Convolutional–Recurrent Modeling of Limit Order Book Dynamics with Structured Feature Representations

Internal Research Note

November 17, 2025

Abstract

This document describes the design of a convolutional–recurrent model for predicting the future intensity of upward and downward price moves in a limit order book. The focus is on the representation of order book states as structured feature tensors that can be consumed by a convolutional neural network (CNN) followed by a long short–term memory (LSTM) layer. We formalize the problem setting, specify the assumptions on data and configuration, and detail the construction of snapshot features and their mapping to the spatial grid used as model input. The objective is to provide a precise and reproducible description of the algorithmic choices that govern how raw order book observations are transformed into model inputs, in a way that is suitable for scientific scrutiny and future extension.

1 Introduction

Modern electronic markets generate high–frequency streams of limit order book states. Each state reflects the outstanding buy and sell interest at multiple price levels. Trading strategies and execution algorithms often depend on predicting short–horizon price movements and their intensities, rather than only direction. Data–driven approaches based on deep learning architectures are natural candidates for this task, provided that the representation of the order book is expressive enough to capture relevant microstructural patterns while remaining computationally tractable.

The system under consideration aims to predict, for each observation time, the distribution over discrete intensity classes of future upward and downward price moves. The prediction target is represented by two output heads, corresponding respectively to the upward and downward move intensity, each parameterized as a categorical distribution over a fixed set of magnitude bins. These bins are defined by configuration parameters that specify positive percentage thresholds for the future maximum price movement. The design of the input representation must therefore preserve information that is relevant for these targets, such as the shape of the book, depth at multiple levels, and imbalances between the bid and ask sides.

The present note focuses on the design of the convolutional–recurrent model and, in particular, on the construction of the feature tensor that encodes the state of the order book at each observation time. The goal is to describe the assumptions, the mathematical formulation, and the configuration interface in a way that allows a practitioner to reproduce the model and understand the trade–offs implicit in each design choice.

2 Problem setting and assumptions

We consider a single traded instrument with a limit order book observed at discrete times. Let $t \in \{1, \ldots, T\}$ index observation times, and let S_t denote the state of the order book at time t for the instrument of interest. At each time t, for each side $s \in \{b, a\}$ corresponding to bid and ask, and for each level $\ell \in \{1, \dots, L\}$, we observe a price $P_t^{s,\ell}$ and an available quantity $Q_t^{s,\ell}$. Level $\ell = 1$ corresponds to the best bid or best ask, and higher levels correspond to decreasing competitiveness on that side of the book.

The predictive task is defined with respect to a fixed prediction horizon Δt expressed in seconds or in an equivalent time unit. For each observation time t, we define the maximum upward and downward price moves realized over the future interval $[t, t + \Delta t]$ relative to a reference price at time t, such as the mid price. These maximum moves are expressed as percentage changes. The configuration specifies positive magnitude thresholds $0 < b_1 < b_2 < \cdots < b_K$ that partition the positive real line into $K + 1$ bins. The upward intensity class at time t is defined by assigning the maximum upward move percentage to one of these bins, including the bin corresponding to no or negligible movement. The downward intensity class is defined analogously using the magnitude of downward moves.

We assume that the system operates in a regime where configuration is explicit and centralized. All numerical values that govern the model structure, feature construction, and target definition are supplied by a configuration file in YAML format. The code does not introduce numerical defaults for such parameters. If a required configuration key is missing or of incompatible type, the system fails with an explicit error before training or evaluation proceeds. This design ensures that experimental runs are fully reproducible and that any change in the model or feature space is reflected in the configuration.

From an engineering perspective, we further assume that the data set has been preprocessed to define chronological splits into training, validation, and test sets, and that all computations respect this temporal structure. In particular, any statistics used for feature normalization are computed using training data only and are then reused for validation and test data. This prevents information leakage across splits and preserves the integrity of the evaluation.

3 Representation of order book snapshots

At each observation time t, the raw order book state S_t is transformed into a fixed–shape tensor X_t that will serve as input to the convolutional network. The representation is designed to capture information across multiple price levels and to distinguish the bid and ask sides while maintaining a regular grid structure that is well aligned with convolutional operations.

The construction proceeds as follows. A configuration parameter L specifies the maximum number of price levels to include on each side of the book. For each side s, we consider the sequence of prices and quantities $(P_t^{s,1}, Q_t^{s,1}), \dots, (P_t^{s,L}, Q_t^{s,L})$. If fewer than L levels are available on a side at time t, the missing levels are represented by a consistent placeholder, such as zero or a neutral value after normalization, depending on the chosen transformation. The configuration further specifies a set of per–level feature functions that map $(P_t^{s,\ell}, Q_t^{s,\ell})$ and, when needed, global quantities such as the mid price into scalar features. Typical examples of such features include the price relative to the mid price, the raw quantity, the cumulative quantity up to level ℓ on a given side, and normalized measures of depth.

Formally, let $\mathcal{F}$ denote the collection of selected per–level feature functions. For each $f \in \mathcal{F}$, side s, and level ℓ, we compute a scalar value $f_t^{s,\ell} = f(S_t, s, \ell)$. The configuration defines $|\mathcal{F}|$, the number of per–level features, and determines the precise mathematical form of each f. In addition to per–level features, the design may include aggregate features that depend on the best levels of both sides, such as the spread or a measure of order book imbalance. These aggregate features are scalar functions of S_t and are incorporated into the tensor representation through an additional mapping described below.

The spatial grid is organized so that each row corresponds to a level index ℓ and each column corresponds to a specific combination of side and feature. For a given level ℓ, the row of the grid contains the feature values $f_t^{b,\ell}$ for all $f \in \mathcal{F}$, followed by the feature values $f_t^{a,\ell}$ for all $f \in \mathcal{F}$. If there are $F = |\mathcal{F}|$ per–level features and two sides, the resulting grid has height $H = L$ and width $W = 2F$. The tensor X_t is then constructed as a three–dimensional array in $\mathbb{R}^{H \times W \times C}$, where C denotes the number of channels. In the initial design, all features are placed in a single channel, so that $C = 1$, and the values described above populate this channel. Aggregate features that do not vary with level can be incorporated either by repeating them across all levels in dedicated columns or by embedding them in additional channels. The choice is governed by configuration and must be consistent with the expectations of the convolutional architecture.

The mapping from raw order book data to the tensor X_t is deterministic given the configuration. The spatial arrangement is fixed once the set of per–level features, the number of levels, and the placement of aggregate features are specified. This explicit design allows the practitioner to reason about which microstructural patterns the convolutional layers can potentially capture, such as local slopes of the order book, clustering of liquidity near the mid price, or asymmetries between the bid and ask sides.

4 Normalization and statistical treatment

The numerical scales of prices and quantities can vary substantially over time and across instruments. To stabilize training and to facilitate the use of a single model architecture across different data sets, the feature representation is normalized according to a scheme defined in the configuration. The system supports several normalization strategies, such as standardization to zero mean and unit variance (often referred to as z–score normalization) and min–max scaling to a fixed interval. The choice of normalization is specified by a configuration key, and the code does not default to any particular scheme in the absence of this key.

For a given choice of normalization, the procedure operates in two phases. During preprocessing of the training data, the system computes the necessary statistics of each feature, such as the mean and standard deviation for z–score normalization, using only the training subset as defined by the chronological split. These statistics are stored in the metadata associated with the data set. When constructing tensors for validation and test data, the system reuses the stored statistics and applies the same normalization transformation without recomputing the statistics. This approach ensures that the validation and test sets are treated consistently with the training set and that no information from future data leaks into the training process through the normalization step.

The normalization of aggregate features follows the same principles. Each aggregate feature is treated as a distinct dimension with its own statistics. If aggregate features are replicated across levels in the spatial grid, the same normalized value is used in each corresponding cell, preserving the semantics of a global quantity while maintaining compatibility with the grid structure.

5 Convolutional–recurrent architecture

Once the tensor representation X_t has been constructed and normalized, it is provided as input to a convolutional neural network followed by an LSTM layer. The architecture is defined through configuration parameters that specify the number of convolutional layers, the number of filters in each layer, the kernel sizes, the pooling sizes, activation functions, and dropout rates. The configuration does not introduce hidden defaults; any missing parameter that is required for model construction results in an error.

The convolutional stack operates on X_t and produces a sequence of intermediate feature maps. Each convolutional layer applies a set of two–dimensional filters that convolve across levels and across the feature dimension. This allows the network to learn local patterns in the order book structure, such as the rate at which quantities decay with level or local asymmetries between the bid and ask sides. Max pooling layers reduce the spatial resolution while preserving the strongest responses, which helps achieve invariance to small shifts and reduces the dimensionality of subsequent layers.

After the final convolutional and pooling layers, the resulting feature map is reshaped into a sequence that forms the input to the LSTM layer. In the current design, the spatial dimensions are flattened into a one–dimensional sequence of feature vectors, each associated with a particular spatial location in the original grid. The LSTM processes this sequence and produces a fixed–dimensional representation that summarizes the most informative spatial patterns discovered by the convolutional layers. Although the LSTM is often used to model temporal sequences, here it is repurposed to model ordered spatial patterns. This design choice is motivated by the desire to capture dependencies along structured dimensions without introducing a separate temporal axis in the model input.

The output of the LSTM is then passed through one or more fully connected layers with non–linear activation functions. These layers project the representation into a space that is appropriate for the final prediction task. The last stage of the architecture comprises two separate dense output heads. Each head maps the final hidden representation to a vector of logits of dimension $K + 1$, corresponding to the upward or downward intensity classes defined by the magnitude thresholds. A softmax activation is applied to each head, yielding a categorical distribution over intensity bins for upward and downward moves respectively.

6 Training objective and configuration

The model is trained to approximate the joint distribution of upward and downward move intensities at each observation time. The training objective is formulated as the sum of the categorical cross–entropy losses for the two heads. For each sample, the true upward intensity class is encoded as a one–hot vector of dimension $K + 1$, and the same is done for the downward intensity class. The loss function for the upward head is the cross–entropy between the true one–hot vector and the predicted probability vector, and similarly for the downward head. The total loss is the sum of these two quantities, possibly averaged over the batch.

Optimization is performed using a variant of stochastic gradient descent, such as the Adam optimizer, as specified in the configuration. The learning rate, optimizer type, loss function identifier, and the list of evaluation metrics are all drawn from the configuration file and must be present for the model to be compiled. In particular, for a model with two output heads, the metrics configuration must map appropriately to each head. The system interprets a list of metric names as a base set to be applied symmetrically to both heads or allows a more explicit mapping when the configuration provides a dictionary from head names to metric lists.

Throughout training, the data are split chronologically into training and validation sets. The model is fitted on the training set, and performance is monitored on the validation set using the configured metrics. Model checkpoints, learning curves, and other artifacts may be logged to an experiment tracking system, but the details of that integration lie outside the scope of this note. The critical point is that both the architecture and the feature representation are fully determined by the configuration and the design outlined above, allowing for rigorous experimentation and comparison across different configurations.

7 Conclusion

This document has presented a detailed description of the CNN–LSTM architecture and the associated feature representation used to model limit order book dynamics for the purpose of predicting future price move intensities. The central design idea is to encode each order book snapshot as a structured grid of features across price levels and sides, normalized according to statistics computed on training data, and to process this grid with convolutional and recurrent layers before producing two separate intensity distributions. By enforcing a strict separation between configuration and implementation and by avoiding implicit numerical defaults, the system facilitates reproducible research and careful ablation studies on the impact of each representational and architectural choice.