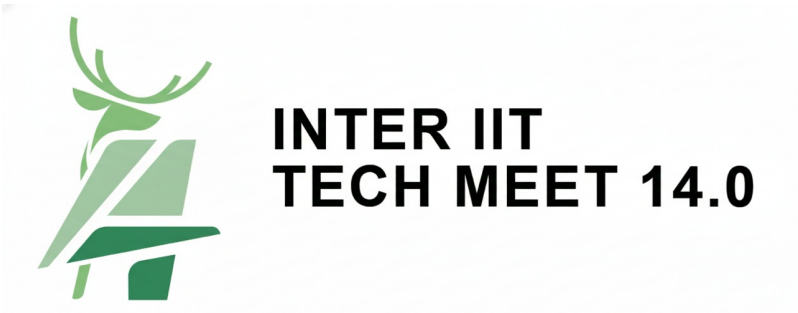
Algorithmic Strategy Development on Multi-Feature Time Series

INTER IIT TECH MEET 14.0

Idea Summary

Team 33



Contents

1 Introduction

In modern financial markets, speed and precision determine profitability. This project develops a systematic intraday trading framework using **Reinforcement Learning** to optimize sequential decision-making under real-world constraints. Rather than predicting prices, the strategy employs **Proximal Policy Optimization (PPO)** to learn an optimal policy that maximizes cumulative risk-adjusted returns. The approach integrates extensive feature engineering—including Heikin-Ashi smoothing, KAMA filtering, momentum oscillators (RSI, CMO, CCI), volatility measures (ATR), and trend identification systems—to construct a multi-scale state representation capturing price dynamics, volatility regimes, and momentum signals across varying market conditions.

2 Problem Formulation

The objective is to develop a systematic intraday trading strategy for two anonymized datasets, **EBX** and **EBY**, that identifies profitable patterns, closes all trades daily, adapts consistently across datasets, and maximizes risk-adjusted returns under realistic constraints including transaction costs and position limits.

The strategy treats trading as a decision-making problem where an agent chooses actions (**Hold**, **Buy**, or **Sell**) based on market observations. These observations include recent price history, technical indicators measuring momentum and volatility, trend signals, time-of-day patterns, and the agent's current position and profit. The agent learns through trial and error, receiving rewards for profitable trades and penalties for losses, excessive trading, poor risk management, thereby developing behaviors that balance profit generation with risk control.

3 Experimented Approaches

3.1 Approach 1: Trend Classification with Heikin Ashi & MA Ribbon

Our initial strategy focused on robust trend identification by integrating two key techniques: **Heikin Ashi (HA) smoothing** and a **Johnny's MA Ribbon.**

The process began with an iterative HA transformation to filter market noise. This is achieved by recalculating the candle values, where the HA-Close is the average of the current bar's OHLC ($HA_{Close} = \frac{O+H+L+C}{4}$), and the HA-Open is the midpoint of the previous HA-candle's body ($HA_{Open} = \frac{\text{Prev. } HA_{Open}+\text{Prev. } HA_{Close}}{2}$). This averaging mechanism creates a smoother price series that highlights the underlying trend.

Following this smoothing, the Moving Averages Ribbon - a set of Moving Averages, MA_i, with increasing periods p_i (where $p_1 < p_2 < \cdots < p_n$) - was used to classify the market into four distinct phases based on its configuration relative to a reference moving average, MA_{ref}:

- **Strong Uptrend:** MAs are perfectly fanned out in ascending order, all above the reference line (MA_{ref}).
- **Uptrend Pullback:** MAs remain above MA_{ref} but lose their strict ascending order, signaling consolidation.
- **Strong Downtrend:** MAs are perfectly fanned out in descending order, all below the reference line.
- **Downtrend Bounce:** MAs remain below MA_{ref}, losing their descending order, signaling short-term rally.

While this rule-based classification produced a visually clean representation of momentum as seen in Figure 1, its primary weakness was a decisive failure in **choppy, non-trending markets.**

3.2 Approach 2: Adaptive Trend Filtering with KAMA and CHOP Index

Our second approach introduced adaptive filters to distinguish trending from choppy markets, directly addressing the prior method's failure. This strategy was centered on combining two distinct indicators: the **Choppiness Index (CHOP)** for direct regime identification and the **Kaufman Adaptive Moving Average (KAMA)** for adaptive smoothing.

- **Choppiness Index (CHOP):** First, we used the CHOP index as a direct, normalized measure of choppiness. It compares the Average True Range (ATR) over a period n to the total price range, with high values indicating a sideways market and low values indicating a strong trend. The formula is:

$$\text{CHOP} = 100 \cdot \frac{\log_{10}\left(\sum_{i=t-n}^{t} \text{ATR}_i\right) - \log_{10}\left(\max(H_t, \ldots, H_{t-n}) - \min(L_t, \ldots, L_{t-n})\right)}{\log_{10}(n)}$$

  Trades were only considered when **CHOP** $<$ **43.2** indicating a trending market.

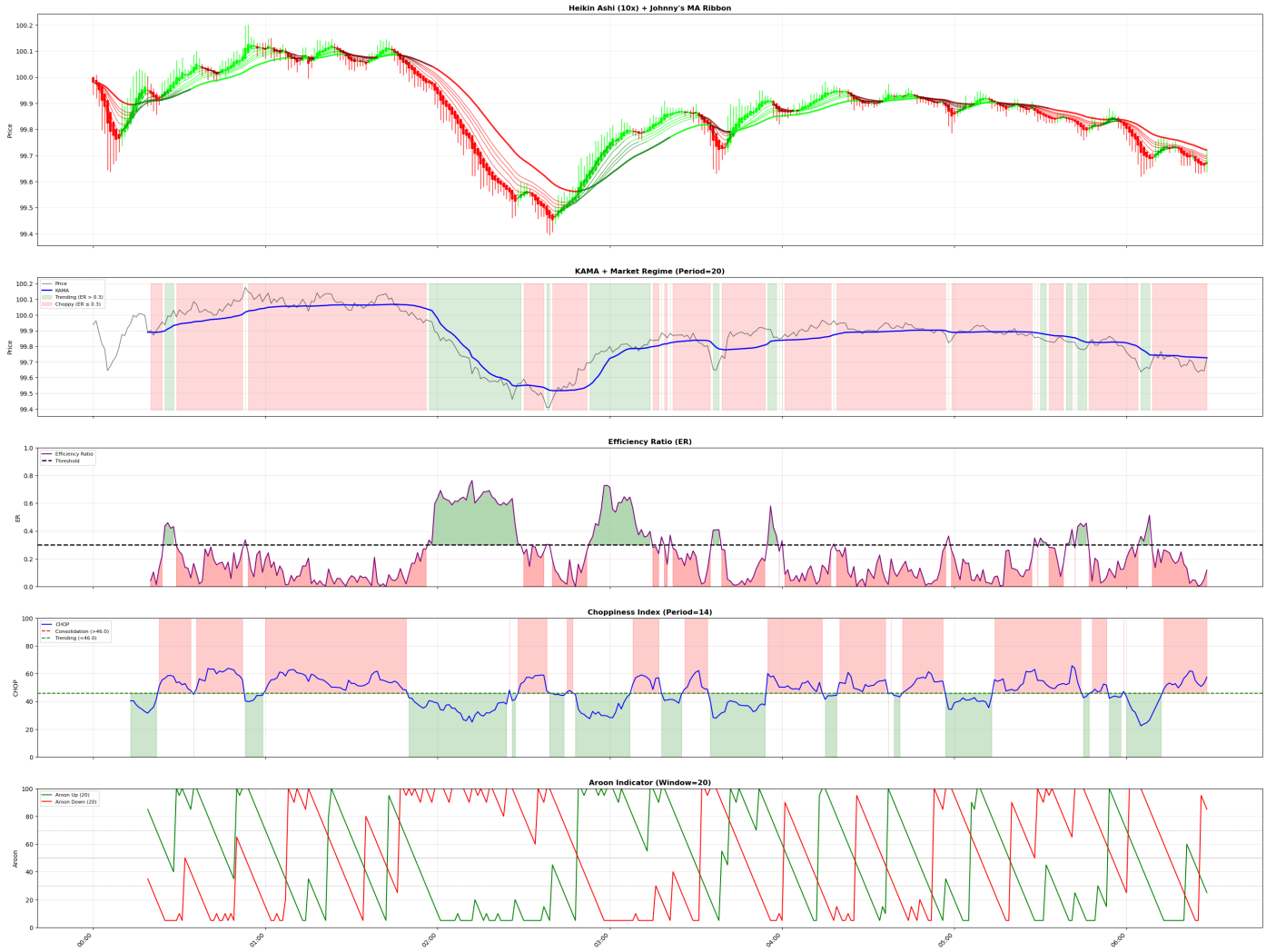


Figure 1: This figure uses five synchronized charts to define the market state: Heiken-Ashi Candles with MA Ribbons, KAMA trend line, the Efficiency Ratio (ER), the Choppiness Index (CHOP), and the Aroon Indicator.

- **Kaufman Adaptive Moving Average (KAMA):** Within trending regimes identified by CHOP, we used KAMA to generate signals. KAMA's speed is governed by the **Efficiency Ratio (ER)**, which is high (> 0.7) in trends and low (< 0.3) in chop. The regimes are visualized in the Figure 1.

$$ER = \frac{|\text{Close}_t - \text{Close}_{t-n}|}{\sum_{i=t-n}^{t} |\text{Close}_i - \text{Close}_{i-1}|} \quad , \quad SC_t = [ER_t \cdot (SC_{\text{fast}} - SC_{\text{slow}}) + SC_{\text{slow}}]^2$$

  The ER is used to calculate a dynamic smoothing constant (SC), which in turn is used to compute the KAMA recursively:

$$KAMA_t = KAMA_{t-1} + SC_t \cdot (\text{Price}_t - KAMA_{t-1})$$

  Buy/sell signals were triggered when the KAMA's slope exceeded fixed thresholds ($\theta_{buy}, \theta_{sell}$).

Despite the addition of the CHOP filter, the strategy's reliance on **fixed slope thresholds** for KAMA remained a critical flaw. A low-volatility trend might be correctly identified by CHOP but still fail to trigger a KAMA signal. Although this combined approach failed as a standalone strategy, it validated the **CHOP Index, ER, and KAMA** as powerful feature engineering components, which were retained for the final model.

Figure 2: The figure shows the regime predictions, synchronized with the Regime Probability distribution (bottom), which identifies high-confidence trending (green) and consolidating/reversing (red) market environments.

3.3 Approach 3: Variational Auto-Encoder (VAE)

Building upon our Mid-Term submission, that strategy has been significantly refined into an end-to-end probabilistic machine learning pipeline. The model is now trained on a richer feature set of nearly 200 indicators including the features discussed in the previous sections such as Heiken-ashi, KAMA and CHOP derived from one-minute candles, along with additional indicators like Aroon up/down. This methodology ensures the model learns to identify only statistically significant directional movements.

To manage this high-dimensional data, we now employ a hybrid VAE-XGBoost architecture. A Variational Autoencoder (VAE) compresses the feature set into a dense, 64-dimensional latent vector, acting as a powerful non-linear feature extractor. The VAE's frozen encoder transforms features into these latent vectors, which are then fed into an XGBoost classifier to output the final probability distribution for the three market states – **Bull**, **Bear** and **Sideways**.

The execution layer translates these probabilities into risk-managed trades. It uses the **Efficiency Ratio (ER)** as a regime filter to permit entries only during trending conditions. A key refinement is **confidence-based position sizing:** high-probability signals ($\geq 90\%$) receive maximum allocation (100 units), while lower-confidence signals are scaled down (40 and 30 units). The Figure 2 shows the predicted regimes and their corresponding confidence. The predictions initially start with low confidence and gradually becomes confident as we get trend confirmation.

4 Final Strategy

Intraday trading is modeled as a sequential decision-making problem in which an agent interacts with a market environment and learns a policy that maximizes expected cumulative reward. The setup is represented as a **Markov Decision Process (MDP)** defined by:

- **State (Observation)**: At each timestep, the agent receives a fixed-length feature representation constructed from (i) a short rolling lookback window of market/technical features and (ii) current-step features, augmented with the agent's internal state such as current position (flat/long/short) and mark-to-market (unrealized) PnL, discussed in detail in section 4.3.
- **Action Space**: A compact discrete action space consisting of three actions: **Hold**, **Buy**, and **Sell**, which maps directly to trading decisions (maintain position, enter long/exit short, enter short/exit long).
- **Transition Dynamics**: At each step, open positions are updated based on price evolution and enforces exit conditions such as stop-loss, trailing-stop logic, and end-of-day liquidation.

- **Reward**: The reward function is shaped to align learning with risk-managed profitability. It incorporates rewards for profitable trades and penalties for undesirable behavior like excessive trading and adverse exits.

4.1 Rationale for Our Current Architecture

- **Reinforcement Learning** is selected because the trading objective is not a single-step prediction task but the optimization of a sequence of decisions under constraints. Trading performance is path-dependent: entry timing, holding behavior, and exit timing interact with transaction costs and risk controls (stop-loss, trailing exits, end-of-day liquidation). Reinforcement Learning incorporates these mechanics naturally through reward function, enabling direct optimization of cumulative trading reward instead of optimizing prediction losses.
- **Proximal Policy Optimization (PPO)** is selected for policy optimization due to its stability. PPO utilizes parallel rollout collection, where multiple workers simultaneously gather diverse trajectories (state, action, reward) from different experiences. This strategy ensures the resulting training batch is diversified and less correlated, yielding a more robust gradient that is less biased by recent, specific events, thus reliably guiding the policy toward a global optimum.
- **An MLP policy** is used as it is well matched to tabular/engineered inputs and can learn nonlinear feature interactions with a simple neural network without using unnecessarily complex architectures such as RNNs/Transformers.
- The **Actor–Critic** architecture is used as it improves the stability and supports smoother convergence by training two distinct networks: the **Actor**, which determines the policy $\pi(a \mid s)$, and the **Critic**, which estimates the state value $V(s)$. This design enables advantage-based updates where the Critic's estimate $V(s)$, transforms raw rewards R_t into the Advantage Function $A(s, a) = R_t - V(s)$. This allows the Actor to receive refined feedback on whether its chosen action was better or worse than expected.

4.2 RL Hyperparameters and Parameterization

Hyperparameters are selected to balance exploration, stability, and learning speed in an intraday environment.

4.2.1 Hyperparameter Tuning for Training Stability

Policy and Value Function Architecture: In this setup, separate feed-forward networks are used for both the policy (actor) and value function (critic), each with two hidden layers. This simple architecture favors faster training and smoother convergence.

Rollout Collection and Optimization: The rollout collection process involves using parallel environments to collect multiple trajectories at the same time. This parallelization increases the throughput of data collection and stabilizes optimization by introducing batch diversity. The rollout horizon, denoted by n_{steps}, is chosen to be long enough to capture significant market structure in each policy update while allowing frequent updates on-policy.

The batch size and number of epochs (denoted by batch_size and n_{epochs}) are selected to ensure stable gradient updates. Larger batch sizes improve the stability of gradient estimation, while multiple epochs help improve the efficiency of the samples. The Proximal Policy Optimization (PPO) algorithm includes clipping to avoid destructive over-updates on the same data.

Discounting and Advantage Estimation: The discount factor (γ) controls how much the agent values future rewards compared to immediate ones. Setting it close to 1 makes the agent patient and focused on long-term profitability, which is important since intraday trading involves sequences of decisions that affect final outcomes. A value closer to 0 would make the agent shortsighted, caring only about immediate gains. The **Generalized Advantage Estimation (GAE)** parameter (λ_{gae}) helps balance accuracy versus stability when evaluating how good each action was. This is especially useful when trading rewards are noisy and unpredictable, as it smooths out the learning process and prevents erratic behavior.

4.2.2 Optimization Techniques and Learning Rate Management

Normalization and Reproducibility: During training, both observation and reward normalization are applied to stabilize gradient scales and reduce sensitivity to differences in feature and reward magnitudes. This helps maintain a consistent learning process across varied market conditions. Additionally, deterministic seeding is employed across the training stack, ensuring that experiments can be reproduced with the same setup for consistency and debugging.

Learning Rate Decay: Instead of using a constant learning rate, a decayed learning rate is used to improve convergence. During the early stages of training, a larger step size helps the model discover profitable behaviors faster and explore a broader policy space. As training progresses, the step size is gradually reduced to allow for fine-tuning, reducing oscillations caused by noisy gradients and improving the robustness of the final policy. The learning rate is updated according to a linear schedule, where the learning rate at any training step t is given by:

$$\mathrm{lr}(t) = \frac{\mathrm{lr}_{\max} - \mathrm{lr}_{\min}}{\mathrm{T}} \cdot \mathrm{t}$$

where $\mathrm{lr}_{\min} = 10^{-4}$, $\mathrm{lr}_{\max} = 5 \times 10^{-3}$, and $\mathrm{T} = 1,000,000$ is the total number of training steps. A minimum learning rate floor ensures that updates do not become too small to be effective in later stages of training.

4.3 State Space

The state space serves as the agent's perceptual interface to the market environment, and its design fundamentally constrains the complexity of policies that can be learned and the efficiency with which learning occurs. In the context of algorithmic trading, the state space must capture market microstructure including price dynamics, volatility characteristics, momentum signals, and broader regime indicators. The state space is constructed as follows:

$$\mathbf{s}^{(t)} = \left[\mathbf{f}^{(t)}_{\text{lookback}}, \mathbf{f}^{(t)}_{\text{scalar}}, \mathbf{f}^{(t)}_{\text{agent}}\right]$$

where lookback features maintain a window of $w = 10$ historical timesteps, scalar features represent current values only, and agent features encode the current trading state. We organize our features into six distinct regime categories, each designed to capture specific aspects of market microstructure and dynamics.

1. **Price Action and Agent Features:** These features capture raw price movements and candle structure, providing the foundation for understanding immediate market dynamics. We compute logarithmic returns of the previous candle's Open, High, Low, and Close prices. We also included the agent's internal state, namely the position $\in \{1, 0, -1\}$ and mark-to-market PnL to make position-aware decisions.

$$r^{(t)} = \ln\left(\frac{P^{(t-1)}}{P^{(t-2)}}\right), \text{ where } P \text{ is the price}$$

2. **Momentum and Oscillator Features:** Momentum indicators such as RSI, CMO and CCI calculated over the lookback periods n (candles) $\in \{3, 5, 10, 20\}$, measure the velocity of price changes across multiple time-frames, helping identify overbought/oversold conditions, identify divergences between short and long-term momentum, and anticipate mean reversion opportunities. RSI values above 70 indicate overbought conditions; below 30 indicates oversold. CCI identifies cyclical turns and extreme prices. CMO ranges from -100 to +100, providing a centered momentum measure that distinguishes between trending and mean-reverting regimes.

$$\mathrm{RSI}_n = 100 - \frac{100}{1 + \frac{\mathrm{AvgGain}_n}{\mathrm{AvgLoss}_n + \epsilon}} \quad , \mathrm{CCI}_n = \frac{P^{(t)} - \mathrm{SMA}_n(P)}{0.015 \times \mathrm{MAD}_n(P)} \quad , \mathrm{CMO}_n = 100 \times \frac{\sum \mathrm{Gains}_n - \sum \mathrm{Losses}_n}{\sum \mathrm{Gains}_n + \sum \mathrm{Losses}_n + \epsilon}$$

3. **Volatility Features:** Volatility features quantify market uncertainty and risk, crucial for position sizing intuition and regime detection. Features like ATR and Rolling standard deviation normalized by price enable the agent to distinguish between low-volatility consolidation periods (where breakouts are likely) and high-volatility trending periods (where momentum strategies excel).

$$\mathrm{ATR}_n = \frac{1}{n}\sum_{i=1}^{n} \mathrm{TR}^{(t-i)}, \quad \mathrm{ATR}_{\text{norm}} = \frac{\mathrm{ATR}_n}{P^{(t-1)} + \epsilon} \quad , \sigma_n = \frac{\sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(P^{(t-i)} - \bar{P}_n)^2}}{P^{(t-1)} + \epsilon}$$

4. **Trend Identification Features:** These features identify the prevailing market direction and trend strength, essential for aligning trades with dominant market forces. Features discussed in Section 3 like Heiken-ashi OHLC prices, Moving Averages Ribbon based on Heiken-ashi closing prices and Aroon up/down for classifying trend into bearish and bullish trend and KAMA, CHOP and ER for classifying market into trending and sideways, were included with additional feature engineering. Trending and sideways market were included as encoded features using KAMA and CHOP. ER value above 0.3 indicates a trending matket :

$$\mathrm{ER}_{\text{binary}} = \mathbb{1}[\mathrm{ER}_{20} > 0.3] \quad \& \quad \mathrm{CHOP}_{\text{binary}} = \mathbb{1}[\mathrm{CHOP}_{14} < 43.2]$$

Additionally, features like Heiken-ashi trend (bull or bear), candle body size and upper and lower wick were also included as body size tends to be very small at fractals, while upper and lower wick tends to zero for trending markets. Aroon crossovers signal trend changes – extreme values of Aroon up/down indicate strong directional conviction.

$$\text{Aroon}_{\text{down}} = \frac{n - \text{periods since } L_{\min}}{n} \times 100 \quad \& \quad \text{Aroon}_{\text{up}} = \frac{n - \text{periods since } H_{\max}}{n} \times 100$$

Furthermore, trends from moving averages of heiken-ashi close price of different look-back periods $p \in \{2, 3, 5, 8, 12, 15, 18\}$ were also included as encoded features. The ribbon provides a multi-scale view of trend health. Alignment of regimes of all MAs indicates strong trends, while mixed regimes suggest transitions.

$$\text{Regime} = \begin{cases} 1 \text{ (Strong Bullish Trend)} & \text{if } \Delta\text{EMA} \geq 0 \text{ and EMA} > \text{EMA}_{ref} \\ 2 \text{ (De-accelerating Bullish Trend)} & \text{if } \Delta\text{EMA} < 0 \text{ and EMA} > \text{EMA}_{ref} \\ 3 \text{ (Strong Bearish Trend)} & \text{if } \Delta\text{EMA} \leq 0 \text{ and EMA} < \text{EMA}_{ref} \\ 4 \text{ (De-accelerating Bearish Trend)} & \text{if } \Delta\text{EMA} > 0 \text{ and EMA} < \text{EMA}_{ref} \end{cases}$$

5. **Temporal Features:** Intraday patterns exhibit significant time-of-day effects. We encode time using cyclical transformations:

$$H_{\sin} = \sin\left(\frac{2\pi \cdot \text{hour}}{24}\right), \quad H_{\cos} = \cos\left(\frac{2\pi \cdot \text{hour}}{24}\right), \quad M_{\sin} = \sin\left(\frac{2\pi \cdot \text{minute}}{60}\right), \quad M_{\cos} = \cos\left(\frac{2\pi \cdot \text{minute}}{60}\right)$$

Cyclical encoding preserves the continuous nature of time and allows the agent to learn time-dependent patterns such as opening volatility and closing dynamics.

The state space design gives the agent a clear view of the market by using Multi-Scale Analysis to spot patterns over both short and long periods, while Regime Diversity ensures it recognizes different market types. To prevent confusion, Noise Reduction smoothes out random price jumps to reveal the true trend, and Normalization adjusts the numbers so the strategy works regardless of the asset's price. The agent also uses Temporal Awareness to understand time-based patterns and Position Awareness to make smarter decisions based on what it currently owns and its current profit or loss.

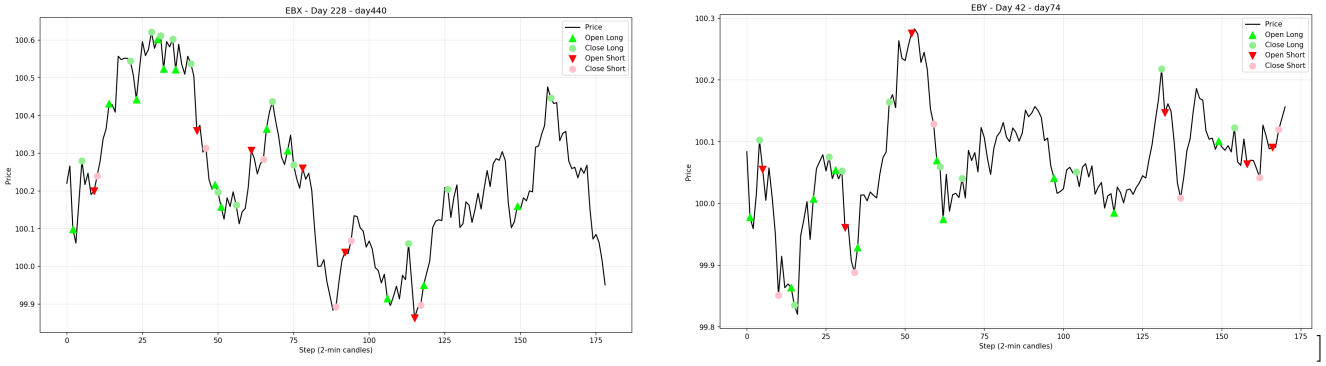


Figure 3: Example plots showing opening and closing position of the trades.

4.4 Reward Function

The reward function is a critical component of our RL trading agent, as it directly shapes the agent's behavior and trading strategy. The following reward components are included in our reward function:

1. **Profit and Loss (PnL) Reward :** When the agent closes a position, it receives a reward proportional to the realized return expressed in basis points (bps). To instill loss aversion in the agent, we apply a higher penalty for loss making by multiplying the reward with a loss multiplier λ_{loss}.

$$\text{pnl}_{\text{bps}} = \left(\frac{p_{\text{exit}} - p_{\text{entry}}}{p_{\text{entry}}}\right) \times 10000 \quad \text{and} \quad r_{\text{pnl}} = \begin{cases} \text{pnl}_{\text{bps}} & \text{if pnl}_{\text{bps}} > 0 \\ \lambda_{\text{loss}} \cdot \text{pnl}_{\text{bps}} & \text{if pnl}_{\text{bps}} \leq 0 \end{cases}$$

2. **Stop loss hit penalty :** We implement a trailing stop mechanism to protect unrealized profits and provide a dynamic exit strategy. Let the price at time t be $p^{(t)}$, and define the trailing stop level as p_{trail}. The trailing stop mechanism is expressed as:

$$p_{\text{trail}} = \begin{cases} p^{(t)}_{\text{high}} \times (1 - \delta_{\text{trail}}) & \text{for long positions} \\ p^{(t)}_{\text{low}} \times (1 + \delta_{\text{trail}}) & \text{for short positions} \end{cases} \quad \text{where} \begin{cases} p^{(t)}_{\text{high}} = \max(p^{(t-1)}_{\text{high}}, p^{(t)}) \\ p^{(t)}_{\text{low}} = \min(p^{(t-1)}_{\text{low}}, p^{(t)}) \end{cases}$$

   We set the trailing stop loss level at $\delta_{\text{trail}} = 5$ bps and the static stop loss level at $\delta_{\text{stop}} =$ -5 bps. When the trailing stop is triggered, the position is automatically closed with penalty $r_{\text{trail}} = \beta_{\text{trail}}$. When the static stop loss is triggered we impose a larger penalty given by $r_{\text{stop}} = \beta_{\text{stop}}$, to prevent catastrophic losses and discourage the agent from entering positions with high probability of immediate adverse movement.

3. **Missed Opportunity Penalty :** To encourage the agent to participate in significant market movements and avoid excessive passivity, we penalize remaining flat during volatile periods. We compute the absolute price movement over a lookback window $w = 5$ candles. If the price movement is more than the threshold $\theta_{\text{opp}} = 5$ bps we penalize for missing profitable moves, otherwise we give a small reward for not over-trading in calm markets.

$$r_{\text{opportunity}} = \begin{cases} \beta_{\text{miss}} & \text{if } \Delta_w > \theta_{\text{opp}} \\ \beta_{\text{wait}} & \text{if } \Delta_w \leq \theta_{\text{opp}} \end{cases} \quad \text{where} \quad \Delta_w = \frac{|p^{(t)} - p^{(t-w)}|}{p^{(t-w)}}$$

4. **Trade Entry Penalty :** To discourage excessive trading frequency and thereby excessive transaction fees, we also apply a fixed penalty upon entering any new position, given by $r_{\text{entry}} = \beta_{\text{entry}}$.
5. **End-of-Day Penalty :** This penalty encourages the agent to close trades before the trading session ends. The penalty is given by $r_{\text{eod}} = \beta_{\text{eod}}$.

The Table 1 summaries the rewards components. The complete reward at each timestep is the sum of all applicable components, included only when its corresponding triggering condition is satisfied at time t :

$$R^{(t)} = r_{\text{pnl}} + r_{\text{trail}} + r_{\text{stop}} + r_{\text{opportunity}} + r_{\text{entry}} + r_{\text{eod}}$$

Table 1: Reward Function Components and Parameters

Component	**Parameter**	**Value**	**Purpose**
Profitable Exit	—	$+r_{\text{pnl}}$	Reward realized gains
Losing Exit	λ_{loss}	$4.0 \times r_{\text{pnl}}$	Penalize losses asymmetrically
Stop Loss Hit	β_{stop}	-100	Severe penalty for poor entries
Trailing Stop Hit	β_{trail}	-10	Mild penalty for mechanical exits
Trade Entry	β_{entry}	-5	Discourage overtrading
Missed Opportunity	β_{miss}	-2	Penalize inaction in volatile markets
Wait Bonus	β_{wait}	$+0.1$	Reward patience in calm markets
End-of-Day Hold	β_{eod}	-10	Discourage overnight positions

5 Discussion and Results

The reinforcement learning agent, demonstrated exceptional profitability and robustness across two tickers, EBX (255 days) and EBY (140 days). The core strength of the approach was its exceptional risk management, facilitated by the use of tight stop loss and trailing stop loss mechanisms, which resulted in the strategy essentially performing **scalping**. This led to the exceptionally high annualized returns and Calmar Ratio mentioned in Table 2. The strategy's performance was validated through 10 independent simulation runs for each ticker, randomly splitting the days into 50% for training and 50% for4 testing, confirming the robustness of the learned policy. This random selection of days resulted in wide variation in max drawdown and the calmar ratio across the individual runs.

Table 2: Average Annualized Performance Summary

Ticker	**Ann.PnL(%)**	**Sharpe**	**Sortino**	**Calmar**	**Max DD**	**Avg.Trades/day**	**Win Rate**
EBX(255 days)	81.61	7.30	38.15	70.96	1.15	13.77	69.33
EBY(140 days)	77.97	4.91	23.17	63.91	1.22	18.85	69.08