ArchetypeTrader: Reinforcement Learning for Selecting and Refining Learnable Strategic Archetypes in Quantitative Trading

Chuqiao Zong, Molei Qin, Haochong Xia, Bo An

Nanyang Technological University, Singapore
{zong0005, molei001, haochong001}@e.ntu.edu.sg, boan@ntu.edu.sg

Abstract

Quantitative trading using mathematical models and automated execution to generate trading decisions has been widely applied across financial markets. Recently, reinforcement learning (RL) has emerged as a promising approach for developing profitable trading strategies, especially in highly volatile markets like cryptocurrency. However, existing RL methods for cryptocurrency trading face two critical drawbacks: 1) Prior RL algorithms segment markets using handcrafted indicators (e.g., trend or volatility) to train specialized sub-policies. However, these coarse labels oversimplify market dynamics into rigid categories, biasing policies toward obvious patterns like trend-following and neglecting nuanced but lucrative opportunities. 2) Current RL methods fail to systematically use demonstration data. While some approaches ignore demonstrations altogether, others rely on "optimal" yet overly granular trajectories or human-crafted strategies, both of which can overwhelm learning and introduce significant bias, resulting in high variance and large profit drawdowns. To address these problems, we propose ArchetypeTrader, a novel reinforcement learning framework that automatically selects and refines data-driven trading archetypes distilled from demonstrations. The framework operates in three phases: 1) We use dynamic programming (DP) to generate representative expert trajectories and train a vector-quantized (VQ) encoder-decoder architecture to distill these demonstrations into discrete, reusable strategic archetypes through self-supervised learning, capturing nuanced market-behavior patterns without human heuristics. 2) We then train an RL agent to select contextually appropriate archetypes from the learned codebook and reconstruct action sequences for the upcoming horizons, effectively performing demonstration-guided strategy reuse. 3) We finally train a policy adapter that leverages hindsight-informed rewards to dynamically refine the archetype actions based on real-time market observations and performance, enabling more fine-grained decision-making and yielding profitable and robust trading strategies. Extensive experiments on four popular cryptocurrency trading pairs demonstrate that ArchetypeTrader significantly outperforms state-of-the-art approaches in both profit generation and risk management.

Introduction

With a market capacity exceeding 90 trillion dollars, the global financial sector continues to draw a vast range of participants seeking steady gains and opportunistic profits. Over the past decade, quantitative trading has gained particular traction by leveraging increasingly sophisticated tools for data analysis and decision-making. Among these, reinforcement learning (RL) is promising because of its ability to process high-dimensional financial data and to address complex sequential decision-making problems (Deng et al. 2016; Zhang, Zohren, and Stephen 2020; Liu et al. 2020a). In contrast to rule-based approaches that rely on human experts' insights, RL allows the trading agent to learn adaptive policies directly from interactions with the market environment, which is especially effective in volatile markets.

Although RL has achieved great success in quantitative trading, existing algorithms exhibit several critical shortcomings, mainly including: 1) Many existing methods treat the financial market as a homogeneous and stationary process (Briola et al. 2021; Jia et al. 2019). Consequently, the learned policies perform poorly on volatile instruments like cryptocurrencies, which experience frequent dynamic changes. Another common approach segments market conditions using human-designed indicators, such as market trends (e.g., bullish, bearish) or volatility, and trains specialized sub-policies for each regime (Qin et al. 2023; Zong et al. 2024). However, these coarse, human-engineered labels tend to oversimplify market dynamics and bias the sub-policies toward superficial trading behaviors like "trend-following" while neglecting nuanced yet profitable trading opportunities, thereby limiting overall strategy effectiveness. 2) Many RL-based trading methods fail to leverage demonstration data effectively. Some ignore demonstrations altogether (Zhu and Zhu 2022; Zou et al. 2024), while others adopt fully "optimal" yet overly granular trajectories (Qin et al. 2023; Zong et al. 2024) or rely on human-devised heuristic strategies (Liu et al. 2020b), both of which can overwhelm learning with noise or introduce bias. As a result, these approaches often exhibit high variance and suboptimal performance, having missed the opportunity to incorporate proven, high-return trading behaviors from the outset.

To address these challenges, we propose ArchetypeTrader, a novel RL framework learning to dynamically select and

refine strategic archetypes, which are discrete and reusable trading strategies derived from demonstrations. Unlike existing approaches that rely on monolithic policies or human-designed market segmentation, ArchetypeTrader operates through three tightly integrated phases: 1) In the first phase, a dynamic programming (DP) planner generates demonstration trajectories over fixed horizons without predefined human heuristics. A vector-quantized encoder-decoder is then trained to compress these trajectories into a compact codebook of strategically meaningful trading archetypes. 2) In the second phase, ArchetypeTrader trains an RL-driven archetype selector to pick the most suitable archetype for the current market regime at the start of each horizon. 3) In the third phase, a step-wise policy adapter refines the selected archetype's actions using the latest market observations and intra-horizon performance. Optimized not only for increasing total profit but also for minimizing regret that measures the gap to top hindsight opportunities within each horizon, our adapter is able to provides fine-grained, impactful adjustments without abandoning the archetype's overall intent.

Our key contributions are three-fold: 1) we introduce a self-supervised method to discover discrete trading archetypes from DP-generated demonstrations without human heuristics, which can be reused to provide reasonable trading actions; 2) we design a two-layer control scheme: an RL selector that activates the right archetype per horizon, and a regret-aware adapter that performs high-impact, step-level refinements, which provides a profitable and robust trading strategy; 3) comprehensive experiments on 4 popular cryptocurrencies demonstrate that ArchetypeTrader significantly outperforms state-of-the-art baselines in both profit and risk management metrics.

Related Work

In this section, we briefly introduce the existing quantitative trading methods, which are based on traditional financial technical analysis or reinforcement learning algorithms.

Traditional Financial Methods

Classic technical analysis assumes that recurring price-volume patterns foreshadow future moves (Murphy 1999) and has spawned a vast family of handcrafted indicators (Kakushadze 2016). Examples range from order-flow imbalance (Chordia, Roll, and Subrahmanyam 2002) capturing short-term market direction to momentum measures such as MACD (Hung 2016; Krug, Dobaj, and Macher 2022) reflecting potential trends. In highly non-stationary markets like cryptocurrency, however, such signals often become noisy and misleading (Liu et al. 2020b; Qin et al. 2023; Li, Zheng, and Zheng 2019).

RL for Quantitative Trading

Early work ports off-the-shelf algorithms such as DQN (Mnih et al. 2015) and PPO (Schulman et al. 2017) to financial markets, while subsequent variants improve stability or representation power. For example, CDQNRP (Zhu and Zhu 2022) adds random perturbations to steady DQN training, and CLSTM-PPO (Zou et al. 2024) augments PPO with an LSTM state encoder for high-frequency stock trading. However, these stationary policies struggle in the regime-switching, high-volatility markets (e.g., crypto).

To capture longer-horizon dynamics, recent methods separate data with handcrafted trend/volatility indicators and apply hierarchical reinforcement learning (HRL) to achieve stable performance. EarnHFT (Qin et al. 2023) trains trend-specific agents and a router for agent selection in high-frequency cryptocurrency trading. MacroHFT (Zong et al. 2024) builds context-aware sub-policies and fuses them via a memory-augmented hyper-agent. Nevertheless, reliance on human labels biases behavior toward superficial "trend-following" and ignores latent opportunities, while the inability to leverage hindsight demonstrations often yields unstable, sub-optimal returns.

Problem Formulation

In this section, we first introduce the essential financial concepts and fundamental MDP formulation of cryptocurrency trading. We then position our task against current RL trading methods, pinpoint their core limitations, and motivate the pursuit of a more robust solution.

Financial Background & MDP Formulation

Market observations: an M-level limit-order book $b_t = \{(p_i^b, q_i^b), (p_i^a, q_i^a)\}_{i=1}^M$, the OHLCV bar $x_t = (p_t^o, p_t^h, p_t^l, p_t^c, v_t)$ and a set of technical indicators $y_t = \psi(x_{t-w+1:t}, b_{t-w+1:t})$ computed over backward window w.

Position: asset amount $P_t \in \{-m, 0, m\}$ (short, flat, long).

Execution Loss: executing position change $\Delta P_t = P_{t+1} - P_t$ incurs $O_t = C(|\Delta P_t|) - |\Delta P_t| p_t^{\text{mark}} + \delta |\Delta P_t| p_t^{\text{mark}}$, where $C(\cdot)$ is the LOB fill cost, p_t^{mark} is the mark price, and δ is the commission rate.

Net Value: the sum of cash and the market value of cryptocurrency, calculated as $V_t = V_{ct} + P_t \times p_t^{\text{mark}}$, where V_{ct} is the cash value. The purpose of quantitative trading is to maximize the final net value V_t after trading a single asset for a time period.

We formulate cryptocurrency trading as an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where state $s_t \in \mathcal{S}$ is the market observations, action $a_t \in \mathcal{A} = \{0, 1, 2\}$ sets the target position $P_t = m\,(a_t - 1)$, transition $\mathcal{T}$ follows the streamed market, reward $\mathcal{R}$ is measured by net value difference $r_t^{\text{step}} = V_t - V_{t-1} = P_t\,(p_{t+1}^{\text{mark}} - p_t^{\text{mark}}) - O_t$, and $\gamma \in [0, 1)$ is the discount factor for future returns. A trading policy $\pi(a|s)$ specifies target positions given states. The objective is to find the optimal policy π^* maximizing expected discounted return $J = E_\pi \left[\sum_{t=0}^{+\infty} \gamma^t r_t^{\text{step}}\right]$.

Problem Statement

Existing approaches (Qin et al. 2023; Zong et al. 2024) attempt to handle the non-stationary and high-volatility cryptocurrency market by assigning a high-level policy to select regime-specific sub-policies. However, they suffer from two hurdles: (i) human-engineered regime labels (e.g.,

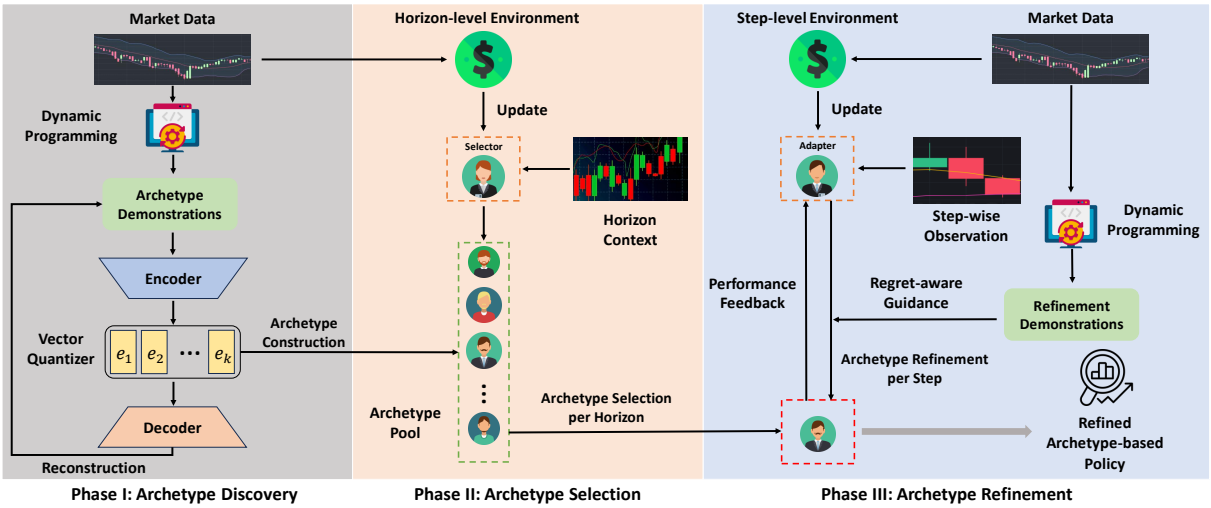

Figure 1: Phase I extracts archetypes via a VQ encoder-decoder, Phase II selects archetypes based on horizon context, and Phase III adapts the policy using a regret-aware mechanism with hindsight demonstrations.

bullish/bearish) oversimplify dynamics and bias agents toward rigid trend-following; (ii) demonstrations are either ignored or used in ways that inject noise—fully "optimal" DP traces or handcrafted rules. A systematic, noise-filtered way to exploit demonstrations without human intervention therefore remains an open problem.

ArchetypeTrader

To overcome the pitfalls of human-engineered segmentation and underutilized demonstrations, we propose ArchetypeTrader, a hierarchical framework with three training phases illustrated in Fig. 1: 1) In phase one, we design a DP planner to generate demonstration trajectories. A VQ-based encoder–decoder then compresses these trajectories into a discrete set of archetypes capturing key trading behaviors. 2) In phase two, a horizon-level RL agent selects the most suitable archetype whose expert insights best match current market features. 3) In phase three, a step-level policy adapter refines the archetype's base actions in response to real-time market observations and interim performance, ensuring robustness and agility under volatile conditions.

Archetype Discovery

We propose a self-supervised learning pipeline that extracts compact and reusable trading archetypes directly from high-quality trading trajectories, eliminating reliance on human design or heuristics. Concretely, we first sample n data chunks of fixed length h from the training dataset and apply a DP planner (Algorithm 1) to identify profitable trading actions for each chunk. Unlike prior approaches that attempt to capitalize on every profitable fluctuation (Qin et al. 2023; Zong et al. 2024), we deliberately limit each data chunk to a single trade to emphasize the most significant and impactful opportunities within each horizon. By capturing only the primary movements, our demonstration trajectories filter out small, short-lived fluctuations, thereby reducing noise, simplifying the subsequent learning, and providing a clean foundation for coherent and reusable trading archetypes.

Algorithm 1: Single-trade DP planner

Input: price series P of length N, action set $\mathcal{A}$
Output: $\{\hat{a}_t\}_0^{N-1}$

$V[N{+}1, |\mathcal{A}|, 2] \leftarrow 0,\ \Pi[N, |\mathcal{A}|, 2] \leftarrow -1$
for $t = N-1$ **downto** 0 **do**
  for $i \in \mathcal{A},\ c \in \{0,1\}$ **do**
    $\Pi[t,i,c] = \arg \max_{j \in \mathcal{A}\ :\ c+\mathbf{1}[i\neq j]\leq 1} \big(r_t(i \to j) + \gamma V[t+1, j, c+\mathbf{1}[i \neq j]]\big)$
    $V[t,i,c] \leftarrow r_t(i \to \Pi[t,i,c]) + \gamma V[t{+}1, \Pi[t,i,c], c+\mathbf{1}[i \neq \Pi[t,i,c]]]$
  end for
end for
$i \leftarrow 1\ , c \leftarrow 0$
for $t = 0$ **to** $N-2$ **do**
  $i' \leftarrow \Pi[t,i,c];\ \hat{a}_t \leftarrow \mathcal{A}[i']$
  $c \leftarrow \min\{1, c+\mathbf{1}[i \neq i']\};\ i \leftarrow i'$
end for
$\hat{a}_{N-1} \leftarrow \hat{a}_{N-2}$; **return** $\{\hat{a}_t\}_0^{N-1}$

For each sampled horizon represented by market observations $\mathbf{s}_{\text{demo}} = (s^{\text{demo}}_{0:h-1})$, the DP planner outputs a demonstration action sequence $\mathbf{a}_{\text{demo}} = (a^{\text{demo}}_{0:h-1})$, where $a^{\text{demo}} \in \{0,1,2\}$ denotes short, flat or long positions respectively. By construction, exactly one timestep $t \in [0, h-1]$ satisfies $a^{\text{demo}}_t - a^{\text{demo}}_{t-1} \neq 0$, guaranteeing a single trade per horizon. The resulting reward sequence is $\mathbf{r}_{\text{demo}} = (r^{\text{demo}}_{0:h-1})$ by executing the demonstration actions over the horizon. We collect the resulting demonstration trajectories into tuples $\tau = (\mathbf{s}_{\text{demo}}, \mathbf{a}_{\text{demo}}, \mathbf{r}_{\text{demo}})$ and compile them into a dataset $D = \{\tau_i\}_{i=0}^{n-1}$, which forms the training base for archetype extraction.

We now seek to distill the demonstration trajectories into

representative trading archetypes. Inspired by skill-based RL approaches in robotics (Pertsch, Lee, and Lim 2021), we adopt an encoder-decoder framework: the encoder ingests each demonstration chunk and projects it to a continuous latent vector, while the decoder reconstructs the original action sequences given the state inputs and the learned representation. However, a purely continuous representation can be challenging for subsequent RL-based trading, primarily for two reasons: 1) it expands the search space, making it harder for the RL selector in the second phase to effectively explore and select an optimal archetype from a continuous manifold; 2) it complicates strategy clustering, resulting in fragmented archetypes and inconsistent policy behavior. To address these limitations, we incorporate a VQ module (Van Den Oord, Vinyals et al. 2017) to learn more meaningful and generalizable archetype embeddings. By discretizing the encoder's continuous latent outputs into a finite codebook, each demonstration chunk is abstracted to one of a limited set of discrete codes, forming a concise and highly reusable set of archetypes.

In practice, each demonstration trajectory is passed through an LSTM-based encoder,

$$q_{\theta_e}(z_e|\mathbf{s}_{\text{demo}}, \mathbf{a}_{\text{demo}}, \mathbf{r}_{\text{demo}}) \tag{1}$$

which produces a continuous embedding z_e. Next, this embedding is quantized by selecting the nearest entry from a learnable codebook $\varepsilon = \{e_0, ...e_{K-1}\}$, i.e.,

$$k = \underset{0\le i\le K-1}{\operatorname{argmin}} \|z_e - e_i\|^2, \quad z_q = e_k \tag{2}$$

yielding a discrete latent representation z_q. Finally, a decoder

$$p_{\theta_d}(\hat{\mathbf{a}}_{\text{demo}}|\mathbf{s}_{\text{demo}}, z_q) \tag{3}$$

reconstructs the original actions $\mathbf{a}_{\text{demo}}$ from states $\mathbf{s}_{\text{demo}}$ and the quantized embedding z_q. We train this model by minimizing the combined loss function:

$$L = L_{rec} + \|\text{sg}[z_e] - z_q\|^2 + \beta_0 \|z_e - \text{sg}[z_q]\|^2 \tag{4}$$

where L_{rec} measures the reconstruction loss of demonstration actions, the rest terms enforce commitment to the chosen code and keep the codebook entries close to the encoder output, and $\text{sg}[\cdot]$ denotes the stop-gradient operation. By optimizing this loss, the codebook vectors $\{e_i\}$ learn to effectively capture reusable trading patterns, i.e., the archetypes, that effectively summarize the demonstration trajectories. In later phases, selecting an archetype based on current market observations and decoding its latent code into a concrete action sequence allows us to deploy these archetypes as coherent, data-driven trading strategies.

Archetype Selection

To effectively apply the learned strategic archetypes to trading, we train an RL-based archetype selector to choose the optimal archetype based on the current market observations and execute the micro-actions reconstructed by the previously trained decoder. Specifically, we lift the basic MDP defined in the previous section to a horizon-level MDP $\mathcal{M}_{\text{sel}} = \langle S_{\text{sel}}, A_{\text{sel}}, R_{\text{sel}}, \gamma \rangle$. For a fixed horizon $H = [t, t+h-1]$, state s^{sel} is defined as the state vector s_t defined in the previous section, captured at the first bar of the horizon. Action $a^{\text{sel}} \in \{0, 1, ..., K-1\}$ denotes the selected archetype from previously learned archetype set $\varepsilon = \{e_{0:k-1}\}$ via the selection policy $\pi_\phi^{\text{sel}}(a^{\text{sel}}|s^{\text{sel}})$. Once an archetype is chosen, its archetype code $e_{a^{\text{sel}}}$ is fed into the frozen decoder $p_{\theta_d}(a^{\text{base}}|s, e_{a^{\text{sel}}})$, which emits stepwise micro actions $(a^{\text{base}}_{t:t+h-1})$ based on the upcoming states $(s_{t:t+h-1})$ for the next h steps. We use the sum of stepwise reward over the whole archetype horizon H as the reward function for archetype selection, which is calculated as $r_t^{\text{sel}} = \sum_{\tau=t}^{t+h-1} r_\tau^{\text{step}}$. The policy π_ϕ is optimized to maximize the expected return while also encouraging consistency with demonstration trajectories. Concretely, we define the following objective:

$$J = E_{\pi_\phi^{\text{sel}}}\left[\sum_{t=0}^{\infty}(\gamma^t r_t^{\text{sel}} - \alpha KL(\hat{a_t^{\text{sel}}}||\pi_\phi^{\text{sel}}(a_t^{\text{sel}}|s_t^{\text{sel}})))\right] \tag{5}$$

where r_t^{sel} is the cumulative return obtained over the horizon, $\hat{a_t^{\text{sel}}}$ is the ground-truth archetype label assigned by the VQ encoder to the demonstration chunk of this horizon, and α is a hyperparameter balancing the environment reward against the KL-divergence penalty. This penalty encourages the selection policy to remain near the demonstrated archetype choices, yet still allows it to adapt as it gains experience in the live environment.

Archetype Refinement

A horizon-level choice of archetype already delivers a coherent trading plan. However, two key limitations arise when market conditions shift or the chosen archetype is suboptimal. First, a single decision at the beginning of the horizon cannot react to rapid intra-chunk changes, potentially missing short-lived trading opportunities or failing to perform timely stop-loss actions. Second, if the archetype selection policy chooses an ill-suited archetype at the horizon's start, the agent is mostly limited to suboptimal actions for the entire horizon. Yet, in many cases, on-the-fly performance monitoring can quickly reveal that the chosen archetype is misaligned with ongoing market dynamics. Without an effective adaptation method, the agent endures unnecessary losses or foregoes profits until the horizon ends. To handle these issues, we propose a step-level archetype refinement method by employing a policy adapter, which leverages ongoing market observations and partial archetype performance to fine-tune the archetype's base actions.

For each horizon H, we freeze the horizon-level selection policy π_ϕ^{sel} and decode the corresponding micro action sequence $\mathbf{a}_{\text{base}} = (a^{\text{base}}_{t:t+h-1})$ by the chosen archetype. The archetype refinement process is formulated as a step-level MDP $\mathcal{M}_{\text{ref}} = \{S_{\text{ref}}, A_{\text{ref}}, R_{\text{ref}}\}$. The archetype adaptation agent observes the real-time state s_τ^{ref} at time $\tau \in [t, t+h-1]$, which consists of two parts: step-wise market observations $s_\tau^{\text{ref1}} = s_\tau$ and archetype information $s_\tau^{\text{ref2}} = [e_{a_t^{\text{sel}}}, a_\tau^{\text{base}}, R_\tau^{\text{arche}}, \tau_{\text{remain}}]$, where $e_{a_t^{\text{sel}}}$ is the se-

lected archetype embedding, a_τ^{base} is the current micro action constructed by the archetype, $R_\tau^{\text{arche}} = \sum_{i=t}^{\tau} r_i^{\text{step}}$ is the cumulative reward under the archetype's base policy and $\tau_{\text{remain}} = t+h-\tau$ is the number of steps left in the horizon. The adapter generates a refinement signal $a_\tau^{\text{ref}} \in \{-1, 0, 1\}$ that shifts the archetype's base action a_τ^{base} while ensuring the original trades are never overridden:

$$a_\tau^{\text{final}} = \begin{cases} a_\tau^{\text{base}}, & \text{if } a_\tau^{\text{base}} \neq a_{\tau-1}^{\text{base}} \text{ or } a_\tau^{\text{ref}} = 0, \\ 0, & \text{if } a_\tau^{\text{ref}} = -1, \\ 2, & \text{if } a_\tau^{\text{ref}} = 1, \end{cases} \tag{6}$$

Executing a_τ^{final} yields the realized step-wise profit r_τ^{step}. The refinement policy $\pi_\omega^{\text{ref}}(a_\tau^{\text{ref}}|s_\tau^{\text{ref}})$ is trained to issue an override at most once within each horizon, preventing the adjustment from deviating excessively from the selected archetype's intended strategy. Formally, $a_\tau^{\text{ref}} \neq 0$ for a single $\tau \in [t, t+h-1]$ per horizon and $a_\tau^{\text{ref}} = 0$ for other timesteps. To enable the refinement policy to jointly consider real-time market signals and the archetype's evolving performance, we apply adaptive layer normalization (Peebles and Xie 2023; Zong et al. 2024) to condition market state s_τ^{ref1} on archetype context s_τ^{ref2} in practice.

However, because only a single-step adaptation is allowed per horizon, the policy must carefully identify the most impactful moment to intervene rather than squandering its opportunity on minor market fluctuations or prematurely overriding an archetype strategy that may yield greater returns later. To focus the refinement policy on meaningful adjustments, we compute the top-5 hindsight-optimal adaptations via DP and introduce a novel regret-aware reward function that penalizes the agent not only for failing to outperform the base strategy but also for missing out on the best possible adaptation. Concretely, the DP solver returns:

$$O_{\text{top5}} = \{(\tau_{\text{opt}}^n, a_{\text{opt}}^n, R_{\text{opt}}^n)\}_{n=1}^5 \tag{7}$$

where τ_{opt}^n is the adaptation timestep, $a_{\text{opt}}^n \in \{-1, 1\}$ is the adaptation action, and R_{opt}^n is the horizon's cumulative return of executing the adaptation. Instead of directly using the immediate step-wise reward r_τ^{step}, our regret-augmented reward function for the policy adaptation agent is formulated as follows:

$$r_\tau^{\text{ref}} = \begin{cases} (R - R_{\text{base}}) \ + \ \beta_1(R - R_{\text{opt}}^1) & \text{if } a_\tau^{\text{ref}} \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where $R = \sum_{\tau=0}^{h-1} r_\tau^{\text{step}}$ is the horizon's cumulative return of taking action a_τ^{ref}, R_{base} is the horizon's return under the base archetype policy, R_{opt}^1 is the horizon's maximum return from the top DP-identified adaptation and β_1 is the hyper-parameter controlling the tolerance for suboptimality. This reward structure encourages profitable adjustments relative to the base strategy while penalizing the gap from the best possible adaptation. By striking this balance, the policy avoids wasting its single adaptation opportunity on negligible improvements and instead aims to deploy its intervention at the most valuable moment. Because only one adaptation is allowed per horizon, the RL episode terminates as soon as the adapter chooses a non-zero action. Finally, the objective function for training the refinement policy is given by

$$J' = E_{\pi_\omega^{\text{ref}}}\left[\sum_{\tau=0}^{h-1}(\gamma^\tau r_\tau^{\text{ref}} - \beta_2 L(\hat{a_\tau^{\text{ref}}}, \pi_\omega^{\text{ref}}(a_\tau^{\text{ref}}|s_\tau^{\text{ref}})))\right] \tag{9}$$

where $\hat{a_\tau^{\text{ref}}}$ denotes the optimal adaptation action ($\hat{a_\tau^{\text{ref}}} = a_{\text{opt}}^n$ if $\tau = \tau_{\text{opt}}^n$, and 0 otherwise), and L is the cross-entropy loss guiding the refinement policy toward optimal behavior. By optimizing this objective, we encourage the policy adapter to perform high-impact adjustments to the base archetype actions, ultimately yielding a more profitable and robust trading strategy.

Experiments

Datasets

To evaluate the effectiveness of our method, we use 10-minute data with orderbook depth $M = 25$ of BTC/ETH/-DOT/BNB against USDT. The dataset spans from 2021-06-01 to 2023-05-31 for training, 2023-06-01 to 2023-12-31 for validation, and 2024-01-01 to 2024-09-01 for testing.

Evaluation Metrics

We evaluate our proposed method on 6 different financial metrics, including one profit criterion, two risk criteria, and three risk-adjusted profit criteria listed below.

- **Total Return (TR)** is the overall return rate of the entire trading period, which is defined as $TR = \frac{V_T - V_1}{V_1}$, where V_T is the final net value and V_1 is the initial net value.
- **Annual Volatility (AVOL)** is the variation of return over one year measured as $\sigma[r] \times \sqrt{m}$, where $r = [r_1, r_2, ..., r_T]$ is return, $\sigma[\cdot]$ is standard deviation, $m = 52560$ is the number of timesteps.
- **Maximum Drawdown (MDD)** measures the largest loss from any peak to show the worst case.
- **Annual Sharpe Ratio (ASR)** measures the amount of extra return a trader gets per unit of increased risk, calculated as $ASR = E[r]/\sigma[r] \times \sqrt{m}$.
- **Annual Calmar Ratio (ACR)** measures the risk-adjusted return calculated as $ACR = \frac{E[r]}{MDD} \times m$.
- **Annual Sortino Ratio (ASoR)** measures risk with downside deviation (DD): $SoR = \frac{E[r]}{DD} \times \sqrt{m}$.

Baselines

To benchmark our method, we select 8 baselines including standard RL (**DQN** (Mnih et al. 2015), **PPO** (Schulman et al. 2017), **CDQNRP** (Zhu and Zhu 2022), **CLSTM-PPO** (Zou et al. 2024)), hierarchical RL (**EarnHFT** (Qin et al. 2023), **MacroHFT** (Zong et al. 2024)), and rule-based strategies (**IV** (Chordia, Roll, and Subrahmanyam 2002), **MACD** (Krug, Dobaj, and Macher 2022)).

		Profit	Risk-Adjusted Profit			Risk Metrics				Profit	Risk-Adjusted Profit			Risk Metrics	
Market	Model	TR(%)↑	ASR↑	ACR↑	ASoR↑	AVOL(%)↓	MDD(%)↓	Market	Model	TR(%)↑	ASR↑	ACR↑	ASoR↑	AVOL(%)↓	MDD(%)↓
BTC	CLSTM-PPO	–40.10	–1.81	–1.42	–2.35	38.29	49.02	DOT	CLSTM-PPO	–53.21	–1.23	–1.26	–1.84	71.35	69.66
	PPO	16.38	0.64	0.96	1.05	51.18	33.97		PPO	-51.82	-1.13	-1.18	-1.73	73.08	69.66
	CDQNRP	15.13	0.63	0.95	0.90	52.93	35.01		CDQNRP	30.54	1.28	1.43	1.85	35.61	31.81
	DQN	14.91	0.60	0.90	0.97	52.45	35.01		DQN	–53.97	–1.25	–1.29	–1.90	71.98	69.84
	MACD	–15.96	–0.38	–0.55	–0.63	38.81	26.34		MACD	–51.06	–1.87	–1.58	–2.62	48.57	57.49
	IV	–15.44	–0.35	–0.39	–0.55	46.27	41.44		IV	-40.59	–1.51	–1.49	–2.02	42.99	43.59
	EarnHFT	24.56	1.00	1.52	1.58	39.28	25.72		EarnHFT	16.95	1.29	1.38	1.89	19.49	18.21
	MacroHFT	11.84	0.58	0.71	0.92	41.32	34.39		MacroHFT	10.15	0.57	0.54	0.72	51.58	56.72
	ArchetypeTrader	76.25	2.66	3.76	4.48	33.57	23.75		ArchetypeTrader	147.48	3.19	8.72	6.08	44.41	16.26
ETH	CLSTM-PPO	4.87	0.39	0.49	0.53	39.44	31.26	BNB	CLSTM-PPO	-14.77	-0.42	-0.42	-0.54	35.98	36.27
	PPO	–10.94	0.02	0.02	0.03	63.85	49.25		PPO	-3.91	0.08	0.09	0.10	37.96	31.62
	CDQNRP	–10.74	0.07	0.10	0.09	65.56	49.25		CDQNRP	-18.66	-0.63	-0.59	-0.92	36.06	38.20
	DQN	–28.14	–0.60	–0.64	–0.92	57.21	53.68		DQN	-24.94	-5.99	-1.35	-1.66	6.89	25.19
	MACD	27.45	1.13	2.53	1.94	39.80	17.80		MACD	59.81	1.80	3.38	3.19	44.46	23.60
	IV	5.39	0.36	0.35	0.52	38.94	39.47		IV	88.32	1.95	3.96	3.17	53.15	26.24
	EarnHFT	–0.92	0.16	0.20	0.25	44.90	36.19		EarnHFT	–0.40	–0.33	–0.37	–0.16	1.77	1.60
	MacroHFT	22.30	0.75	0.89	1.13	58.11	48.77		MacroHFT	36.13	1.00	1.32	1.46	58.80	44.35
	ArchetypeTrader	41.93	1.37	1.86	2.26	43.61	32.13		ArchetypeTrader	39.24	1.34	3.53	2.41	42.31	16.03

Table 1: Performance comparison on 4 crypto markets with 8 baselines. Best/2nd/3rd highlighted in pink/green/blue.

Experiment Setup

Experiments are conducted on 4 RTX-4090 GPUs. All trades incur a commission of $\delta = 0.02\%$, and the max position m varies by asset (BTC 8, ETH 100, DOT 2500, BNB 200). For archetype discovery, we sample 30k DP trajectories (horizon length $h = 72$) and train a 128-unit VQ encoder–decoder (archetype vector dimension=16, number of archetypes $K = 10$, $\beta_0 = 0.25$) for 100 epochs. Specifically, archetype dimension is set smaller than network dimension to create information bottleneck, forcing the VQ module to distill high-level trading strategies. For archetype selection, an RL selector with $\alpha = 1$ is optimized for 3M steps, and the checkpoint with the best validation performance is retained. For archetype refinement, the regret-aware adapter is trained for 1M steps with $\beta_2 = 1$ and asset-specific β_1 tuned over $\{0.3, 0.5, 0.7\}$ (0.5 for BTC/-DOT, 0.7 for ETH/BNB) via validation. We emphasize that all DP modules are applied solely for training and disabled during inference to avoid future information leakage.

Results and Analysis

Table 1 summarizes the performance of ArchetypeTrader alongside baseline methods across four cryptocurrency markets. Overall, ArchetypeTrader attains the highest profit and best risk-adjusted returns in all cryptocurrency markets except BNBUSDT. Even in the predominantly bullish BNB market, where ArchetypeTrader does not fully exploit the upward trend, our method still achieves competitive profit and risk control capacity. By contrast, RL approaches that rely on uniform policies (DQN, CDQNRP, PPO, CLSTM-PPO) struggle to adapt to shifting market regimes, leading to inconsistent profit generation. Meanwhile, rule-based methods (IV, MACD) perform on specific datasets yet suffer catastrophic losses on others, indicating a lack of robustness. Although hierarchical RL methods (EarnHFT, MacroHFT) demonstrate relatively stable returns by segmenting markets into sub-policies, these sub-policies remain biased toward certain market dynamics by human-crafted indicators and fail to capture more nuanced, profitable opportunities. ArchetypeTrader avoids hand-crafted segmentation by discovering and refining archetypes directly from the effective use of demonstration data, resulting in a broader and more flexible set of trading behaviors that ultimately yield better risk management and higher overall profitability.

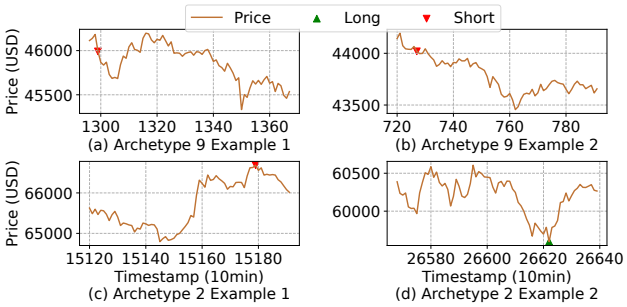

Figure 2: Trading behaviors of different archetypes

Interpretability Analysis of Archetypes

To illustrate that our data-driven archetypes capture meaningful expert behaviors beyond standard hand-crafted segmentation, we visualize several trading signals prescribed by the selected archetypes on BTCUSDT, as shown in Figure 2. For Archetype 9 (Figure 2(a, b)), examples 1 and 2 reveal a "short-and-hold" strategy suitable for bearish market horizons: the archetype short-sells near the horizon start, anticipating a downward trend. Thus, when the market is poised for further decline, the selection policy would probably pick this archetype. From examples 1 and 2 of Archetype 2 (Figure 2(c, d)), it is apparent that the archetype is geared toward shorting near local peaks and going long near local troughs, aiming to capture subsequent price reversals. By choosing this archetype, the resulting policy's core behavior is iden-

tifying short-term extremes and trading against them, which is a typical counter-trend or mean-reversion style. Crucially, although Archetype 2 often exhibits these short-term extremes, it does not merely repeat identical positions. Instead, it extracts a high-level similarity of trading against the current price movement, leading to different specific trades but a consistent mean-reversion effect. These examples show how our learned archetypes retain intuitive and expert-like trading behaviors rather than following rigid trends.

	BTCUSDT			ETHUSDT		
Model	TR(%)↑	ASR↑	MDD(%)↓	TR(%)↑	ASR↑	MDD(%)↓
Continuous	-33.42	-1.73	40.47	-30.88	-1.07	49.14
Clustering	47.84	1.75	15.87	-28.50	-0.94	49.57
VQ	57.47	2.02	19.91	12.42	0.60	29.91
	DOTUSDT			BNBUSDT		
Model	TR(%)↑	ASR↑	MDD(%)↓	TR(%)↑	ASR↑	MDD(%)↓
Continuous	22.09	0.96	30.89	20.47	0.91	19.81
Clustering	36.84	1.21	30.30	-14.31	-0.72	23.74
VQ	105.65	2.77	16.78	21.56	0.89	20.20

Table 2: Performance of models with different archetype embeddings. Underlined results are best.

	BTCUSDT			ETHUSDT		
Model	TR(%)↑	ASR↑	MDD(%)↓	TR(%)↑	ASR↑	MDD(%)↓
w/o-Ref	57.47	2.02	19.91	12.42	0.60	29.91
w/o-Reg	30.24	1.10	22.40	40.60	1.31	41.56
Original	76.25	2.66	23.75	41.93	1.37	32.13
	DOTUSDT			BNBUSDT		
Model	TR(%)↑	ASR↑	MDD(%)↓	TR(%)↑	ASR↑	MDD(%)↓
w/o-Ref	105.65	2.77	16.78	21.56	0.89	20.20
w/o-Reg	97.55	2.34	20.52	-4.29	0.07	23.92
Original	147.48	3.19	16.26	39.24	1.34	16.03

Table 3: Performance of models with different refinement modules. Underlined results represent the best.

Ablation Study

To assess the effectiveness of our framework's core components, we conduct various ablation experiments examining: vector quantization (VQ) for archetype embeddings, archetype refinement (Ref) via a policy adapter, and regret regularization (Reg) in the refinement phase. In Table 2, we compare three archetype embedding methods: continuous embeddings, clustered embeddings (k-means and k=10), and our vector-quantized discrete embeddings under the no-refinement setting for a fair comparison. The results show that discrete VQ embeddings consistently yield higher returns and better risk management metrics, highlighting the advantages of learnable discrete archetypes that the selector can easily interpret and switch among. Table 3 and Figure 3 present the performance for three refinement approaches: no refinement, refinement without regret, and refinement with regret (our original model). It can be observed that adding refinement via a policy adapter captures extra trading opportunities and corrects suboptimal archetype executions, leading to higher profit and better risk control. Moreover, incorporating a hindsight regret penalty ensures that refinements occur only at the most valuable moments while removing this term can even cause deficits compared with the archetype selector's base policy, indicating that regret guidance is crucial for meaningful intra-horizon interventions.

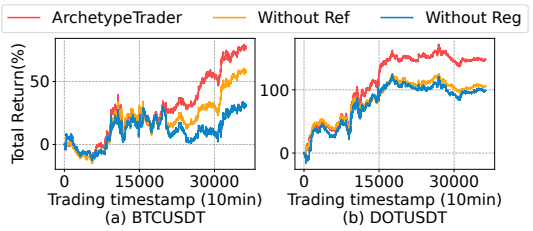

Figure 3: Performance of ArchetypeTrader and two variations of different refinement methods

Hyper-parameter Sensitivity

We evaluate how some hyper-parameters shape performance on DOTUSDT dataset (Table 4). A small codebook of archetypes under-represents the diverse trading motifs found in the demonstrations; a large one overloads the selector and adapter. Similarly, horizons that are too short trigger frequent, noise-driven shifts, whereas very long horizons postpone updates and blunt responsiveness. Intermediate values for both K and h give the best trade-off between expressiveness and agility.

Fixed $K = 10$, vary h		**Fixed $h = 72$, vary K**	
h	TR(%)↑	K	TR(%)↑
36	55.43	5	39.26
72	147.48	10	147.48
144	73.69	20	28.68

Table 4: Hyperparameter sensitivity: horizon length h and archetype number K. Underlined results represent the best.

Conclusion

In this paper, We present ArchetypeTrader, a novel RL framework that discovers, selects, and refines strategic archetypes for automated cryptocurrency trading. Firstly, ArchetypeTrader discovers representative and reusable strategic archetypes from DP–generated demonstrations via a vector-quantized encoder-decoder. A horizon-level RL agent then selects the best archetype for each chunk based on market context, while a step-level adapter refines archetype actions within each horizon through regret-aware updates. Comprehensive experiments on four major cryptocurrency pairs demonstrate that ArchetypeTrader consistently outperforms multiple state-of-the-art trading algorithms in profit-making while maintaining outstanding risk management ability in cryptocurrency trading.

Acknowledgments

This research is supported by the Joint NTU-WeBank Research Centre on Fintech, Nanyang Technological University, Singapore.

References

Briola, A.; Turiel, J.; Marcaccioli, R.; Cauderan, A.; and Aste, T. 2021. Deep reinforcement learning for active high frequency trading. *arXiv preprint arXiv:2101.07107*.

Chordia, T.; Roll, R.; and Subrahmanyam, A. 2002. Order imbalance, liquidity, and market returns. *Journal of Financial Economics*, 65(1): 111–130.

Deng, Y.; Bao, F.; Kong, Y.; Ren, Z.; and Dai, Q. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3): 653–664.

Hung, N. H. 2016. Various moving average convergence divergence trading strategies: A comparison. *Investment Management and Financial Innovations*, (13, Iss. 2): 363–369.

Jia, W.; Chen, W.; Xiong, L.; and Hongyong, S. 2019. Quantitative trading on stock market based on deep reinforcement learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Kakushadze, Z. 2016. 101 formulaic alphas. *Wilmott*, 2016(84): 72–81.

Krug, T.; Dobaj, J.; and Macher, G. 2022. Enforcing network safety-margins in industrial process control using MACD indicators. In *European Conference on Software Process Improvement*, 401–413. Springer.

Li, Y.; Zheng, W.; and Zheng, Z. 2019. Deep robust reinforcement learning for practical algorithmic trading. *IEEE Access*, 7: 108014–108022.

Liu, X.-Y.; Yang, H.; Chen, Q.; Zhang, R.; Yang, L.; Xiao, B.; and Wang, C. D. 2020a. FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*.

Liu, Y.; Liu, Q.; Zhao, H.; Pan, Z.; and Liu, C. 2020b. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2128–2135.

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.

Murphy, J. J. 1999. *Technical Analysis of the Futures Markets: A Comprehensive Guide to Trading Methods and Applications, New York Institute of Finance*. Prentice-Hall.

Peebles, W.; and Xie, S. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4195–4205.

Pertsch, K.; Lee, Y.; and Lim, J. 2021. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, 188–204. PMLR.

Qin, M.; Sun, S.; Zhang, W.; Xia, H.; Wang, X.; and An, B. 2023. Earnhft: Efficient hierarchical reinforcement learning for high frequency trading. *arXiv preprint arXiv:2309.12891*.

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

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30.

Zhang, Z.; Zohren, S.; and Stephen, R. 2020. Deep reinforcement learning for trading. *The Journal of Financial Data Science*.

Zhu, T.; and Zhu, W. 2022. Quantitative trading through random perturbation Q-network with nonlinear transaction costs. *Stats*, 5(2): 546–560.

Zong, C.; Wang, C.; Qin, M.; Feng, L.; Wang, X.; and An, B. 2024. MacroHFT: Memory augmented context-aware reinforcement learning on high frequency trading. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4712–4721.

Zou, J.; Lou, J.; Wang, B.; and Liu, S. 2024. A novel deep reinforcement learning based automated stock trading system using cascaded lstm networks. *Expert Systems with Applications*, 242: 122801.