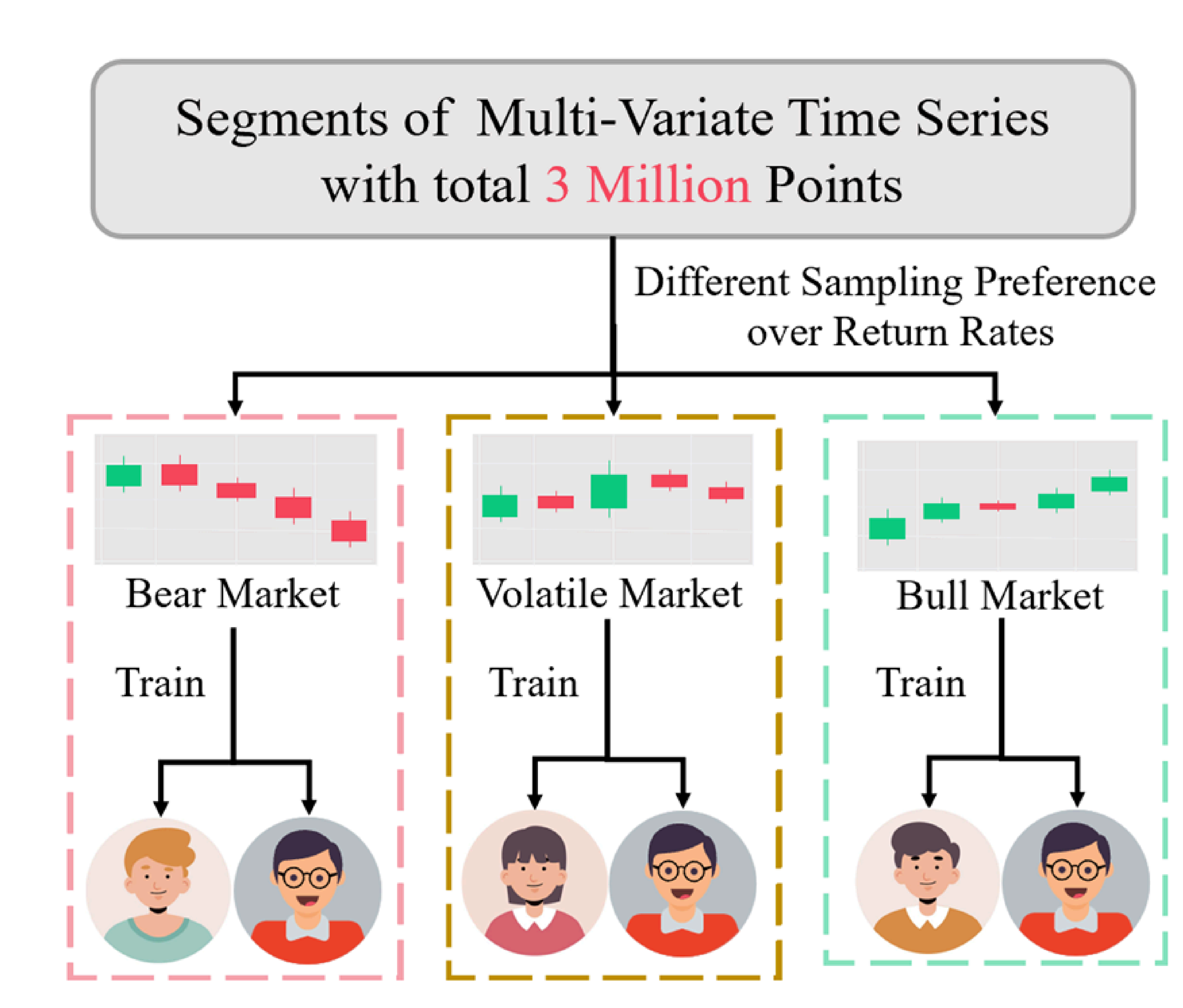
Stage 2: Building Diverse Agents

1. Seprate dataset into chunks

2. Train agents with different market trends by giving different priorities to each chunk:

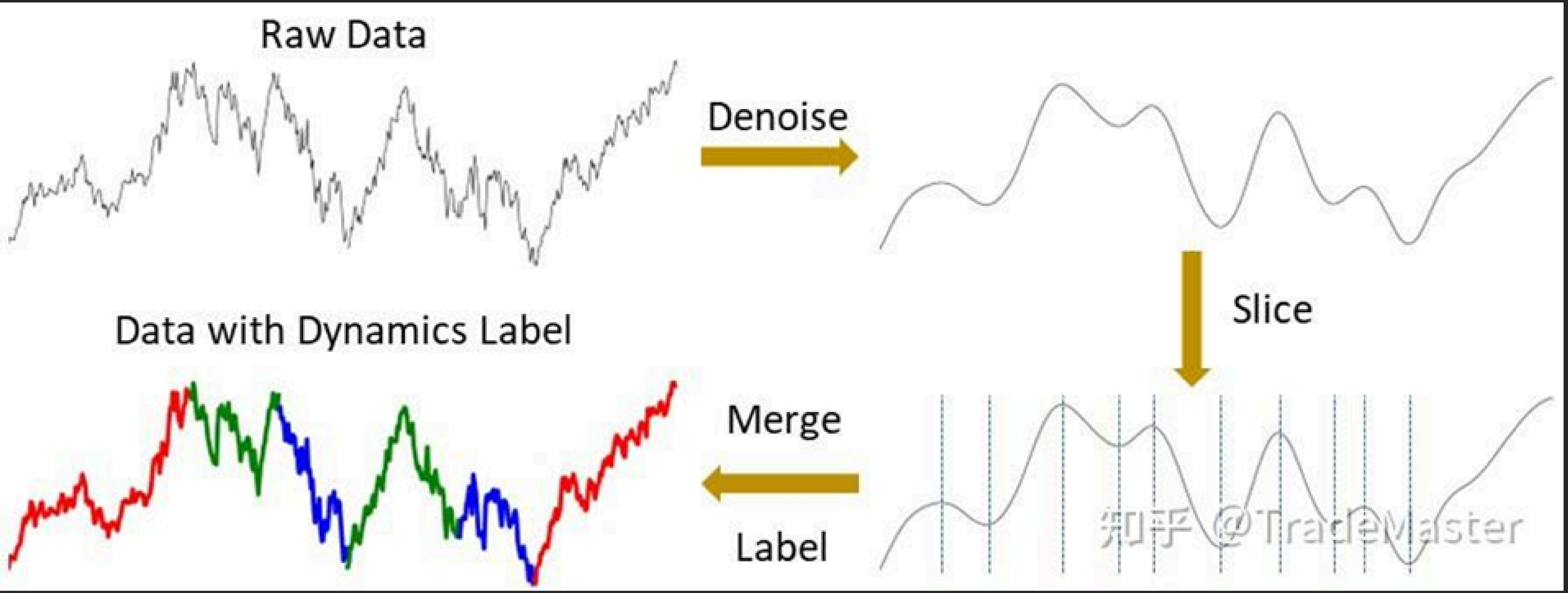
$$f(x) = \begin{cases} \frac{e^{\beta r}}{pdf(r)} & \text{if } Q_{\frac{\theta}{2}}(R) \leq r \leq Q_{1-\frac{\theta}{2}}(R) \\ e^{\beta r} & \text{if } r \geq Q_{1-\frac{\theta}{2}}(R) \vee r \leq Q_{\frac{\theta}{2}}(R) \end{cases}$$

- r → return
- β → Preference parameter (high/low return)
- pdf(r) → Probability density of return r
- Q_θ(R) → Quantile function
- θ → Risk threshold



Stage 2 (Cont.): Construction of Agent Pool

1. Market Segmentation & Labelling



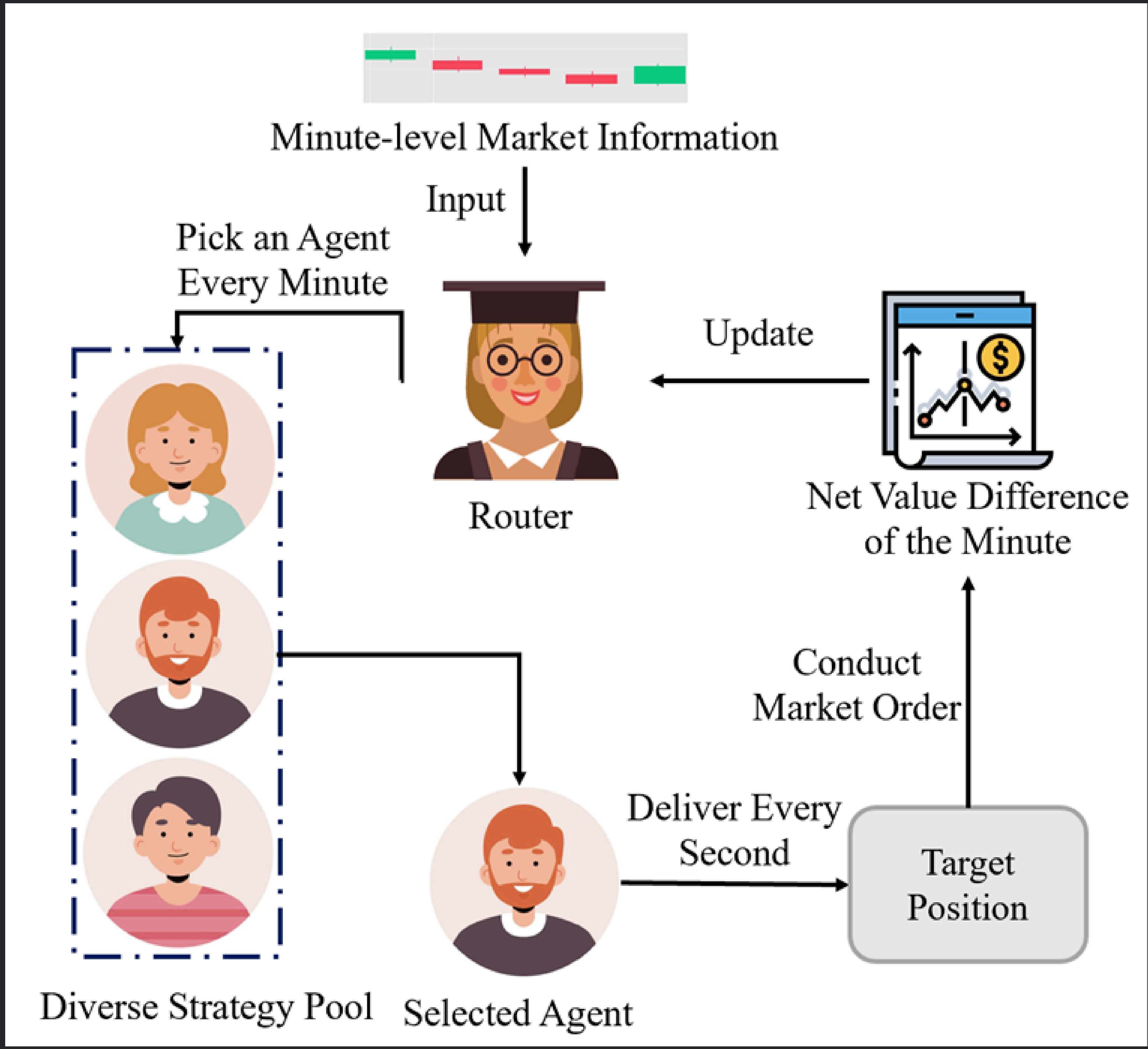
2. Evaluate the trained agents and choose the top ones to construct <u>an agent pool of *(m, n)*</u>

$m \rightarrow$ number of market trends

$n \rightarrow$ initial positions

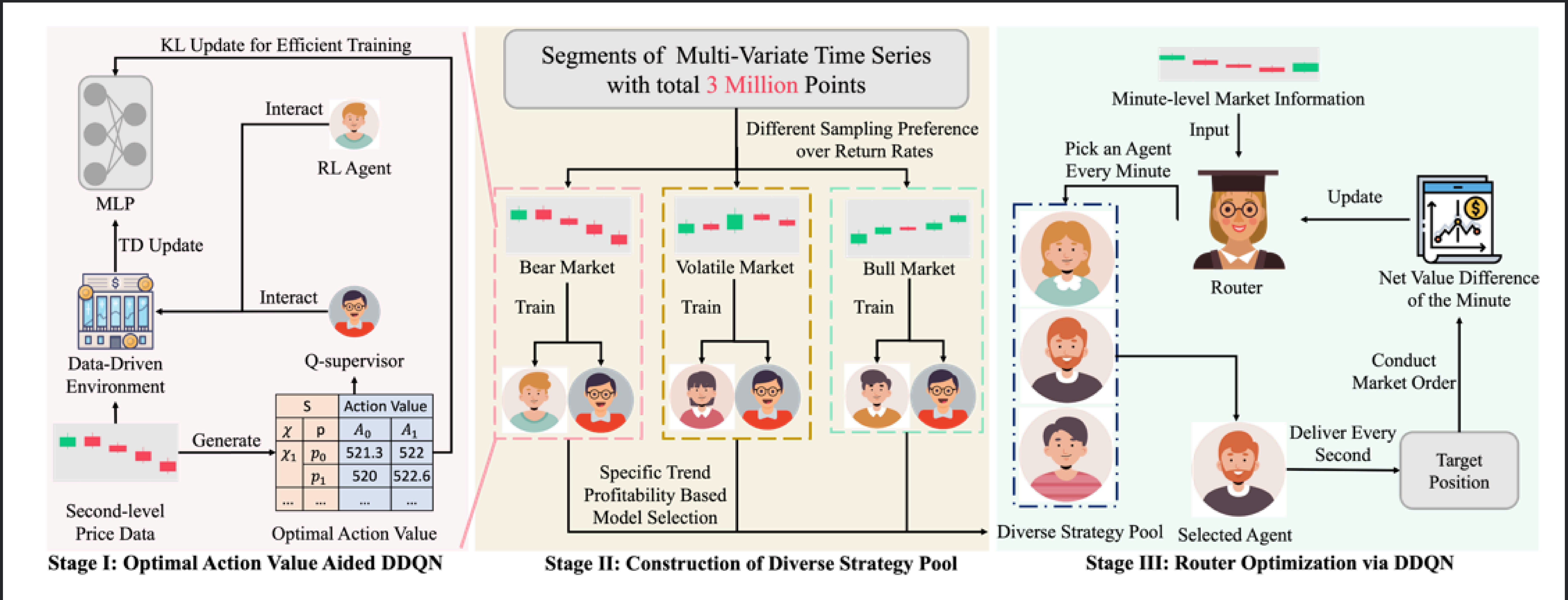
Stage 3: Dynamic Routing Optimization

- Train a DDQN for *Router*
- We reduce the number of possible low-level agents to *m*
  - using the agents with the <u>same initial position</u>



Summary

Summary



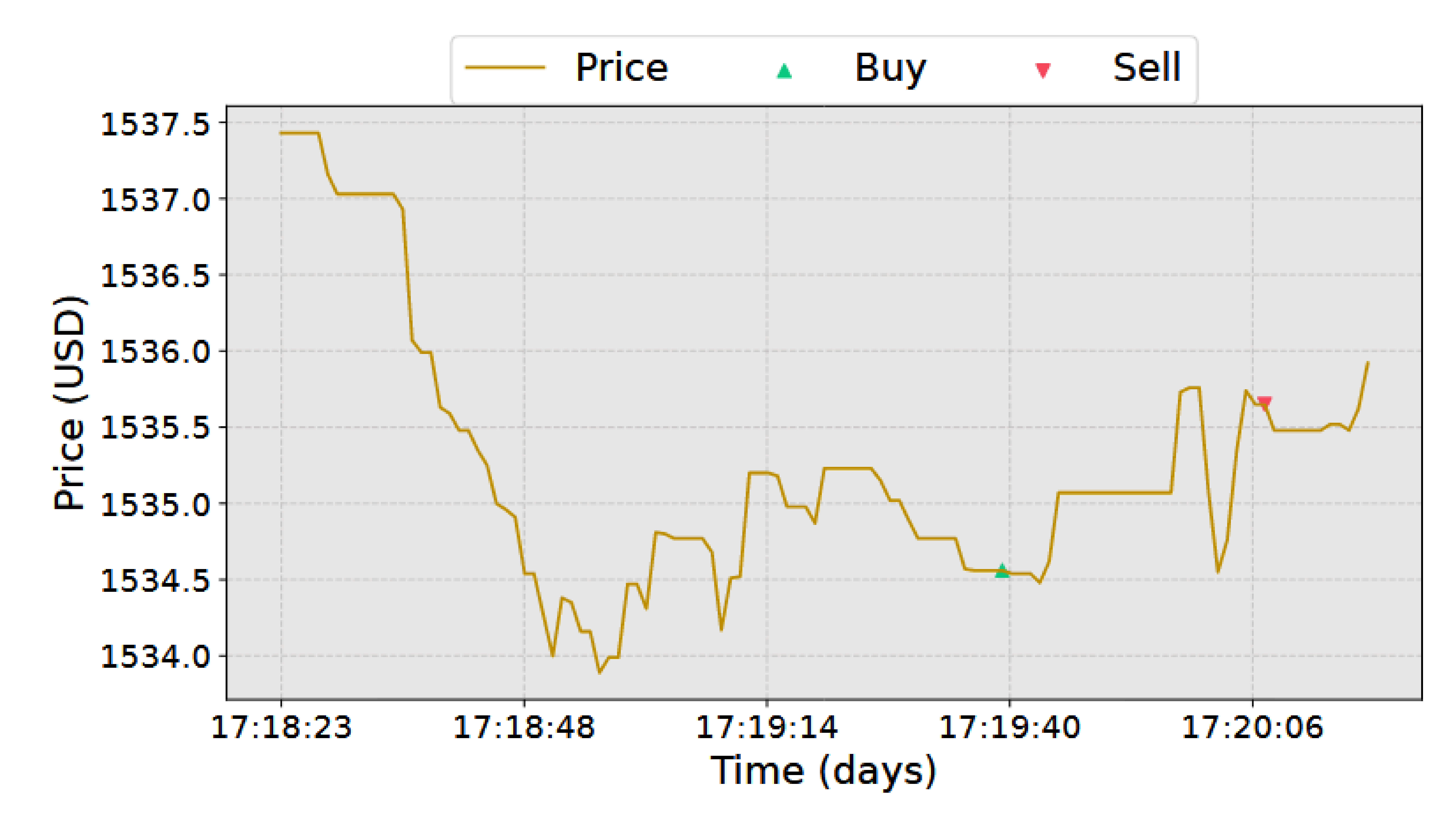
KL Update for Efficient Training
MLP
Interact
RL Agent
TD Update
Interact
Data-Driven Environment
Q-supervisor
S
Action Value
χ
p
A0
A1
X1
p0
521.3
522
p1
520
522.6
...
Generate
Second-level Price Data
Optimal Action Value
Stage I: Optimal Action Value Aided DDQN
Segments of Multi-Variate Time Series with total 3 Million Points
Different Sampling Preference over Return Rates
Bear Market
Volatile Market
Bull Market
Train
Train
Train
Specific Trend Profitability Based Model Selection
Stage II: Construction of Diverse Strategy Pool
Minute-level Market Information
Input
Pick an Agent Every Minute
Router
Update
Net Value Difference of the Minute
Conduct Market Order
Deliver Every Second
Target Position
Diverse Strategy Pool
Selected Agent
Stage III: Router Optimization via DDQN

Results

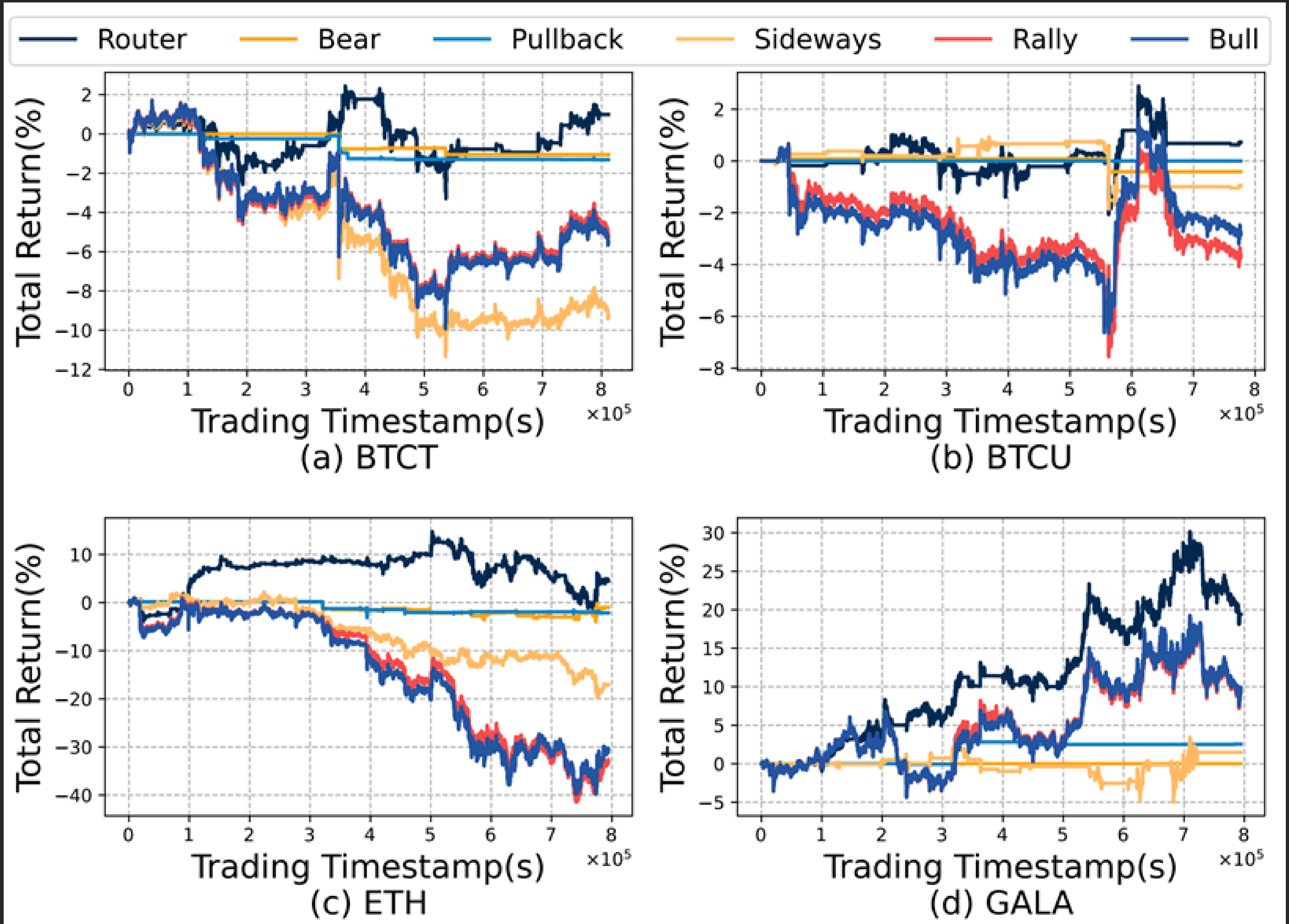
Experiments Results

		Prof↑	RAP↑	Risk↓			Prof↑	RAP↑	Risk↓
Market	Model	TR(%)	SR	MDD(%)	Market	Model	TR(%)	SR	MDD(%)
	DRA	-4.56	-4.28	9.24		DRA	-2.65	-4.82	5.84
	PPO	-3.61	-5.25	6.41		PPO	-0.60	-14.74	*0.65*
	CDQNRP	*-2.83*	*-2.91*	7.38		CDQNRP	-0.60	-19.52	**0.61**
BTCU	DQN	-3.48	-12.37	*4.09*	BTCT	DQN	*0.47*	**4.21**	0.66
	MACD	-6.07	-10.11	9.98		MACD	-4.02	-5.80	6.44
	IV	-2.99	-3.78	8.32		IV	-12.01	-17.83	12.66
	EarnHFT	**0.72**	**1.22**	**3.07**		EarnHFT	**0.99**	*1.34*	5.61
	DRA	-33.37	-9.06	45.88		DRA	10.56	4.77	10.60
	PPO	-22.61	-10.11	31.17		PPO	10.56	4.77	10.60
	CDQNRP	-6.82	-24.41	**6.96**		CDQNRP	5.22	4.51	*5.41*
ETH	DQN	-11.02	-9.47	*13.79*	GALA	DQN	2.94	3.55	**3.78**
	MACD	*-4.29*	*-1.78*	16.35		MACD	2.37	1.79	9.84
	IV	-27.42	-12.27	33.96		IV	*13.95*	*6.74*	9.91
	EarnHFT	**4.52**	**2.92**	13.89		EarnHFT	**19.41**	**9.77**	9.26

Experiments Results

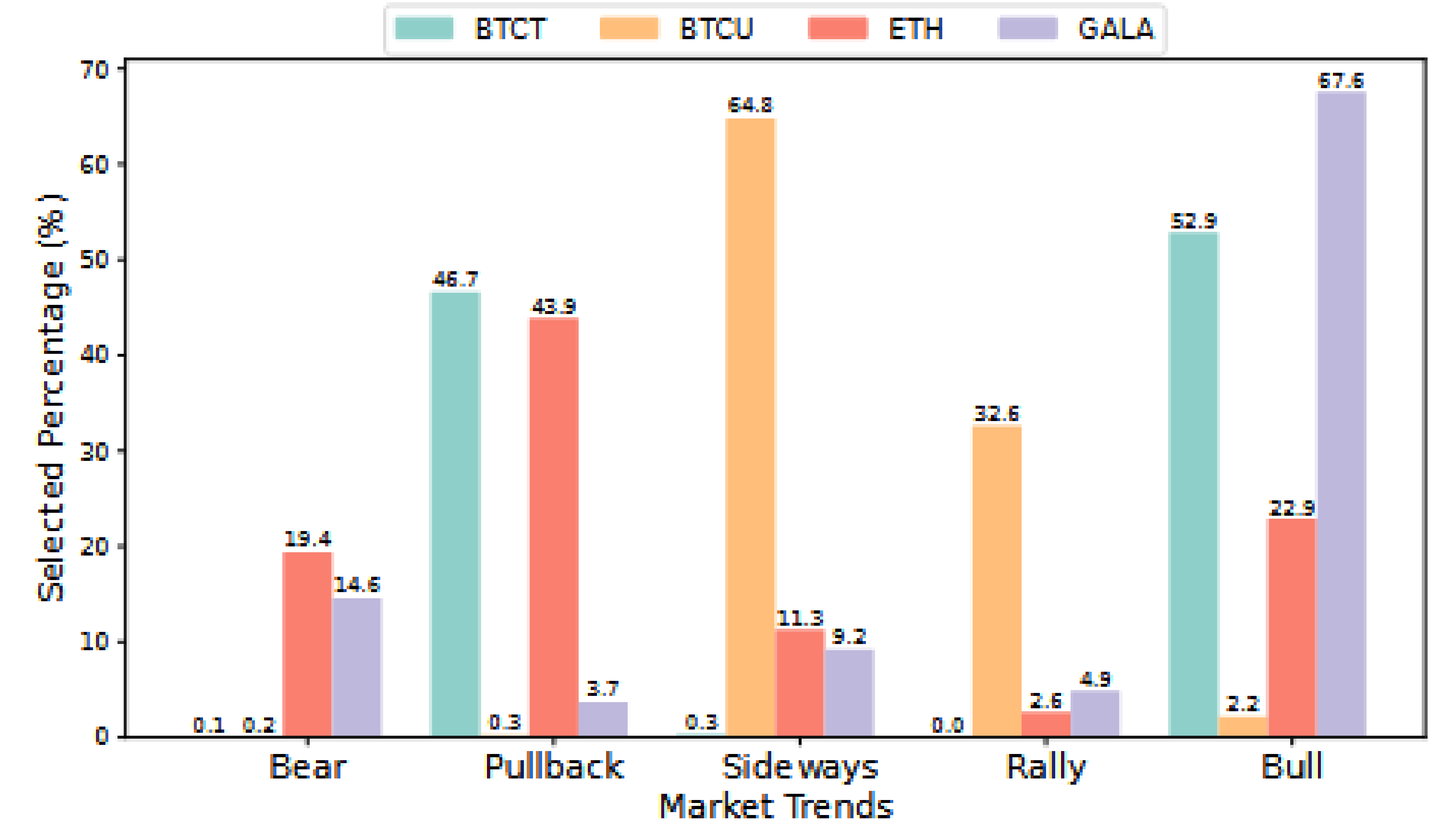


Experiments Results



Experiments Results

Dataset	Dynamics	Seconds	From	To
BTC/TUSD	Sideways	4057140	23/03/30	23/05/15
BTC/USDT	Sideways	3884400	22/09/01	22/10/15
ETH/USDT	Bear	3970800	22/05/01	22/06/15
GALA/USDT	Bull	3970740	22/07/01	22/08/15



Thanks!

Appendix

Algorithm 1: Construction of Optimal Action Value

Input: Multivariate Time Series $\mathcal{D}$ with Length N, Commission Fee Rate δ, Action Space A

Output: A Table Q^* Indicating Optimal Action Value at Time t, Position p and Action a.

Initialize Q^* with shape (N,$|A|$,$|A|$) and all elements 0.
for $t \leftarrow N-1$ to 1 **do**
    for $p \leftarrow 1$ to $|A|$ **do**
        for $a \leftarrow 1$ to $|A|$ **do**
            $Q^*[t,p,a] \leftarrow \max_{a'} Q^*[t+1,a,a'] + a \times p_{t+1}^{b1} - (p \times p_t^{b1} + E_t(p-a))$.
        end for
    end for
end for
return Q^*

Algorithm 2: Efficient RL with Q-Teacher

Input: Multivariate Time Series $\mathcal{D}$ with Length N, Commission Fee Rate δ, Action Space A
Output: Network Parameter θ

Initialize experience replay R, network Q_θ, target network $Q_{\theta'}$ and construct the optimal action value using Algorithm 1 and trading environment Env.
Initialize trading environment Env
for $t = 1$ to $N - 1$ **do**
    Choose action a_ϵ using ϵ-greedy policy.
    Store transition $(s, a_\epsilon, r, s', Q^*)$ in D
end for
Reinitialize trading environment Env
for $t = 1$ to $N - 1$ **do**
    Choose action a_o that $\operatorname{argmax}_a Q^*[t, p, a]$.
    Store transition (s, a_o, r, s', Q^*) in R
end for
Sample transitions $(s_j, a_j, r_j, s'_j, Q^*_j)$
Calculate L following Equation 2, do its gradient descent on θ and update $\theta' = \tau\theta + (1 - \tau)\theta'$.
return Q_θ

Algorithm 3: Market Segmentation & Labelling

Input: A Time Series $\mathcal{D}$ with Length N
Parameter: Risk threshold θ, Label number M
Output: Labels indicating the trend they belong to for every point in time series D

1: $D' \leftarrow$ denoising high frequency noise D.
2: Divide D' according to its extrema into segments S.
3: Merge adjacent segments in S if DTW (Muda, Begam, and Elamvazuthi 2010) and slop difference are small enough until S is stable.
4: Calculate threshold $H = Q_{1-\frac{\theta}{2}}(R)$, $L = Q_{\frac{\theta}{2}}(R)$
5: Calculate the upper bond and lower bonds of slopes for each label based on the quantile and the threshold.
6: Label each segment based on the bonds.
7: Return the label corresponding to each segment.

The Effectiveness of optimal value supervisor (OS) & optimal actor (OA)

OA	OS	GALA CS	GALA RS	GALA AHL	ETH CS	ETH RS	ETH AHL
✓	✓	78848	4.43	448	102400	12.32	81.3
✓		102400	0.24	38.7	102400	-1.40	4.15
	✓	4608	2.89	147	30720	4.87	35.8
		30720	-0.01	284	30720	-29.6	39.1

CS → Convergence Steps

RS → Converged Reward Sum

AHL → Average Holding Length