# BOLT

**B**ayes-optimized **O**ff-policy **L**iquidity **T**rading

Hyoseung Kang

## Executive Summary


- **Goal:** Fuse high–resolution CEX order–book (DOM) data and on–chain liquidity flow to generate structural intraday alpha.
- **Engine:** Hybrid Soft Actor–Critic (HSAC) enhanced with a Transformer–based Critic that evaluates long–horizon context.
- **Strengths:** Off–policy sample reuse, entropy–driven exploration, and Transformer attention for delayed/sparse reward.
- **Challenges:** *Sequence length tuning*, *credit assignment between discrete and continuous actions*, and *computational cost* of Transformer inference.


## Concept

**BOLT** fuses high–resolution CEX order–book data with on–chain liquidity flow to extract structural alpha, then deploys positions via low–spread, high–leverage ECN CFDs. **Target:** intraday executions.

## Architecture

| | |
|---|---|
| **Signal** | Binance BTCUSDT DOM, Execution Traces, and BSC Scan Liquidity Flow |
| **Engine** | Bayesian Error Minimisation, Attention Mechanism, and Off–policy Inductive Bias |
| **Execution** | BTCUSD ECN CFD allows 1:500 leverage, low–commission, and $< 0.1$ pip spread |

## Learning Framework: HSAC with Transformer Critic

To meet the long–horizon, sparse–reward nature of intraday liquidity trading,[1] we upgrade standard Hybrid Soft Actor–Critic (HSAC) by replacing the conventional $Q$–network with a *Transformer–based Critic*. The actor remains an entropy–regularised Gaussian policy for the continuous lot–size parameter, while discrete trade type is still chosen via arg max.

### Training Loop

1. *Collect.* Observe state $s$, sample $a_c \sim \pi(a_c|s)$,
   choose $a_d = \arg\max_d Q(s, a_d, a_c)$, receive reward $r$ and next state $s'$, store $(s, a_d, a_c, r, s')$.
2. *Critic Update.* Sample mini–batch,
   compute where $a_c^{\sim}\pi(\cdot|s')$ and $a_d^{=}\arg\max_d Q^{\text{tgt}}(s', d, a_c^{)}$.
   Optimise $\sum(y - Q)^2$.
3. *Actor Update.* Optimise $\mathcal{L}_{\text{actor}} = \mathbb{E}\big[\alpha, \log \pi(a_c|s) - Q(s, a_d^*, a_c)\big]$ with $a_d^* = \arg\max_d Q(s, d, a_c)$.
4. *Target Soft Update.* $\theta^- \leftarrow \tau\theta + (1 - \tau)\theta^-$ with $\tau = 0.005$.

### Transformer Critic Details

- **Input Sequence:** tokens $(s_{t-k}, a_{t-k}, r_{t-k}), \ldots, (s_t, a_t)$. Positional encodings preserve order.
- **Output Head:** last token embedding $\rightarrow$ MLP $\rightarrow$ $Q$–value.
- **TD Target:** $y_t = r_t + \gamma Q^-(s_{t+1}, a_{t+1})$ with causal mask.
- **Stability:** twin critics + clipped target, gradient clipping at 1.0.

## Strengths and Remaining Challenges

**Sample Efficiency** Off–policy replay buffer allows extensive re–use of rare liquidity events.

**Long–Horizon Reasoning** Transformer attention integrates multi–scale DOM and on–chain patterns.

**Risk Control** Entropy regularisation and clipped Q reduce over–confident trades.

**Challenges** Context length selection, per–action credit assignment, and Transformer inference cost remain open problems; see also Critic–Guided DT (AAAI 2024) for alternative design.

## Related Work

- Transformer critic in RL: Chunking the Critic, Decision Q.
- Decision Transformer with critic guidance: CGDT.
- Hybrid action RL: Parameterized Action DDPG.

## Future Work

Immediate directions include (i) curriculum learning over market regimes, (ii) explainable attention maps for trader oversight, and (iii) distributed Transformer critic with fused LOB and on–chain feature encoders.

[1] Transformer critic idea adapted from Chunking the Critic (2025).

# Contents

# 1 Critical Note

## 1.1 Predction-Profitability Alignment

***2025-06-20: Having looked at the BTCUSDT chart with 5-minute candles, I am increasingly convinced of the validity of my sophisticated vectorized return prediction idea in this section.***

Follow-up is here.

Neural networks designed for short-term trading typically adopt a variation of the following formulations for their prediction output:

1. **Classification Approach**

   Given the current time $t$, the model is trained to classify the future price movement based on a predefined set of variable-length prediction horizons, $\{t + \tau_k\}_{k=1}^{T}$, where $\tau_k$ can take arbitrary values such as $\{1\text{s}, 3\text{s}, 10\text{s}, 30\text{s}\}$.

   The corresponding price differences $\{p_{t+\tau_k} - p_t\}$ are mapped to discrete labels $\{\texttt{up}, \texttt{down}, \texttt{stable}\}$, enabling the neural network to perform a multi-class classification task.

2. **Time Series (Vector) Generation Approach**

   In this approach, the neural network is trained to predict the following *future return time series*:

   $$\left\{ \frac{p_{t+\tau_k}}{p_t} \right\}_{k=1}^{T},$$

   where $\tau_{k+1} - \tau_k$ is consistent throughout $k = 1, \ldots, T-1$.

While these two approaches are frequently adopted in short-term trading models, each has its own strengths and limitations depending on the modeling objective and trading strategy. The following table summarizes their key advantages and limitations for a clearer comparison.

Table 1: Advantages and Limitations of Classification and Vector Generation Approaches

| **Approach** | **Advantage** | **Limitation** |
|---|---|---|
| Classification | Reduces the dimensionality of the prediction output. Offers high parameter and sample efficiency; may achieve a relatively better accuracy with fewer parameters even under limited data. | Temporal sparsity occurs as intermediate price actions not included in the predefined $\tau$ set are lost. Even with accurate predictions, fine-grained signals for entry/exit timing may be lacking for trading decisions. |
| Vector Generation | Provides a sequence of returns at uniformly spaced future time points. Useful not only for trading decisions, but also for risk assessment, position sizing, and trajectory-based strategy design. | As the prediction horizon length increases, sample efficiency degrades and solvability requires more model parameters and larger datasets. This creates a trade-off with the advantages of the classification approach. |

**Our Goal of Overcoming the Common Limitation of Both Approaches.** Despite the differences between these two approaches, they share a fundamental limitation when it comes to predicting far into

the future. As the prediction horizon extends further, the irreducible error increases according to the Bayes error theorem. This is fundamentally driven by the declining signal-to-noise ratio of distant future price movements, rendering neural networks inherently incapable of accurately predicting very distant outcomes. Moreover, while the ultimate goal of neural networks for short-term trading is profitability, most commonly adopted prediction outputs fail to incorporate profitability as an explicit objective. Therefore, our objective is to achieve ***Prediction–Profitability Alignment.*** We aim to define a new form of prediction output that not only eliminates the need to specify uniformly spaced future return horizons—thereby overcoming the limitations of the vector generation approach—but also addresses the low-resolution drawback inherent in the classification approach.

**Our Proposed Prediction Output Formulation.** Both the previously discussed classification and vector generation approaches define prediction outputs strictly in terms of discrete time samples. While one could theoretically predict the coefficients of a polynomial function to define a continuous function over time, in practice, financial market data only provides observable discrete samples. Consequently, defining the prediction output as a discrete vector is more practically aligned with the nature of the ground truth and with the requirements of evaluation.

However, as previously discussed, defining a high-dimensional discrete time series as the prediction output introduces inherent limitations from both the Bayes error theorem and the solvability perspective. As the dimensionality of the output increases, the irreducible error also increases, and the sample complexity required for effective learning grows rapidly.

Therefore, our objective is to design a prediction output that possesses ***a lower dimensionality than the raw observable ground truth data, while remaining directly aligned with profitability.*** In particular, this prediction output should be actionable for trading decisions and simultaneously valid as a representation of the state space for a reinforcement learning agent. Such a representation should naturally support not only supervised learning, but also seamless integration into downstream decision-making processes.

In this context, ***the horizon set*** $\mathcal{T}$, defined as

$$\mathcal{T} := \{\tau_k\}_{k=1}^{T}, \tag{1}$$

specifies ***the future prediction horizons*** to be considered. At this stage, we do not predefine the specific values of $\tau_k$. While prior research and practical trading systems often rely on ***manually selected*** $\tau_k$ values based on the designer's intuition or trading experience, such an approach lacks guarantees of ***optimality*** or ***redundancy minimization*** from a data scientific perspective. We therefore propose to construct the horizon set $\mathcal{T}$ in a data-driven manner through ***Exploratory Data Analysis*** (EDA). Specifically, each $\tau_k$ within $\mathcal{T}$ should represent a sufficiently distinct horizon, and the associated prediction target should provide meaningful trading decision insights, as determined through empirical data analysis. This approach aims to ***reduce redundancy across horizons, eliminate uninformative horizons, and increase the information density*** of the prediction output. Moreover, a data-driven $\mathcal{T}$ offers the potential to adapt optimally to varying market regimes, thereby serving as a key design element for achieving ***Prediction–Profitability Alignment***.

Given the constructed horizon set $\mathcal{T}$, our prediction output formulation does not aim to predict pointwise returns at specific future timestamps. Instead, it predicts the cumulative return accumulation up to each $\tau_k$—that is, the ***signed return area up to time*** $\tau_k$. Formally, the prediction output is defined as:

$$\left\{ \sum_{\ell=1}^{\tau_k} \left( \frac{p_{t+\ell}}{p_t} - 1 \right) \right\}_{k=1}^{T}. \tag{2}$$

This formulation enables the prediction output to implicitly capture ***intermediate market dynamics***

that are not explicitly represented in pointwise prediction approaches, resulting in a more informative and ***profitability-aligned*** signal. An additional advantage of this formulation is its robustness to low-precision computation (e.g., FP16), due to the accumulation of small relative returns.

### 1.2 Specialized RL Agents

A common approach to training a reinforcement learning (RL) agent focuses on maximizing profit as the reward signal. However, it is also possible to train a separate RL agent specialized in risk management. For example, an agent initialized in a state where the account balance is declining due to unrealized losses can be trained to learn how to recover from such drawdowns. Subsequently, the profit-seeking (alpha-centric) agent and the risk-management agent can collaborate through ensemble methods, as demonstrated in [1].

## 2 MacroHFT 2024 Nanyang Tech [1]

$$
\begin{aligned}
\boldsymbol{b}_t^M &\coloneqq \left\{ \left( p_t^{b_i}, q_t^{b_i} \right), (p_t^{a_i}, q_t^{a_i}) \right\}_{i=1}^{M} && \in \mathbb{R}^{M \times 4} \\
\boldsymbol{x}_t &\coloneqq \left( p_t^o, p_t^h, p_t^l, p_t^c, v_t \right)^\top && \in \mathbb{R}^{5 \times 1} \\
y_t &\coloneqq \phi \left( \boldsymbol{x}_t, \boldsymbol{b}_t, \ldots, \boldsymbol{x}_{t-h+1}, \boldsymbol{b}_{t-h+1} \right) && \in \mathbb{R}
\end{aligned}
$$

· $\boldsymbol{b}_t^M$: Snapshot of the level-$M$ limit order book (LOB) at time $t$, where $M = 5$ in [1].

· $\boldsymbol{x}_t$: The typical OHLCV at time $t$, representing aggregated market executions.

· $y_t$: A TA at time $t$, computed from $h$ snapshots in the past time horizon of LOB and OHLCV .

· The index $i$ corresponds to a level in the LOB snapshot.

· The symbols $b_i$ and $a_i$ denote the bid and ask, respectively.

· The symbols $p$ and $q$ denote price and quantity, respectively.

| # | Indicator | Calculation Formula | Source |
|---|---|---|---|
| 1 | max_oc | $y_{\text{max\_oc}} = \max(p_t^o, p_t^c)$ | $\boldsymbol{x}_t$ |
| 2 | min_oc | $y_{\text{min\_oc}} = \min(p_t^o, p_t^c)$ | $\boldsymbol{x}_t$ |
| 3 | kmid | $y_{\text{kmid}} = p_t^c - p_t^o$ | $\boldsymbol{x}_t$ |
| 4 | kmid2 | $y_{\text{kmid2}} = (p_t^c - p_t^o) \,/\, (p_t^h - p_t^l)$ | $\boldsymbol{x}_t$ |
| 5 | klen | $y_{\text{klen}} = p_t^h - p_t^l$ | $\boldsymbol{x}_t$ |
| 6 | kup | $y_{\text{kup}} = (p_t^h - y_{\text{max\_oc}})$ | $\boldsymbol{x}_t$ |
| 7 | kup2 | $y_{\text{kup2}} = (p_t^h - y_{\text{max\_oc}}) \,/\, (p_t^h - p_t^l)$ | upper shadow ratio $\in [0,1]$ |
| 8 | klow | $y_{\text{klow}} = (y_{\text{min\_oc}} - p_t^l)$ | $\boldsymbol{x}_t$ |
| 9 | klow2 | $y_{\text{klow2}} = (y_{\text{min\_oc}} - p_t^l) \,/\, (p_t^h - p_t^l)$ | lower shadow ratio $\in [0,1]$ |
| 10 | ksft | $y_{\text{ksft}} = (2 \cdot p_t^c - p_t^h - p_t^l)$ | $\boldsymbol{x}_t$ |
| 11 | ksft2 | $y_{\text{ksft2}} = (2 \cdot p_t^c - p_t^h - p_t^l) \,/\, (p_t^h - p_t^l)$ | $\boldsymbol{x}_t$ |
| 12 | volume | $y_{\text{volume}} = \sum_{i=1}^{M} \left(q_t^{b_i} + q_t^{a_i}\right)$ | $\boldsymbol{b}_t^M$ |
| 13 | bid_i_size_n | $\text{bid}_i\text{\_size\_n} = q_t^{b_i} / y_{\text{volume}}$, $i \in \{1, \ldots, M\}$ | $\boldsymbol{b}_t^M$ |
| 14 | ask_i_size_n | $\text{ask}_i\text{\_size\_n} = q_t^{a_i} / y_{\text{volume}}$, $i \in \{1, \ldots, M\}$ | $\boldsymbol{b}_t^M$ |
| 15 | wap1 | $y_{\text{wap1}} = \left(q_t^{a_1} p_t^{b_1} + q_t^{b_1} p_t^{a_1}\right) / \left(q_t^{a_1} + q_t^{b_1}\right)$ | $\boldsymbol{b}_t^M$ |
| 16 | wap2 | $y_{\text{wap2}} = \left(q_t^{a_2} p_t^{b_2} + q_t^{b_2} p_t^{a_2}\right) / \left(q_t^{a_2} + q_t^{b_2}\right)$ | $\boldsymbol{b}_t^M$ |
| 17 | wap_balance | $y_{\text{wap\_balance}} = \lvert y_{\text{wap1}} - y_{\text{wap2}} \rvert$ | $\boldsymbol{b}_t^M$ |
| 18 | buy_spread | $y_{\text{buy\_spread}} = \lvert p_t^{b_1} - p_t^{b_5} \rvert$ | $\boldsymbol{b}_t^M$ |
| 19 | sell_spread | $y_{\text{sell\_spread}} = \lvert p_t^{a_1} - p_t^{a_5} \rvert$ | $\boldsymbol{b}_t^M$ |
| 20 | buy_volume | $y_{\text{buy\_volume}} = \sum_{i=1}^{M} q_t^{b_i}$ | $\boldsymbol{b}_t^M$ |
| 21 | sell_volume | $y_{\text{sell\_volume}} = \sum_{i=1}^{M} q_t^{a_i}$ | $\boldsymbol{b}_t^M$ |
| 22 | volume_imbalance | $y_{\text{volume\_imbalance}} = \frac{y_{\text{buy\_volume}} - y_{\text{sell\_volume}}}{y_{\text{buy\_volume}} + y_{\text{sell\_volume}}}$ | $\boldsymbol{b}_t^M$ |
| 23 | price_spread | $y_{\text{price\_spread}} = 2 \cdot \left(p_t^{a_1} - p_t^{b_1}\right) / \left(p_t^{a_1} + p_t^{b_1}\right)$ | $\boldsymbol{b}_t^M$ |
| 24 | sell_vwap | $y_{\text{sell\_vwap}} = \sum_{i=1}^{M} \text{ask}_i\text{\_size\_}n \cdot p_t^{a_i}$ | $\boldsymbol{b}_t^M$ |
| 25 | buy_vwap | $y_{\text{buy\_vwap}} = \sum_{i=1}^{M} \text{bid}_i\text{\_size\_}n \cdot p_t^{b_i}$ | $\boldsymbol{b}_t^M$ |
| 26 | log_return_bidi_price | $\log\left(p_t^{b_i} / p_{t-1}^{b_i}\right)$, $i = 1, 2$ | temporal from $\boldsymbol{b}_t^M$ |
| 27 | log_return_aski_price | $\log\left(p_t^{a_i} / p_{t-1}^{a_i}\right)$, $i = 1, 2$ | temporal from $\boldsymbol{b}_t^M$ |
| 28 | log_return_wap1 | $\log\left(y_{\text{wap1}}^{t} / y_{\text{wap1}}^{t-1}\right)$ | temporal |
| 29 | log_return_wap2 | $\log\left(y_{\text{wap2}}^{t} / y_{\text{wap2}}^{t-1}\right)$ | temporal |
| 30 | trend_features | $\boldsymbol{y} = [p_t^{a_i}, p_t^{b_i}, y_{\text{buy\_spread}}, y_{\text{sell\_spread}}, y_{\text{wap1}}, y_{\text{wap2}}, y_{\text{sell\_vwap}}, y_{\text{buy\_vwap}}, y_{\text{volume}}]^\top$ | |
| | | $\boldsymbol{y}_{\text{trend}} = \frac{\boldsymbol{y} - \text{RollingMean}(\boldsymbol{y}, 60)}{\text{RollingStd}(\boldsymbol{y}, 60)}$ | derived from $\boldsymbol{b}_t^M$, $\boldsymbol{x}_t$ history |

Table 2: Calculation formulas for technical indicators used in MacroHFT [1]

# 3 HFformer: Mid-price from LOB passes ADF Test [2]

**Insight from HFformer [2].** LOB-based log returns exhibit stationarity, characterized by a stable mean and variance, which enables reliable pattern learning during training.

**Definition:**

· Execution Strength (ES, Korean Standard) := $^{\text{Market Buy Volume}}/_{\text{Market Sell Volume}}$

**TODO:**

· Come up with my definition of ES.

· Define equation (3) also incorporating ES.

**ADF Test on Log Returns from LOB.** The authors [2] compute a *weighted mid price* (WMP) from the top-$k$ bid/ask levels of the LOB, ***excluding any executed trades on the chart***:

$$\text{WMP}_t := \frac{\sum_{i=1}^{k=20} P_i^{\text{ask}} Q_i^{\text{ask}} + \sum_{i=1}^{k=20} P_i^{\text{bid}} Q_i^{\text{bid}}}{\sum_{i=1}^{k=20} Q_i^{\text{ask}} + \sum_i Q_i^{\text{bid}}} \tag{3}$$

From this, they compute log return (a unit-less ratio):

$$r_t := \log\left(\frac{\text{WMP}_t}{\text{WMP}_{t-1}}\right) = \log\left(\text{WMP}_t\right) - \log\left(\text{WMP}_{t-1}\right)$$

To test the stationarity of $r_t$, they perform an Augmented Dickey-Fuller (ADF) test. The regression used is:

$$\Delta r_t = \alpha + \beta t + \gamma r_{t-1} + \sum_{i=1}^{p} \delta_i \Delta r_{t-i} + \varepsilon_t,$$

where The test checks whether the autoregressive coefficient $\gamma$ is significantly less than zero.

| **Element** | **Meaning** | **Category** |
|---|---|---|
| $p_t$ | Log return series from WMP | Input (series) |
| Lag order $p$ | Manually or automatically selected lag order | User parameter |
| $\gamma$, $\alpha$, $\beta$, $\delta_i$ | Regression coefficients estimated from fitting | To be fitted |
| $p$-value$^{(*)}$ | Result of $t$-test on $\gamma$ for stationarity | To be fitted |
| $\Delta p_t$ | First-order difference: $p_t - p_{t-1}$ | Computable |

Table 3: ADF test formulation: variable roles and types

**Hypothesis:**

- $H_0$: $\gamma = 0$ (unit root exists; non-stationary)
- $H_1$: $\gamma < 0$ (no unit root; stationary)

The fitted $p$-value $\ll 0.05$ implies rejection of $H_0$, indicating that LOB-based log returns exhibit stationarity (i.e., stable mean and variance), and hence are suitable for pattern learning. A simple Python example is shown in Code-1.

### 3.1 Archite & Optimization Technic

***Layer Normalization*** normalizes the mean and variance across feature dimensions within each sample, stabilizing output distributions and improving gradient flow for faster convergence. Mathematically, it maintains consistent input scales for subsequent layers, smoothing the loss surface and stabilizing weight update directions. It is widely used in Transformers and RNNs, especially when batch sizes are small or input distributions fluctuate, such as in time-series prediction. However, it may be less suitable for CNNs where spatial structure matters, or in tasks sensitive to noise or already using BatchNorm. While primarily applied at the architectural level, care must be taken with its placement relative to residual paths and position embeddings, as the ordering can significantly affect performance.

Hell

**To Read:**

· [BertiK25] TLOB Dual Attn LOB

· [GuoLH23] MM with DRL from LOB

# A ADF Test Python Example [2]

```python
import numpy as np
from statsmodels.tsa.stattools import adfuller

# Price time series (e.g., BTC price)
price = np.array([100, 102, 105, 110, 114, 113, 115, 118, 120, 122],
    dtype=float)

# Compute log returns
log_return = np.diff(np.log(price))

# Perform Augmented Dickey-Fuller test
result = adfuller(log_return)

# Print test results
print(f"ADF Statistic: {result[0]:.4f}")
print(f"p-value: {result[1]:.4f}")
print(f"Lags Used: {result[2]}")
print(f"# Observations: {result[3]}")
print("Critical Values:")
for key, value in result[4].items():
    print(f"   {key}: {value:.4f}")

# Interpretation of result
if result[1] < 0.05:
    print("stationary")
else:
    print("non-stationary")
```

Code-1: ADF Test on Log Returns of Price Series

## B Tehnical Notes

- Should we choose ***10-level or 20-level DOM*** as the Input Data to NN?
  We can only tell by ***experimentally comparing the performances*** of NN models.
- Time-and-Price ***Delta*****:** Normalized Input Features to NN in the Context of
  Execution Strength against Order Book Balance, also allowing ***FP-16***
- ***Predict Return Regression" followed by Imitation Learning (IL)*****:** Train the model to predict the normalized returns. Analyze the prediction accuracy, and then, extend the approach using Imitation Learning. ***Manual labelling is unclear not only costly.***
- ***Multi-level Time Frames*** to capture Important Prices for Symbol: Develop a General Notion, such as Resistance and Support
- ***Partially Take Profit,*** and ***Re-entry after an Early Cut-loss***
- Uncertainty Feedback via LoRA & Prunning if necessary
- The notion of ***"The Harmony in Bid/Ask of DOM"*** from CWA is related
  to the so-called ***order imbalance,*** defined as: $^{(\texttt{Bid}-\texttt{Ask})}/_{(\texttt{Bid}+\texttt{Ask})}$.

## References

[1] C. Zong, C. Wang, M. Qin, L. Feng, X. Wang, and B. An, "MacroHFT: Memory Augmented Context-aware Reinforcement Learning On High Frequency Trading," in *Proc. 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, (Barcelona, Spain), pp. 4712–4721, Aug. 2024.

[2] F. Barez, P. Bilokon, A. Gervais, and N. Lisitsyn, "Exploring the Advantages of Transformers for High-frequency Trading ," *arXiv preprint*, 2023. Access: https://arxiv.org/abs/2302.13850.