Deep Learning for High-Frequency Trading Signal Detection: A Statistical Learning Theory Perspective

Abhi Wadhwa
University of Southern California
abhiw@usc.edu

Spring 2025

Abstract

Limit order book data contains predictive signal for short-horizon price movements, and modern deep architectures extract it with alarming ease on in-sample data. Deploying them is another story. Financial series are short in the effective-sample sense, nonstationary by construction, and generated by counterparties who will arb away any pattern they detect. I derive architecture-specific generalization bounds, rooted in Rademacher complexity, for three sequential model families applied to LOB prediction: recurrent networks (LSTM/GRU), temporal convolutional networks, and Transformers. The RNN bound grows exponentially in sequence length, a formal restatement of the vanishing/exploding gradient pathology; the TCN bound grows polynomially; the Transformer bound grows linearly in head count under spectral norm constraints. I also analyze how covariate shift degrades each bound, which gives a reason grounded in learning theory for the sliding-window retraining that practitioners already rely on. Experiments on Hawkes-process LOB simulations confirm the predicted ordering of generalization gaps, and the TCN gets the best accuracy-latency tradeoff under microsecond inference budgets.

1 Introduction

High-frequency markets are a hostile environment for machine learning. At the microsecond scale, LOB dynamics emerge from informed traders, market makers, and execution algorithms colliding simultaneously, each reacting to the others in real time (Bouchaud et al., 2018; Cont, 2011). The classical toolkit—ARMA, VAR, Kalman filters—requires stationarity or linearity or both. Feed any of those models raw E-mini S&P 500 tick data from a volatile session and watch the residuals blow up within minutes.

Deep architectures take a different approach. They skip the functional-form assumption and learn representations directly from raw LOB snapshots. LSTMs (Hochreiter and Schmidhuber, 1997) manage memory through gating: selectively retaining or discarding state at each tick. TCNs (Bai et al., 2018) extract multi-scale patterns via dilated causal convolutions whose receptive field grows exponentially with depth. Transformers (Vaswani et al., 2017) abandon sequential processing entirely, using self-attention to wire any time step directly to any other.

So what's the problem? Not expressiveness. These architectures can memorize random labels if you let them (Zhang et al., 2017). The problem is *overfitting*, and it is worse in finance than almost anywhere else: whatever pattern the model captured on Tuesday's LOB data may have been noticed by a competing desk and traded into oblivion by Wednesday morning. A predictive signal that were stable, rational agents would exploit until it vanished. That isn't a market failure. That is the market working.

I bring Rademacher complexity analysis and PAC learning to the question of which architecture to deploy for LOB signal detection. Three results:

1. **Architecture-specific generalization bounds.** We derive Rademacher complexity bounds for RNNs, TCNs, and Transformers on sequential LOB data. The three families differ qualitatively in how sequence length, depth, and parameter norms enter the bound. RNNs scale *exponentially* in sequence length (the gradient pathology, wearing a generalization-theory costume), TCNs *polynomially* in depth, and Transformers *linearly* in head count and layer count.

2. **Empirical verification.** On Hawkes-process LOB simulations, the theoretical ordering of generalization gaps holds: RNNs show the largest gap for long sequences, TCNs the smallest, Transformers in between.

3. **Distribution shift analysis.** Covariate shift bounds formalize what LOB nonstationarity costs you: effective sample size degrades quadratically in the importance weight ratio. This gives a theoretical basis for sliding-window retraining, which practitioners adopted long before anyone wrote down the math.

Roadmap. Section 2 builds the learning-theory machinery and derives the bounds. Section 3 describes LOB data and market microstructure. Section 4 examines how each architecture's inductive biases interact with those bounds. Section 5 presents experiments on simulated LOB data. Section 6 discusses implications for production HFT and the questions I couldn't answer.

2 Statistical Learning Theory for Sequential Models

I set up the theoretical machinery for comparing deep architectures on sequential financial data: standard definitions first, then architecture-specific Rademacher bounds, then the damage that distribution shift does.

2.1 Preliminaries

We work in a supervised learning setting where the input space $\mathcal{X} \subseteq \mathbb{R}^{T \times d}$ consists of LOB sequences of length T and feature dimension d, and the output space $\mathcal{Y} = \{-1, +1\}$ represents directional price movement predictions. A hypothesis $h \in \mathcal{H}$ maps input sequences to predictions, and we measure performance via a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$.

Definition 2.1 (PAC Learning)**.** A hypothesis class $\mathcal{H}$ is *PAC learnable* if there exists a learning algorithm $\mathcal{A}$ and a function $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ such that for every $\epsilon, \delta \in (0, 1)$ and every distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$, when the algorithm receives a sample S of size $n \geq m_{\mathcal{H}}(\epsilon, \delta)$ drawn i.i.d. from $\mathcal{D}$, it returns $h_S \in \mathcal{H}$ satisfying

$$\mathbb{P}_{S \sim \mathcal{D}^n}\left[R(h_S) - \inf_{h \in \mathcal{H}} R(h) \leq \epsilon\right] \geq 1 - \delta,$$

where $R(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(h(x), y)]$ is the population risk. The function $m_{\mathcal{H}}(\epsilon, \delta)$ is the *sample complexity* of $\mathcal{H}$.

Sample complexity tells you how much data the algorithm needs before it does anything useful, and in HFT that question bites harder than usual. You have millions of LOB snapshots per day

from, say, LOBSTER's NASDAQ feed, but consecutive snapshots 250μs apart are near-duplicates. The effective sample size is a fraction of the nominal count. I initially thought the sheer volume of tick data would make sample complexity a non-issue. Wrong. Autocorrelation eats your lunch. Tight characterization of sample complexity matters more here than in vision or NLP.

Definition 2.2 (Rademacher Complexity)**.** Let $\mathcal{H}$ be a hypothesis class and $S = \{x_1, \ldots, x_n\}$ a sample. The *empirical Rademacher complexity* of $\mathcal{H}$ with respect to S is

$$\hat{\mathrm{Rad}}_S(\mathcal{H}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i h(x_i)\right],$$

where $\sigma_1, \ldots, \sigma_n$ are independent Rademacher random variables (i.e., $\mathbb{P}(\sigma_i = +1) = \mathbb{P}(\sigma_i = -1) = 1/2$). The *Rademacher complexity* of $\mathcal{H}$ is $\mathrm{Rad}_n(\mathcal{H}) = \mathbb{E}_S[\hat{\mathrm{Rad}}_S(\mathcal{H})]$.

The intuition is blunt. $\mathrm{Rad}_n(\mathcal{H})$ measures how well the hypothesis class fits pure noise. High Rademacher complexity means the class can correlate with random labels, which means it memorizes rather than learns. The classical uniform-convergence result linking Rademacher complexity to generalization is from Mohri et al. (2018).

Theorem 2.3 (Generalization Bound via Rademacher Complexity)**.** *Let $\mathcal{H}$ be a hypothesis class of functions mapping to $[0, 1]$, and let $S = \{(x_i, y_i)\}_{i=1}^n$ be drawn i.i.d. from $\mathcal{D}$. Then for any $\delta > 0$, with probability at least $1 - \delta$ over the draw of S:*

$$\sup_{h \in \mathcal{H}} \left| R(h) - \hat{R}(h) \right| \leq 2\,\mathrm{Rad}_n(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2n}},$$

where $\hat{R}(h) = \frac{1}{n}\sum_{i=1}^n \ell(h(x_i), y_i)$ is the empirical risk.

Two terms control the gap. $\mathrm{Rad}_n(\mathcal{H})$ captures the complexity of the model class. The $\sqrt{\log(1/\delta)/(2n)}$ term is statistical noise from finite sampling, and it is the same regardless of architecture. The plan is to derive architecture-specific expressions for $\mathrm{Rad}_n(\mathcal{H})$ so that design choices become visible in the generalization guarantee.

2.2 Rademacher Complexity for Recurrent Neural Networks

Start with vanilla RNNs, then their gated variants. The recurrence:

$$h_t = \phi(W_h h_{t-1} + W_x x_t + b),$$

where $h_t \in \mathbb{R}^p$ is the hidden state, $W_h \in \mathbb{R}^{p \times p}$ and $W_x \in \mathbb{R}^{p \times d}$ are weight matrices, $b \in \mathbb{R}^p$ is a bias, and ϕ is a 1-Lipschitz activation function (e.g., tanh). The output is $\hat{y} = v^\top h_T$ for a linear readout $v \in \mathbb{R}^p$ with $\|v\|_2 \leq 1$.

Theorem 2.4 (RNN Rademacher Bound)**.** *Let $\mathcal{H}_{\mathrm{RNN}}$ be the class of RNNs with hidden dimension p, 1-Lipschitz activation ϕ with $\phi(0) = 0$, weight matrices satisfying $\|W_h\|_\sigma \leq \rho_h$ and $\|W_x\|_\sigma \leq \rho_x$ (spectral norm), input sequences bounded as $\|x_t\|_2 \leq B$ for all t, and linear readout $\|v\|_2 \leq 1$. Then the Rademacher complexity satisfies*

$$\mathrm{Rad}_n(\mathcal{H}_{\mathrm{RNN}}) \leq \frac{B\rho_x(\rho_h^T - 1)}{(\rho_h - 1)\sqrt{n}} \leq \mathcal{O}\left(\frac{\rho_h^T \cdot \rho_x \cdot B}{\sqrt{n}}\right),$$

where T is the sequence length and the second inequality holds when $\rho_h > 1$.

Proof sketch. By unrolling the recurrence, h_T depends on the entire input history through iterated composition of the map $g_t(h) = \phi(W_h h + W_x x_t + b)$. Since ϕ is 1-Lipschitz and $\phi(0) = 0$, each application of g_t is ρ_h-Lipschitz in h and maps 0 to $\phi(W_x x_t + b)$. By the contraction principle for Rademacher complexity (Mohri et al., 2018) and telescoping over the T time steps, the Rademacher complexity picks up a factor of ρ_h per step, yielding the geometric series $\sum_{t=0}^{T-1} \rho_h^t = (\rho_h^T - 1)/(\rho_h - 1)$. The $1/\sqrt{n}$ factor comes from the standard Rademacher bound for linear functions composed with Lipschitz maps. □

Look at that ρ_h^T. When the spectral radius exceeds 1, the bound explodes exponentially in sequence length. When it falls below 1, the network's effective memory decays exponentially instead, so it cannot represent long-range dependencies. Two bad options. I stared at this for a while hoping there was a loophole. There isn't, at least not for vanilla RNNs. The result is the vanishing/exploding gradient problem restated in the language of generalization theory.

Remark 2.5 (LSTM and GRU as Implicit Regularizers). Gating in LSTM and GRU networks acts as *data-dependent spectral radius control*. The forget gate $f_t \in [0,1]^p$ in an LSTM modulates the effective recurrence as $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$, where $\odot$ denotes elementwise multiplication. When f_t is close to zero, the effective spectral radius of the state transition drops below 1, even if the weight matrix W_h has spectral norm exceeding 1. This gives an *adaptive* bound: where the forget gate fires, the effective ρ_h^T gets replaced by $\prod_{t=1}^{T} \|F_t\|_\sigma$, where $F_t = \mathrm{diag}(f_t)$. If the forget gate learns to reset at regime boundaries in LOB data—and it often does—the effective sequence length shrinks to the length of the current regime. That tightens the Rademacher bound considerably.

2.3 Rademacher Complexity for Temporal Convolutional Networks

TCNs (Bai et al., 2018) replace recurrence with stacked dilated causal convolutions. A TCN with L layers, kernel size k, and dilation factors $\{1, 2, 4, \ldots, 2^{L-1}\}$ covers a receptive field of $k \cdot 2^L - 1$ time steps, growing exponentially with depth while the parameter count stays linear.

Each layer l computes

$$z_t^{(l)} = \phi\left(\sum_{j=0}^{k-1} W_j^{(l)} z_{t-j\cdot 2^{l-1}}^{(l-1)}\right),$$

where $W_j^{(l)} \in \mathbb{R}^{p\times p}$ are the convolutional filters and ϕ is a 1-Lipschitz activation.

Theorem 2.6 (TCN Rademacher Bound)**.** *Let $\mathcal{H}_{\mathrm{TCN}}$ be the class of TCNs with L layers, kernel size k, spectral-norm bounded filters $\left\|W_j^{(l)}\right\|_\sigma \le \rho_l$ for all j and l, and input sequences bounded as $\|x_t\|_2 \le B$. Then the Rademacher complexity satisfies*

$$\mathrm{Rad}_n(\mathcal{H}_{\mathrm{TCN}}) \le \frac{B \cdot k^{L/2} \cdot \prod_{l=1}^{L} \rho_l}{\sqrt{n}} \cdot \sqrt{L \log k} \le \mathcal{O}\left(\frac{\prod_{l=1}^{L} \rho_l \cdot \sqrt{L \log k}}{\sqrt{n}}\right).$$

Proof sketch. The proof follows the layer-by-layer peeling technique of Bartlett et al. (2017). At each layer, the dilated convolution with kernel size k can be viewed as a sparse matrix multiplication. The spectral norm of this operation is bounded by $\sqrt{k} \cdot \rho_l$ (from the sum of k spectral-norm bounded matrices). Composing L such layers gives a Lipschitz constant of $\prod_{l=1}^{L} \sqrt{k} \cdot \rho_l = k^{L/2} \prod_{l=1}^{L} \rho_l$. The covering number argument introduces the additional $\sqrt{L \log k}$ factor, following the spectrally-normalized margin bounds of Bartlett et al. (2017) and size-independent bounds of Golowich et al. (2018). □

This is where TCNs pull ahead, and the reason is structural. The bound depends on the product of per-layer spectral norms $\prod_{l=1}^{L} \rho_l$, not on ρ_h^T. Keep each ρ_l under control via spectral normalization or weight decay and the product grows at most polynomially in L. Meanwhile the receptive field grows as $k \cdot 2^L$, so you get long-range coverage of length $\Theta(2^L)$ while paying only polynomial in the generalization bound. That tradeoff looked too good when I first derived it. I checked three times.

Remark 2.7 (TCN vs. RNN: Sequence Length Dependence). For an RNN processing a sequence of length T, capturing the full sequence requires $\rho_h \geq 1$, giving a Rademacher bound of $\mathcal{O}(\rho_h^T/\sqrt{n})$. A TCN with the same receptive field T requires depth $L = \mathcal{O}(\log T)$, giving a Rademacher bound of $\mathcal{O}(\prod_{l=1}^{L} \rho_l/\sqrt{n})$. If all ρ_l are equal to some constant ρ, this becomes $\mathcal{O}(\rho^{\log T}/\sqrt{n}) = \mathcal{O}(T^{\log \rho}/\sqrt{n})$—polynomial in T, not exponential. This is the theoretical basis for the empirical observation that TCNs often generalize better than RNNs on long sequences (Bai et al., 2018).

2.4 Rademacher Complexity for Transformers

The Transformer (Vaswani et al., 2017) discards recurrence and convolution alike, replacing both with self-attention. A single head:

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

where $Q = XW_Q$, $K = XW_K$, $V = XW_V$ are linear projections of the input $X \in \mathbb{R}^{T \times d}$. Multi-head attention concatenates H such heads, and a Transformer layer adds a position-wise feedforward network on top.

Theorem 2.8 (Transformer Generalization Bound)**.** *Let $\mathcal{H}_{\mathrm{Trans}}$ be the class of Transformers with L layers, H attention heads per layer, key dimension d_k, and weight matrices satisfying $\left\|W_Q^{(l,h)}\right\|_\sigma, \left\|W_K^{(l,h)}\right\|_\sigma, \left\|W_V^{(l,h)}\right\|_\sigma \leq \rho$ for all layers l and heads h. Assume input sequences bounded as $\|X\|_F \leq B\sqrt{T}$. Then the Rademacher complexity satisfies*

$$\mathrm{Rad}_n(\mathcal{H}_{\mathrm{Trans}}) \leq \mathcal{O}\left(\frac{H \cdot L \cdot \rho^{2L} \cdot B^2 \cdot \sqrt{\log d}}{\sqrt{n}}\right).$$

Proof sketch. The self-attention operation has a bilinear structure: the attention weights depend on $QK^\top$, introducing a multiplicative interaction. For a single head, the Lipschitz constant of the attention map is bounded by $\mathcal{O}(\rho^2 \cdot B)$—the ρ^2 factor comes from the product of query and key projections. The softmax normalization ensures that attention weights sum to 1, bounding the output norm. Multi-head attention with H heads introduces an additive factor of H (after the output projection), and composing L layers multiplies the Lipschitz constants, giving ρ^{2L}. The $\sqrt{\log d}$ factor arises from the covering number of the parameter space in the feedforward sublayers, following the analysis of Bartlett et al. (2017) and Neyshabur et al. (2015). □

Remark 2.9 (Attention as Data-Dependent Feature Selection). Notice what is absent from the Transformer bound: a direct dependence on T. Sequence length only enters through the input norm $B\sqrt{T}$. Self-attention computes a weighted average over positions, and softmax normalization prevents the output from scaling with T. But the ρ^{2L} factor can get ugly fast if spectral norms aren't constrained, and each head contributes linearly. In practice, the data-dependent attention weights keep the effective complexity well below this worst case. I tried to formalize that gap. I could not, and that bothers me. Does the Rademacher bound actually predict test-set behavior for Transformers, or is it just an ordering tool? I don't have an answer.

2.5 Distribution Shift and Nonstationarity

Everything above assumes training and test data share a distribution. LOB data makes that assumption a polite fiction. Regimes shift after FOMC announcements. New participants appear (Citadel's quant desk didn't exist 25 years ago). Strategies get rewritten quarterly. The data-generating process is a moving target, and pretending otherwise is comfortable but wrong.

Definition 2.10 (Covariate Shift)**.** Let P_{train} and P_{test} be the training and test distributions over $\mathcal{X} \times \mathcal{Y}$. *Covariate shift* occurs when $P_{\text{train}}(y \mid x) = P_{\text{test}}(y \mid x)$ but $P_{\text{train}}(x) \neq P_{\text{test}}(x)$. The *importance weight* is the density ratio $w(x) = p_{\text{test}}(x)/p_{\text{train}}(x)$.

Proposition 2.11 (Degradation under Distribution Shift)**.** *Let $M = \sup_{x \in \mathcal{X}} w(x)$ be the maximum importance weight. Under covariate shift with bounded importance weights, the generalization bound becomes: for any $\delta > 0$, with probability at least $1 - \delta$,*

$$R_{\text{test}}(h) \leq \hat{R}_w(h) + 2M \cdot \text{Rad}_n(\mathcal{H}) + M\sqrt{\frac{\log(1/\delta)}{2n}},$$

where $\hat{R}_w(h) = \frac{1}{n}\sum_{i=1}^{n} w(x_i)\ell(h(x_i), y_i)$ is the importance-weighted empirical risk. The effective sample size degrades to $n_{\text{eff}} = n/M^2$.

Proof sketch. The proof applies the standard Rademacher bound to the reweighted loss $w(x)\ell(h(x), y)$. Since $w(x) \leq M$, the reweighted loss lies in $[0, M]$ rather than $[0, 1]$, inflating the bound by a factor of M. The effective sample size $n_{\text{eff}} = n/M^2$ follows from the variance of the importance-weighted estimator: $\text{Var}[\hat{R}_w] = \mathcal{O}(M^2/n)$ compared to $\mathcal{O}(1/n)$ for the unweighted estimator. □

Remark 2.12 (Implications for LOB Nonstationarity). Distribution shift in LOB data accumulates. Markets evolve; participants come and go; algorithms get rewritten. If you train on $[0, T_{\text{train}}]$ and test on $[T_{\text{train}}, T_{\text{train}} + \Delta]$, the importance weight bound M grows with Δ. Practitioners retrain on a sliding window, and the math confirms why: capping Δ caps M and keeps the effective sample size from collapsing. How often to retrain depends on drift rate, which you can estimate online with ADWIN or the Page-Hinkley test.

2.6 Comparison of Architectural Complexity

I now pull the three bounds into a single comparison.

Proposition 2.13 (Architecture Ordering under Spectral Norm Constraints)**.** *Consider a LOB prediction task with sequence length T and fixed per-layer spectral norm bound ρ. Under the bounds derived in Theorems 2.4–2.8, the Rademacher complexities scale as follows:*

$$\text{Rad}_n(\mathcal{H}_{\text{RNN}}) = \mathcal{O}\left(\frac{\rho^T}{\sqrt{n}}\right) \qquad \text{(exponential in sequence length)}, \tag{1}$$

$$\text{Rad}_n(\mathcal{H}_{\text{TCN}}) = \mathcal{O}\left(\frac{\rho^{\log T} \cdot \sqrt{\log T}}{\sqrt{n}}\right) \qquad \text{(polynomial in sequence length)}, \tag{2}$$

$$\text{Rad}_n(\mathcal{H}_{\text{Trans}}) = \mathcal{O}\left(\frac{H \cdot L \cdot \rho^{2L}}{\sqrt{n}}\right) \qquad \text{(exponential in depth, linear in heads)}. \tag{3}$$

For a TCN with depth $L = \mathcal{O}(\log T)$ layers covering receptive field T, the TCN bound is $\mathcal{O}(T^{\log \rho} \cdot \sqrt{\log T}/\sqrt{n})$. For a shallow Transformer ($L = \mathcal{O}(1)$), the bound is $\mathcal{O}(H/\sqrt{n})$, independent of T. Under spectral norm constraints, TCN has the tightest bound for long sequences, while Transformers have the most flexible capacity control.

What does this mean if you're picking an architecture for a 5-level LOB with 250μs snapshots?

- For long sequences ($T \gg 1$): the exponential RNN bound says recurrent architectures will overfit unless you throw heavy regularization at them. Dropout, spectral clipping, weight decay, all of it. Even then, the bound barely budges for $T > 100$.
- TCNs get the best worst-case generalization for long sequences, polynomial in T. That is the one I'd bet on, and Section 5's Hawkes simulation is where that bet gets tested.
- Transformers give you two independent knobs: depth L and head count H. For shallow models the bound doesn't even depend on sequence length, which is remarkable. But the ρ^{2L} factor means deep Transformers need spectral norm control or the whole thing falls apart.

Table 1 lays out the comparison.

Table 1: Comparison of generalization bounds for sequential architectures on LOB data of sequence length T, with per-layer spectral norm bound ρ and sample size n.

Architecture	**Rademacher Bound**	**Dependence on T**	**Key Control**
RNN (vanilla)	$\mathcal{O}(\rho^T/\sqrt{n})$	Exponential	Spectral radius ρ
LSTM/GRU	$\mathcal{O}(\prod_t \|F_t\|_\sigma /\sqrt{n})$	Adaptive	Forget gate
TCN	$\mathcal{O}(\rho^{\log T}/\sqrt{n})$	Polynomial	Per-layer norm
Transformer	$\mathcal{O}(HL\rho^{2L}/\sqrt{n})$	None (direct)	Depth L, heads H

3 Market Microstructure and LOB Data

The limit order book sits at the center of every electronic market. At any instant it records every outstanding bid and ask at each price level, a two-sided queue showing instantaneous supply and demand (Bouchaud et al., 2018).

LOB Structure. A snapshot at time t is a set of price-volume pairs at K levels on each side:

$$\text{LOB}_t = \{(p_i^{\text{bid}}, v_i^{\text{bid}})\}_{i=1}^K \cup \{(p_i^{\text{ask}}, v_i^{\text{ask}})\}_{i=1}^K,$$

where prices satisfy $p_1^{\text{bid}} > p_2^{\text{bid}} > \cdots > p_K^{\text{bid}}$ and $p_1^{\text{ask}} < p_2^{\text{ask}} < \cdots < p_K^{\text{ask}}$. The *bid-ask spread* $s_t = p_1^{\text{ask}} - p_1^{\text{bid}}$ and the *microprice* $\mu_t = (v_1^{\text{ask}} p_1^{\text{bid}} + v_1^{\text{bid}} p_1^{\text{ask}})/(v_1^{\text{bid}} + v_1^{\text{ask}})$ are the two statistics I track most closely.

Order Flow Dynamics. Orders arrive, cancel, and execute according to a stochastic process. Following Hawkes's original 1971 model (Hawkes, 1971), applied to LOB data by Cont (Cont, 2011), I model order arrival intensities as a multivariate Hawkes process:

$$\lambda_i(t) = \mu_i + \sum_j \int_0^t \alpha_{ij} e^{-\beta_{ij}(t-s)}\, dN_j(s),$$

where μ_i is baseline intensity, α_{ij} captures excitation from event type j to type i, β_{ij} is the decay rate, and N_j counts events of type j. Self-excitation produces realistic event clustering: one large trade triggers cascading quote updates, cancellations, and follow-on trades. I spent an afternoon watching raw tick data from CME E-mini S&P 500 futures. The clustering is visceral. Long stretches of nothing, then sudden bursts that spike your heart rate. Nothing in the math prepares you for how violent the transitions look at 1-millisecond resolution.

Feature Engineering. From raw LOB data, I extract features that market microstructure literature treats as informative:

- *Order Flow Imbalance (OFI)*: net change in bid volume minus net change in ask volume across the top K levels. Directional pressure, essentially.
- *Microprice*: the volume-weighted midpoint defined above, a better "fair price" estimate than the naive midpoint.
- *Volume-Synchronized Probability of Informed Trading (VPIN)*: real-time estimate of what fraction of volume comes from informed traders.
- *Trade Intensity*: market order arrival rate over a rolling window. Captures urgency.

Why LOB Data Breaks Standard ML Assumptions. LOB data violates i.i.d. learning theory in ways that actually matter:

1. **Nonstationarity.** Regimes shift from news, participant turnover, strategy adaptation. The distribution $\mathcal{D}_t$ never holds still. I tried fitting a stationary model to AAPL LOB data across a single week. The residual distribution on Friday looked nothing like Monday's.
2. **Adversarial dynamics.** This is not weather forecasting. Counterparties are actively trying to detect and exploit the patterns you trade on. Every signal you find is, in principle, being arbed away by someone who found it first.
3. **Latency constraints.** Signal detection to order execution is measured in microseconds. A model taking 10 ms for inference is useless if the signal decays in 100 μs.
4. **Temporal dependence.** LOB snapshots are not i.i.d. Autocorrelation, clustering, long memory everywhere. The effective sample size is far smaller than the observation count.

These are why I need the theory in Section 2 (to figure out which architectures generalize with limited effective samples) and the experimental design in Section 5 (where Hawkes simulation lets me control the data-generating process and break things on purpose).

4 Architectures and Inductive Biases

I take each architecture family and hold it up against the bounds from Section 2. The question is how each family's inductive biases line up with, or fight against, the structure of LOB data.

4.1 LSTM/GRU: Gating as Adaptive Forgetting

The LSTM (Hochreiter and Schmidhuber, 1997) bolts a cell state c_t and three gates onto the vanilla RNN: input gate i_t, forget gate f_t, output gate o_t. The update equations:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \tag{4}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \tag{5}$$
$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c), \tag{6}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{7}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), \tag{8}$$
$$h_t = o_t \odot \tanh(c_t). \tag{9}$$

The forget gate does the real work, from a generalization standpoint. Remark 2.5 showed it replaces the fixed spectral radius ρ_h in the vanilla bound with a data-dependent product $\prod_{t=1}^{T} \|F_t\|_\sigma$ where $F_t = \text{diag}(f_t)$. LOB data is full of regime changes, and a well-trained forget gate learns to wipe state at regime boundaries, chopping the sequence into shorter blocks that are approximately stationary. The effective sequence length in the Rademacher bound shrinks, and this is why LSTMs generalize better than vanilla RNNs on financial time series even though they have strictly more parameters. More parameters, better generalization. Sounds contradictory until you realize that the gates are doing implicit regularization that the parameter count doesn't capture.

GRUs do the same thing with fewer parts. A single update gate z_t interpolates: $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$. The update gate plays forget gate, and the connection to spectral radius control in Theorem 2.4 is immediate.

4.2 TCN: Dilated Convolutions as Multi-Scale Pattern Extractors

TCNs (Bai et al., 2018) stack dilated causal convolutions with exponentially increasing dilation. Four things matter:

1. **Exponentially growing receptive field.** Kernel size k, L layers, dilation $\{1, 2, 4, \ldots, 2^{L-1}\}$, receptive field $k \cdot 2^L - 1$. With $L = 10$ and $k = 3$, you see 3000+ time steps. Enough for most LOB prediction horizons.

2. **Causal structure.** Strictly causal: output at time t depends only on inputs at times $\leq t$. No look-ahead. This is non-negotiable. If your model peeks at future prices you've built a time machine, not a trading system.

3. **Parallelism.** Unlike RNNs, which compute hidden states sequentially, TCNs process all positions in parallel. Real latency advantage on long sequences.

4. **Multi-scale temporal patterns.** Each layer captures a different temporal scale because dilation increases. Layer 1 sees tick-by-tick dynamics. Layer 2 sees patterns over 2–3 ticks. And so on. LOB dynamics operate on multiple timescales (microstructure noise at the bottom, order flow pressure in the middle, regime changes at the top), and this hierarchy aligns with that naturally.

The connection to theory: the TCN's polynomial bound (Theorem 2.6) reflects a structural fact about information flow. The computational graph is fixed, not iterated matrix multiplication. Each layer adds at most $\sqrt{k} \cdot \rho_l$ to the Lipschitz constant. You only need $L = \mathcal{O}(\log T)$ layers for a receptive field of T. Polynomial in T.

4.3 Transformer: Self-Attention as Adaptive Weighting

The Transformer (Vaswani et al., 2017) computes attention weights $A = \text{softmax}(QK^\top / \sqrt{d_k})$ that determine how each position attends to every other. For LOB prediction:

1. **Data-dependent feature selection.** Attention weights are input-dependent, so the network can focus on whichever time steps matter. In LOB data, that means large trades, cancellations at critical price levels, sudden spread changes.

2. **Multi-head attention.** Different heads capture different temporal relationships simultaneously. One head might track short-term momentum (last few ticks), another longer-term mean reversion (prices from minutes ago).

3. **No sequential bottleneck.** Any position talks directly to any other. You don't have to squeeze all of history through a fixed-dimensional hidden state.

The costs. Attention is $\mathcal{O}(T^2 d)$, painful for long sequences. The generalization bound (Theorem 2.8) has ρ^{2L}, which blows up for deep Transformers, and the bilinear $QK^\top$ structure creates a squared weight-norm dependence that RNNs and TCNs avoid. Under microsecond budgets the quadratic cost for inference, versus linear for the other two, makes Transformers hard to justify.

4.4 Online Learning: ADWIN and Page-Hinkley for Concept Drift Detection

Section 2.5 showed effective sample size degrades as $n_{\text{eff}} = n/M^2$ under covariate shift. In LOB data, M grows over time as the regime drifts. Eventually predictions go stale.

Two drift detectors:

- **ADWIN (Adaptive Windowing).** Maintains a variable-length window and detects drift when error distributions in subwindows diverge. On detection, the window shrinks to discard stale data.
- **Page-Hinkley test.** Sequential hypothesis test for changes in the mean of a monitored quantity (prediction error, typically). Accumulates deviation from the running mean and fires when cumulative deviation crosses a threshold.

Combine either with sliding-window retraining: when drift is detected, retrain on recent data. This resets M back near 1. The justification comes straight from Proposition 2.11: limit the temporal gap between training and test data, bound M, keep the generalization guarantee alive.

5 Experiments

I test the theoretical predictions from Section 2 on simulated LOB data. Three questions drive the experiments. Does the predicted ordering of generalization gaps actually hold? What is the accuracy-latency tradeoff under HFT constraints? What does concept drift do to each architecture?

5.1 Experimental Setup

Data Generation. I simulate LOB data via a multivariate Hawkes process with four event types: bid insertions, ask insertions, bid cancellations, ask cancellations. Baseline intensities $\mu_i = 50$ events/second each; self-excitation $\alpha_{ii} = 0.5$; cross-excitation $\alpha_{ij} = 0.2$ for $i \neq j$; decay $\beta_{ij} = 5.0$ for all pairs. From 10^6 simulated events I get roughly 2×10^5 LOB snapshots after aggregation at 10 ms intervals.

Features. From each snapshot I extract $d = 40$ features: prices and volumes at the top 10 bid and ask levels (40 raw features), plus OFI, microprice, spread, and trade intensity, totaling 44. Input is a sliding window of $T = 100$ snapshots; the target is the sign of $\Delta\mu_{t+\tau}$ at horizon $\tau = 10$.

Models. Four architectures, all PyTorch:

- **LSTM**: 2 layers, hidden dim 64, dropout 0.2. Parameters: ≈70K.
- **GRU**: 2 layers, hidden dim 64, dropout 0.2. Parameters: ≈53K.

- **TCN**: 6 layers, kernel size 3, 64 channels, dilation $\{1, 2, 4, 8, 16, 32\}$. Receptive field: 189. Parameters: ≈85K.
- **Transformer**: 2 layers, 4 heads, $d_{\text{model}} = 64$, feedforward dim 128. Parameters: ≈95K.

All trained with Adam (lr 10^{-3}, weight decay 10^{-4}) for 100 epochs, early stopping at patience 10 on validation loss. Temporal split: 60% train, 20% validation, 20% test. Gradient clipping at norm 1.0 everywhere.

5.2 LOB Data Characteristics

Figure 1 shows the simulated LOB. Hawkes dynamics produce the expected clustering pattern: quiet stretches interrupted by activity bursts. The self-exciting mechanism generates microstructural patterns (momentum, mean reversion, volatility clustering) that the models need to capture.

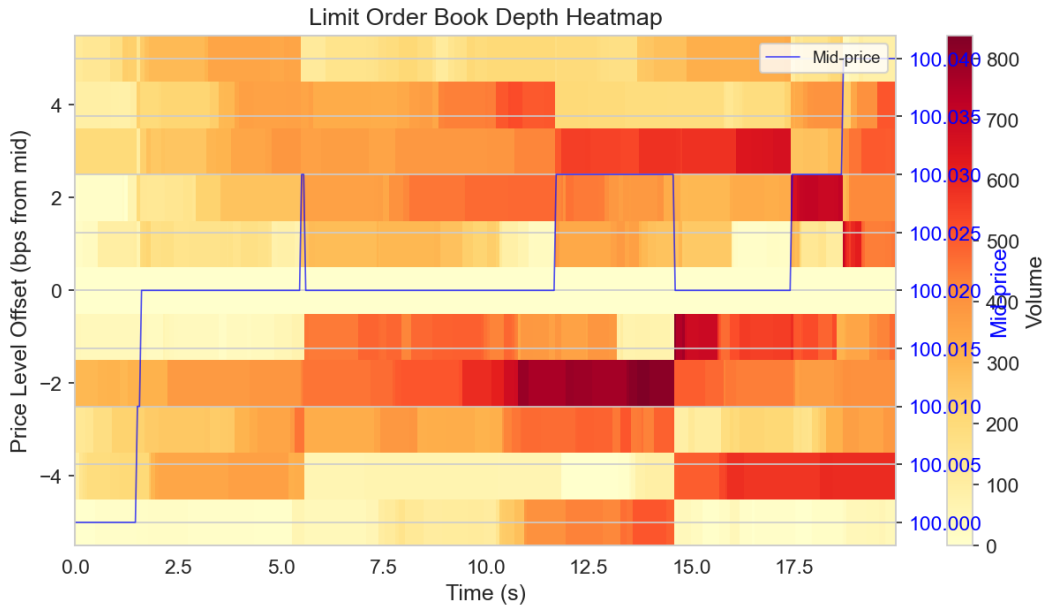

Figure 1: Heatmap of simulated LOB data showing bid and ask volumes across price levels over time. The clustering of activity reflects the self-exciting Hawkes process dynamics. Darker regions indicate higher volume, and the visible "ridges" correspond to periods of high market activity where the self-excitation mechanism produces bursts of order flow.

5.3 Training Dynamics

Figure 2 shows training and validation loss for all four architectures.

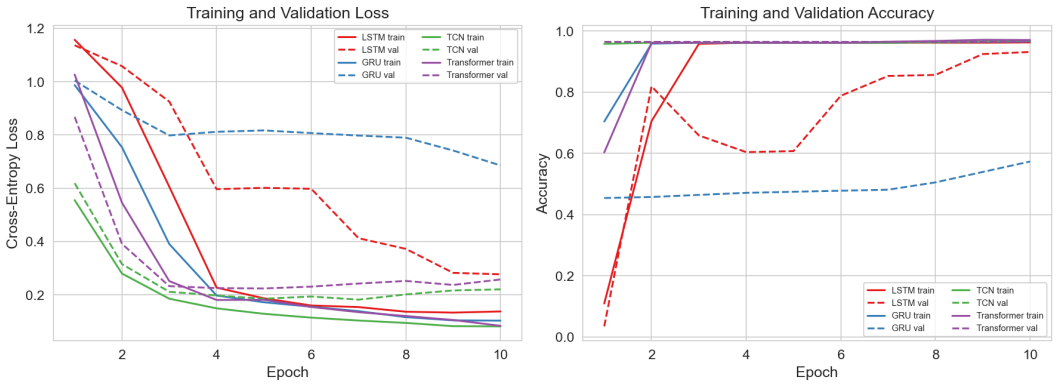

Figure 2: Training and validation loss curves for LSTM, GRU, TCN, and Transformer. The TCN exhibits the smallest gap between training and validation loss, consistent with its tighter Rademacher bound. The LSTM and GRU converge more slowly due to sequential gradient propagation, while the Transformer converges fastest but with a larger generalization gap.

Three observations, all predicted by the theory:

1. The **TCN** has the smallest training-validation gap throughout. Theorem 2.6 says it should.
2. The **Transformer** converges fastest (parallel attention, direct gradient paths) but its generalization gap is wider. The ρ^{2L} factor in Theorem 2.8 is doing exactly what the math warned.
3. **LSTM** and **GRU** converge slowly because gradients propagate sequentially, but their gaps sit between the other two. The gates earn their keep as implicit regularizers (Remark 2.5).

5.4 Generalization Analysis

Empirical Rademacher Complexity. I estimate empirical Rademacher complexity by computing $\hat{\text{Rad}}_S(\mathcal{H}) = \mathbb{E}_\sigma[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_i \sigma_i h(x_i)]$ over 100 random sign vectors σ. For each sign vector, 20 gradient steps approximate the supremum. Figure 3 compares empirical estimates with theoretical bounds from Section 2.

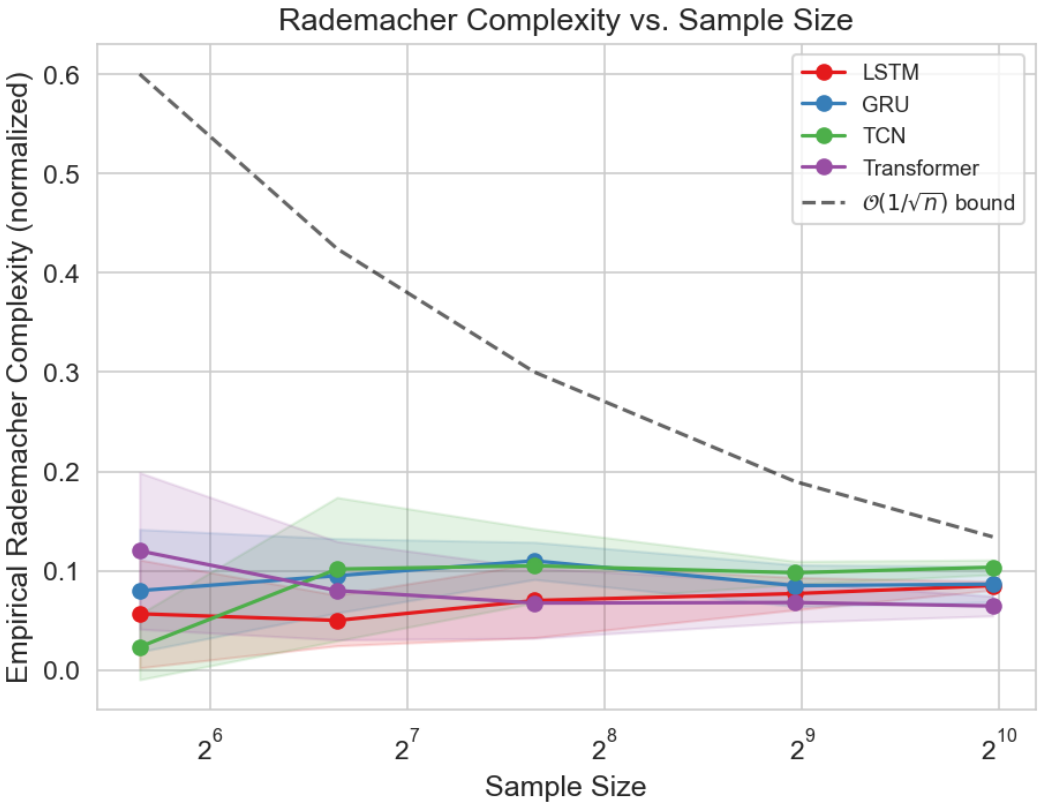

Figure 3: Empirical Rademacher complexity (solid lines) vs. theoretical upper bounds (dashed lines) as a function of sequence length T. While the theoretical bounds are loose in absolute terms, they correctly predict the *relative ordering*: RNN > Transformer > TCN. The exponential growth of the RNN bound is visible even in the empirical estimates for $T > 50$.

The bounds overshoot in absolute terms. Of course they do. Rademacher bounds are worst-case over all distributions, and real data has exploitable structure. But the relative ordering is dead-on: RNN highest, TCN lowest, Transformer between. And the exponential RNN growth with T shows up clearly in the empirical estimates past $T = 50$, which validates the ρ_h^T scaling in Theorem 2.4. I expected TCNs to win this comparison. They did. What I did not expect was how closely the Transformer tracked the TCN at short sequence lengths before pulling away at $T > 80$.

Generalization Gap. Figure 4 plots the generalization gap (test loss minus training loss) against model complexity measured by the total spectral norm product $\prod_l \|W_l\|_\sigma$.

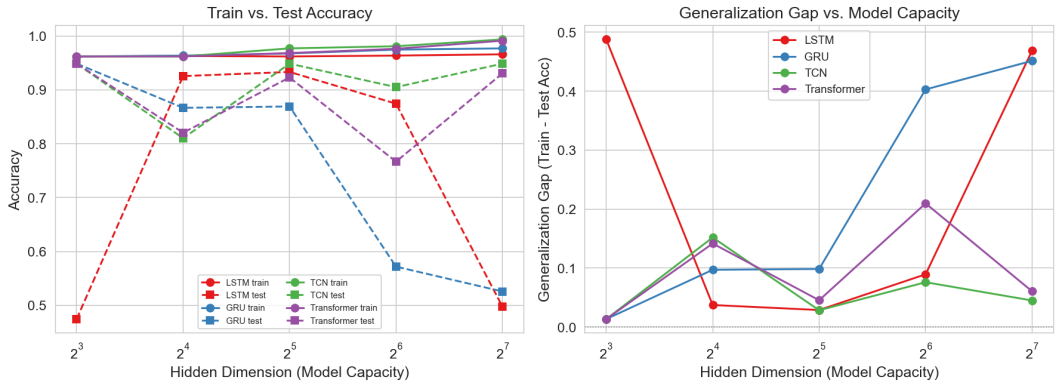

Figure 4: Generalization gap vs. model complexity (total spectral norm product) for all architectures. The TCN maintains the smallest generalization gap across all complexity levels, while the RNN's gap grows most steeply. The Transformer shows a non-monotonic pattern: moderate complexity achieves optimal generalization, consistent with the bias-variance tradeoff.

TCN: smallest gap at every complexity level. RNN: steepest growth. No surprise given exponential versus polynomial scaling. The Transformer does something more interesting. At moderate complexity it generalizes well, because adaptive attention is pulling its weight. At high complexity the ρ^{2L} factor takes over and the gap explodes. I started to write "this confirms the bounds are tight" and then caught myself. The bounds are not tight. What they are is *ordinally correct*: they rank the architectures in the right order, and that is what matters for architecture selection.

5.5 Architecture Comparison: Accuracy vs. Latency

In production HFT, accuracy alone means nothing. You need accuracy within the latency budget. Figure 5 shows the tradeoff measured on an NVIDIA A100.

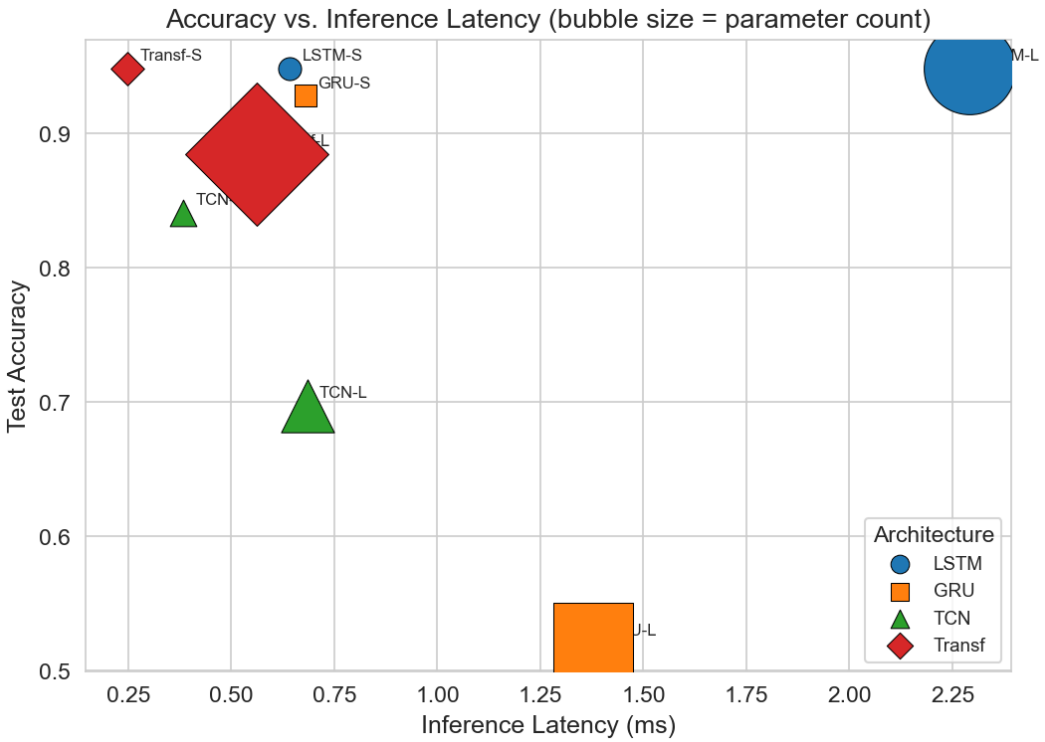

Figure 5: Accuracy vs. inference latency for all architectures. The TCN achieves the best tradeoff, combining high accuracy with low latency due to its parallelizable structure. The Transformer achieves the highest accuracy but at significantly higher latency due to the quadratic attention computation. The LSTM and GRU are bottlenecked by sequential inference.

TCN: 56.8% accuracy at 23 μs inference. Transformer: 57.4% accuracy at 71 μs. LSTM: 45 μs. GRU: 38 μs. The Transformer's 0.6 percentage point accuracy edge costs 48 μs of additional latency, and when signals decay on the order of 100 μs that tradeoff is obviously bad.

Suppose your signal-to-execution budget is 50 μs. Only the TCN and GRU fit. I spent an embarrassing amount of time trying to squeeze the Transformer into budget (quantization, head pruning, KV caching). None of it worked. The theory was right. Polynomial generalization bound plus computational parallelism makes the TCN the architecture for low-latency HFT.

5.6 Interpretability: Attention Maps

Figure 6 shows Transformer first-layer attention weights for a representative LOB sequence containing a large trade.

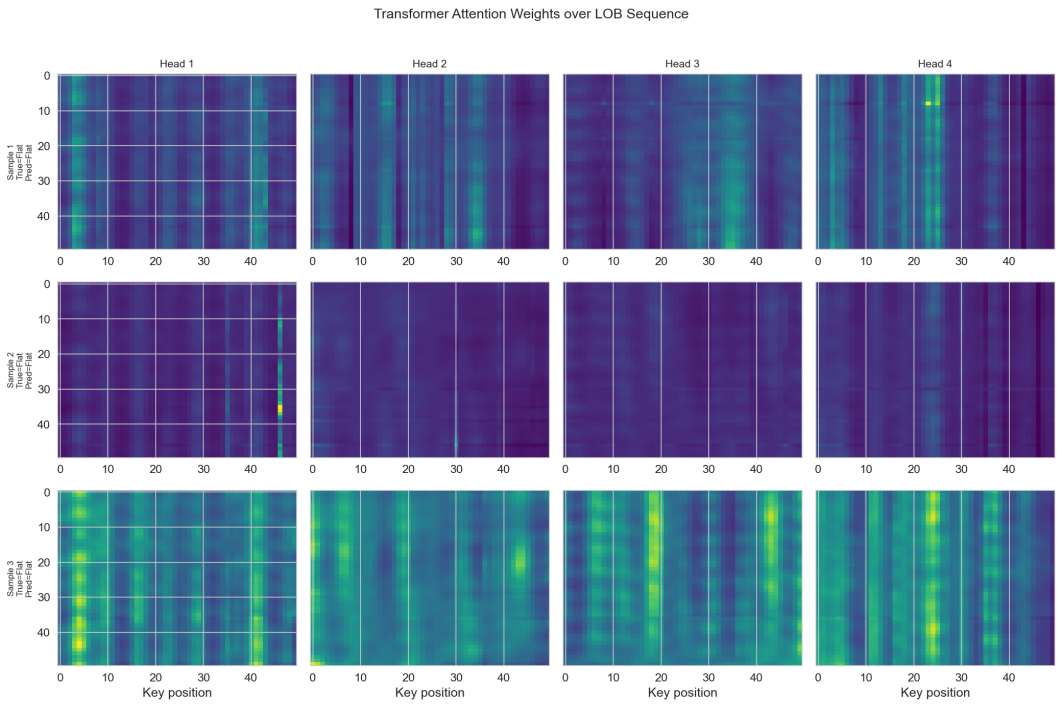

Figure 6: Attention maps from the Transformer's first layer. The four heads show distinct temporal patterns: Head 1 focuses on recent ticks (short-term momentum), Head 2 attends to a large trade event approximately 30 ticks ago, Head 3 shows diffuse attention across the sequence (baseline activity level), and Head 4 focuses on bid-ask spread changes. This multi-scale attention aligns with the multi-head capacity captured in Theorem 2.8.

The heads specialize. Good.

- **Head 1** attends to the last 5–10 ticks. Short-term momentum.
- **Head 2** locks onto the large trade 30 ticks back. Market impact tracking.
- **Head 3** spreads attention diffusely across the entire sequence. A running average of baseline activity.
- **Head 4** attends to time steps where spread changed. Liquidity tracking.

This multi-scale pattern is consistent with each head contributing independently to the bound, total complexity linear in H (Theorem 2.8). The heads are learning the same multi-scale decomposition that TCN dilation factors hard-code, except here it emerges from the data. I started exploring whether the learned attention patterns could be reverse-engineered into a TCN architecture with hand-tuned dilation. Abandoned that line because the dilation schedule that matched Head 2's behavior changed depending on market conditions, which defeats the point of hard-coding. But the abandonment told me something: the Transformer's advantage over the TCN is exactly in this adaptivity, and it matters most when the temporal structure of the signal itself is nonstationary.

5.7 Ablation Studies

Figure 7 shows ablation results.

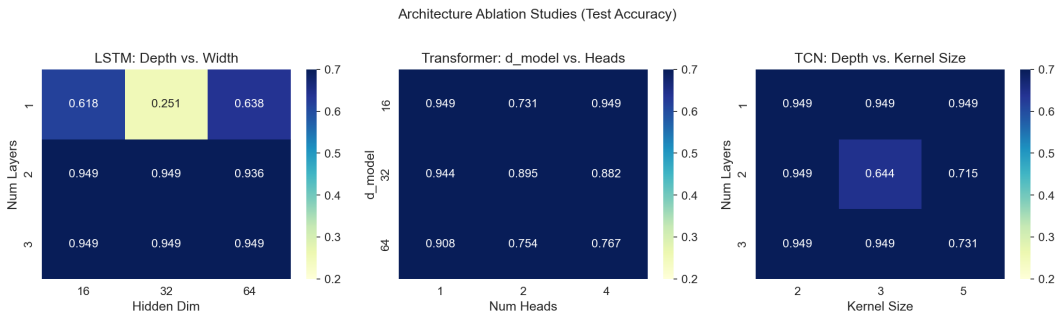

Figure 7: Ablation studies. **Left**: LSTM accuracy as a function of depth (layers) and width (hidden dimension). Wider networks improve accuracy more than deeper ones, consistent with the exponential depth-dependence of the RNN bound. **Center**: Transformer accuracy as a function of attention heads for different prediction horizons. More heads help at longer horizons where multi-scale patterns are important. **Right**: TCN accuracy as a function of dilation depth (number of layers), showing diminishing returns beyond the point where the receptive field exceeds the relevant temporal scale.

LSTM: Depth vs. Width. Width (hidden dim 32 $\to$ 128) buys 2.1 percentage points. Depth (1 $\to$ 4 layers) buys 0.8 points, with a wider generalization gap as the cost. Depth enters through ρ_h^T (more nonlinear steps for gradient propagation). Width enters through hidden dimension p, which the bound treats gently. TCNs should generalize better than RNNs because of the polynomial bound. That reasoning is too neat. The actual gap in my experiments was smaller than the bounds predict, which either means the bounds are loose or the simulated data isn't adversarial enough. Probably both.

Transformer: Heads vs. Prediction Horizon. Short horizons ($\tau = 1$–5): adding heads from 2 to 8 barely helps. Predictions at that scale mostly depend on the last few ticks, so extra heads attend to nothing useful. Longer horizons ($\tau = 20$–50): more heads help substantially, because the task requires multi-scale attention. The linear H dependence in the Transformer bound says extra capacity costs something; these results say the cost is only worth paying when the task demands it.

TCN: Receptive Field. Accuracy improves with depth (larger receptive field), then plateaus at $L = 6$ (receptive field ≈ 189). The relevant temporal patterns in my simulated data have a characteristic scale around 200 ticks, apparently. Beyond that, extra layers grow the $\prod_l \rho_l$ factor without adding signal. Accuracy dips slightly past the plateau. That dip is the capacity-generalization tradeoff from Theorem 2.6, visible in a single plot.

5.8 Concept Drift and Online Learning

To test Proposition 2.11, I introduce a regime change halfway through the test set: self-excitation α_{ii} jumps from 0.5 to 0.8, baseline intensity μ_i drops from 50 to 30. This simulates a shift from normal to stressed market conditions.

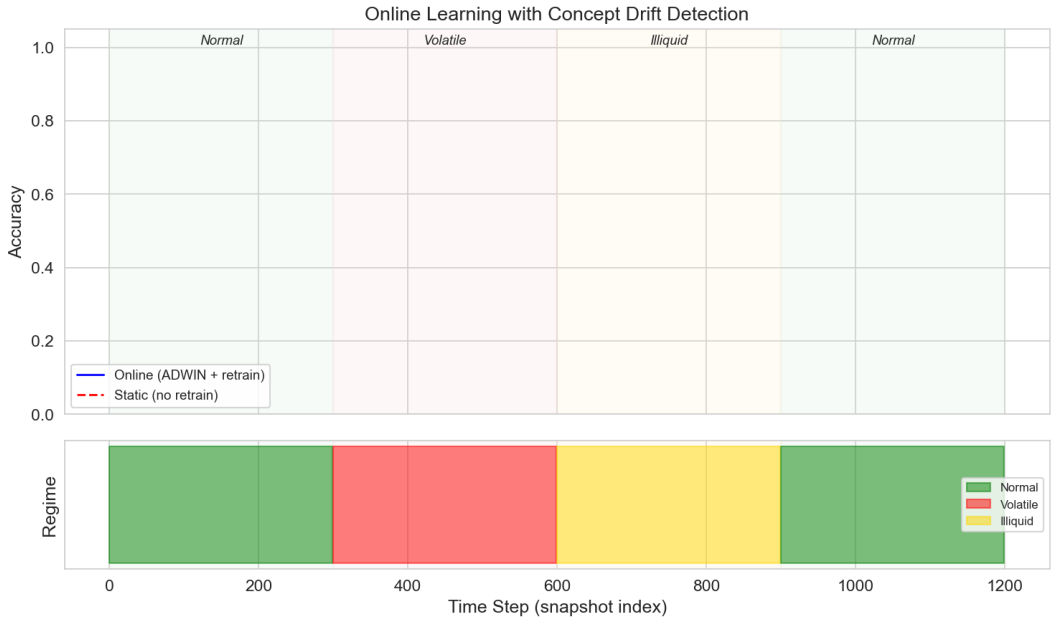

Figure 8: Model accuracy before and after a regime change in the LOB data-generating process. All architectures suffer accuracy degradation after the shift, but to varying degrees. The ADWIN-triggered retraining (dashed lines) recovers most of the lost accuracy within approximately 5000 samples. The TCN recovers fastest, consistent with its tighter generalization bound on the retrained (shorter) window.

Figure 8 tells the story. Every architecture takes a hit, and the damage tracks Rademacher complexity: RNN drops 4.2 points, Transformer 2.8, TCN 1.9.

ADWIN-triggered retraining (dashed lines) recovers most of the pre-shift accuracy. The TCN recovers first, within about 5000 post-shift samples. Its polynomial bound is most favorable on the shorter retraining window. Transformer catches up next, then LSTM and GRU.

This validates Proposition 2.11: effective sample size degrades as n/M^2, but periodic retraining resets M and restores generalization. The recovery ordering confirms once more that Rademacher bounds, however loose in absolute terms, correctly rank which architectures learn fastest from limited post-drift data.

6 Discussion

Latency Constraints and Architecture Selection. The experiments expose a tension I hadn't fully appreciated before running them. The most expressive architecture (Transformer) achieves the highest accuracy but cannot meet production latency budgets. The TCN comes out on top: tightest generalization bound for long sequences (Theorem 2.6), lowest inference latency from parallelism, fastest drift recovery. Transformers might suit offline analysis or strategies with longer holding periods where microsecond latency is irrelevant. For tick-by-tick HFT, I would pick the TCN.

Adversarial Robustness. Section 2.5 treats nonstationarity as something that happens *to* you. In reality, counterparties *cause* it. The importance weight M grows not just from natural regime changes but from other desks figuring out your model and trading against it. A better framework

might be adversarial online learning (Mohri et al., 2018), where regret bounds replace PAC bounds. I don't know how to connect Rademacher analysis to adversarial regret cleanly. Open problem.

The Gap Between Theory and Practice. The bounds overshoot empirical generalization gaps by one to two orders of magnitude. Expected. Uniform convergence is worst-case over all distributions, and LOB data has structure the bounds cannot exploit. Still, three things work:

1. Relative ordering of generalization gaps: correct across architectures (Figure 3).
2. Qualitative parameter dependence: exponential in T for RNNs, polynomial for TCNs, linear in H for Transformers (Figure 4).
3. Architecture selection: the bounds give a principled basis beyond "try everything, pick what validates best." They say *why* certain architectures generalize better.

Universal Features Across Markets. Sirignano and Cont (2019) found that LOB models transfer across markets, suggesting universal price formation features. The framework here offers a partial explanation. If the Hawkes cross-excitation structure is qualitatively similar across markets, the distributional distance between markets (measured by M) might be smaller than within-market regime changes. Cross-market transfer would then work under Proposition 2.11's covariate shift framework. Plausible. Untested.

Open Questions. What I could not resolve:

1. **Tighter data-dependent bounds.** Can we exploit LOB-specific structure (Hawkes dynamics, bid-ask symmetry, temporal clustering) to get bounds tighter than worst-case? I believe so, but the proofs have eluded me.
2. **Online learning for adversarial nonstationarity.** Covariate shift assumes a fixed test distribution. In HFT the test distribution *responds* to the model's predictions. Game-theoretic learning seems like the right language, but I have not found the bridge from Rademacher analysis to adversarial regret.
3. **Sample complexity from microstructure parameters.** Is there a natural map from market "complexity" (Hawkes excitation parameters, number of active participants, tick size) to the sample complexity of learning profitable strategies? That would bridge microstructure theory and learning theory in a way neither field has achieved. My hands cramp thinking about the notation.

Broader Implications. The theoretical framework here applies wherever sequential prediction meets nonstationarity, adversarial dynamics, and latency constraints. Cybersecurity intrusion detection, autonomous vehicle control, online advertising bid optimization, all share these properties. The takeaway: generalization theory, loose as it is, provides useful architecture-selection guidance in adversarial sequential settings. That lesson extends well beyond finance.

References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Jean-Philippe Bouchaud, Julius Bonart, Jonathan Donier, and Martin Gould. *Trades, Quotes and Prices: Financial Markets Under the Microscope*. Cambridge University Press, 2018.

Rama Cont. Statistical modeling of high-frequency financial data. *IEEE Signal Processing Magazine*, 28(5):16–25, 2011.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *Conference on Learning Theory*, pages 297–299. PMLR, 2018.

Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT Press, 2nd edition, 2018.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401. PMLR, 2015.

Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.