Machine Learning and Market Microstructure in Digital Asset Markets

Day 4 of 5

Prof. Jörg Osterrieder

PhD Seminar: Digital Finance Research

Spring 2026

PhD Seminar Series: Digital Finance Research

Where We Are

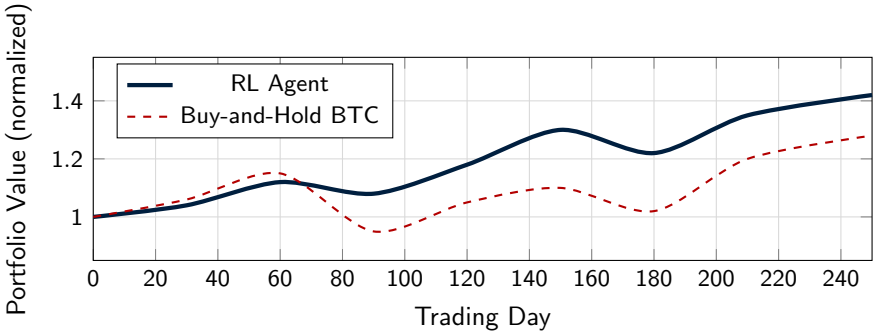
Days 1–3 Recap

- Day 1: Institutional landscape, AMMs, CFMM geometry
- Day 2: Derivative pricing, Heston–Kou, implied vol
- Day 3: EIP-1559, MEV game theory, tokenomics, PBS

Today's Roadmap

1. Market microstructure theory (Kyle, VPIN, Amihud)
2. Reinforcement learning for trading
3. Transformers and limit order book modeling
4. Applications: SHAP, backtesting, flash crash prediction

Can a Machine Learn to Trade Bitcoin?



Illustrative equity curves. RL agent uses PPO with microstructure features. Based on concepts from FinRL [2].

FinRL: Reinforcement Learning for Finance

FinRL framework (open-source):

- Standardized environment for training RL agents on financial data
- Supports: PPO, A2C, DDPG, SAC, TD3
- State space: prices, volumes, technical indicators, microstructure features
- Action space: portfolio weights or order quantities

Typical workflow:

1. Preprocess market data $\rightarrow$ construct state representation
2. Train agent on historical data (in-sample)
3. Evaluate on out-of-sample period
4. Compare against benchmarks (buy-and-hold, equal-weight, MACD)

Survey findings [2]: RL shows strongest performance gains in *market making* and *optimal execution*, less consistent for pure alpha generation.

Why Does the RL Agent Sometimes Fail?

Regime-dependent performance:

Market Regime	RL Agent	Buy-and-Hold
Trending (bull)	Comparable	Strong
Mean-reverting	Strong	Weak
High volatility / crash	Depends on training	Very weak
Regime change	Often fails	Unaffected

Root causes of failure:

- **Non-stationarity**: the data-generating process changes
- **Distribution shift**: test regime $\neq$ training regime
- **Overfitting**: agent memorizes training patterns
- **Transaction costs**: in-sample profits vanish after costs

Two Fundamental Questions

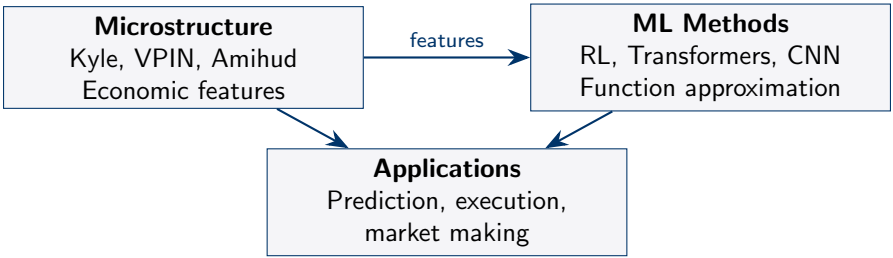
1. **What information matters?**
   - Price returns? Volume? Order book shape? Informed trading?
   - $\Rightarrow$ **Market microstructure theory** provides the economic foundation for feature selection

2. **How do we extract it?**
   - Linear models? Random forests? Deep nets? RL policies?
   - $\Rightarrow$ **Machine learning methods** provide the computational tools

Today's Thesis

The best ML for finance combines *economic structure* (microstructure theory) with *flexible function approximation* (deep learning). Neither alone is sufficient.

Today: Microstructure Theory + ML Methods



Core references: [3], [1], [7], [2]

Outline

Kyle (1985): The Foundational Model

[3]: the most influential model of *informed trading* and *price impact*. Published in *Econometrica*.

Central question: How does private information get incorporated into asset prices through the trading process?

Three types of agents:

1. **Informed trader**: knows the true asset value v
2. **Noise traders**: trade for liquidity reasons; aggregate demand $u \sim \mathcal{N}(0, \sigma_u^2)$
3. **Market maker**: competitive, risk-neutral, sets prices to break even conditional on total order flow

Key assumption: The market maker cannot distinguish informed from noise order flow — only the *total* order flow is observed.

Reference text: [6]

Kyle Model: Setup

Asset: true value $v \sim \mathcal{N}(\mu, \sigma_v^2)$, unknown to market maker and noise traders.

Timeline:

1. Informed trader observes v and submits order x
2. Noise traders submit aggregate order $u \sim \mathcal{N}(0, \sigma_u^2)$
3. Market maker observes total order flow $y = x + u$
4. Market maker sets price $p = p(y)$
5. Trades execute at price p

Information structure:

- v and u are independent
- Market maker knows the distributions of v and u but not their realizations
- The informed trader knows v but not u

Informed Trader's Optimization

The informed trader chooses order size x to maximize expected profit:

Informed Trader's Problem

$$\max_x \ \mathbb{E}\big[(v-p)(x) \ \big| \ v\big]$$

Substituting the linear pricing rule $p = \mu + \lambda(x+u)$:

$$\max_x \ \mathbb{E}\big[(v-\mu-\lambda(x+u))\cdot x \ \big| \ v\big]$$

Since $\mathbb{E}[u]=0$:

$$\max_x \ (v-\mu)\cdot x - \lambda x^2$$

First-order condition:

$$x^*(v) = \frac{v-\mu}{2\lambda}$$

The informed trader trades *proportionally* to the mispricing $(v-\mu)$ but moderates to reduce price impact.

Market Maker: Linear Pricing Rule

The competitive market maker sets price equal to expected value conditional on total order flow:

Market Maker's Pricing Rule

$$p = \mathbb{E}[v \mid y] = \mu + \lambda \cdot y = \mu + \lambda(x + u)$$

This is the **regression of** v **on** y:

$$\lambda = \frac{\mathrm{Cov}(v, y)}{\mathrm{Var}(y)} = \frac{\mathrm{Cov}(v, x^*(v) + u)}{\mathrm{Var}(x^*(v) + u)}$$

Substituting $x^*(v) = \frac{v-\mu}{2\lambda}$ and solving for λ:

Kyle's Lambda

$$\lambda = \frac{\sigma_v}{2\sigma_u}$$

Kyle's Lambda: The Price Impact Coefficient

$$\boxed{\lambda = \frac{\sigma_v}{2\sigma_u}}$$

Interpretation:

- λ is the **price impact per unit of order flow**
- Higher σ_v (more private information) $\Rightarrow$ higher λ (less liquidity)
- Higher σ_u (more noise trading) $\Rightarrow$ lower λ (more liquidity, informed trader can hide)

Equilibrium properties:

- Prices are *semi-strong efficient*: $p = \mathbb{E}[v \mid y]$
- The informed trader's expected profit: $\Pi = \frac{\sigma_v \sigma_u}{2}$
- Market maker earns zero expected profit (competitive)
- Noise traders bear the cost: $\mathbb{E}[\text{noise loss}] = \frac{\sigma_v \sigma_u}{2}$

This λ is the ancestor of all modern price impact models.

Adapting Kyle to Crypto Markets

The Kyle model was built for equity markets. Crypto markets differ:

Feature	**Traditional**	**Crypto**
Trading hours	6.5 hrs/day	24/7/365
Venues	1–2 exchanges	100+ CEX + DEX
Order routing	Reg NMS/MiFID	No best-execution rule
Informed traders	Insiders, analysts	MEV bots, on-chain analysts
Noise traders	Retail, index funds	Retail, yield farmers
Market makers	Citadel, Virtu	Wintermute, on-chain LPs

Key modifications needed:

- Multi-venue λ: fragmented liquidity $\Rightarrow$ venue-specific price impact
- MEV extractors as a *new class of informed agents* (information = mempool visibility)
- AMM LP as passive market maker with different adverse selection dynamics

VPIN: Volume-Synchronized Probability of Informed Trading

VPIN (Easley, López de Prado, O'Hara): a real-time estimator of the probability that trading is information-driven.

Construction:

1. Partition the trading timeline into **volume buckets** of equal size V (not equal time)
2. In each bucket i, classify volume as *buy* (V_i^B) or *sell* (V_i^S) using the bulk volume classification rule
3. Compute order imbalance in each bucket: $|V_i^B - V_i^S|$

Intuition: Informed traders generate *directional* order flow; noise traders generate balanced flow. High imbalance $\Rightarrow$ likely informed trading.

See also [4] for ML applications of microstructure features.

VPIN: The Algorithm

Bulk volume classification for bucket i:

$$V_i^B = V \cdot \Phi\left(\frac{\Delta p_i}{\sigma_{\Delta p}}\right), \qquad V_i^S = V - V_i^B$$

where Δp_i is the price change within bucket i and Φ is the standard normal CDF.

Rolling VPIN over n buckets:

VPIN Formula

$$\text{VPIN} = \frac{\sum_{i=1}^{n} |V_i^B - V_i^S|}{n \cdot V}$$

Range: $\text{VPIN} \in [0, 1]$

- $\text{VPIN} \approx 0$: balanced flow (noise-dominated)
- $\text{VPIN} \to 1$: highly directional flow (information-driven)

VPIN: Properties and Crypto Applications

Statistical properties:

- Volume synchronization eliminates time-of-day effects
- No trade-by-trade tick classification needed (unlike PIN)
- Can be updated in real time as new volume buckets complete

Crypto applications:

- VPIN spikes before exchange outages and flash crashes
- Cross-venue VPIN: informed trading migrates to the venue with lowest λ
- On-chain VPIN: apply to DEX volume (where all trades are public)
- [1]: VPIN among the top predictive features for crypto market dynamics

Connection to Kyle: VPIN estimates the fraction of order flow that is informed ($\propto \alpha$ in the Easley–O'Hara sequential trade model), which drives Kyle's λ.

Roll Measure: Estimating the Bid-Ask Spread

Roll (1984): infer the effective spread from the serial covariance of price changes.

Assumption: Observed price $p_t = m_t + c \cdot q_t$ where m_t is the efficient price and $q_t \in \{-1, +1\}$ is the trade direction (i.i.d.).

Price change: $\Delta p_t = \Delta m_t + c(q_t - q_{t-1})$

Under the model: $\mathrm{Cov}(\Delta p_t, \Delta p_{t-1}) = -c^2$, so

Roll Spread Estimator

$$\text{Roll} = 2\sqrt{-\mathrm{Cov}(\Delta p_t, \Delta p_{t-1})}$$

Crypto context:

- Useful for DEXs where bid-ask spreads are not directly observable
- Negative autocovariance $\Rightarrow$ bid-ask bounce present
- When $\mathrm{Cov} > 0$ (momentum), Roll is undefined $\Rightarrow$ set to zero or use alternative estimators

Amihud Illiquidity Ratio

Amihud (2002): a simple, data-efficient measure of price impact (illiquidity).

Amihud Illiquidity

$$\text{ILLIQ}_T = \frac{1}{T}\sum_{t=1}^{T}\frac{|r_t|}{\text{Volume}_t}$$

where r_t is the return on day t and Volume_t is the dollar (or token) volume.

Interpretation:

- Measures "price change per unit of volume traded"
- Higher ILLIQ $\Rightarrow$ more illiquid (larger price impact per dollar)
- Proxy for Kyle's λ when only daily data are available

In crypto:

- Highly variable across tokens and exchanges
- Small-cap tokens: ILLIQ can be $100\times$ that of BTC/ETH
- [5]: cross-exchange arbitrage profits are related to ILLIQ differentials

ML with Microstructure Features (Easley et al. 2024)

[1]: combine microstructure measures with ML methods to predict crypto market dynamics.

Feature set:

1. VPIN (informed trading probability)
2. Amihud ILLIQ (price impact)
3. Roll spread (effective spread)
4. Volume imbalance (buy vs. sell pressure)
5. Realized volatility (5-min, 1-hr, daily)
6. Order flow toxicity (VPIN derivatives)

Methods: Random forest, gradient boosting, neural networks

Prediction targets: returns, volatility, liquidity regime changes

Key Finding: Microstructure Measures Predict Crypto Dynamics

Easley et al. (2024) main results across BTC, ETH, BNB, SOL, XRP:

- Microstructure features improve out-of-sample R^2 by **3–8 percentage points** over price-only models
- **VPIN** is the single most important feature for predicting short-horizon returns (1–60 minutes)
- **Amihud ILLIQ** dominates for volatility prediction
- Feature importance is *time-varying*: VPIN matters more during high-volatility regimes
- **Economic significance**: a simple trading strategy based on microstructure signals earns positive risk-adjusted returns (before transaction costs)

Takeaway for ML Practitioners

Microstructure-informed features should be standard inputs for any crypto ML pipeline. Raw OHLCV data alone is insufficient.

Outline

Reinforcement Learning for Finance: The MDP Formulation

Trading as a Markov Decision Process (MDP):

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \gamma)$$

- $\mathcal{S}$: state space (market observations)
- $\mathcal{A}$: action space (trading decisions)
- $P(s'|s,a)$: transition dynamics (market evolution)
- $r(s,a,s')$: reward function
- $\gamma \in (0,1)$: discount factor

Agent's objective:

$$\max_{\pi} \ \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t \, r_t\right]$$

where $\pi : \mathcal{S} \to \mathcal{A}$ is the policy.

Survey: [2] — comprehensive review of RL in financial decision-making.

State Space: What the Agent Observes

$$S_t = \Big(\underbrace{\boldsymbol{L}_t}_{\text{LOB state}} , \underbrace{I_t}_{\text{inventory}} , \underbrace{t}_{\text{time}} \Big)$$

LOB state $\boldsymbol{L}_t$ (limit order book):

- Top K bid prices and sizes: $(p_1^b, q_1^b), \ldots, (p_K^b, q_K^b)$
- Top K ask prices and sizes: $(p_1^a, q_1^a), \ldots, (p_K^a, q_K^a)$
- Recent trade history: prices, sizes, aggressor side
- Microstructure features: VPIN, spread, imbalance

Inventory I_t: agent's current position (in units of the asset)

Time t: time remaining until end of horizon (for execution tasks)

Dimensionality: With $K = 10$ levels, $S_t \in \mathbb{R}^d$ where $d \approx 40$–100 depending on features.

Action Space: Trading Decisions

Market making:

$$a_t = (\delta_t^b, \delta_t^a, q_t^b, q_t^a)$$

where $\delta_t^{b/a}$ are bid/ask offsets from mid-price and $q_t^{b/a}$ are order sizes.

Optimal execution:

$$a_t = v_t \in [0, V_{\max}]$$

the volume to execute at time t (continuous action).

Portfolio allocation:

$$a_t = (w_t^1, \ldots, w_t^n), \quad \sum_i w_t^i = 1$$

portfolio weights across n assets.

Action space design choices:

- **Discrete**: buy/sell/hold (DQN-friendly, coarse)
- **Continuous**: exact quantities (PPO/SAC, more realistic)
- **Parameterized**: discrete type + continuous parameter

Reward Function: Risk-Adjusted P&L

Standard Reward

$$r_t = \underbrace{\mathrm{PnL}_t}_{\text{mark-to-market P\&L}} - \underbrace{\phi \cdot \mathrm{Risk}_t}_{\text{risk penalty}}$$

Common specifications:

- $\mathrm{PnL}_t = I_t \cdot \Delta p_t +$ realized spread $-$ transaction costs
- $\mathrm{Risk}_t = I_t^2$ (quadratic inventory penalty)
- $\phi > 0$: risk aversion parameter

Alternative rewards:

- Sharpe ratio: $r_t = \frac{\mathrm{PnL}_t}{\sigma_{\mathrm{PnL}}}$ (but non-stationary normalization)
- CARA utility: $r_t = -e^{-\phi \cdot \mathrm{PnL}_t}$
- Implementation shortfall: $r_t = -(p_t^{\text{exec}} - p_0^{\text{arrival}}) \cdot v_t$

Design matters: Reward shaping is the most impactful hyperparameter in financial RL.

Policy Gradient: REINFORCE Algorithm

Parametric policy: $\pi_\theta(a|s)$ with parameters θ.

Objective: $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$

Policy Gradient Theorem (Williams 1992)

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot R_t]$$

where $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$ is the return-to-go.

REINFORCE update:

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot R_t$$

Problem: High variance — R_t is a noisy Monte Carlo estimate.

Variance reduction: Subtract a baseline $b(s_t)$:

$$\nabla_\theta J(\theta) = \mathbb{E}[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot (R_t - b(s_t))]$$

When $b(s_t) = V^\pi(s_t)$, this leads to the *actor-critic* family.

Actor-Critic: PPO and A2C

Idea: Learn both a policy (*actor*) and a value function (*critic*) simultaneously.

Advantage function:

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t)$$

A2C (Advantage Actor-Critic):

$$\nabla_\theta J = \mathbb{E}\Big[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \hat{A}_t\Big]$$

PPO (Proximal Policy Optimization):

$$L^{\text{PPO}}(\theta) = \mathbb{E}\Big[\min\Big(\rho_t(\theta)\hat{A}_t,\ \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\Big)\Big]$$

where $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ and $\epsilon \approx 0.2$.

Why PPO for finance: Stable updates, works with continuous actions, less sensitive to hyperparameters than TRPO.

Deep Q-Networks for Discrete Actions

Q-learning: learn the action-value function $Q^*(s, a) = \max_\pi \mathbb{E}_\pi[R_t | s_t = s, a_t = a]$.

DQN (Mnih et al. 2015): approximate Q^* with a neural network $Q_\omega(s, a)$.

Bellman target:

$$y_t = r_t + \gamma \max_{a'} Q_{\omega^-}(s_{t+1}, a')$$

where ω^- are target network parameters (updated periodically).

Loss:

$$L(\omega) = \mathbb{E}\Big[\big(Q_\omega(s_t, a_t) - y_t\big)^2\Big]$$

For trading:

- Action space: $\{-K, -K+1, \ldots, 0, \ldots, K\}$ (discrete lot sizes)
- Works well for execution (buy k lots vs. wait)
- Scales poorly to large continuous action spaces

Almgren–Chriss Optimal Execution Framework

Classical benchmark for optimal execution against which RL agents are compared.

Setup: Sell X shares over $[0, T]$ with temporary and permanent price impact.

Price dynamics:

$$S_t = S_0 - g \cdot \sum_{\tau=0}^{t} v_\tau - h(v_t)$$

where g is permanent impact per share and $h(v_t) = \eta \cdot v_t$ is temporary impact.

Objective (mean-variance):

$$\min_{\{v_t\}} \ \mathbb{E}[\text{Cost}] + \phi \cdot \text{Var}[\text{Cost}]$$

Optimal trajectory (closed-form):

$$x_t^* = X \cdot \frac{\sinh\big(\kappa(T - t)\big)}{\sinh(\kappa T)}$$

where $\kappa = \sqrt{\phi\sigma^2/\eta}$ and x_t is the remaining inventory at time t.

HJB Equation for Optimal Execution

Continuous-time formulation with jump-diffusion price dynamics:

$$\mathrm{d}S_t = \mu\,\mathrm{d}t + \sigma\,\mathrm{d}W_t + J\,\mathrm{d}N_t - g \cdot v_t\,\mathrm{d}t$$

where N_t is a Poisson process (jumps) with intensity λ_J and jump size J.

Value function $V(t, x, S)$ for remaining inventory x:

HJB Equation

$$0 = \frac{\partial V}{\partial t} + \sup_{v \geq 0}\left[-h(v) \cdot v + (\mu - g \cdot v)\frac{\partial V}{\partial S} + \tfrac{1}{2}\sigma^2\frac{\partial^2 V}{\partial S^2} - v\frac{\partial V}{\partial x} + \lambda_J\,\mathbb{E}\big[V(t, x, S + J) - V\big]\right]$$

Why this matters for RL: The HJB provides the *optimal benchmark*; RL aims to approximate its solution without knowing the model parameters $(\mu, \sigma, g, h, \lambda_J)$.

Adapting Execution to Crypto Markets

Challenges unique to crypto:

1. **24/7 trading**: No market open/close — execution horizon is a continuous choice
2. **Fragmented liquidity**: split across CEXs, DEXs, L1s, L2s $\Rightarrow$ cross-venue execution
3. **MEV risk**: large orders in DEX mempools attract sandwich attacks
   - Cost of MEV: additional slippage $\approx \lambda_{\text{MEV}} \cdot v_t$
   - Must be included in execution cost model
4. **Stochastic fees**: gas fees add a random component to transaction costs (especially during congestion)
5. **Settlement risk**: finality times vary (Ethereum $\sim$12 min for economic finality)

Modified cost function:

$$\text{Cost} = \underbrace{g \cdot X}_{\text{permanent}} + \underbrace{\eta \sum_t v_t^2}_{\text{temporary}} + \underbrace{\sum_t \text{MEV}_t(v_t)}_{\text{MEV cost}} + \underbrace{\sum_t \text{gas}_t}_{\text{fees}}$$

RL Performance by Task (Survey Results)

Key findings from [2]:

Task	Relative Gain	Notes
Market making	$+++$	Highest gains; learns adaptive spreads
Optimal execution	$++$	Beats Almgren–Chriss in non-Gaussian
Portfolio allocation	$+$	Mixed; often $\leq$ mean-variance
Alpha generation	$+/-$	Inconsistent across assets/periods

Why market making works best:

- Clear reward signal (spread capture minus inventory risk)
- Stationary-enough environment (microstructure regularities)
- Well-defined action space (bid/ask quotes)
- Frequent feedback (many trades per episode)

When RL Fails: Pitfalls in Financial Applications

Common Failure Modes

1. **Overfitting to training period**
   Agent learns regime-specific patterns that don't generalize
2. **Non-stationarity**
   Market dynamics shift; the MDP itself changes over time
3. **Distribution shift**
   Test-time states lie outside training distribution
4. **Reward hacking**
   Agent exploits simulator artifacts (e.g., infinite liquidity)
5. **Sample inefficiency**
   Financial data is limited; cannot generate unlimited episodes

Mitigation strategies:

- Walk-forward validation (expanding window)
- Domain randomization (perturb market parameters)

Outline

The Self-Attention Mechanism

Core operation of the Transformer:

Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$

where $Q \in \mathbb{R}^{n \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$, $V \in \mathbb{R}^{n \times d_v}$ are query, key, and value matrices.

Multi-head attention:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$.

For financial time series:

- Each time step is a "token"
- Attention learns which *past observations* are most relevant for predicting the next step
- Long-range dependencies captured without recurrence

Why Transformers Outperform LSTMs for Financial Data

Property	LSTM	Transformer
Long-range dependencies	Degrades	Constant via attention
Parallelization	Sequential	Fully parallel
Variable-length input	Natural	Via masking
Interpretability	Hidden state (opaque)	Attention weights
Training speed	Slow (sequential)	Fast (parallel)
Irregular sampling	Difficult	Positional encoding

Key advantage for LOB data:

- LOB snapshots arrive at irregular intervals
- Transformer attention can weight *event-driven* vs. *time-driven* features adaptively
- Multi-head attention captures multiple time scales simultaneously (tick-level, second-level, minute-level patterns)

Positional Encoding for Temporal Structure

Transformers have no built-in notion of sequence order. **Positional encoding** injects temporal information.

Sinusoidal encoding (Vaswani et al. 2017):

$$\text{PE}(t, 2k) = \sin\left(\frac{t}{10000^{2k/d}}\right), \quad \text{PE}(t, 2k+1) = \cos\left(\frac{t}{10000^{2k/d}}\right)$$

where t is position and k is the dimension index.

For financial data, adaptations include:

- **Timestamp encoding**: encode actual wall-clock time (captures intraday patterns)
- **Event-time encoding**: position = event index (trade count, volume bucket count)
- **Learned encoding**: let the model learn optimal positional representation from data
- **Relative encoding**: encode time *differences* (time between events)

Sirignano & Cont (QF 2025): Deep LOB Forecasting

[7]: state-of-the-art deep learning for limit order book prediction.

Task: Predict future mid-price movement from LOB snapshots.

Input: Sequence of LOB snapshots, each containing K levels of bid/ask prices and quantities.

Target:

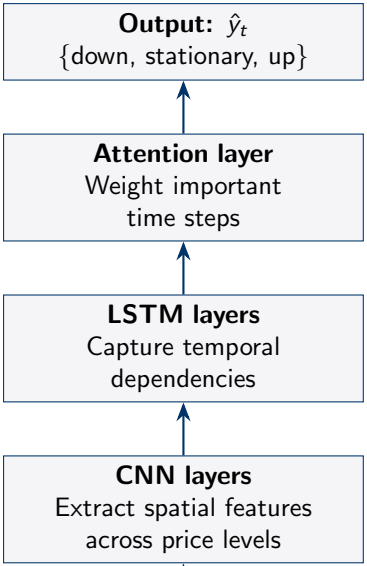
$$y_t = \text{sign}\left(\frac{m_{t+\Delta} - m_t}{m_t}\right) \in \{-1, 0, +1\}$$

where m_t is the mid-price and Δ is the prediction horizon.

Key results:

- Out-of-sample accuracy: 65–72% for $\Delta = 10$ events
- Significant improvement over linear and shallow ML baselines
- Features from deeper LOB levels (levels 5–10) contribute meaningfully — not just best bid/ask

DeepLOB Architecture: CNN + LSTM



Sequence of LOB feature vectors
$\rightarrow$ temporal patterns

Convolve across bid/ask levels
$\rightarrow$ order book features

Hawkes Processes for Order Arrivals

Self-exciting point processes model the clustering of market events (orders, trades, cancellations).

Univariate Hawkes Process

$$\lambda(t) = \mu + \int_0^t \alpha\, e^{-\beta(t-s)}\, \mathrm{d}N(s)$$

Parameters:

- $\mu > 0$: baseline intensity (exogenous event rate)
- $\alpha > 0$: excitation parameter (each event boosts intensity)
- $\beta > 0$: decay rate (excitation fades exponentially)
- Stationarity condition: $\alpha/\beta < 1$ (branching ratio)

Branching ratio α/β: fraction of events that are *endogenous* (triggered by previous events). In liquid markets, $\alpha/\beta \approx 0.7$–0.9: most events are reactions to prior events, not exogenous news.

Mutually-Exciting Processes for Bid/Ask Dynamics

Extension: A bivariate Hawkes process where bid and ask events excite each other.

Bid arrival intensity:

$$\lambda^b(t) = \mu^b + \int_0^t \alpha^{bb} e^{-\beta^{bb}(t-s)} \, \mathrm{d}N^b(s) + \int_0^t \alpha^{ba} e^{-\beta^{ba}(t-s)} \, \mathrm{d}N^a(s)$$

Ask arrival intensity:

$$\lambda^a(t) = \mu^a + \int_0^t \alpha^{ab} e^{-\beta^{ab}(t-s)} \, \mathrm{d}N^b(s) + \int_0^t \alpha^{aa} e^{-\beta^{aa}(t-s)} \, \mathrm{d}N^a(s)$$

Key patterns in data:

- $\alpha^{bb}, \alpha^{aa} > 0$: self-excitation (order clustering)
- $\alpha^{ba}, \alpha^{ab} > 0$: cross-excitation (bid activity triggers ask activity and vice versa)
- Captures the "hot potato" effect in high-frequency trading

Extension: Advanced Architectures (Self-Study)

Flagged for Self-Study

The following recent architectures extend the deep LOB paradigm.

1. **PrimoGPT + PrimoRL:**
   - GPT-style autoregressive model pre-trained on LOB sequences
   - Fine-tuned with RL for specific trading tasks
   - Combines generative pre-training with task-specific optimization
2. **StockFormer:**
   - Transformer with cross-asset attention (learns inter-stock dependencies)
   - Multi-scale temporal encoding (tick, minute, daily)
   - Reported improvements over DeepLOB on mid-price prediction

Recommended reading:

- [4] Ch. 18–19 for ML pipeline best practices
- [2] for the RL survey

Outline

Applications & Industry

From Theory to Practice

Microstructure Signals: Which Measures Predict Crypto Returns?

Summary of predictive power (based on [1]):

Feature	Returns	Volatility	Liquidity
VPIN	Strong	Moderate	Weak
Amihud ILLIQ	Weak	Strong	Strong
Roll spread	Weak	Moderate	Strong
Realized vol	Moderate	Strong	Moderate
Volume imbalance	Strong	Weak	Weak
LOB depth ratio	Moderate	Moderate	Strong

Key insight: No single measure dominates across all prediction targets. Combining microstructure features via ML (random forest, gradient boosting) yields the best out-of-sample performance.

VPIN as an Early Warning for Flash Crashes

Hypothesis: VPIN spikes *before* large price dislocations because informed traders act first.

Evidence:

- **May 6, 2010** (equity flash crash): VPIN reached historically high levels 1–2 hours before the crash
- **Crypto application**: VPIN computed on BTC/USDT spikes 30–90 minutes before major liquidation cascades
- Cross-venue: VPIN on one exchange can predict events on another (informed traders choose execution venue)

Early warning system:

$$\text{Alert if } \text{VPIN}_t > \mu_{\text{VPIN}} + k \cdot \sigma_{\text{VPIN}}$$

where $k \approx 2$–3 (calibrated to historical false positive rate).

Practical use: Risk management — reduce position size or widen spreads when VPIN is elevated.

ML Demo Concept: Random Forest + SHAP Explanations

Pipeline:

1. Construct feature matrix: VPIN, ILLIQ, Roll, realized vol, volume, returns (lagged)
2. Target: next-hour return sign $\in \{-1, 0, +1\}$
3. Train random forest (500 trees, max depth 8)
4. Evaluate: accuracy, F1, AUC on out-of-sample data
5. **Explain predictions with SHAP**

Why random forests:

- Handle non-linearities and feature interactions natively
- Robust to outliers (common in crypto)
- Feature importance is built in (Gini, permutation)
- SHAP values provide *instance-level* explanations

Reference: [4] for ML best practices in financial applications.

SHAP: SHapley Additive exPlanations

Based on cooperative game theory (Shapley 1953):

SHAP Value for Feature i

$$\phi_i = \sum_{S \subseteq \{1,\ldots,n\} \setminus \{i\}} \frac{|S|!\,(n - |S| - 1)!}{n!} \left[f(S \cup \{i\}) - f(S) \right]$$

Interpretation:

- ϕ_i = average marginal contribution of feature i across all possible feature coalitions S
- $f(S)$: model prediction using only features in S
- Satisfies: $\sum_i \phi_i = f(\boldsymbol{x}) - \mathbb{E}[f(\boldsymbol{x})]$ (prediction = sum of contributions)

For financial models:

- "This BTC sell prediction is driven primarily by high VPIN ($\phi_{\text{VPIN}} = -0.12$) and elevated Amihud ($\phi_{\text{ILLIQ}} = -0.08$)"
- Enables regulatory explainability and model debugging

Backtesting Pitfalls

The Five Deadly Sins of Backtesting

1. **Lookahead bias**: using future information in feature construction (e.g., end-of-day volume at intraday decision time)
2. **Survivorship bias**: only using tokens that survived to the present (ignores delisted tokens, failed projects)
3. **Transaction costs**: ignoring slippage, fees, market impact $\Rightarrow$ paper profits vanish in practice
4. **Multiple testing**: trying 1000 strategies, reporting the best one (inflated Sharpe). Deflated Sharpe ratio [4]:

$$\text{DSR} = \mathbb{P}\big(\text{SR}^* > 0 \;\big|\; \text{tried } N \text{ strategies}\big)$$

5. **Regime overfitting**: tuning to a specific market regime (e.g., 2020–2021 bull run) that does not repeat

Summary & Day 5 Preview

Today's key results:

1. Kyle's $\lambda = \sigma_v/(2\sigma_u)$: price impact from informed trading; extends to crypto with MEV and fragmentation
2. VPIN, Amihud, Roll: computable microstructure measures that predict crypto dynamics ([1])
3. RL strongest for market making and execution; fragile for alpha
4. Transformers capture long-range LOB dependencies; Hawkes processes model event clustering
5. SHAP provides economically interpretable ML explanations

Required Reading:

- [3] — Continuous auctions and insider trading
- [1] — ML with microstructure in crypto
- [2] — RL in financial decision making (survey)

Day 5 Preview: DeFi Lending, Systemic Risk, and Network Analysis — Eisenberg–Noe, liquidation cascades, protocol interconnections.

References I

[1] David Easley, Maureen O'Hara, Liyan Yang, and Zhuo Zhang. *Microstructure and Market Dynamics in Crypto*. SSRN 4814346. 2024.

[2] Petter N. Kolm and Gordon Ritter. *Reinforcement Learning in Financial Decision Making*. 2025. arXiv: 2512.10913 [q-fin.CP].

[3] Albert S. Kyle. "Continuous Auctions and Insider Trading". In: *Econometrica* 53.6 (1985), pp. 1315–1335.

[4] Marcos Lopez de Prado. *Advances in Financial Machine Learning*. Wiley, 2018.

[5] Igor Makarov and Antoinette Schoar. "Trading and Arbitrage in Cryptocurrency Markets". In: *Journal of Financial Economics* 135.2 (2020), pp. 293–319.

[6] Maureen O'Hara. *Market Microstructure Theory*. Wiley, 1995.

[7] Justin Sirignano and Rama Cont. "Deep Limit Order Book Forecasting". In: *Quantitative Finance* (2025).