LOBERT: Generative AI Foundation Model for Limit Order Book Messages

Eljas Linna
Data Science Research Centre
Tampere University
eljas.linna@tuni.fi

Kestutis Baltakys
Data Science Research Centre
Tampere University
kestutis.baltakys@tuni.fi

Alexandros Iosifidis
Data Science Research Centre
Tampere University
alexandros.iosifidis@tuni.fi

Juho Kanniainen
Data Science Research Centre
Tampere University
juho.kanniainen@tuni.fi

Abstract

Modeling the dynamics of financial Limit Order Books (LOB) at the message level is challenging due to irregular event timing, rapid regime shifts, and the reactions of high-frequency traders to visible order flow. Previous LOB models require cumbersome data representations and lack adaptability outside their original tasks, leading us to introduce LOBERT, a general-purpose encoder-only foundation model for LOB data suitable for downstream fine-tuning. LOBERT adapts the original BERT architecture for LOB data by using a novel tokenization scheme that treats complete multi-dimensional messages as single tokens while retaining continuous representations of price, volume, and time. With these methods, LOBERT achieves leading performance in tasks such as predicting mid-price movements and next messages, while reducing the required context length compared to previous methods.

1 Introduction

The Limit Order Book (LOB) stands as the cornerstone of modern digital markets, serving as the primary mechanism for price discovery and a critical indicator of market liquidity and risk. The ability to accurately model and predict the LOB message flow is crucial as its features underpin market making, optimal execution, and other high-frequency trading activities. Moreover, other quantities describing security price dynamics can be derived from it. However, modeling the LOB at the message level brings a unique and difficult set of challenges: Extreme irregularity in time intervals varying by a factor of millions, rapid regime shifts with cascading effects caused by a single message, and amplification of these dynamics through the interaction of agents with the visible LOB message flow where trading algorithms react to the actions of each other.

While we believe using Generative AI with LOB messages has promise by capturing the most atomic data of LOB markets, previous generative approaches suffer from slow inference, narrow scope of application, and the need to split each message into components that must be predicted sequentially (see Appendix A). To address the shortcomings, we introduce LOBERT, a general-purpose *encoder-only* foundation model which adapts the highly successful BERT model from [4] for message-level LOB modeling. Its novelty is fourfold. First, we propose a *one-token-per-message* tokenizer that consolidates side, type, and coarse price/volume cues (with a roundness flag), while preserving *continuous* price, volume, and time through a Piecewise Linear–Geometric scaling, avoiding the token explosion of prior generators that split each message into 22–24 sub-tokens. Second, we adapt BERT to irregular event streams via *continuous-time rotary* attention and a *Masked Message Modeling* objective that jointly masks messages and their surrounding snapshots to prevent label leakage. Third, we introduce a *hybrid discrete–continuous* decoding head and a *Combined* inference scheme that uses token bins to bound the regressors, yielding higher marginal fidelity than token-only or regressor-only variants. Fourth, we demonstrate practical *efficiency and transfer*: by predicting a single token per message, LOBERT reduces effective context length by roughly an order of magnitude on our data (i.e., $\approx$20$\times$ fewer tokens per sequence) while fine-tuning cleanly to diverse downstreams (e.g., next-message and mid-price direction) with small task-specific heads. To our knowledge, this specific combination of tokenization, time-aware encoder pretraining, and hybrid decoding has not been explored for LOB messages.

2 Methodology

2.1 Message preprocessing

The main principle behind LOBERT's preprocessing is to produce discrete tokens that represent the most common messages and an approximation of the less common messages, e.g. "new sell order far above the best price with medium volume". This is combined with continuous representation of the information, allowing precisely pinpointing rare values without having to excessively bloat the vocabulary of discrete values.

First, the **price level** of each message is transformed to stationary price difference measured in ticks from the best opposing price, i.e. the best bid for sell orders and the best ask for buy orders. The price difference is quantized to levels (0, 1, 2, 3, 5, 10) while a duplicate of the price difference is scaled through Piecewise Linear-Geometric Scaling (PLGS, see Appendix B) with parameters $\tau_{\text{start}} = 10$ ticks, $\tau_{\text{max}} = 20$, and $\tau_{\text{clip}} = 1000$ ticks. This means the scaling function produces linear output growth up to 10 ticks (50% of the maximum scaled value), after which growth slows geometrically toward the asymptotic limit. The quantization levels are chosen to distribute the values as evenly as possible across the most common price movements observed in the training data.

Second, the **volume** is processed with the same method of quantizing to levels (0, 50, 100, 200) and duplicating into a continuous scaled value through PLGS with $\tau_{\text{start}} = 200$ units, $\tau_{\text{max}} = 400$, and $\tau_{\text{clip}} = 1500$ units. The quantization levels are chosen by analyzing the distribution of volumes and observing that volumes are heavily concentrated at round values such as 50, 100, and 200 (over 60% of all volume values are exactly 100 units in the dataset). We also include an additional binary indicator to denote whether the volume is exactly on the quantized level or between levels ("Y" or "N"), which proves essential for reconstructing the original volume values during inference.

Third, the arrival **time** of each message is converted to time difference measured from the previous message. Since we did not identify clear repeating patterns in time difference distribution beyond its heavy-tailed nature, we represent it solely with a continuous value scaled through PLGS with parameters $\tau_{\text{start}} = 1\text{ms}$, $\tau_{\text{max}} = 50$, and $\tau_{\text{clip}} = 250\text{ms}$. The final normalized values sit in a range of [0, 1].

Finally, the **side** (buy or sell) and **message type** (new order, edit, delete, execution, hidden order) are combined with the quantized price difference and volume values, along with the volume round indicator, into discrete tokens using a colon delimiter. Together with special tokens for padding, masking, and unknown messages, this process generates 293 distinct message tokens from the training data. A special stringification rule is applied to execution messages (type 4) by converting their price level to 0 since they always occur at the best available price. We now have a vocabulary representing the most important messages, and continuous values supporting them.

2.2 Order Book snapshot preprocessing

Along with the messages, LOBERT processes a **snapshot** of the Order Book immediately after the message, i.e., the immediate effect of each message is included in its corresponding snapshot. A single snapshot is an array of 40 values corresponding to the 10 best bid prices and their volumes, and 10 best ask prices and their volumes. Volumes are scaled using an exponential transformation:

$$v_{\text{scaled}} = 1 - e^{-v_{\text{raw}}/k} \tag{1}$$

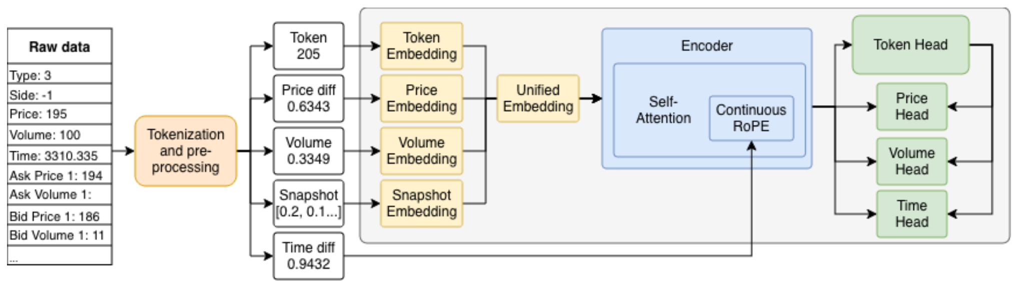
where v_{raw} is the raw volume at a given price level and $k = 2000$ is a scaling parameter that controls the transformation rate. This maps unbounded volume values to $[0, 1)$ with diminishing sensitivity to extremely large volumes. Prices are normalized as tick distances from the opposing best quote, clipped at a maximum distance of $D_{\max} = 20$ ticks and normalized to $[0, 1]$ through division by the maximum distance. This representation captures the relative depth and spread structure of the order book in a compact, normalized form. See Appendix 2.2 for further details.

2.3 Model architecture, inputs and outputs

LOBERT's architecture (see Figure 1) builds upon the principles of the original BERT model, implemented using PyTorch. A key innovation is its *multi-modal embedding layer*, which handles the hybrid discrete–continuous nature of LOB data by merging representations from four sources: (1) discrete message tokens, (2) continuous price differences, (3) continuous volume values, and (4) optionally, LOB snapshot representations integrated through a learned gating mechanism. These embeddings are combined additively to produce a unified representation for each position. In addition to learned positional embeddings, we incorporate continuous Rotary Position Embedding (ROPE) within the self-attention mechanism that operates on cumulative time differences between consecutive messages. This hybrid approach enables the model to capture both sequential ordering and the actual temporal distances between events, addressing the non-uniform nature of market dynamics.

The transformer encoder consists of stacked layers, each containing multi-head self-attention followed by a position-wise feed-forward network with GELU activation and intermediate dimension expansion. Layer normalization and residual connections are applied after each sub-layer, with dropout ($p = 0.1$) for regularization. The model outputs pass through multiple specialized heads: a linear classification head for discrete token prediction (cross-entropy loss), and three regression heads for continuous values (weighted MSE losses). Uniquely, these regression heads employ a *dual-input design*, concatenating both token prediction logits and final hidden states, enabling them to leverage discrete predictions when generating continuous outputs. This architecture facilitates hybrid inference strategies that combine token-based categorical information with refined continuous regression.

Figure 1: Simplified architecture diagram highlighting the unique properties of LOBERT



2.4 Training and fine-tuning

The model's output structure is also multi-faceted. A standard token prediction head is responsible for predicting the next discrete message token, optimized using a cross-entropy loss function. In parallel, three regression heads are appended after the main transformer layers to predict the continuous values for the given message's price, volume, and time difference. These heads are trained using a Mean Squared Error (MSE) loss, allowing the LOBERT to simultaneously learn the "language" of the most common LOB messages while retaining full granularity.

Our training methodology employs a two-phase strategy designed to first build a robust understanding of the "language" of the LOB message stream and then to refine this understanding for predictive tasks. The initial phase consists of Masked Message Modeling (MMM), analogous to BERT's Masked Language Modeling. In this stage, we train a bidirectional version of LOBERT to reconstruct randomly masked messages within a sequence. To ensure a balanced learning process, the regression losses for price, volume, and time are weighted to be approximately equal after the first epoch of training, while the token prediction loss is given a higher weight to emphasize the importance of identifying the main characteristics of messages. To avoid leaking the masked messages to the model via LOB snapshots around the masked position, the snapshots are also masked for 90% of positions randomly. Following the MMM phase, we transition to a fine-tuning phase for tasks such as next-message prediction and mid-price prediction. In these phases, all snapshots are visible to the model.

3 Experiments and results

In order to demonstrate LOBERT's adaptability, we have conducted fine-tuning experiments for next-message prediction and direct mid-price prediction. Note that due to issues in data access at the time of writing, we could only utilize millisecond-level time differences rather than nanoseconds and thus the time aspect is only partially covered in the experiments. We use 80 trading days of Nasdaq ITCH feed data on AAPL, INTC, MSFT and FB stocks between 2015.05.11 and 2015.09.01 for training, and 10 days between 2015.09.02 and 2015.09.16 for validation. The final 10 days of data from 2015.09.17 to 2015.09.30 are reserved for testing and producing the results discussed here. In total, the dataset has 470M messages divided into 919k non-overlapping sequences of 512 messages each.

For training LOBERT, we use an AdamW optimizer configured with a learning rate of 5×10^{-5} and weight decay of 0.01. Weight decay is disabled for bias terms and layer normalization parameters following standard practice. The learning rate schedule employs Cosine Annealing with Warm Restarts, where the initial restart period T_0 is set to 40,000 steps with a multiplicative factor $T_{\text{mult}} = 2$ for subsequent cycles, and a minimum learning rate of 5×10^{-6}. Training is conducted for 10 epochs with a batch size of 32, and model evaluation is performed every $15{,}000$ training steps on a held-out validation set. For next-message and mid-price prediction, the model weights are initialized from the pre-trained MMM checkpoint, and the best-performing model is selected based on the lowest validation loss.

3.1 Next-message prediction experiment

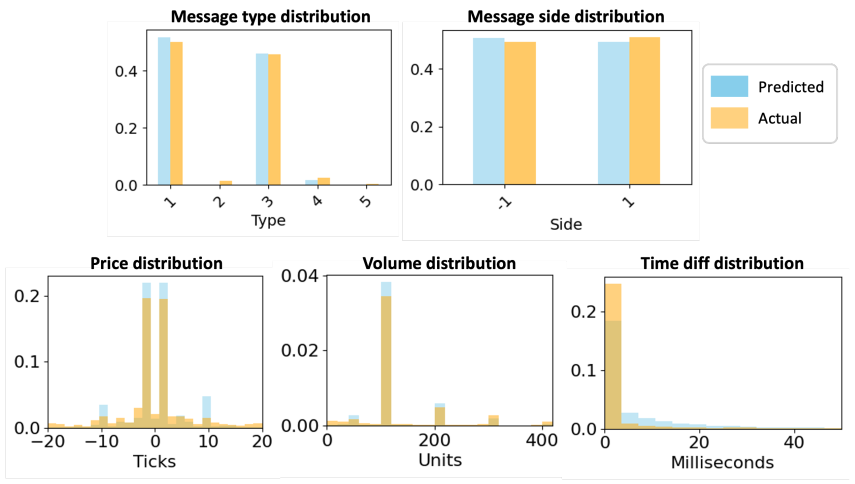
The pre-trained MMM model is adopted for causal modeling by using a triangular mask which allows the model to see only past messages, and the label is set to the first message in the hidden section for each sequence. To compare LOBERT with a previous leading model in next-message prediction, we have reconstructed the S5 model as per [7] with 1.2M parameters and without Book Module which processes the LOB snapshots, and trained LOBERT with 1.1M parameters, both with and without Book Module. The models were trained on the same data with roughly an equal amount of training time, and their outputs were transformed with the same processing steps discussed in section 2.1. The results for individual predicted messages are compared between the two models by measuring the accuracy of their components (see Table 1). The accuracy of LOBERT is considerably higher for each separated component and the full message when considered as a whole.

Table 1: Accuracy comparison of predicting the tokenized components of individual messages.

Model	**Message type**	**Side or direction**	**Price quantized**	**Volume quantized**	**Full message**
S5	50.8%	52.2%	18.6%	32.1%	6.1%
LOBERT	**60.6%**	**65.1%**	**51.8%**	**72.1%**	**26.4%**
LOBERT with Book Module	**61.9%**	**65.4%**	**53.8%**	**72.2%**	**27.8%**

To ensure that LOBERT does not overfit to small niches, we inspect the overall distribution of predicted values against the real distribution in the same test dataset (see Figure 2). LOBERT predictions matche the distribution with Pearson correlations of *0.55* for price, *0.37* for volume and *0.52* for time. One notable deviation stands out at price level 10, which we hypothesize to be caused by conflicts between the token head and price regression head, and we will investigate it further.

Figure 2: Value distributions of price, volume, time difference, type and side for LOBERT model with Book Module enabled.



3.2 Message processing comparison

We compare three message-processing modes that differ in how the discrete token head and the continuous regression heads are utilized at inference time: (i) **Combined**, which uses the predicted token's quantization range as lower and upper bounds for the regressor; (ii) **Token only**, which maps the predicted token to its quantized price (bin-start) and volume (bin-centre) while discarding the regressors; and (iii) **Regressor only**, which ignores the token prediction and uses the three regression outputs directly. Note that even though Time Difference is only predicted with regressor, we include it in the table for reference on performance. To quantify distributional fidelity, we evaluate marginal discrepancies between predictions and ground truth on the test split using the Wasserstein-1 distance (W1), Jensen–Shannon divergence (JSD), and Total Variation distance (TVD) for *Price*, *Volume*, and *Time Difference*. Lower is better for all metrics.

Across all three variables and all three metrics, the **Combined** mode consistently achieves the smallest discrepancy relative to both ablations (see Table 2). This pattern indicates a complementary effect: the token head captures high-probability structural motifs (e.g., side/type and coarse price–volume quantiles), while the regressors refine continuous residuals and rare values, reducing discretization artefacts present in the token-only pathway and stabilizing the regressor-only pathway. These gains substantiate the design choice of retaining both channels in LOBERT's output layer.

Table 2: Distributional discrepancy (lower is better) between predicted and true marginals for *Price*, *Volume*, and *Time Difference*. Reported metrics are Wasserstein-1 (W1), Jensen–Shannon divergence (JSD), and Total Variation distance (TVD). Time difference is only predicted with regressors but we include the results here for reference on performance.

	Price			Volume			Time Difference		
Method	**W1**	**JSD**	**TVD**	**W1**	**JSD**	**TVD**	**W1**	**JSD**	**TVD**
Combined	***10.04***	***0.2111***	***0.1839***	***76.86***	***0.1696***	***0.1200***	*N/A*	*N/A*	*N/A*
Token only	*15.44*	*0.5566*	*0.6134*	*85.36*	*0.2446*	*0.1585*	*N/A*	*N/A*	*N/A*
Regressor only	*10.85*	*0.3959*	*0.4767*	*86.09*	*0.6623*	*0.8395*	*10.658*	*0.2681*	*0.3270*

3.3 Mid-price prediction experiment

We adapt LOBERT for mid-price direction by adding a classification head on top of the MMM-pretrained encoder. The task predicts the direction of the average tick-normalized mid-price movement over a message-based horizon. Let $m(t)$ be the mid-price at message t, and define the horizon set $H = \{10, 50, 100\}$ messages (message-based rather than time-based). We use the average future tick-normalized mid-price $\bar{m}_h(t) = \frac{1}{h}\sum_{i=1}^{h} m(t+i)$ and assign a ternary label

$$\ell_h(t) = \begin{cases} +1, & \bar{m}_h(t) - m(t) > \tau(h), \\ -1, & \bar{m}_h(t) - m(t) < -\tau(h), \\ 0, & \text{otherwise,} \end{cases}$$

where $\tau(h)$ is a horizon-dependent threshold (in the same units as m). We choose a logarithmically increasing threshold to reflect the sublinear growth of typical price moves with horizon length:

$$\tau(h) \;=\; \frac{1}{1000}\,\text{round}\Big(100\,\log_2 \tfrac{h}{10}\Big).$$

The averaging in $\bar{m}_h(t)$ reduces microstructure noise relative to end-of-horizon labeling.

We train a 3-way cross-entropy head where the inputs include snapshots and messages up to time t, and the target is $\ell_h(t)$. We report *macro-F1* and *coverage* under *selective prediction*: a prediction is emitted only if the *maximum softmax probability* exceeds a confidence threshold $\tau_c \in \{0.3, 0.4, \ldots, 0.9\}$. Results are averaged over four securities (AAPL, INTC, MSFT, FB). As a baseline, we re-train DEEPLOB [16] under the same data and horizons.

Table 3 shows that LOBERT consistently outperforms or matches DeepLOB across horizons and confidence thresholds. We also observe that LOBERT is more responsive to confidence filtering: as τ_c increases from 0.3 to 0.9, macro-F1 improves markedly (e.g., $H = 100$: $0.55 \to 0.88$) while coverage decreases ($1.00 \to 0.10$), indicating effective ranking of easy vs. hard cases. See Appendix F for asset-specific curves.

Table 3: Mid-price prediction F1 metrics (selective F1 at a confidence threshold) and coverage for prediction horizons of 10, 50, and 100. The F1 and coverage values are averaged over four securities: FB, INTC, MSFT, and AAPL. Detailed results are provided in the Appendix. Note that selective F1 should be read as the model's F1 score only on the cases where it was confident enough to make a prediction.

Confidence threshold	$H = 10$				$H = 50$				$H = 100$			
	LOBERT		DeepLOB		LOBERT		DeepLOB		LOBERT		DeepLOB	
	F1	Cover.	F1	Cover.	F1	Cover.	F1	Cover.	F1	Cover.	F1	Cover.
0.3	*0.51*	1.00	0.44	1.00	*0.56*	1.00	0.53	1.00	*0.55*	1.00	0.55	1.00
0.4	*0.52*	0.95	0.45	0.98	*0.56*	0.96	0.53	0.99	*0.56*	0.96	0.55	0.99
0.5	*0.55*	0.79	0.46	0.90	*0.60*	0.77	0.54	0.89	*0.60*	0.77	0.56	0.92
0.6	*0.60*	0.61	0.47	0.77	*0.67*	0.51	0.52	0.70	*0.68*	0.48	0.55	0.66
0.7	*0.68*	0.47	0.49	0.67	*0.77*	0.33	0.53	0.60	*0.77*	0.29	0.55	0.53
0.8	*0.75*	0.37	0.49	0.58	*0.86*	0.23	0.56	0.50	*0.85*	0.18	0.58	0.41
0.9	*0.82*	0.29	0.48	0.50	*0.84*	0.14	0.60	0.40	*0.88*	0.10	0.61	0.28

LOBERT's inference is currently slower than DeepLOB's by 53%. However, Transformer-based architectures such as LOBERT have a large arsenal of potential mechanisms to increase throughput by several multiples, which we plan to implement in future versions. See Appendix D for details on inference speed.

4 Discussion and conclusion

Our work on LOBERT demonstrates the significant potential of adapting methodologies from the world of natural language processing to model the dynamics of the LOB. By treating the stream of LOB events as a unique language, we leverage the well-established pattern-recognition capabilities of the BERT architecture. The initial results from our next-message and mid-price prediction experiments are promising, suggesting that a model can indeed learn the intricate "grammar" of market data. Thus, LOBERT foundation model can be adapted to countless use cases with a small amount of fine-tuning, opening the door for a completely new way of creating models in LOB domain, similar to how the original BERT model changed language modeling.

Despite the advancements, we acknowledge the limitations in our current work. Namely, our experiments so far do not guarantee the model's ability to produce realistic long-term sequences. Our future work will validate this capability through extensive sequence-level analysis. Furthermore, we have not yet completed the study on the breadth of task types which LOBERT can be adapted to.

LLMs are also known to struggle with tracking the state of entities through changes, and LOBERT may face a similar challenge in tracking the size and location of orders on the book when the price levels move. Our current implementation simplifies this by not modeling individual order IDs. Additionally, the computational cost and inference time of the transformer architecture are non-trivial.

In conclusion, LOBERT represents a promising new direction for modeling LOB dynamics. By adapting the BERT framework, we have developed a model capable of understanding the fundamental language of market messages, unifying discrete and continuous event data into a single, cohesive framework. We will now focus on addressing the current limitations and expanding the model's capabilities. Before the conference, our roadmap includes conducting comprehensive sequence prediction experiments to validate the model's generative quality and expanding the fine-tuning coverage study. The performance of the model will be improved with more efficient attention mechanisms and enhanced training techniques introduced in later adaptations of BERT. To further enrich the model's understanding, we will incorporate crucial market state information, such as the time of day, the current bid-ask spread, and full LOB snapshots.

References

[1] Leonardo Berti and Gjergji Kasneci. Tlob: A novel transformer model with dual attention for price trend prediction with limit order book data. *arXiv preprint arXiv:2502.15757*, 2025.

[2] Leonardo Berti, Bardh Prenkaj, and Paola Velardi. Trades: Generating realistic market simulations with diffusion models. *arXiv preprint arXiv:2502.07071*, 2025.

[3] Rama Cont, Mihai Cucuringu, Jonathan Kochems, and Felix Prenzel. Limit order book simulation with generative adversarial networks. *Available at SSRN 4512356*, 2023.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[5] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. In *International conference on machine learning*, pages 5506–5518. PMLR, 2021.

[6] Junjie Li, Yang Liu, Weiqing Liu, Shikai Fang, Lewen Wang, Chang Xu, and Jiang Bian. Mars: a financial market simulation engine powered by generative foundation model. In *The Thirteenth International Conference on Learning Representations*, 2025.

[7] Peer Nagy, Sascha Frey, Silvia Sapora, Kang Li, Anisoara Calinescu, Stefan Zohren, and Jakob Foerster. Generative ai for end-to-end limit order book modelling: A token-level autoregressive generative model of message flow using a deep state space network. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 91–99, 2023.

[8] Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. In *Machine learning and AI in finance*, pages 5–15. Routledge, 2021.

[9] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

[10] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[11] Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE transactions on neural networks and learning systems*, 30(5):1407–1418, 2018.

[12] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *2017 IEEE 19th conference on business informatics (CBI)*, volume 1, pages 7–12. IEEE, 2017.

[13] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

[14] Aaron Wheeler and Jeffrey D Varner. Marketgpt: Developing a pre-trained transformer (gpt) for modeling financial time series. *arXiv preprint arXiv:2411.16585*, 2024.

[15] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models, 2022.

[16] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019.

[17] Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. Bert loses patience: fast and robust inference with early exit. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc., 2020.

A Related work

Previous attempts to model the LOB have spanned a range of machine learning techniques. Early efforts focused on supervised learning, framing the problem as a classification task to predict mid-price movements [8, 11, 12, 16]. More recently, GANs have been applied for LOB snapshot prediction [3], and generative models such as [7] based on sequence modeling as with [9] by splitting each message into 22 tokens and predicting them one by one. [14] presents a generative pre-trained transformer for financial data simulation but it is also slowed down by splitting messages into 24 tokens. [2] introduces a transformer-based diffusion engine for LOB simulations, offering flexible generation but with the cost of heavy computational requirement by sampling requires hundreds of iterative steps. [1] proposes a dual-attention transformer for price trend prediction, improving accuracy, yet focusing narrowly on trend forecasting rather than full LOB message generation. In a concurrent line of work, [6] introduces MarS, a financial market simulation engine powered by a generative foundation model (LMM) that also operates at the order level to create interactive and controllable market simulations.

B Piecewise Linear-Geometric Scaling

To handle the wide dynamic range of LOB features while preserving both precision at small values and representation of extreme values, we employ Piecewise Linear-Geometric Scaling. For a raw input value x (measured in original units: ticks for price, units for volume, or milliseconds for time), with parameters τ_{start} (transition point in input units), τ_{max} (maximum scaled output value), and τ_{clip} (input clipping threshold), the transformation proceeds in two stages.

First, we compute an intermediate scaled value $s(x)$:

$$s(x) = \begin{cases} x & \text{if } x \leq \tau_{\text{start}} \\ \tau_{\text{start}} + \sum_{k=0}^{\lfloor x-\tau_{\text{start}} \rfloor} \mu^k & \text{if } \tau_{\text{start}} < x \leq \tau_{\text{clip}} \\ \tau_{\text{start}} + \sum_{k=0}^{\lfloor \tau_{\text{clip}}-\tau_{\text{start}} \rfloor} \mu^k & \text{if } x > \tau_{\text{clip}} \end{cases} \tag{2}$$

where the geometric decay factor μ is determined by requiring that the infinite series converges to τ_{max}:

$$\mu = 1 - \frac{1}{\tau_{\text{max}} - \tau_{\text{start}}} \tag{3}$$

This ensures that $\lim_{x\to\infty} s(x) = \tau_{\text{start}} + \frac{1}{1-\mu} = \tau_{\text{max}}$.

Second, we normalize the intermediate scaled value to the range $[0, 1]$:

$$x_{\text{scaled}} = \frac{s(x)}{\tau_{\text{max}}} \tag{4}$$

This two-stage process provides several advantages: (i) linear growth for $x \leq \tau_{\text{start}}$ preserves sensitivity to small, frequent values; (ii) geometric decay for $x > \tau_{\text{start}}$ compresses extreme outliers; (iii) the ratio $\tau_{\text{start}}/\tau_{\text{max}}$ directly controls the proportion of the output range [0,1] dedicated to linear scaling; and (iv) asymptotic convergence prevents unbounded values from dominating the learned representations. For continuous input values between discrete steps, linear interpolation is applied within each unit interval.

C Snapshot scaling

The volume at each price level is transformed using an exponential scaling function to normalize the distribution:

$$v_{\text{scaled}} = 1 - e^{-v_{\text{raw}}/k} \tag{5}$$

where v_{raw} is the raw volume at a given price level and k is a scaling parameter that controls the transformation rate. This transformation maps unbounded volume values to the range $[0, 1)$, with the exponential function providing diminishing sensitivity to extremely large volumes.

Prices are normalized by representing price levels as distances from the best opposite quote. The distance from each price level to the best opposite quote is normalized to capture relative position in the order book. For bid prices:

$$d_{\text{bid},i} = \frac{\min\left(\left\lfloor \frac{p_{\text{ask}}^{(1)} - p_{\text{bid}}^{(i)}}{\Delta p} \right\rceil - 1, D_{\max}\right)}{D_{\max}} \tag{6}$$

where $p_{\text{ask}}^{(1)}$ is the best ask price, $p_{\text{bid}}^{(i)}$ is the i-th bid price level, Δp is the minimum tick size, and $D_{\max}$ is the maximum distance threshold (clip_dist). The term $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.

Similarly, for ask prices:

$$d_{\text{ask},i} = \frac{\min\left(\left\lfloor \frac{p_{\text{ask}}^{(i)} - p_{\text{bid}}^{(1)}}{\Delta p} \right\rceil - 1, D_{\max}\right)}{D_{\max}} \tag{7}$$

where $p_{\text{bid}}^{(1)}$ is the best bid price and $p_{\text{ask}}^{(i)}$ is the i-th ask price level. Both distance metrics are clipped to $[0, 1]$ through the minimum operation and subsequent normalization by $D_{\max}$.

D Inference speed

The present implementation of LOBERT prioritizes predictive quality over latency. On a single NVIDIA V100 GPU, the model attains a throughput of **281.87** predictions per second, corresponding to **47%** of DEEPLOB's throughput under the same evaluation conditions with batch size of 1. While these figures reflect a quality-focused design choice, the Transformer backbone offers several well-understood avenues for accelerating inference without altering task formulation.

Quantization. Reducing arithmetic precision from FP32 to lower-bit formats (e.g., INT8) can substantially increase throughput by improving compute and memory efficiency. In prior work on Transformer encoders, quantization has yielded *up to* $4\times$ speed-ups with minimal accuracy impact on supported hardware [5]. Applying post-training or quantization-aware variants to LOBERT is therefore a promising first lever for latency reduction.

Structured pruning. Structured removal of computation—such as pruning full layers, attention heads, or blocks within the feed-forward sublayers—reduces end-to-end FLOPs and wall-clock time while preserving dense tensor layouts that deploy efficiently. Reported gains reach *up to* $2\times$ at modest accuracy cost when pruning is paired with brief recovery fine-tuning [15]. Given LOBERT's compact depth, conservative layer- or width-pruning regimes are natural candidates.

Early-exit strategies. Adaptive inference equips intermediate layers with auxiliary classifiers and halts computation once confidence is sufficient. On average, such mechanisms have achieved $\sim 1.57\times$ speed-ups, with the actual gain governed by the distribution of "easy" inputs and the exit threshold [17]. Integrating early exits into LOBERT preserves the model's capacity for difficult cases while trimming work on routine ones.

Efficient attention. For longer input windows, the quadratic cost of standard self-attention becomes a bottleneck. Efficient attention formulations that reduce the effective complexity can deliver *up to* $1.5\times$ speed-ups at moderate sequence lengths while maintaining predictive quality [13]. These methods are especially relevant if future deployments require extending the historical context processed per inference step.

E Continuous Rotary Position Embedding

We extend the standard Rotary Position Embedding (RoPE) mechanism introduced in [10] to handle continuous time by replacing discrete position indices with cumulative time differences. For each attention head with dimension d_h, we define frequency components $\theta_i = 10000^{-2i/d_h}$ for $i \in \{0, 1, \ldots, d_h/2 - 1\}$. Given cumulative time $t \in \mathbb{R}^+$ at each sequence position, we compute rotation angles $\phi_i(t) = t \cdot \theta_i$. The continuous RoPE transformation is then applied to query and key vectors $\mathbf{q}, \mathbf{k} \in \mathbb{R}^{d_h}$ by rotating consecutive dimension pairs: for each pair (q_{2i}, q_{2i+1}), we compute $(q'_{2i}, q'_{2i+1}) = (q_{2i} \cos\phi_i(t) - q_{2i+1} \sin\phi_i(t), q_{2i} \sin\phi_i(t) + q_{2i+1} \cos\phi_i(t))$, and analogously for $\mathbf{k}$. This formulation preserves the relative position encoding property of RoPE while naturally accommodating irregular time intervals between events in the Limit Order Book message stream.

F Asset specific F1 and coverage scores with confidence thresholding

Figures 3 and 4 describe how the F1 score of predictions increases as we increase the minimum threshold required for action. LOBERT reacts more aggressively to confidence thresholds, as seen in the steep upwards curvature, while DeepLOB's confidence is more uniform. At the same time, the coverage of LOBERT decreases.

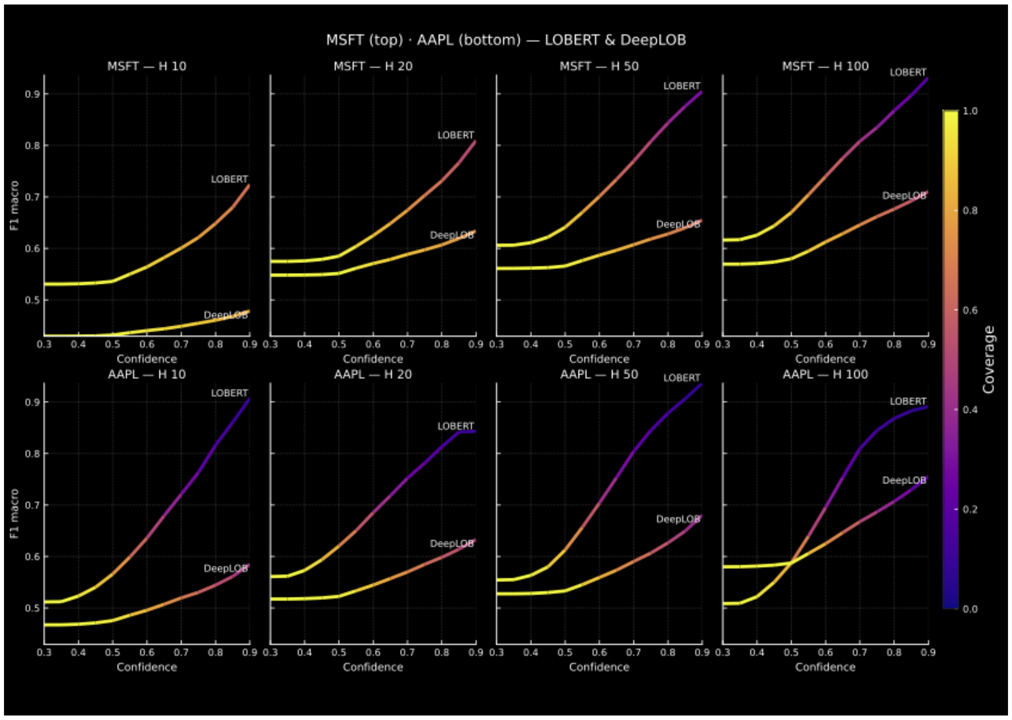


Figure 3: Comparison of LOBERT and DeepLOB for mid-price prediction plotted by confidence threshold, F1 macro score and coverage for MSFT and AAPL assets.

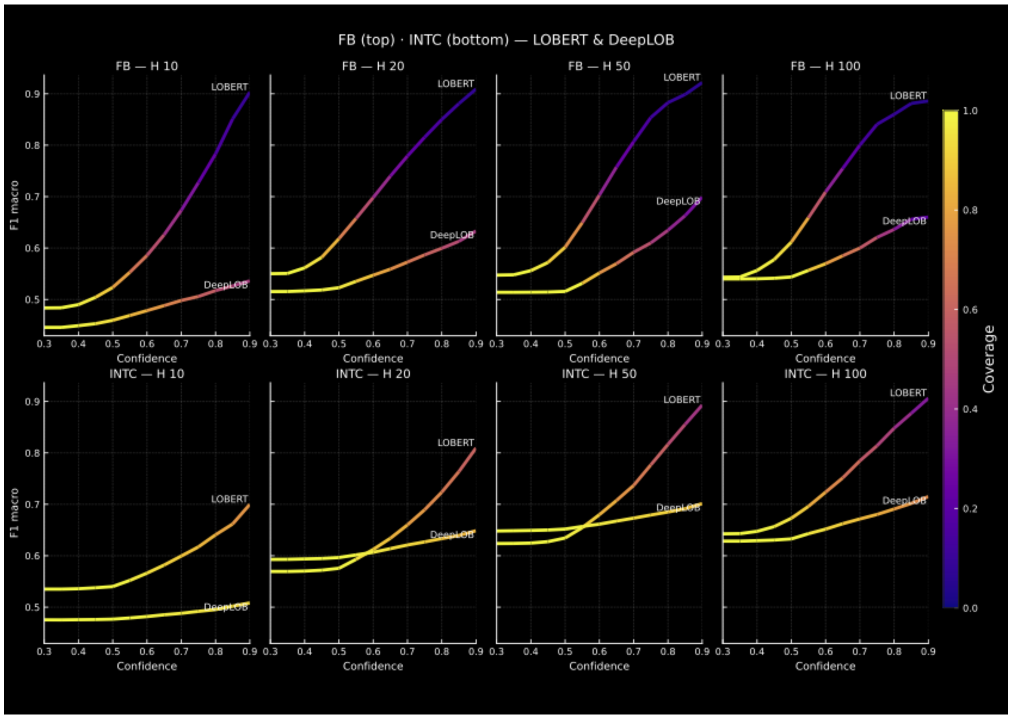


Figure 4: Comparison of LOBERT and DeepLOB for mid-price prediction plotted by confidence threshold, F1 macro score and coverage for FB and INTC assets.