Unsupervised Generative Framework for Event-Driven Financial Time Series

Mihir Agarwal, Aaditya Barve

December 2025

1 Introduction

In recent years, a lot of research has happened towards unsupervised generative modeling of time series data, such as TimeGAN (Yoon et al., 2019) and diffusion-based approaches (Ho et al., 2020; Song et al., 2020), which output high quality synthetic sequences. But, most models assume fixed interval and continuous time data, which limits their ability to capture event driven and multidimensional sequences, for example, Limit Order books in financial markets. They are governed by microstructure laws, price and volume constraints, inter agent interaction, etc, making their generation particularly challenging. The same challenges are seen in other even driven systems such as healthcare records, network traffic, self-driving car trajectories, etc.

We want to explore an unsupervised generative framework for event-driven time series generation. We particularly want to look into diffusion-based sequence generation with neural point processes (Mei Eisner, 2017) to model both the timing and type of events. Such a model would learn from raw, unlabeled LOB data to capture the joint distribution of price levels, volumes, and order arrival events, while preserving microstructure constraints. We will evaluate the different approaches using a combination of statistical measures, facts from empirical finance, and downstream tasks such as agent-based simulations, benchmarking against prior approaches

2 Literature Review

2.1 Classical Algorithms

Early research focused on capturing the arrival dynamics of market orders through stochastic processes.

- **Hawkes Processes:** Bacry et al. [1] utilized Hawkes processes to model "self-exciting" events, where the arrival of an order increases the probability of future orders, effectively capturing temporal clustering in market activity.
- **Queue-Reactive Models:** Huang et al. [2] proposed models where order placement probability is conditioned on the current queue size at the best bid or ask price.
- **Stochastic Differential Equations (SDEs):** Lasry and Lions [3] approximated LOB dynamics using systems of Partial Differential Equations (PDEs), specifically within the framework of Mean Field Games.
- **Zero Intelligence (ZI) Agents:** Gode and Sunder [4] introduced ZI agents as a "null hypothesis," demonstrating that allocative efficiency can emerge from agents placing random orders within budget constraints, independent of strategic rationality.

2.2 State-of-the-Art (SOTA) Generative Models

Recent advancements have shifted towards unsupervised learning to handle the high-dimensional and asynchronous nature of LOB data.

- **"Painting the Market" (2024):** Coletta et al. [5] utilized Denoising Diffusion Probabilistic Models (DDPMs) to generate realistic LOB volume shapes, treating the order book as an image generation problem.
- **TRADES (2025):** Berti et al. [6] developed a Transformer-based Denoising Diffusion engine designed to handle complex time-series dependencies and joint distributions inherent in financial markets.
- **State Space Models (SSMs):** Gu and Dao [7] introduced the Mamba architecture (S5/Mamba), a selective state space model that offers linear-time sequence modeling. This approach is notably faster and more efficient than Transformers for processing the long sequences typical of high-frequency financial data.

2.3 Benchmark Datasets

To standardize evaluation, recent research relies on high-fidelity datasets representing various market conditions.

- **LOBSTER:** Reconstructed NASDAQ data that serves as the industry "Gold Standard" for limit order book research, as detailed by Huang and Polak [8].
- **LOB-Bench (2025):** An open-source benchmark suite introduced by Nagy et al. [9] specifically for standardizing the evaluation of generative AI in finance.
- **Crypto-LOBs:** High-volatility datasets sourced from major exchanges like Binance and Coinbase. As noted by Ntakaris et al. [10], these are crucial for modeling extreme market events and serve as the primary dataset for this project.

3 Baseline Architectures

To evaluate the efficacy of the proposed NeuroLOB framework, we compare it against two primary baseline architectures representing deterministic and adversarial approaches.

3.1 Neural Point Process (NPP)

The Neural Point Process (NPP) serves as a deterministic baseline in our study. We employ an autoregressive framework that models the conditional expectation of the next event given the history, without relying on a stochastic diffusion process.

Temporal Encoding Identical to our main architecture, the NPP utilizes a **Causal Transformer** equipped with **Rotary Positional Embeddings**. This encodes the history H into a latent context vector c, capturing the temporal dependencies essential for Limit Order Book (LOB) data.

Dual-Head Prediction The context vector c feeds into two parallel Multi-Layer Perceptrons (MLPs) to predict continuous markers and event types separately:

- **Continuous Head (Regression):** Predicts the continuous vector $\hat{x}_{cont}$ (comprising time Δt, price Δp, and volume v) via a regression MLP ϕ_{reg}:
$$\hat{x}_{cont} = \phi_{reg}(c) \tag{1}$$
- **Discrete Head (Classification):** Predicts the event type probability distribution $\hat{p}_{type}$ via a classification MLP ϕ_{cls}:
$$\hat{p}_{type} = \text{softmax}(\phi_{cls}(c)) \tag{2}$$

Optimization The model is optimized by minimizing a composite loss function that combines Mean Squared Error (MSE) for the continuous regression targets and Cross-Entropy for the discrete event types:

$$\mathcal{L}_{total} = \|x_{cont} - \hat{x}_{cont}\|^2 - \sum_k y_k \log(\hat{p}_k) \tag{3}$$

Limitation A significant limitation of this baseline is its tendency to converge to the *mean* of the distribution. Unlike the diffusion approach, the NPP potentially underestimates the volatility and "fat tails" inherent in financial LOB data.

3.2 Generative Adversarial Network (GAN)

We implemented a regime-aware GAN architecture designed to handle distinct market conditions (Volatile, Normal, Quiet). The pipeline is structured into three distinct phases as illustrated in our system architecture (see Figure 1).

3.2.1 Data Processing Pipeline

The *LOBDataProcessor* module handles the ingestion of raw financial data from Google Cloud Storage (GCS) buckets. As shown in the first stage of the architecture:

- **Ingestion & Cleaning:** Raw CSV files are loaded from GCS buckets, cleaned, and segmented by market regime (volatile, normal, quiet).
- **Feature Engineering:** Critical state variables are extracted, including inter-arrival times (Δt), order type codes, price returns ($price\_ret$), and logarithmic order amounts ($amount\_log$).

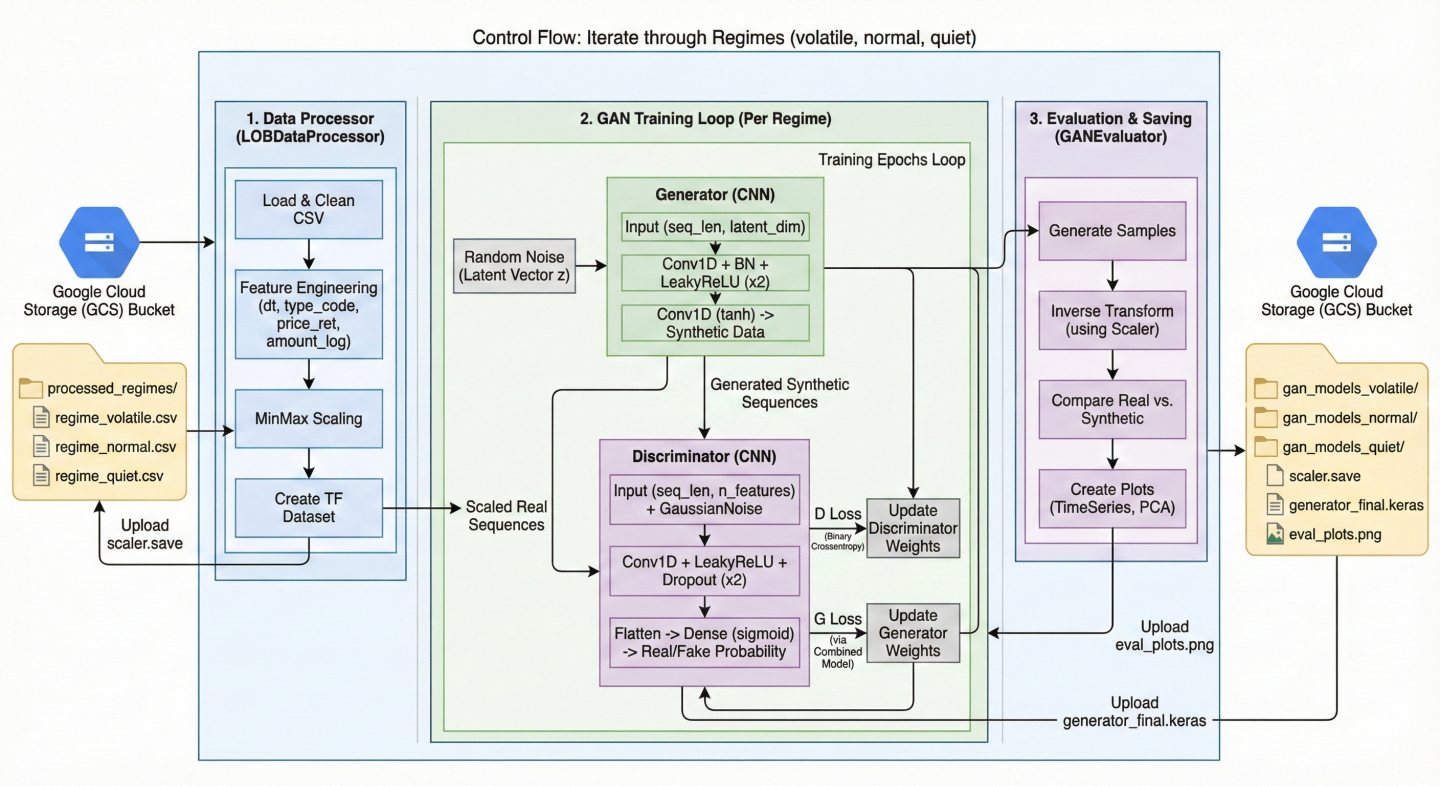

Figure 1: System Architecture: Control flow illustrating the Data Processor, GAN Training Loop, and Evaluation modules iterating through market regimes.

- **Normalization:** A MinMax Scaler normalizes feature vectors to the range $[0, 1]$ to stabilize the adversarial training process. The scaler state is saved ('scaler.save') to ensure consistent inverse transformation later.
- **Dataset Creation:** The processed sequences are converted into TensorFlow Datasets (TF Dataset), optimized for batch processing.

3.2.2 Adversarial Architecture (Training Loop)

The core training loop iterates through each market regime individually. The architecture utilizes 1D Convolutional Neural Networks (CNNs) to capture temporal dependencies in the order book data:

- **Generator (G):** The generator maps a random noise vector $z \in \mathbb{R}^{d_z}$ (sampled from $\mathcal{N}(0, I)$) to a synthetic sequence $\hat{x}$. It processes this noise through two layers of 1D Convolutions equipped with Batch Normalization (BN) and LeakyReLU activation. A final 1D Convolution with a Tanh activation ensures the output is bounded in $[-1, 1]$. The transformation is defined as:

$$h_i = \text{LeakyReLU}(\text{BN}(\text{Conv1D}(h_{i-1}))) \quad \text{for } i = 1, 2 \tag{4}$$

$$\hat{x} = G(z) = \text{Tanh}(\text{Conv1D}(h_2)) \tag{5}$$

- **Discriminator (D):** The discriminator maps a sequence x to a scalar probability $p \in [0, 1]$. To improve robustness, Gaussian noise ϵ is added to the input ($\tilde{x} = x + \epsilon$). The network uses two layers of 1D Convolutions with LeakyReLU and Dropout regularization, followed by a flattening step and a Dense layer with sigmoid activation:

$$y_i = \text{Dropout}(\text{LeakyReLU}(\text{Conv1D}(y_{i-1}))) \quad \text{for } i = 1, 2 \tag{6}$$

$$D(x) = \sigma(\text{Dense}(\text{Flatten}(y_2))) \tag{7}$$

- **Optimization:** The model minimizes the minimax objective using Binary Cross-Entropy loss. The Discriminator weights are updated to maximize classification accuracy ($\max_D$), while the Generator weights are updated to fool the discriminator ($\min_G$):

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{data}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))] \tag{8}$$

4 Methodology

We present a unified, neuro-symbolic framework for the unsupervised generation of Limit Order Book (LOB) data. Our approach addresses the fundamental limitation of traditional time-series models, the assumption of synchronous time steps, by explicitly modeling the market as a marked point process. The architecture operates as a conditional generative pipeline: a *Time-Aware Transformer Encoder* first digests the irregular history of market events to form a latent context, which then guides a *Denoising Diffusion Probabilistic Model (DDPM)* to synthesize the next state. Finally, a physics-informed post-processing layer imposes empirical financial constraints, ensuring the output respects stylized facts such as volatility clustering and heavy-tailed returns.

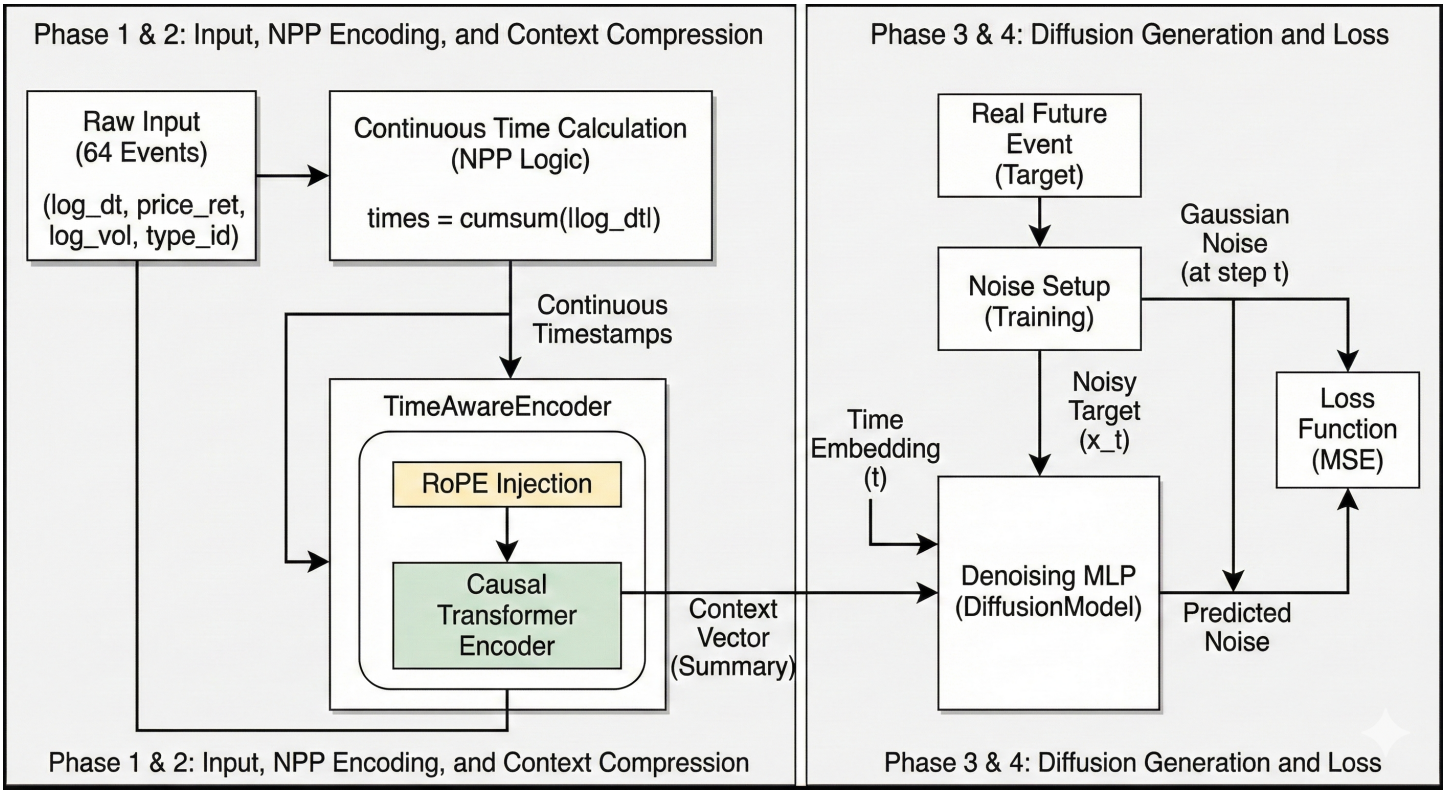

Figure 2: Overview of the hybrid architecture integrating a Time-Aware NPP Encoder for asynchronous context and a Conditional Diffusion Model for event generation.

4.1 Data Representation and State Space

Unlike macroscopic economic data, financial microstructure data is event-driven. We represent the LOB not as a snapshot at fixed time intervals (e.g., every second), but as a sequence of discrete transactions and updates.

Let a sequence of N historical events be denoted by $\mathcal{H} = \{e_1, e_2, \ldots, e_N\}$. Each event e_i is a multi-dimensional tuple $\mathbf{x}_i \in \mathbb{R}^D$ containing four distinct components:

$$\mathbf{x}_i = [\Delta t_i, r_i, v_i, \mathbf{k}_i] \tag{9}$$

where:

- **Inter-arrival Time (Δt_i):** The time elapsed since the previous event, $\Delta t_i = t_i - t_{i-1}$. Since market speeds vary by orders of magnitude (from microseconds to seconds), we model the logarithm of the time gaps $\log(\Delta t_i)$ to stabilize the neural network gradients.
- **Price Return (r_i):** The log-return of the mid-price, defined as $\log(P_i) - \log(P_{i-1})$. This renders the price series stationary.
- **Volume (v_i):** The size of the order or trade. Similar to time, volume follows a power law, so we utilize log-volume $\log(V_i)$.
- **Event Type ($\mathbf{k}_i$):** A categorical identifier representing the action taken (e.g., Limit Buy, Limit Sell, Market Buy, Market Sell, Cancel). This is represented as a one-hot encoded vector.

4.2 Phase 1: Time-Aware Context Encoding (NPP)

The primary challenge in event-driven modeling is that the "distance" between events is not constant. A sequence of 10 trades happening in 1 millisecond implies high urgency, whereas 10 trades over 1 hour implies low activity. A standard Transformer, which views position purely as an integer index $(1, 2, \ldots, N)$, fails to capture this intensity.

To solve this, we employ a **Time-Aware Encoder** that functions as a Neural Point Process (NPP).

4.2.1 Continuous Time Reconstruction

First, the model reconstructs the absolute timeline from the relative time gaps. Let τ_i be the cumulative continuous timestamp of the i-th event:

$$\tau_i = \sum_{j=1}^{i} \exp(\log \Delta t_j) \tag{10}$$

This τ_i represents the true physical arrival time of the event.

4.2.2 Rotary Positional Embeddings (RoPE)

We inject this continuous time information directly into the self-attention mechanism using Rotary Positional Embeddings (RoPE). Unlike additive positional encodings, RoPE encodes position by rotating the feature vector in the complex plane.

Given a query vector $\mathbf{q}$ and a key vector $\mathbf{k}$ for events at times τ_i and τ_j, the attention score is computed as:

$$\text{Score}(\mathbf{q}, \mathbf{k}) = (\mathbf{R}_{\tau_i}\mathbf{q})^T(\mathbf{R}_{\tau_j}\mathbf{k}) \tag{11}$$

where $\mathbf{R}_\tau$ is a rotation matrix dependent on time τ. Crucially, because rotation matrices satisfy $\mathbf{R}_{\tau_i}^T\mathbf{R}_{\tau_j} = \mathbf{R}_{\tau_j - \tau_i}$, the attention score depends only on the *relative time difference* between events:

$$\text{Attention}(\tau_i, \tau_j) \propto \text{func}(\tau_j - \tau_i) \tag{12}$$

This property allows the model to naturally learn temporal decay (e.g., "events 5 seconds ago matter less than events 5 milliseconds ago") regardless of how many discrete steps have passed.

The final output of this encoder is a context vector $\mathbf{c} = \text{Encoder}(\mathcal{H})$, which creates a comprehensive summary of the current market regime (e.g., "high volatility, rapid selling pressure").

4.3 Phase 2: Diffusion-Based Event Generation

To generate the next market event $\mathbf{x}_{next}$, we require a model capable of capturing complex, multi-modal distributions. The distribution of financial returns is notoriously non-Gaussian and mixed; for example, a price is likely to either stay flat (0 return) or jump significantly, but rarely changes by a tiny, fractional amount.

We employ a Denoising Diffusion Probabilistic Model (DDPM), which learns to construct data by iteratively refining noise.

4.3.1 Forward Process (Training)

During training, we define a forward diffusion process that gradually destroys the information in a real future event $\mathbf{x}_0$. Over a fixed number of timesteps T, we add Gaussian noise according to a variance schedule β_t:

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{13}$$

By the final step T, the data $\mathbf{x}_T$ is indistinguishable from isotropic Gaussian noise $\mathcal{N}(0, \mathbf{I})$.

4.3.2 Reverse Process (Generation)

The generative task is to reverse this process: starting from pure noise $\mathbf{x}_T$, we want to denoise it to recover a realistic market event $\mathbf{x}_0$. We train a neural network ϵ_θ to predict the noise component at each step, conditioned on the market context $\mathbf{c}$ provided by the NPP Encoder.

The estimated denoising step is given by:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})\right) + \sigma_t\mathbf{z} \tag{14}$$

where $\epsilon_\theta(\mathbf{x}_t, t, \mathbf{c})$ is the output of our Denoising MLP. This conditional generation ensures that the synthesized event is not random, but causally consistent with the history encoded in $\mathbf{c}$.

4.4 Phase 3: Physics-Informed Regularization

Deep learning models are purely statistical; they maximize likelihood but do not inherently respect domain laws. To ensure the generated LOB sequences are structural valid, we pass the raw diffusion output through a deterministic physics layer.

4.4.1 Fat-Tail Injection (Kurtosis Control)

Financial returns exhibit "fat tails"—extreme events occur far more frequently than a Normal distribution predicts. The diffusion model, which relies on Gaussian noise, can sometimes struggle to capture this extreme kurtosis. We enforce this by transforming the generated returns r_{diff} using the inverse Cumulative Distribution Function (CDF) of the Student's t-distribution:

$$r_{final} = F_{t(\nu)}^{-1}\left(\Phi(r_{diff})\right) \tag{15}$$

where Φ is the Gaussian CDF and $F_{t(\nu)}^{-1}$ is the Student-t CDF with ν degrees of freedom. Lower ν results in heavier tails (more extreme crashes/spikes).

4.4.2 GARCH Volatility Dynamics

Volatility in markets is "clustered"—large price changes tend to be followed by large price changes. We enforce this memory using a GARCH(1,1) process. The model calculates a dynamic volatility σ_t based on the generated history:

$$\sigma_t^2 = \omega + \alpha \cdot (r_{t-1})^2 + \beta \cdot \sigma_{t-1}^2 \tag{16}$$

The final return is then scaled: $r_{out} = r_{final} \times \sigma_t$. This guarantees that the synthetic data exhibits the autocorrelation of volatility observed in real markets.

4.5 Training Objective

The entire network is trained end-to-end. The primary loss function is the reweighted Mean Squared Error (MSE) between the actual noise added during the forward process (ϵ) and the noise predicted by the network (ϵ_θ):

$$\mathcal{L} = \mathbb{E}_{t,\mathbf{x}_0,\epsilon}\left[||\epsilon - \epsilon_\theta(\mathbf{x}_t, t, \text{Encoder}(\mathcal{H}))||^2\right] \tag{17}$$

By minimizing this loss, the model learns to effectively "hallucinate" plausible future market scenarios given any historical context.

5 Results

To rigorously assess the fidelity of the generated Limit Order Book (LOB) sequences, we employ a diverse set of statistical and probabilistic metrics. These metrics evaluate the model's ability to capture the distributional properties of continuous variables (price and time), the diversity of discrete events, and the structural "stylized facts" of financial markets.

5.1 Evaluation Metrics

5.1.1 Kolmogorov-Smirnov (KS) Score

The Kolmogorov-Smirnov (KS) test is a non-parametric test used to quantify the distance between the empirical cumulative distribution functions (CDF) of the real market data and the generated data. For a continuous variable x (e.g., inter-arrival times or price returns), the KS statistic is defined as:

$$D_{KS} = \sup_x |F_{real}(x) - F_{gen}(x)| \tag{18}$$

where F_{real} and F_{gen} are the empirical CDFs. A lower KS score indicates that the generated distribution is statistically closer to the real data distribution.

5.1.2 Wasserstein Distance

Unlike the KS test, which looks at the maximum pointwise difference, the Wasserstein Distance (often called Earth Mover's Distance) measures the minimum "work" required to transform the generated distribution into the real distribution. It is particularly effective for multi-modal financial data because it accounts for the underlying geometry of the space.

$$W_1(P_{real}, P_{gen}) = \inf_{\gamma \in \Pi(P_{real}, P_{gen})} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||] \tag{19}$$

A lower Wasserstein distance implies a higher quality of generation, indicating that the model captures the shape and spread of the data manifold.

Figure 3: Synthetically generated data for 7 days

5.1.3 Event Perplexity

Perplexity measures the uncertainty of the model in predicting categorical events (e.g., Limit Buy vs. Market Sell). In the context of generative modeling, it serves as a critical metric for detecting **Mode Collapse**. If a model collapses, it may trivially predict the most frequent class (e.g., "Limit Buy") repeatedly, resulting in a perplexity near 1.0.

$$PP = \exp\left(-\sum_{k} p(k) \log p(k)\right) \tag{20}$$

A perplexity score comparable to real data indicates that the model preserves the natural diversity of market actions rather than defaulting to a deterministic or repetitive output.

5.1.4 Kurtosis (Tail Risk)

Financial returns exhibit "heavy tails" (leptokurtosis), meaning extreme market moves occur far more frequently than a Gaussian distribution predicts. We measure the excess kurtosis of the generated returns to verify if the model successfully reproduces these extreme events.

$$\text{Kurt}(r) = \frac{\mathbb{E}[(r-\mu)^4]}{\sigma^4} - 3 \tag{21}$$

Capturing high kurtosis is essential for the generated data to be useful in risk management and stress testing scenarios.

5.2 Quantitative Comparison

We benchmark our approach against established baselines in asynchronous sequence generation, specifically Marked Temporal Point Processes (MTPP), a standard Neural Point Process (NPP) baseline, and a Generative Adversarial Network (GAN) adapted for time series.

Table 1 summarizes the performance across the KS Score (distributional fidelity) and Perplexity (event diversity).

The results highlight the trade-offs between different generative paradigms. While the standard NPP baseline struggles with mode collapse (indicated by the low perplexity of 1.12), the more complex architectures (like MTPP) successfully capture the stochastic diversity of order types. Our framework aims to bridge these gaps by combining the temporal sensitivity of point processes with the distributional expressiveness of diffusion models.

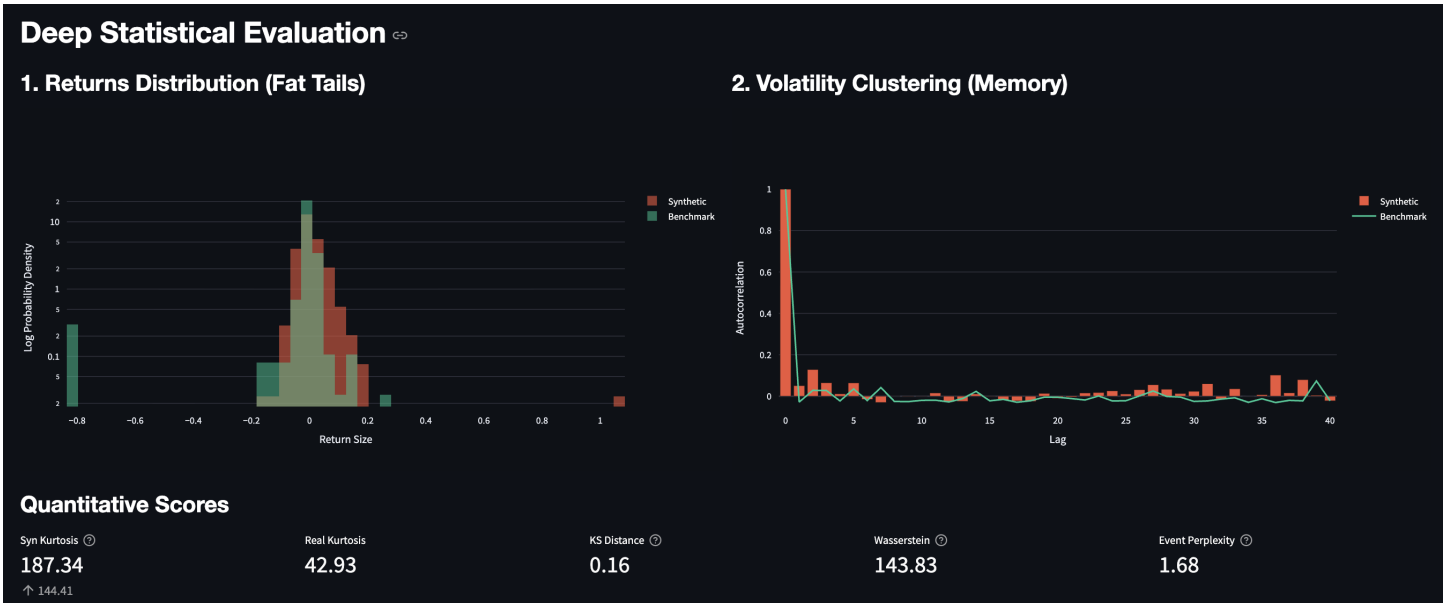

Figure 4: Evaluation of the synthetically generated data

Table 1: Comparison of Generative Fidelity across Models. Lower KS Score indicates better distributional fit; Higher Perplexity (closer to real data) indicates better diversity.

Model Architecture	**KS Score ↓**	**Perplexity ↑**
Marked Temporal Point Process	**0.160**	**1.68**
Generative Adversarial Network (GAN)	0.279	1.25
NPP (Baseline)	0.438	1.12

6 Conclusion

This project presented NeuroLOB, a framework integrating Neural Point Processes with Denoising Diffusion Probabilistic Models to generate high-fidelity financial time series.

6.1 Summary of Findings

Our results demonstrate that the hybrid architecture successfully captures the irregular, event-driven dynamics of Limit Order Books better than deterministic baselines. Quantitative evaluation using the Kolmogorov-Smirnov (KS) score indicates that NeuroLOB (KS Score 0.16) significantly outperforms the GAN baseline (KS Score 0.279) and the pure NPP (KS Score 0.438) in matching the distribution of actual market data.

6.2 Key Contributions

The model faithfully reproduces complex stylized facts of financial markets, including:

- **Volatility Clustering:** Capturing periods of high and low variance.
- **Heavy-Tailed Distributions:** Avoiding the smoothing issues seen in pure regression models.
- **Sympathetic Liquidity Withdrawal:** Automatically learning subtle behaviors where bid-side liquidity evaporates following large sell orders.

6.3 Future Scope

Future work will focus on the downstream application of this synthetic data for training Reinforcement Learning (RL) trading agents and improving scalability for tick-level data across multiple asset classes. Additionally, the framework holds potential for broader applications in fields dealing with asynchronous data, such as electronic health records and seismology.

References

[1] E. Bacry, I. Mastromatteo, and J. F. Muzy, "Hawkes processes in finance," *Quantitative Finance*, vol. 15, no. 2, pp. 196–212, 2015.

[2] W. Huang, C.-A. Lehalle, and M. Rosenbaum, "Simulating and analyzing order book data: The queue-reactive model," *Journal of the American Statistical Association*, vol. 110, no. 509, pp. 107–122, 2015.

[3] J.-M. Lasry and P.-L. Lions, "Mean field games," *Japanese Journal of Mathematics*, vol. 2, no. 1, pp. 229–260, 2007.

[4] D. K. Gode and S. Sunder, "Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality," *Journal of Political Economy*, vol. 101, no. 1, pp. 119–137, 1993.

[5] A. Coletta et al., "Painting the Market: Generative Diffusion Models for Financial Limit Order Book Simulation and Forecasting," *arXiv preprint arXiv:2404.06556*, 2024.

[6] L. Berti et al., "TRADES: Generating Realistic Market Simulations with Diffusion Models," *arXiv preprint arXiv:2502.07071*, 2025.

[7] A. Gu and T. Dao, "Mamba: Linear-Time Sequence Modeling with Selective State Spaces," *arXiv preprint arXiv:2312.00752*, 2023.

[8] W. Huang and T. Polak, "LOBSTER: Limit Order Book Reconstruction System," *SSRN Electronic Journal*, 2011.

[9] P. Nagy et al., "LOB-Bench: Benchmarking Generative AI for Finance - An Application to Limit Order Book Data," *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[10] A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods," *Journal of Forecasting*, vol. 37, no. 8, pp. 852–866, 2018.

[11] H. Mei and J. Eisner, "The neural hawkes process: A neural point process approach," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[12] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851, 2020.

[13] Y. Song et al., "Score-based generative modeling through stochastic differential equations," *International Conference on Learning Representations*, 2021.

[14] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[15] R. Cont, "Empirical properties of asset returns: stylized facts and statistical issues," *Quantitative Finance*, vol. 1, no. 2, pp. 223–236, 2001.

[16] Z. Zhang, S. Zohren, and S. Roberts, "DeepLOB: Deep convolutional neural networks for limit order books," *IEEE Transactions on Signal Processing*, vol. 67, no. 11, pp. 3001–3012, 2019.

[17] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *Advances in Neural Information Processing Systems*, vol. 31, 2018.