Volatility Prediction with Ultra-High-Frequency Order Book Data

Ayush Singh

March 29, 2025

Abstract

Volatility is a fundamental concept in financial markets, quantifying the magnitude of price fluctuations over time. Accurate volatility forecasting is crucial for trading firms and market makers, as it underpins risk management, option pricing, and strategy formulation. In particular, high-frequency traders and option market-makers rely on short-term volatility predictions to adjust their quotes and hedge risks in real time. This document provides a detailed and mathematically rigorous treatment of volatility prediction using ultra-high-frequency order book data, motivated by a dataset from Optiver containing 167 million observations of top-of-book quotes at 1-second intervals for over 100 stocks. Each stock's data consists of successive 10-minute time intervals, with fields including second $0-599$ seconds, the best bid-ask prices and next level prices, corresponding sizes, and stockID identifier.

1 Introduction

We begin by reviewing foundational financial concepts - the definition and measurement of volatility, the bid-ask spread, and order book microstructure - and their mathematical formulations. We then delve into several volatility prediction models, deriving their key essumptions in depth. These models range from simple benchmarks to classical time-series models and modern machine learning:

1. Historical Average Realized Volatility $HAV - RV$ as a baseline.

2. An $ARMA - GARCH$ model capturing volatility clustering.

3. A simpler Exponentially Weighted Moving Average $EWMA$ model.

4. An advanced long-short term memory $LSTM$ neural network predictor.

For each model, we present rigorous mathematical derivations and discuss statistical properties (e.g. stationarity, memory, assumptions on returns). We also outline how to evaluate and compare these models' performance using appropriate metrics such as Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and the Quasi-Likelihood (QLIKE) loss. Importantly, since any predictive tool must be usable by traders, we discuss interpretability and explainability considerations, highlighting how model outputs can be understood and trusted in a trading context. Finally, we conclude with a step-by-step implementation guide (in prose, without code) describing how to go from raw ultra-high-frequency order book data to a deployed volatility prediction model. This includes data preprocessing, feature engineering, model training, evaluation, and practical deployment considerations.

2 Notation and Setup

Before proceeding, let us establish some notation. We denote by S_t the price of an asset (stock) at time t. Because we are dealing with high-frequency data, t might index time in seconds or finer. In each 10-minute interval (600 seconds) labeled by a time-id, we have observations at times $t = 0, 1, 2, \ldots, 599$ seconds. Define the midpoint of the best bid and ask at time t as the mid-price:

$$M_t = \frac{\text{BidPrice}_t + \text{AskPrice}_t}{2}$$

A simple proxy for instantaneous returns is the log-return of the mid-price: $r_{t_i} = \ln M_{t_i} - \ln M_{t_{i-1}}$ over some short interval (e.g., one second). The dataset's realized volatility for a given 10-minute interval (with time-id τ) can be computed by summing squared high-frequency returns within that interval. Specifically, if $0 = t_0 < t_1 < \cdots < t_n = 600$ seconds are the timestamps in the interval, the realized variance over that interval is:

$$R_\tau = \sum_{i=1}^{n} r_{t_i}^2$$

And the realized volatility is the standard deviation (square root) of realized variance:

$$\sigma_\tau = \sqrt{R_\tau}, \tag{1}$$

In practice, r_{t_i} could be 1-second log-returns of the mid-price, so:

$$R_\tau = \sum_{t=1}^{599} (\ln M_t - \ln M_{t-1})^2$$

This realized volatility σ_τ for each 10-minute window is the target our models will aim to predict based on order book information. We will denote by $\hat{\sigma}_\tau$ a forecast of the volatility for interval τ.

3 Financial Background

Volatility and Its Measurement

Volatility refers to the variability or dispersion of returns of a financial asset. If $r_t = \ln S_t - \ln S_{t-1}$ is the log-return over a short period from $t-1$ to t, the variance of returns $\mathrm{Var}(r_t)$ quantifies volatility. In an elementary sense, volatility σ is the standard deviation of returns. More formally, one often works with conditional volatility σ_t, defined as the standard deviation of the next return conditional on information up to time t.

For example, in a model $r_t = \mu_t + \epsilon_t$ with mean $\mu_t = \mathbb{E}[r_t|\mathcal{F}_{t-1}]$ and shock ϵ_t (zero mean), the conditional variance is $\sigma_t^2 = \mathbb{E}[\epsilon_t^2|\mathcal{F}_{t-1}] = \mathrm{Var}(r_t|\mathcal{F}_{t-1})$. A high σ_t implies that large returns (up or down) are likely at time t, i.e., high market uncertainty.

In high-frequency settings, volatility is often measured via realized volatility, as defined in Eq. (1) above. The realized volatility σ_τ over a short interval (e.g., 10 minutes) is an ex-post measure computed from high-frequency returns

inside that interval. Under ideal conditions (increasing sampling frequency), realized variance converges to the underlying continuous-time integrated variance of the price process. For instance, if S_t follows a continuous semimartingale $d \ln S_t = \mu_t dt + \sigma_t dW_t$ (with W_t a Brownian motion), then as the sampling interval shrinks, $\sum r^2 \to \int \sigma_t^2 dt$ (plus jumps) in probability. Thus, realized volatility is a consistent estimator of true volatility and is widely used as "ground truth" in volatility modeling.

A key empirical property of asset returns is volatility clustering. Large price moves tend to cluster in time, followed by periods of relative calm, rather than being independently and identically scattered. In other words, if volatility was high in the recent past, it is likely to remain high in the near future—and similarly for low volatility. This phenomenon can be formalized by the relation in squared returns or realized volatility series. Volatility clustering underpins the success of time-series models like GARCH (which we derive later) that use past variances or squared returns to predict future variance. It also means volatility is time-varying rather than constant.

Bid-Ask Spread and Liquidity

The bid-ask spread is the difference between the best ask price (lowest selling price) and best bid price (highest buying price) in the order book. If at time t the best bid is B_t and best ask is A_t, the $absolute_t - B_t$. Often a normalized spread $\frac{A_t - B_t}{(A_t + B_t)/2}$ is used to compare across price levels, but for simplicity we refer to the absolute spread. The bid-ask spread is a fundamental indicator of market liquidity and transaction cost: a narrow spread means buyers and sellers are close in price (high liquidity, low trading cost), whereas a wide spread indicates a gap between buy and sell willingness (low liquidity, higher cost). In our dataset, we have both prices and sizes at the top two levels. The Level-1 spread at time t is simply $A_t - B_t$. The sizes $v_b^1(t), v_a^1(t)$ (best bid size and ask size) indicate how many shares/contracts are available at those quotes, and similarly v_b^2, v_a^2 for second level.

Mathematically, the spread can be seen as the difference between two stochastic price processes: A_t (ask price) and B_t (bid price). A zero spread ($A_t = B_t$) would imply perfect liquidity, but in order-driven markets a positive spread persists as compensation for market makers who provide liquidity. Spread is related to volatility in that periods of high volatility often coincide with widening spreads,

as market makers quote more cautiously to avoid being adverse informed traders. Indeed, a high spread is both a result and potential cause of volatility: wide spreads may lead to larger jumps when trades finally execute, and conversely high uncertainty (volatility) makes liquidity providers widen their quotes. In microstructure models, the spread can be decomposed into componenory risk and adverse selection cost - the latter often linked to volatility (higher volatility, greater adverse selection risk). We will later consider features like the average spread or maximum spread in an interval as possible predictors of volatility.

Order Book Microstructure and Order Flow

A **limit order book (LOB)** is the electronic record of all outstanding limit buy and sell orders in the market, organized by price level. At any given time t, the LOB provides a snapshot of supply and demand: it lists prices at which traders are willing to buy (bids), along with the associated sizes (volumes). In most order books, buy orders are sorted in descending price (highest bid at top) and sell orders in ascending price (lowest ask at top). The best bid B_t and best ask A_t are the first levels of the book as described above. The depth of the book refers to the volume available at each price or within a price range.

Order book data contains rich information about short-term price dynamics and volatility. The **order book imbalance** is one important indicator: for example, one can define volume imbalance at the top of the book as:

$$\Psi_t = \frac{v_b^1(t) - v_a^1(t)}{v_b^1(t) + v_a^1(t)} \in [-1, 1], \tag{2}$$

which is the normalized difference between bid volume and ask volume at the best quotes. $\Psi_t \approx 1$ means much more buy volume than sell volume at the top (buying pressure), while $\Psi_t \approx -1$ indicates selling pressure. Large imbalance (in either direction) can predict short-term price movements and volatility, as an extremely one-sided book may precede a price jump (e.g. if buys overwhelm sells, price may uptick). Another related measure is the microprice M_t (sometimes called fair price), which weighs the best prices by relative volume:

$$M_t = \frac{v_b^1(t)}{v_b^1(t) + v_a^1(t)} A_t + \frac{v_a^1(t)}{v_b^1(t) + v_a^1(t)} B_t$$

The microprice will lie between B_t and A_t closer to the side with greater volume, and it often leads the midprice as an indicator of where the price might move given the imbalance.

Order flow refers to the sequence of order book events over time: limit orders being placed, canceled, or executed (market orders). Essentially, it is the evolution of the LOB. Formally, we can denote an order as a tuple $O = (t_O, p_O, v_O)$ with time, price, and volume. The order flow in a time interval $(t - \Delta T, t]$ is the collection of all orders (and trades) in that period. Order flow drives price changes: for example, a large market sell order will consume the bid quotes, potentially moving the midprice down and increasing short-term volatility. High-frequency volatility is intimately linked with order flow dynamics - rapid sequences of trades and order cancellations can cause abrupt price moves. Empirical studies find that measures derived from order flow (like order arrival rates, trade volume, and imbalance) can significantly imprerm volatility forecasts. The rationale is that microstructure effects - such as liquidity droughts or aggressive trading - foreshadow volatility that a macro variable or past price alone will leverage this by considering predictors like the bid-ask spread, volume imbalance, etc., in advanced models.

Mathematically, one can model order flow as a point process. However, for our purposes, we will simply extract features from the LOB data (like Ψ_t or spread) or feed the raw sequence to a machine learning model. The key point is that short-term volatility may be predicted by microstructure features: e.g., a very large bid order at the second level getting pulled (canceled) might signal potential volatility if that liquidity was supporting the price. Traders often visually monitor such order book dynamics in real-time. In this paper, we aim to incorporate this granular information into formal volatility prediction models.

4 Volatility Prediction Models

Historical Average Realized Volatility (HAV-RV)

The historical average realized volatility model is a naive benchmark that predicts future volatility as some average of past realized volatilities. For each stock, one can use the long-run mean of realized volatility (or the mean over a rolling window) as the forecast.

It assumes that volatility has no serial correlation (or that any variation is un-

predictable), so the best guess for tomorrow's volatility is the average of yesterday's. Let σ_τ be the realized volatilities for past intervals $\tau = 1, 2, \ldots, T$ for a given stock (e.g. for previous 10-min windows, or previous days). The historical average forecast for the next interval $T + 1$ is:

$$\hat{\sigma}_{T+1} = \frac{1}{T} \sum_{t=1}^{T} \sigma_t, \tag{3}$$

This could be a full-sample mean or a moving average over a recent window (say last N intervals): $\hat{\sigma}T + 1 = \frac{1}{N} \sum i = T - N + 1^T \sigma_i$. One can also weight more recent observations more heavily (yielding an EWMA model, discussed later). In Eq. (3), all past volatilities are equally weighted in forming the expectation of the next. If we square both sides, we similarly get a forecast for variance:

$$\hat{\sigma}^2_{T+1} = \left(\frac{1}{T} \sum_{t=1}^{T} \sigma_t \right)^2$$

Assumptions: This model implicitly assumes volatility is stationary with a constant mean. If σ_t were i.i.d. with mean m, then $\mathbb{E}[\sigma_{T+1}|\text{past}] = m$ and the sample average is a natural estimator. HAV-RV ignores volatility clustering; it will not react to recent changes. For example, if volatility spiked in the last few intervals, HAV-RV will under-predict the next interval's volatility because it "dilutes" the spike with older data. Conversely, after a sudden drop in volatility, HAV-RV over-predicts until the mean catches up. In statistical terms, it is the prediction from an intercept-only model $\sigma_{t+1} = \bar{\sigma} + \epsilon_{t+1}$.

Derivation of Properties: The expected error of this model depends on the variance of volatility itself. If the true volatility process has mean m and autocorrelation ρ, the HAV-RV (full sample mean) has error $\hat{\sigma}T + 1 - \sigma T + 1$. Its MSE can be shown to be $\mathrm{Var}(\sigma_{T+1})$ if the mean is estimated over a long history (so the estimator itself is nearly constant). There is no adaptive component. Nonetheless, HAV-RV often serves as a benchmark: for instance, out-of-sample R^2 metrics in volatility forecasting are sometimes computed relative to a "historical mean" forecast.

Use in Practice: Traders would rarely rely solely on a long-run average for intraday volatility prediction, but they might use it as a sanity check or as part

of a more complex model. For example, if current short-term volatility deviates greatly from its historical average, it could indicate a regime change or a transient effect. In our context, we will use HAV-RV to gauge the value-added of more sophisticated approaches: any successful model should outperform this baseline in error metrics.

(Note: HAV-RV is analogous to a constant forecast that does not vary with time. It can be improved by allowing the "constant" to differ by market conditions - but that leads into time-series models, which we cover next.)

ARMA-GARCH Model

Financial returns often exhibit heteroskedasticity (changing variance) and mild autocorrelation. The ARMA-GARCH framework combines an ARMA (Auto-Regressive Moving Average) model for the return series (to capture any linear predictable structure in returns) with a GARCH (Generalized Auto-Regressive Conditional Heteroskedasticity) model for the volatility of those returns. Proposed by Engle (ARCH) and generalized by Bollerslev (1986), GARCH is a workhorse for volatility forecasting. It posits that the next-period variance depends on past squared errors and patility clustering. An ARMA component can model any mean reversion or serial correlation in the return levels themselves, though for many assets one assumes the mean return is zero or constant for short horizons.

Mean Equation (ARMA): Let r_t be the return at time t (e.g. 10-minute return). A general ARMA(p, q) model is:

$$r_t = \mu + \sum_{i=1}^{p} \phi_i r_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t, \tag{4}$$

where ε_t is the unpredovation at time t. Here μ is a constant (or could be zero for simplicity), ϕ_i are autoregressive coefficients, and θ_j are moving-average coefficients on past shocks. In vector-matrix form: $r_t - \mu = \Phi(L)(r_t - \mu) + \Theta(L)\varepsilon_t$ where L is the lag operator and Φ, Θ polynomials. The ARMA part handles any predictable structure in r_t (which for high-frequency returns is usually minimal after removing intraday patterns). For our volatility focus, one often assumes $r_t = \sigma_t, z_t$ with z_t i.i.d. standard normal (or Student-t, etc.), i.e. no ARMA in mean (or an AR(1) if needed).

Volatility Equation (GARCH): The ARMA alone does not account for volatility clustering, as $\mathrm{Var}(r_t)$ would be constant. The GARCH model assumes the conditional variance σ_t^2 evolves as an ARMA-like process driven by past squared innovations. The classic GARCH$(1,1)$ model is:

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2, \tag{5}$$

This says the forecast variance at time t (for return r_t) is a constant ω plus a fraction α_1 of yesterday's squared shock (ARCH term) plus a fraction β_1 of yiance (GARCH term). More generally, a GARCH(p,q) has $\sigma_t^2 = \omega + \sum_{i=1}^{q} \alpha_i \varepsilon_{t-i}^2 + \sum_{j=1}^{p} \beta_j \sigma_{t-j}^2$. GARCH$(1,1)$ is widely used and often sufficient to model financial return volatility.

Combined Model: Together, ARMA-GARCH means $r_t = \mu +$(ARMA terms)$+ \varepsilon_t$, and $\varepsilon_t = \sigma_t z_t$ with σ_t^2 following (5). In other words, the mean equation can have memory, and the variance equation has memory.

Assumptions: In GARCH, ε_t (the residuals after fitting any ARMA mean) are assumed to be i.i.d. with mean 0 and variance 1 (standardized) given information up to $t-1$. They often assume z_t is standard Normal for estimation (quasi-maximum-likelihood), but heavy-tailed distributions can be used to capture fat tails in returns. Key constraints on parameters ensure positivity of variance and stationarity: we require $\omega > 0$, $\alpha_1 \geq 0$, $\beta_1 \geq 0$, and $\alpha_1 + \beta_1 < 1$. The last condition $\alpha_1 + \beta_1 < 1$ implies that volatility shocks die out over time (stationary variance), since we can show the unconditional variance $E[\sigma_t^2] = \frac{\omega}{1-\alpha_1-\beta_1}$. If $\alpha_1 + \beta_1$ is very close to 1, volatility has long memory (slowly decaying autocorrelation), a typical situation in financial markets. If $\alpha_1 + \beta_1 = 1$, it's an IGARCH (integrated GARCH) process, with no mean reversion in variance.

Derivation Highlights: The GARCH recursion (5) can be unrolled iteratively:

$$\sigma_t^2 = \omega + \alpha_1, \varepsilon_{t-1}^2 + \beta_1, \sigma_{t-1}^2$$

Substituting σ_{t-1}^2 from the previous period:

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 [\omega + \alpha_1 \varepsilon_{t-2}^2 + \beta_1 \sigma_{t-2}^2]$$
$$\sigma_t^2 = \omega(1+\beta_1) + \alpha_1 \varepsilon_{t-1}^2 + \alpha_1 \beta_1 \varepsilon_{t-2}^2 + \beta_1^2 \sigma_{t-2}^2$$

Repeating this expansion yields an infinite moving average of past shocks:

$$\sigma_t^2 = \omega \sum_{i=0}^{\infty} \beta_1^i + \alpha_1 \sum_{i=1}^{\infty} \beta_1^{i-1} \varepsilon_{t-i}^2.$$

If $\beta_1 < 1$, the geometric sum $\sum_{i \geq 0} \beta_1^i = 1/(1-\beta_1)$ converges. More importantly, this shows σ_t^2 depends on all past squared errors, with geometrically decaying weights $\alpha_1 \beta_1^{i-1}$ for ε_{t-i}^2. Thus, GARCH$(1,1)$ is equivalent to an ARCH(∞) model, meaning it captures a long memory of past volatility shocks using only a few parameters. The presence of ε_{t-1}^2 in (5) means a large shock (outlier return) will immediately boost the next variance σ_t^2 (the "ARCH effect"), while the β_1 term means that variance also carries over further in time (volatility persistence). This reflects volatility clustering.

Statistical Properties: Under the condition $\alpha_1 + \beta_1 < 1$, the process σ_t^2 is covariance stationary with $E[\sigma_t^2] = \omega/(1-\alpha_1-\beta_1)$. The autocorrelation of squared returns ε_t^2 decays at rate $\alpha_1 + \beta_1$; in practice this sum is often close to 1 (e.g. 0.94), implying a very slow decay and strong persistence of volatility. If $\alpha_1 + \beta_1 \geq 1$, the variance of r_t is not finite (non-stationary variance), a case to avoid in modeling. GARCH can also capture the leptokurtosis of $retur_t = \sigma_t z_t$ and σ_t is serially correlated, even if z_t is normal, the unconditional distribution of r_t is heavy-tailed (a mixture of normals). This often matches the empirical fact that asset returns have heavy tails (higher kurtosis than normal). Extensions like EGARCH or GJR-GARCH add asymmetry for the leverage effect (negative returns increasing volatility more than positive ones), but we will not delve into those here.

Forecasting with GARCH: The one-step ahead variance forecast from a GARCH(1,1) model is given by the equation:

$$\mathbb{E}[\sigma_{t+1}^2 | \mathcal{F}_t] = \omega + \alpha_1 \varepsilon_t^2 + \beta_1 \sigma_t^2.$$

For most of the mean reversion in variance, the forecast will converge toward the long-run variance. Specifically, the h-step ahead forecast (for $h > 1$) is:

$$\mathbb{E}[\sigma_{t+h}^2 | \mathcal{F}_t] = \omega + (\alpha_1 + \beta_1) \mathbb{E}[\sigma_{t+h-1}^2 | \mathcal{F}_t].$$

Iteratively solving for this forecast gives:

$$\mathbb{E}[\sigma_{t+h}^2] = \omega \frac{1-(\alpha_1+\beta_1)^h}{1-\alpha_1-\beta_1} + (\alpha_1+\beta_1)^h \sigma_t^2.$$

As $h \to \infty$, this approaches:

$$\frac{\omega}{1-\alpha_1-\beta_1},$$

which is the unconditional variance. When $\alpha_1+\beta_1$ is close to 1, the forecast for even moderately large h remains close to the current σ_t^2, indicating long volatility memory.

In our context, we might apply a GARCH model not to raw returns but to the realized volatility series itself (for daily or intraday vol). However, a more common approach is to apply GARCH to returns and thereby forecast the next period's variance.

For example, one could aggregate our 1-second data to 10-minute returns (the price change over each interval) and fit an ARMA-GARCH on that series to forecast the next 10-minute return variance. Alternatively, one could use a two-step "realized GARCH" (Hansen et al., 2012) where the GARCH latent volatility is informed by the realized vol as animplicity, we consider the standard setup.

Parameter Estimation: ARMA-GARCH parameters $(\phi_i, \theta_j, \omega, \alpha_i, \beta_j)$ are typically estimated via maximum likelihood assuming $z_t \sim N(0,1)$.

The log-likelihood for each time t is:

$$-\frac{1}{2}[\ln(2\pi) + \ln \sigma_t^2 + \varepsilon_t^2/\sigma_t^2]$$

The term $\ln \sigma_t^2 + \varepsilon_t^2/\sigma_t^2$ reveals why certain evaluation metrics (like QLIKE, see later) are used - they relate to the log-likelihood of volatility forecasts. Some care is needed as the likelihoan be tricky (the constraints $\alpha, \beta \geq 0$ and $\alpha + \beta < 1$ are enforced). In practice, software can fit GARCH models readily.

Interpretation: Despite being a statistical model, GARCH has intuitive meaning for traders: α_1 is the reaction to market shocks - if a big move happened, how much do we increase volatility forecast; β_1 is the persistence - how strongly yesterday's volatility carries forward. For example, if α_1 is small and β_1 large, volatility is mostly a slow-moving process not easily shaken by one outlier (typical for stock

indices). If α_1 is large, volatility jumps a lot on each new shock (more reactive, typical for individual stocks or currencies). Thus, GARCH provides interpretable parameters linking to market behavior.

I model can produce a dynamic volatility forecast $\hat{\sigma}_{t+1}$ that reacts to recent data and embodies the cluster effectrecasts to those of other approaches.

Exponentially Weighted Moving Average (EWMA)

Model Overview: The Exponentially Weighted Moving Average (EWMA) model is a simpler, non-parametric approach to modeling time-varying volatility. Popularized by J.P. Morgan's RiskMetrics (1996), it assumes the variance today is an exponentially weighted average of yesterday's variance and yesterday's squared return. In fact, EWMA is a special case of GARCH(1,1) with $\omega = 0$ and $\alpha_1 + \beta_1 = 1$ (and typically α_1 fixed at $1 - \lambda$, $\beta_1 = \lambda$).

Because $\alpha + \beta = 1$, the conditional variance follows a random walk (no mean reversion) – it's always "on the move," hence the term moving average. EWMA is often used in industry due to its simplicity and because it requires choosing only one parameter (the decay factor λ) rather than estimating parameters via complex likelihood.

Model Specification: We define the EWMA volatility update as:

$$\hat{\sigma}_t^2 = \lambda \hat{\sigma}_{t-1}^2 + (1 - \lambda)\tilde{\sigma}_t^2. \tag{6}$$

Here $0 < \lambda < 1$ is the decay factor (or smoothing constant), and $\tilde{\sigma}_t^2$ is a proxy for the variance at time t based on new data. In practice, $\tilde{\sigma}_t^2$ can be the squared return r_t^2 (if using daily returns, for example). In our setting, if we are updating every 10-minute interval, we could let $\tilde{\sigma}_t^2 = \sigma_t^2$ be the realized variance of interval t.

Then (6) says: the forecast variance for interval $t + 1$ is a weighted average of the previous forecast (from t) and the newly observed variance in t, with weight λ on the old forecast. This can be seen as a recursion: starting from some initial $\hat{\sigma}_1^2$, one can generate forecasts for all t. Typically, $\hat{\sigma}_1^2$ might be set to the sample variance of an initialization period.

The parameter λ controls the memory: λ close to 1 means a long memory (slowly decaying weights, so older observations still have influence), whereas a

smaller λ means more weight on recent observations (fast-decaying memory).

Equivalence to GARCH(1,1): Compare (6) with GARCH(1,1):

$$\sigma_t^2 = \omega + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2.$$

If we set $\omega = 0$, $\alpha_1 = 1 - \lambda$, and $\beta_1 = \lambda$, then:

$$\sigma_t^2 = \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{t-1}^2.$$

Assuming we use last period's variance forecast in place of σ_{t-1}^2 (which is natural in forecasting), and identify ε_{t-1}^2 with $\tilde{\sigma}_{t-1}^2$, we see that:

$$\hat{\sigma}_{t+1}^2 = \alpha_1 \tilde{\sigma}_t^2 + \beta_1 \hat{\sigma}_t^2.$$

The only difference from standard GARCH is the missing constant ω and the fact that $\alpha_1 + \beta_1 = 1$. So EWMA is like a GARCH process on the edge of stationarity. It gives more bias to recent shocks and never truly forgets old data (though weights decay geometrically).

Choosing λ: RiskMetrics famously recommended $\lambda = 0.94$ for daily stock market volatility (roughly corresponding to a $1/(1 - 0.94) \approx 16$-day half-life of information). For shorter intervals like 10-minute data, one might choose a higher λ (closer to 1) if volatility is still highly persistent intraday, or calibrate it by minimizing forecast errors.

In general, λ can be optimized by backtesting (minimize MSE of one-step forecasts). Values often used: 0.94 for daily, 0.97 for monthly, etc. For very fast frequencies, a slightly lower λ might be considered to improve responsiveness to regime shifts.

Multi-step Forecasts: A noted limitation of EWMA is that multi-step forecasts are the same as one-step. Since in (6), once you project one step, any further projection yields:

$$\hat{\sigma}_{t+2}^2 = \lambda \hat{\sigma}_{t+1}^2 + (1 - \lambda)\hat{\sigma}_{t+1}^2 = \hat{\sigma}_{t+1}^2.$$

In fact, it can be shown that under EWMA, $\hat{\sigma}_{t+h}^2 = \hat{\sigma}_{t+1}^2$ for all $h \geq 1$. Essentially, the forecast variance is a martingale: the best guess of future volatility

beyond one step is the one-step forecast (no mean reversion). This is realistic for short horizons but problematic for long horizons (since one expects some reversion eventually). However, if our focus is one interval ahead (e.g., 10 minutes), this is not a big issue.

Interpretation: EWMA is easy to explain: "tomorrow's volatility = yesterday's volatility plus an adjustment for yesterday's shock." If $(1 - \lambda)$ is 0.05, it means each new day contributes 5% of the new information, and 95% comes from the previous estimate – so volatility changes gradually unless a huge shock occurs (which makes r_t^2 very large and thus raises the estimate).

Because $\omega = 0$, the long-run average volatility in EWMA drifts based on data; there is no fixed mean it pulls to. This trait makes EWMA very flexible in adapting to regime shifts. The flip side is it can overreact to outliers slightly (pure GARCH would temper the effect if $\alpha+\beta < 1$ by eventually pulling down to $\omega/(1-\alpha-\beta)$).

Use in Practice: Many risk systems still use EWMA for volatility and covariance forecasts (the "RiskMetrics" methodology). It is computationally light and requires no complex estimation – just pick λ. In an intraday context, one could use EWMA on realized volatilities computed every 10 minutes to get an intraday volatility forecast that adapts to current conditions. For example, if a macroeconomic news release at 11:00 caused a burst of volatility, EWMA would incorporate that into higher forecasts for subsequent intervals.

In our implementation, EWMA might serve as a benchmark alongside HAV-RV: it adds one degree of sophistication (time-weighting) over a flat historical average. We will evaluate it by comparing its error in predicting realized σ_{t+1}.

LSTM Neural Network Predictor

Model Overview: The Long Short-Term Memory (LSTM) neural network is a type of recurrent neural network designed to handle sequence data with long-range dependencies. Unlike ARMA-GARCH, which assumes a specific parametric form (linear dynamics for variance), an LSTM can, in principle, learn complex nonlinear patterns from the data.

We consider an LSTM-based model as a representative of advanced models, which could also include Transformer-based models with attention or other deep learning architectures. In volatility prediction, researchers have started applying LSTMs to capture long memory and to incorporate many features (e.g., order

book states) that would be hard to include in traditional models.

Input Features and Network Structure: Instead of explicitly specifying an equation for $\hat{\sigma}_{t+1}$, the LSTM learns a function f such that:

$$\hat{\sigma}_{t+1} = f(X_t),$$

where X_t represents the input features up to time t. For example, X_t could include a time series of recent order book snapshots or features (spreads, mid-price returns, volumes, etc.) over the past K seconds or intervals.

A typical architecture might process 1-second changes or imbalance metrics from the current 10-minute interval and output the predicted realized volatility for the next 10-minute interval. Another approach uses an LSTM on the series of past realized volatilities $\{\sigma_t\}$, but LSTM's power is better utilized with richer inputs. Some studies even transform order book data into images and use CNNs or LSTMs for volatility forecasting, highlighting the ability of these networks to handle high-dimensional inputs.

LSTM Cell Equations: At the heart of the LSTM is a memory cell c_t and a hidden state h_t. The LSTM updates at each time step as follows:

$$\begin{aligned}
\text{Forget gate:} \quad & f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \\
\text{Input gate:} \quad & i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \\
\text{Candidate cell state:} \quad & \tilde{c}_t = \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \\
\text{Cell state:} \quad & c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
\text{Output gate:} \quad & o_t = \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right) \\
\text{Hidden state:} \quad & h_t = o_t \odot \tanh(c_t)
\end{aligned}$$

Here, x_t is the input vector at time t, $\sigma(\cdot)$ is the sigmoid function, $\tanh$ is the hyperbolic tangent, and $\odot$ denotes element-wise multiplication. W, U, and b are the weight matrices and biases for the respective gates. These are the learnable parameters of the model. The dimension of h_t is a hyperparameter representing model capacity.

Explanation of LSTM Mechanics: The forget gate f_t decides which parts of c_{t-1} to retain. The input gate i_t controls how much of the new candidate $\tilde{c}_t$ to incorporate. Thus, c_t accumulates information over time and can represent long-

range dependencies. The output gate o_t determines which parts of c_t are exposed via h_t, which is used for prediction.

If the goal is to produce a single prediction after seeing a sequence (e.g., from interval t), we use the final hidden state h_T and feed it to a dense layer:

$$\hat{\sigma}_{T+1} = w^\top h_T + b.$$

To ensure positivity, one might predict $\ln \sigma^2_{T+1}$ and exponentiate it, or directly predict variance.

Training the LSTM: The model is trained by minimizing a loss function (e.g., mean squared error between $\hat{\sigma}_\tau$ and σ_τ) across the training data. Training is done via backpropagation through time (BPTT), which uses gradient descent on the weights W, U, and b. Regularization or early stopping is often used to avoid overfitting due to high dimensionality and data noise.

Advantages: LSTMs can incorporate multiple inputs per time step, such as imbalance, spread, price returns, and other indicators. The model can learn conditional nonlinear interactions and accommodate sequences of variable lengths. It can also adapt to regime shifts through forget gate behavior.

Interpretability

LSTMs are typically black boxes with no interpretable parameters like α_1 or β_1. However, interpretability can be attempted using feature attribution methods (e.g., SHAP) or by examining gate activations. Some studies use t-SNE to visualize how internal states evolve and respond to market changes.

Transformer Note: A transformer-based predictor uses self-attention to focus on relevant time points for forecasting. While powerful and more interpretable via attention weights, transformers usually require more data. A Temporal Fusion Transformer could also include exogenous inputs such as news or other asset prices.

Summary: The LSTM model does not yield an explicit formula for $\hat{\sigma}_{t+1}$. Instead, it provides a flexible function approximator. In practice, it may capture GARCH-like effects (e.g., large r_t^2 leads to large σ_{t+1}) along with additional nonlinear patterns (e.g., from order book features). It serves as a data-driven model that complements structured approaches.

5 Model Performance Evaluation Metrics

To assess and compare the performance of volatility prediction models, we require appropriate error metrics. Volatility forecasts are evaluated based on how close they are to realized volatility (or variance). We consider three widely used metrics: Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and the Quasi-Likelihood (QLIKE) loss. Each captures a different aspect of performance, and using multiple metrics ensures robust model comparison.

Mean Squared Error (MSE): MSE is a standard measure of forecast accuracy, defined as the average of the squared errors. If $\hat{y}_t$ is the forecast and y_t the true value, then:

$$\text{MSE} = \frac{1}{N}\sum_{t=1}^{N}(\hat{y}_t - y_t)^2. \tag{7}$$

In our case, $y_t = \sigma_t$ (realized volatility) or $y_t^2 = \sigma_t^2$ (realized variance). Typically, we focus on forecasting volatility directly.

Pros: MSE penalizes large errors more due to squaring, making it useful when large mistakes are costly. It is differentiable, which makes it convenient for training models like LSTMs.

Cons: MSE treats over- and under-predictions equally and can be dominated by outliers. In risk settings, under-predicting volatility can be more dangerous than over-predicting, motivating asymmetrical metrics like QLIKE.

Mean Absolute Percentage Error (MAPE): MAPE expresses errors as percentages of the true value:

$$\text{MAPE} = \frac{100\%}{N}\sum_{t=1}^{N}\left|\frac{\hat{\sigma}_t - \sigma_t}{\sigma_t}\right|$$

For example, if the actual volatility is 2% and the forecast is 1%, the percentage error is 50%. This makes MAPE useful for comparing performance across assets or time periods with different volatility scales.

Pros: MAPE is intuitive (e.g., "on average, forecasts are 15% off") and normalizes error by actual magnitude.

Cons: It becomes unstable when σ_t is very small or zero. To address this, one might exclude zero-vol points or add a small ε in the denominator.

QLIKE (Quasi-Likelihood) Loss: QLIKE is an asymmetric loss function tailored for variance forecasts. It is derived from the log-likelihood of a Gaussian model assuming predicted variance:

$$\text{QLIKE}_t = \ln(\hat{\sigma}_t^2) - \ln(\sigma_t^2) + \frac{\sigma_t^2}{\hat{\sigma}_t^2} - 1. \tag{8}$$

When comparing models, constant terms such as $-\ln(\sigma_t^2)$ and -1 can be dropped. The simplified QLIKE becomes:

$$\text{QLIKE}_t = \ln(\hat{\sigma}_t^2) + \frac{\sigma_t^2}{\hat{\sigma}_t^2}$$

Interpretation: If $\hat{\sigma}_t^2 = \sigma_t^2$, then QLIKE becomes $\ln(\sigma_t^2)+1$. Under-prediction of volatility increases the loss substantially, e.g., if $\sigma_t^2/\hat{\sigma}_t^2 = 2$, the ratio term dominates. Over-prediction results in only modest increases due to the logarithmic term.

Pros: QLIKE aligns with risk management preferences — it penalizes underestimation more heavily. It is also robust to noise in the volatility proxy (e.g., realized volatility).

Using Metrics for Comparison: To compare models, we compute MSE, MAPE, and QLIKE on a test set. While MSE and QLIKE often agree, they emphasize different risk aspects. A consistently under-predicting model may still have low MSE but high QLIKE.

An additional performance metric is out-of-sample R^2, defined as:

$$R^2 = 1 - \frac{\sum_{t=1}^{N}(\hat{\sigma}_t - \sigma_t)^2}{\sum_{t=1}^{N}(\bar{\sigma} - \sigma_t)^2}$$

where $\bar{\sigma}$ is the benchmark (e.g., historical mean).

To assess whether differences in forecast accuracy are statistically significant, the Diebold-Mariano (DM) test can be applied to compare average loss (e.g., QLIKE) between two models.

In our implementation, we will use MSE and QLIKE as the main evaluation metrics. For practitioners, it is helpful to visualize predicted vs actual volatility over time and accompany that with summary metrics like these. This combination offers insight into both accuracy and risk sensitivity of each model's predictions.

6 Interpretability and Explainability for Traders

Predictive accuracy alone is not sufficient for a volatility model to be useful on a trading floor — traders must trust and understand the predictions to act on them. This is especially true for complex models like LSTMs or Transformers, which are often seen as "black boxes." Below, we explore the interpretability of different model classes and strategies to enhance explainability for end-users such as traders and risk managers.

Simple Models (HAV-RV, EWMA): These models are inherently interpretable. HAV-RV returns a constant — the long-run average. A trader can easily understand and contextualize the forecast: "The stock's average 10-minute volatility is 1.5%, and that's our forecast."

EWMA uses only one parameter λ. Explaining that "we weight yesterday's volatility at 94% and the latest observation at 6%" is straightforward. Traders can even manually adjust forecasts based on market context (e.g., if a known event is upcoming). The downside is that these models are inflexible and require human intervention to respond to changing conditions.

GARCH Models: GARCH(1,1) offers interpretable parameters: ω, α_1, and β_1. Traders often interpret α_1 as the "news impact" and β_1 as "persistence." For example, if $\alpha_1 = 0.1$ and $\beta_1 = 0.85$, one might say:

> 15% of volatility comes from the long-term mean (since $\alpha + \beta = 0.95$ leaves 5% to mean reversion), and each new shock moves volatility by 10% of its magnitude.

GARCH forecasts are traceable: a trader can identify why the model sees a calm regime (e.g., "no big returns recently"). Extended GARCH models (e.g., with leverage terms) are less interpretable but still significantly more transparent than deep learning models.

LSTM / Deep Learning Models: These are the least interpretable by default. They consist of many weights without immediate economic meaning. If an LSTM predicts a volatility spike, it is unclear whether it responded to an imbalance, a trade surge, or other factors. This "opaqueness" reduces trust among traders. Several strategies can mitigate this:

Feature Importance and Sensitivity: Techniques like SHAP values or Layer-wise Relevance Propagation can decompose the output into contributions from each input. For example, an LSTM using order book data might reveal that a drop in bid size five seconds ago had a high SHAP value, meaning it strongly influenced the prediction.

Attention Mechanisms: In attention-based models (e.g., Transformers), attention weights highlight which time steps influenced the forecast. For example, high weights during recent price spikes or market open suggest that these moments drive the model's prediction.

Partial Dependence and Scenario Analysis: By feeding controlled hypothetical inputs (e.g., order imbalance flip vs no flip), we can evaluate how the model's prediction changes. If volatility rises in the first case, we infer that the imbalance pattern is influential — this mimics trader-like scenario thinking.

Simplified Surrogate Models: Local surrogate models (e.g., linear regressions fitted on LSTM behavior near the current state) can approximate the model's local response. One might state: "Volatility increases by 0.2 when the spread widens by 0.1," which is intuitive.

Visualization of Internal States: t-SNE plots of LSTM hidden states can reveal clusters such as "high-volatility-uptrend," "low-volatility-stable," etc. These regime clusters help explain model outputs in intuitive terms.

Balancing Accuracy and Interpretability: There is often a trade-off between accuracy and interpretability. One practical approach is to distill deep model insights into simpler terms. For instance, use the LSTM for prediction but display simplified explanations:

- **LSTM baseline**: 1.8%
- **+0.3% adjustment**: wide spread detected
- **+0.4% adjustment**: trade imbalance spike

Alternatively, combine interpretable models with deep models: use GARCH for baseline, and let LSTM generate real-time adjustments.

Use-case for Traders: Suppose a trader sees a dashboard displaying a predicted 10-minute volatility of 2.5% for stock XYZ. The dashboard could explain:

- "This is higher than usual."
- "Contributing factors: Recent trade imbalance (+0.5%), widened spread (+0.2%)."

If the model were:

- **HAV-RV**: "Historical average volatility = 2.5%." (static, easy to understand)
- **GARCH**: "Baseline = 2.0% (due to yesterday's calm), +0.5% due to today's jump."
- **LSTM**: Requires feature-level explanation or visual insights as above.

To ensure usability in trading environments, volatility models must be interpretable or explainable. Either use inherently transparent models, or supplement complex models with interpretability layers. A slightly less accurate but interpretable model may be more effective than a more accurate but opaque one. Trust is key in model adoption — especially where real capital is at stake.

7 Implementation Guide (Step-by-Step)

We outline how to implement a volatility prediction model using ultra-high-frequency order book data. The objective is to forecast 10-minute realized volatility per stock using a dataset such as Optiver's, with the final system deployable in a trading environment.

Step 1: Data Preprocessing *Load and Merge Data:* Organize the dataset by stock and time intervals (e.g., 10-minute intervals identified by `time_id`). Each interval includes up to 600 rows (one per second). Ensure data is sorted chronologically.

Clean Data: Handle missing timestamps, forward-fill last known bid/ask when needed, and ensure all prices are numeric and sizes are integers.
Compute Mid-Price and Log-Return: For each second, define mid-price as $M_t = (B_t + A_t)/2$, then compute log-returns within each interval as $r_t = \ln M_t - \ln M_{t-1}$. *Compute Realized Volatility:* For each interval τ, compute realized variance $R_\tau = \sum_{t \in \tau} r_t^2$ and realized volatility $\sigma_\tau = \sqrt{R_\tau}$; this serves as the target variable.
Remove Outliers/Bad Data: Exclude intervals with anomalies (e.g., zero or extreme volatility), to avoid model distortion.

Step 2: Feature Engineering Derive features that help predict $\sigma_{\tau+1}$ based on prior intervals:

- *Lagged Volatility:* Include σ_τ as a feature.
- *Longer-Horizon Volatility:* Include past hour or daily volatility.
- *Spread Metrics:* Compute average/percentile spread per interval; include spread trend.
- *Order Book Imbalance:* Use Eq. (2) to compute Ψ_t, summarize across interval.
- *Trade Intensity Proxy:* Count mid-price changes in τ.
- *Volume at Best Quotes:* Measure average or minimum $v_b^1(t) + v_a^1(t)$.
- *Momentum Indicators:* Use net mid-price change in interval.
- *Time-of-Day / Exogenous Info:* Include dummy variables for open/close or news.

Combine all into feature vectors X_τ and standardize features where necessary.

Step 3: Model Training Train models of increasing complexity:

- *HAV-RV Benchmark:* Compute historical mean volatility for each stock. No model training is needed; use as baseline.
- *EWMA:* Choose decay λ (e.g., 0.97). Compute:

$$\hat{\sigma}_{t+1}^2 = \lambda \hat{\sigma}_t^2 + (1 - \lambda)\sigma_t^2$$

- *GARCH(1,1):* Fit per stock or pooled using returns or realized vol. Estimate $(\omega, \alpha_1, \beta_1)$ via MLE.
- *LSTM:* Use past k intervals' feature vectors to predict σ_τ. Prepare sequences, split train/val/test chronologically. Train using MSE or QLIKE loss. Use early stopping. Tune hyperparameters: hidden units, sequence length, learning rate. For scale, consider subsampling or using GPU.

Optionally, try Transformers or tree-based models (e.g., XGBoost). Include AR(1) or linear regression for comparison.

Step 4: Model Evaluation and Selection Evaluate on test set using:

- **MSE**, **MAPE**, and **QLIKE** (see Section 4 for definitions).
- Compare all models: *HAV-RV, EWMA, GARCH, LSTM*, etc.
- Check performance across stocks and regimes (e.g., high-volatility periods).
- Use the Diebold-Mariano test to compare forecast errors (e.g., LSTM vs. GARCH).
- Consider interpretability: prefer simpler models if performance is close.

Step 5: Deployment and Use-Case

- *Real-time Prediction:* Build a live system that updates every 10 minutes. For EWMA/GARCH, update in-place. For LSTM, keep last k intervals' features in memory to feed into the model.
- *Output Delivery:* Provide forecasts via dashboard or alert system. Include explanations (e.g., "Volatility up due to widened spread").
- *Integration with Trading Strategies:* Use forecasts to adjust quotes, hedge ratios, or trade timing. Run backtests: e.g., how would P&L change if strategy adjusted for predicted vol?
- *Monitoring and Maintenance:* Track performance over time. Re-estimate GARCH parameters or retrain LSTM monthly. Check for drift.
- *Risk Controls:* Implement override logic (e.g., ignore forecast if major event imminent). Cap minimum volatility. Include manual oversight.

- *Continual Learning:* Update models with new data periodically. For GARCH, use recursive estimation. For LSTM, use rolling retraining windows.

Example Walk-Through: Suppose it is 3:00pm and we want to forecast volatility for the 3:10pm interval for stock ABC. Our pipeline collects data from 2:50–3:00, constructs the feature vector, and feeds it into the model (e.g., LSTM using features from 2:00–3:00). The model outputs $\hat{\sigma}_{3:00-3:10} = 1.8\%$. Compared to historical average (1.0%) and same-time yesterday (1.2%), this is elevated. Order book is thin and a Fed announcement is due at 3:15. Traders widen quotes or reduce exposure. After 3:10, the actual realized volatility is 2.0% — the model was accurate, increasing trader trust. This repeats every interval.

Conclusion: By following this end-to-end pipeline — from preprocessing to deployment — one can build a high-frequency volatility forecasting system that is statistically sound and practically usable. Integrating traditional financial models with modern machine learning ensures both interpretability and predictive power, making the model valuable for real-time trading applications.