DeepLOB on Synthetic Limit Order Book Data: A Practical Study in Microstructure Prediction

Joshua de Freitas

November 14, 2025

Abstract

Modern electronic markets generate large volumes of limit order book (LOB) data. Deep learning architectures such as DeepLOB have shown that it is possible to predict short-horizon price movements directly from sequences of LOB snapshots. This project implements an end-to-end DeepLOB-style pipeline on synthetic LOB data: simulation, tensorisation, model training, evaluation and backtesting. In addition to a DeepLOB architecture based on convolutional and recurrent modules, a Temporal Convolutional Network (TCN) is implemented and compared on the same dataset. The goal is not to produce a production-ready trading strategy, but to build a clear, inspectable framework for experimenting with market microstructure models.

1 Introduction

Electronic markets operate through continuous double auctions. At any point in time, the state of the market is summarised by the *limit order book* (LOB), which contains all outstanding buy (bid) and sell (ask) orders at discrete price levels. The book is updated whenever orders arrive, are cancelled or are matched into trades.

Short-horizon changes in the mid-price—the average of the best bid and best ask—are closely tied to the dynamics of the book. Traders and market makers use these microstructure signals to adjust quotes, manage inventory and control risk. This makes the LOB an attractive input for predictive models.

The aim of this project is to implement and study a DeepLOB-style architecture on a controlled, synthetic dataset. Rather than starting from noisy exchange data with many microstructure details, a simulated LOB is used to:

- understand the full data pipeline end-to-end,
- test different architectures under identical conditions, and
- produce a portfolio-ready codebase that can later be extended to real data.

2 Limit Order Book Representation

A limit order book stores, at each price level, a price and a quantity for both the bid and ask side. In this project, the simulated LOB exposes L levels on each side. Each snapshot contains:

- bid prices $(b_1, \ldots, b_L)$,
- bid sizes $(q_1^b, \ldots, q_L^b)$,
- ask prices $(a_1, \ldots, a_L)$,

- ask sizes $(q_1^a, \ldots, q_L^a)$.

The *mid-price* at time t is defined as

$$\text{mid}_t = \frac{\text{bestBid}_t + \text{bestAsk}_t}{2}. \tag{1}$$

The predictive task is to classify the direction of the mid-price over a short future horizon.

2.1 Simulation

The simulator generates a synthetic mid-price process with small random shocks and constructs a consistent LOB around it. Spread and depth are perturbed to mimic varying liquidity. Each row of the simulated dataset contains the mid-price and a fixed number of bid/ask prices and sizes at different levels.

Although stylised, this setup provides a convenient sandbox: the exact generative process is known, the data is clean, and experiments are fully reproducible.

3 From LOB Snapshots to Tensors

Deep models operate on fixed-size tensors, not on arbitrary time series. The project therefore converts raw LOB data into overlapping windows.

3.1 Window Construction

Let the raw dataset contain N snapshots. For each index t, a window of length T is constructed:

$$X_t = \big(\mathbf{x}_{t-T+1}, \ldots, \mathbf{x}_t\big), \tag{2}$$

where each $\mathbf{x}_k \in \mathbb{R}^F$ is a feature vector derived from the LOB at time k (prices, sizes and optionally mid-price or returns).

This yields a 3D tensor:

$$X \in \mathbb{R}^{N_{\text{samples}} \times T \times F}.$$

3.2 Labels

For each window ending at time t, the label is based on the movement of the mid-price over the next H steps:

$$r_t = \frac{\text{mid}_{t+H} - \text{mid}_t}{\text{mid}_t}. \tag{3}$$

A simple three-class labelling is used:

$$y_t = \begin{cases} +1 & \text{if } r_t > \varepsilon, \\ 0 & \text{if } |r_t| \le \varepsilon, \\ -1 & \text{if } r_t < -\varepsilon, \end{cases}$$

for a small threshold $\varepsilon > 0$.

In the synthetic dataset used here, the final tensor has shape $4\,891 \times 100 \times 15$ (samples $\times$ timesteps $\times$ features).

4 Model Architectures

Two architectures are implemented and trained on the same data.

4.1 DeepLOB-style Model

The DeepLOB-style model is inspired by the architecture proposed by Zhang et al.:

- 1D convolutional layers to extract local patterns across features,
- an inception-style block with multiple kernel sizes to capture patterns at different temporal scales,
- a recurrent layer (LSTM) to model temporal dependencies, and
- a final fully connected classifier with softmax output.

Convolutions are applied along the time dimension. The inception module runs several convolutions with different receptive fields in parallel and concatenates their outputs. The LSTM then sees this enriched representation and produces a sequence of hidden states from which the last state is fed into the classifier.

4.2 Temporal Convolutional Network (TCN)

The TCN replaces the recurrent component with stacked 1D convolutions that are *causal* (no information from the future) and *dilated* (gaps between kernel positions). Dilated convolutions allow the receptive field to grow exponentially with depth, enabling the network to see far back in time with relatively few layers.

In noisy, high-frequency environments, TCNs often train more easily than recurrent networks and can capture long-range dependencies with fewer parameters.

5 Training and Evaluation

The dataset is split into training and validation subsets (80% / 20%). Models are trained with cross-entropy loss and the Adam optimiser. The primary classification metrics are overall accuracy and macro-averaged F1 score.

5.1 Classification Results

Table 1 reports validation accuracy and macro F1 for both models on the synthetic dataset.

Table 1: Classification performance on synthetic LOB data.

Model	Accuracy	Macro F1
DeepLOB	0.6263	0.6264
TCN	0.6444	0.6518

Per-class metrics for the TCN show that the model balances all three classes reasonably well.

Table 2: TCN per-class performance.

Class	Precision	Recall	F1	Support
−1 (down)	0.731	0.653	0.690	1 581
0 (flat)	0.563	0.576	0.569	1 899
+1 (up)	0.669	0.726	0.696	1 411

6 Backtesting

To move beyond pure classification metrics, a simple trading rule is applied:

- If the model predicts *up*, take a long position.
- If the model predicts *down*, take a short position.
- If the model predicts *flat*, stay in cash.

Returns are computed from the realised mid-price movement over the prediction horizon. This yields a sequence of trade returns from which performance statistics are computed: total P&L, average return per trade, win rate, Sharpe ratio and maximum drawdown.

Table 3 summarises the backtest results.

Table 3: Backtest statistics (synthetic data, naive long/short rule).

Metric	DeepLOB	TCN
Total P&L	1.7438	2.1585
Avg return / trade	0.000357	0.000441
Win rate	39.3%	52.8%
Sharpe (per trade)	0.560	0.667
Max drawdown	0.0095	0.0073

On this synthetic dataset, the TCN generates a signal that is not only better in terms of classification, but also yields higher simulated P&L, a higher Sharpe ratio and a lower drawdown.

7 Discussion

Several lessons emerge from this experiment:

- **Representation matters.** The step from raw LOB snapshots to clean, well-shaped tensors is as important as the model itself. The pipeline implemented here—from simulation to window construction and labelling—is reusable for real LOB data.
- **DeepLOB is strong, but TCNs are competitive.** Even on synthetic data, the TCN slightly outperforms the DeepLOB-style architecture, both in classification metrics and backtest results. This aligns with broader experience in time series modelling where convolutional architectures often train faster and generalise well.
- **Accuracy is not the whole story.** Two models with similar accuracy can have very different trading performance. A backtesting layer is essential to understand how model predictions translate into portfolio outcomes.
- **Synthetic data is a useful training ground.** Working in a controlled environment makes it easier to inspect each component of the pipeline. Once the framework is stable, it can be extended to real market feeds.

8 Conclusion and Future Work

This project provides an end-to-end implementation of a DeepLOB-style system on synthetic LOB data, including simulation, tensorisation, model training, evaluation and backtesting. It demonstrates how to turn high-frequency microstructure data into a supervised learning problem and how to benchmark different deep architectures in a consistent way.

Several extensions suggest themselves:

- ingestion of real exchange LOB data,
- more advanced architectures (TCN variants, Transformers),
- regime-aware models that adapt to changing volatility,
- execution-aware backtests incorporating transaction costs and latency,
- integration into a broader research or trading environment.

The codebase is intentionally modular, so that each of these directions can be explored without rewriting the foundation.

Code and Documentation

The full implementation, together with additional documentation and experiment artifacts, is available in the project repository:

`deep-lob/` (data pipeline, models, training, evaluation, backtests)
`docs/deeplob_overview.md` (technical overview)