# LOBRE: TRANSFERABLE LIMIT ORDER BOOK REPRESENTATIONS FOR LOW-DATA REGIMES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Limit order book forecasting models are typically trained for individual prediction tasks, making it difficult to rapidly develop models for applications with limited task-specific labels, such as forecasting within infrequent Limit Order Book (LOB) market regimes. We introduce LOBRE, a transformer encoder pretrained directly on irregular Level-3 message streams to learn reusable representations for downstream forecasting with scarce labels. LOBRE combines Masked Message Modeling (MMM) with Aggregate Activity Forecasting (AAF), a prospective objective that trains a pooled sequence representation to capture information predictive of subsequent market activity. We evaluate transfer across heterogeneous forecasting tasks, multiple target-data budgets, and assets both included in and excluded from pretraining. With the encoder frozen, pretraining improves Macro-F1 over the same architecture trained from scratch in 90/96 configurations on pretraining assets and 88/96 configurations on held-out assets, with mean gains of 0.092 and 0.097, respectively. Frozen adaptation also outperforms full fine-tuning on average and exceeds DeepLOB and TABL benchmarks across all evaluated task–budget comparisons. These results support a pretrain-once, adapt-many approach to limit order book forecasting, in which a shared message-level representation can be reused for new tasks and assets with limited target-specific supervision.

## 1 INTRODUCTION

Limit Order Books (LOB) form the core trading mechanism of modern electronic financial markets. At their most granular level, market activity is an asynchronous stream of messages describing order submission, cancellation, modification, and execution. These events jointly determine prices and liquidity, but are challenging to model because they arrive irregularly, interact through the evolving book state, and arise in non-stationary markets. Models specialized to one security, regime, or forecasting target may therefore require substantial adaptation as conditions change. In practice, the labeled data available for such adaptation can be scarce, for example when a forecasting task concerns an infrequent market regime or an asset not represented in the original training data.

Most machine-learning approaches to LOB forecasting are designed for a specific supervised task. They commonly reconstruct fixed-depth book snapshots and train directly to predict future mid-price movements or related quantities (Tran et al., 2019; Zhang et al., 2019; Passalis et al., 2021). A complementary line of work models the underlying message stream autoregressively for order-flow generation and market simulation (Nagy et al., 2023; Li et al., 2025b). Despite progress in both directions, models are typically optimized for a particular representation or objective, and extending them to new tasks or assets can require learning similar market-microstructure structure repeatedly from task-specific data.

Message-level pretraining offers another possibility. Messages are the most granular observable components from which the LOB state emerges, suggesting that common structure could be learned once from large unlabeled corpora and reused across forecasting problems, analogous to pretrained language models such as BERT (Devlin et al., 2019). Recent work has established reusable representations of reconstructed LOB snapshots and generative foundation models of order flow (Li et al., 2026; Zhong et al., 2025; Li et al., 2025b; Zhang et al., 2026), but it remains less understood whether self-supervised pretraining directly on irregular Level 3 message streams can produce an



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

encoder representation that supports transfer in low-data regimes across heterogeneous forecasting objectives and previously unseen assets.

We investigate this question by introducing LOBRE (*Limit Order Book Regime Encoder*), a transformer encoder designed for adaptation to LOB forecasting tasks in low-data regimes, which combines a BERT-inspired Masked Message Modeling (MMM) technique with a novel Aggregate Activity Forecasting (AAF) objective in pretraining. In the model, each complete market message occupies one sequence position while retaining categorical and continuous information including event type, side, relative price, volume, and inter-arrival time. After pretraining, lightweight task-specific heads operate on the shared encoder representation.

Our central question is *target-data efficiency*: given the same limited amount of labeled target-asset data, how much does pretrained initialization improve downstream forecasting compared with learning the same architecture from scratch? We operationalize label scarcity through target-asset training budgets of one to fifteen trading days; these budgets measure data efficiency rather than reproducing the distribution of any specific market regime. We evaluate this question across multiple forecasting tasks and data budgets, including assets entirely excluded from pretraining, and examine whether the pretrained representation remains effective when the shared base is frozen rather than fully fine-tuned.

Our main contributions are:

- We introduce LOBRE, a pretrained message-level transformer encoder for irregular, mixed discrete–continuous LOB event streams, combining contextual MMM with the prospective AAF objective.
- We show that self-supervised pretraining substantially improves low-data downstream forecasting and that the benefit transfers to unseen assets outside the pretraining corpus.
- We demonstrate that the pretrained representation can be reused through lightweight task-specific heads with the base frozen, and evaluate this transfer across four forecasting tasks, multiple target-data budgets, held-out assets, and full versus head-only adaptation.

## 2 Related Work

**Task-specific LOB forecasting.** Machine learning for limit order books has predominantly focused on supervised prediction from reconstructed book states. Architectures such as TABL and DeepLOB learn temporal and cross-sectional structure for targets such as future mid-price movement (Tran et al., 2019; Zhang et al., 2019), while large-scale evidence for shared structure across securities was provided by Sirignano & Cont (2019). More recent work develops increasingly specialized architectures for LOB structure and non-stationarity: HLOB models dependencies across book levels and studies how predictability varies across securities and horizons (Briola et al., 2025), while transformer-based approaches such as LiT and TLOB explicitly model spatial–temporal dependencies and evaluate adaptation or generalization across changing market conditions (Xiao et al., 2025; Berti & Kasneci, 2025). These approaches demonstrate the value of learning common market structure, but primarily optimize the representation jointly with a particular supervised forecasting objective.

**LOB representation learning.** Closer to our objective, recent work explicitly separates LOB representation learning from downstream prediction. SimLOB uses a Transformer autoencoder to learn compact representations of reconstructed LOB sequences for financial-market simulation and calibration (Li et al., 2026). Building on this direction, Zhong et al. (2025) introduce LOBench, a benchmark for transferable LOB representations that evaluates reconstruction, price-trend prediction, imputation, and cross-dataset transfer, including adaptation with a frozen pretrained encoder and lightweight downstream models. These results establish that reusable representations of reconstructed LOB states can transfer beyond the objective used to learn them. LOBRE instead learns directly from the underlying irregular Level 3 message stream, represents each complete market event as a sequence element, and combines masked message reconstruction with prospective sequence-level pretraining. We evaluate the resulting frozen encoder under limited target supervision across heterogeneous future forecasting tasks, previously unseen securities, and chronologically later market data.

**Message-level generative modeling.** A complementary line of work models the underlying order flow rather than only reconstructed book states. Nagy et al. (2023) formulate LOB modeling as token-level autoregressive generation of order messages coupled with a market simulator, while MarS introduces an order-level generative foundation model for interactive and controllable financial-market simulation (Li et al., 2025b). More recently, M3 jointly represents order events and LOB state and generates future order-flow trajectories for simulation, forecasting, stress testing, and market-impact analysis (Zhang et al., 2026). LOB-Bench evaluates such generative message-level models through distributional fidelity, LOB statistics, and market-impact responses (Nagy et al., 2025). These works demonstrate the value of order-level foundation models as generative market models. In contrast, LOBRE is an encoder representation model: it does not generate market trajectories, but exposes a shared bidirectional representation that can be reused directly by lightweight downstream forecasting heads.

**Transferable time-series pretraining.** More broadly, pretrained time-series models demonstrate that representation learning can be amortized across datasets and tasks. MOMENT studies pretrained representations under limited supervision across multiple time-series tasks (Goswami et al., 2024), while TimesFM and Moirai demonstrate transfer to previously unseen forecasting problems (Das et al., 2024; Woo et al., 2024). In finance specifically, Kronos applies large-scale autoregressive pretraining to tokenized multi-market K-line data and evaluates zero-shot transfer across forecasting, volatility prediction, and generation (Shi et al., 2025). The choice of pretraining objective is also important for future-oriented transfer: TimeSiam, for example, explicitly learns temporal relations between separated subseries rather than relying only on within-window reconstruction (Dong et al., 2024). LOBRE extends this pretraining perspective to irregular, mixed discrete–continuous Level 3 event streams, combining contextual masked-message modeling with AAF supervision that explicitly trains the pooled representation to summarize information predictive of subsequent market activity.

# 3 LOBRE

We introduce LOBRE, a bidirectional transformer encoder designed to learn a reusable representation directly from message-level limit order book data. Its main design principle is to retain the granularity of the original event stream: each complete market message occupies a single sequence position. Pretraining produces a shared base model that can subsequently be reused across forecasting tasks by attaching lightweight task-specific heads.

## 3.1 MESSAGE REPRESENTATION

Pre-processing starts from the standard LOBSTER representation (Huang & Polak, 2011), in which market activity is expressed as a sequence of self-contained messages

$$\mathbf{m}_i = (a_i, d_i, p_i, v_i, t_i)\,, \tag{1}$$

where $a_i$ denotes the activity or event type, $d_i$ the order direction, $p_i$ the absolute price, $v_i$ the volume, and $t_i$ the exchange timestamp.

LOBRE transforms each raw message into a hybrid discrete–continuous representation. The absolute price $p_i$ is first expressed as a relative price difference $\delta p_i$ measured in ticks from the same–side best price. Event type, direction, and quantized price and volume information are then combined into a categorical message token $c_i$. The training corpus contains 665 message combinations which have an occurrence rate above $0.01\%$. Combinations falling below this limit are replaced by a generic UNKNOWN token, and an additional MASK token is reserved for the pretraining procedure described in Section 3.3. This gives the final vocabulary

$$\mathcal{V} = 1, \ldots, 667, \qquad c_i \in \mathcal{V}. \tag{2}$$

Quantization provides a compact categorical description of the message, but would discard potentially relevant numerical precision if used alone. We therefore retain continuous representations of relative price and volume in parallel. The relative price $\delta p_i$ and volume $v_i$ are transformed into scaled values $\tilde{p}_i, \tilde{v}_i \in [0, 1]$ using methods described in Appendix A.1. Absolute timestamps are

replaced by the inter-arrival time $\Delta t_i = t_i - t_{i-1}$, which is similarly mapped to a scaled value $\widetilde{\Delta t}_i \in [0, 1]$.

The resulting representation of message $i$ is therefore

$$\mathbf{x}_i = \left(c_i, \tilde{p}_i, \tilde{v}_i, \widetilde{\Delta t}_i\right) \in \mathcal{V} \times [0, 1]^3. \tag{3}$$

Each original market message corresponds to exactly one sequence position while retaining both a compact categorical identity and its continuous price, volume, and timing information. The final input sequence to LOBRE with window size $L$ is thus

$$\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_L). \tag{4}$$

### 3.2 Encoder and sequence pooling

The input representations are processed by a standard stack of transformer encoder layers with bidirectional self-attention. To account for irregular message arrival, we extend rotary positional embeddings (RoPE) by Su et al. (2024) to continuous time as in Li et al. (2025a), but using scaled cumulative time. For message position $i$, the RoPE rotation is determined by the cumulative elapsed time after clipping and scaling

$$T_i = \sum_{j=1}^{i} \widetilde{\Delta t}_j, \tag{5}$$

rather than the integer sequence index; the clipped and scaled $\widetilde{\Delta t}_j$ is defined in Appendix A.1.

In addition to contextual representations for individual messages, LOBRE learns a single representation of the complete input window. A learned attention pooler assigns weights to the contextualized message states and returns their weighted sum,

$$\mathbf{z} = \sum_{i=1}^{L} \alpha_i \mathbf{h}_i, \qquad \sum_{i=1}^{L} \alpha_i = 1, \tag{6}$$

where $\mathbf{h}_i$ denotes the contextual representation of message $i$. The pooler is part of the pretrained base model rather than a downstream prediction head, so the same pooled representation can be transferred together with the encoder.

### 3.3 Self-supervised pretraining

Pretraining combines two complementary self-supervised objectives. First, *Masked Message Modeling* (MMM), analogous to masked language modeling in BERT (Devlin et al., 2019), randomly hides or scrambles complete messages. This objective encourages the per-message hidden states to capture the local and contextual structure of the observed order flow. MMM alone, however, does not explicitly require the pooled representation $\mathbf{z}$ to summarize information useful for predicting events beyond the input window. We therefore optimize for a second objective, Aggregate Activity Forecasting (AAF), that forecasts aggregate market activity after the end of the sequence at multiple message offsets.

For an input sequence of length $L$, MMM produces predictions at every message position $i \in \{1, \ldots, L\}$. These comprise a categorical distribution over the 667 message tokens, $\hat{\mathbf{p}}_i \in \mathbb{R}^{667}$, and three continuous predictions

$$(\hat{p}_i, \hat{v}_i, \hat{t}_i) \in [0, 1]^3,$$

corresponding to the scaled price difference, volume, and inter-message time, respectively. Although predictions are produced for the full sequence, the MMM loss is evaluated only at the set $\mathcal{M}$ of masked positions, against their original uncorrupted values.

In parallel, the AAF objective operates on the pooled sequence representation and predicts market activity beyond the end of the input window. For $O$ predefined message offsets and $F$ forecast features, its output is

$$\hat{\mathbf{Y}}_{\text{forecast}} \in [-1, 1]^{O \times F}.$$

Each row corresponds to one future offset and contains scaled targets describing aggregate bid- and ask-side volume changes at different book-depth regions, movements of the best bid and ask prices, elapsed time since the end of the input sequence, and the number of executions.

Through the combination of the two prediction tasks, MMM supervises the per-position contextual representations, whereas the AAF directly trains the pooled representation $\mathbf{z}$ to summarize information relevant to future market dynamics which are later exposed to downstream heads. Forecast errors are optimized using a robust regression loss, and the overall pretraining objective is

$$\mathcal{L}_{\text{pre}} = \mathcal{L}_{\text{MMM}} + \mathcal{L}_{\text{AAF}}, \tag{7}$$

where $\mathcal{L}_{\text{AAF}}$ includes the weight $\lambda_{\text{f}}$, which controls the contribution of the prospective forecasting task. Exact masking, normalization, target construction, and loss settings are provided in Appendix A.2.

### 3.4 Downstream adaptation

After pretraining, the message embeddings, transformer encoder, and sequence pooler form the transferable LOBRE base. A downstream task is introduced by attaching a small prediction head to the pooled representation $\mathbf{z}$. For classification tasks, we use a compact residual feed-forward block followed by a task-specific output layer. See Appendix A.3 for the full architecture.

The base can either be kept frozen, so that only the lightweight head is trained, or fine-tuned jointly with the new head. The former setting is particularly important for the intended use of LOBRE: a single pretrained representation can support several forecasting tasks without maintaining and evaluating a separate encoder for each task. Section 4 evaluates both adaptation strategies under limited target-data budgets.

## 4 Experimental Design

Infrequent market regimes are a motivating use case for LOBRE: trading firms may have forecasting tasks for which only limited labeled data are available. Defining those regimes, however, requires specialized trading knowledge, and the rules used by firms are often proprietary. We therefore do not identify or evaluate specific regimes. Instead, we use target-asset training budgets measured in asset–days to test whether LOBRE can adapt with limited task-specific supervision. Because these budgets consist of consecutive trading days not selected for any particular market regime or condition, they test label scarcity but not whether the pretrained representation captures dynamics that are rare in the pretraining corpus; we return to this in Section 6.

The experiments address three questions. First, does self-supervised pretraining reduce the amount of labeled target-asset data needed for a new forecasting task? Second, does the benefit extend to securities excluded from pretraining and to chronologically later observations? Third, can a lightweight downstream head use the pretrained representation effectively while the shared base remains frozen? The principal comparisons hold the downstream data fixed and vary model initialization and adaptation strategy.

Appendix A.3 provides the model architecture and parameter counts; Appendix A.2 describes the optimizer and training procedure.

### 4.1 Data

We use Nasdaq Level 3 message data covering 12 assets over 94 trading days from 2025-10-01 to 2026-02-13, comprising approximately 2.0 billion messages. The assets are divided into six *main* assets used for self-supervised pretraining and six disjoint *held-out* assets used to evaluate cross-asset transfer:

$$\mathcal{A}_{\text{main}} = \{\text{AAPL}, \text{AMD}, \text{AMZN}, \text{ASML}, \text{GOOG}, \text{PLTR}\},$$
$$\mathcal{A}_{\text{held}} = \{\text{MSFT}, \text{META}, \text{INTC}, \text{NFLX}, \text{NVDA}, \text{TSLA}\}.$$

Trading days are partitioned chronologically into training ($\mathcal{T}_{\text{train}}$: 2025-10-01–2026-01-16), validation ($\mathcal{T}_{\text{val}}$: 2026-01-20–2026-01-30), and test ($\mathcal{T}_{\text{test}}$: 2026-02-02–2026-02-13) periods. LOBRE is

pretrained using only $\mathcal{A}_{\text{main}} \times \mathcal{T}_{\text{train}}$, with $\mathcal{A}_{\text{main}} \times \mathcal{T}_{\text{val}}$ used for pretraining validation. Thus, assets in $\mathcal{A}_{\text{held}}$ are entirely absent from pretraining, and $\mathcal{T}_{\text{test}}$ is reserved for final evaluation.

Downstream adaptation is performed separately for each target asset $a \in \mathcal{A}_{\text{main}} \cup \mathcal{A}_{\text{held}}$ under training budgets $B \in \{1, 3, 8, 15\}$ asset–days. For budget $B$, the training set consists of the $B$ most recent days of $\mathcal{T}_{\text{train}}$ for asset $a$, making the budgets nested. Validation uses the first $V(B) = \lceil 0.2B \rceil$ days of $\mathcal{T}_{\text{val}}$ for the same asset, while evaluation always uses all available observations for $a$ in $\mathcal{T}_{\text{test}}$. The resulting allocation is identical across downstream tasks and model variants.

### 4.2 Downstream tasks

We evaluate transfer across a collection of downstream forecasting tasks

$$\mathcal{Q} = \{\tau_1, \ldots, \tau_K\}.$$

Each task $\tau \in \mathcal{Q}$ listed in Table 1 defines a target $y^\tau$ from market activity occurring strictly after the input window. See further details in Appendix A.6 for the target definitions. All tasks use the same classification head structure described in Section 3.4, trained separately for each task with $y^\tau$. None of the downstream labels have been included in the pretraining.

Table 1: Downstream tasks used to evaluate transfer of the pretrained LOBRE representation.

| **Task** | **Target** | **Horizon** | **Classes** |
|---|---|---|---|
| $\tau_{\text{midprice}}$ | Next mid-price move direction | Until next move | 2 (down, up) |
| $\tau_{\text{cross}}$ | Spread-crossing direction | 500 steps | 3 (down, no crossing, up) |
| $\tau_{\text{ofi}}$ | Order-flow imbalance | 100 steps | 3 (bid-side, neutral, ask-side) |
| $\tau_{\text{freq}}$ | Message frequency change | 100 steps | 3 (slower, no change, faster) |

### 4.3 Low-data adaptation and transfer settings

We evaluate downstream adaptation at target-data budgets $B \in \{1, 3, 8, 15\}$ asset–days. Each experiment is defined by a downstream task $\tau \in \mathcal{Q}$, target asset $a \in \mathcal{A}_{\text{main}} \cup \mathcal{A}_{\text{held}}$, and budget $B$. A separate prediction head is trained for every $(\tau, a, B)$ configuration using only the corresponding asset-specific training and validation data defined in Section 4.1. Thus, prediction heads are not shared across tasks, assets, or data budgets.

For main assets $a \in \mathcal{A}_{\text{main}}$, the pretrained base has previously observed unlabeled messages from the same asset, so adaptation measures transfer to a new supervised objective under limited target-specific data. For held-out assets $a \in \mathcal{A}_{\text{held}}$, the asset is entirely absent from pretraining, additionally testing cross-asset transfer.

### 4.4 Model variants and baselines

We evaluate three primary LOBRE variants. *Pretrained–frozen* initializes the base from the MMM+AAF checkpoint and keeps it fixed, optimizing only the task-specific prediction head; this is our primary reusable-representation setting. *Pretrained–unfrozen* uses the same initialization but optimizes both the base and downstream head. *Random initialization* trains an architecturally identical LOBRE and the same downstream head from scratch on the corresponding target data, providing the controlled comparison for the effect of pretraining. Thus, the three variants share the architecture, inputs, labels, and target-data budget, differing only in initialization and whether the pretrained base is updated.

To isolate the contribution of AAF, we additionally evaluate an MMM-only pretrained variant as described in Section 4.5. For external comparison, we train TABL (Tran et al., 2019) and DeepLOB (Zhang et al., 2019) on the same downstream tasks and data configurations.

### 4.5 Pretraining objective ablation

To isolate the contribution of AAF, we additionally pretrain an otherwise identical LOBRE checkpoint using MMM alone. The two checkpoints use the same architecture, pretraining corpus, opti-

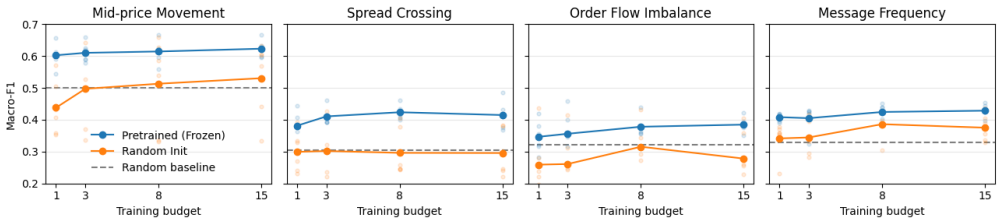



Figure 1: Low-data adaptation on the main assets. Each panel corresponds to one downstream task and shows Macro-F1 against the target-asset training budget for pretrained–frozen and randomly initialized LOBRE. Lines show the mean across assets and faint points show individual-asset results.

mization schedule, and downstream heads; the only difference is the inclusion of $\mathcal{L}_{\mathrm{AAF}}$. We evaluate both checkpoints primarily with the pretrained base frozen, since AAF is introduced specifically to train the pooled representation $\mathbf{z}$ consumed by downstream heads. Comparing MMM-only and MMM+AAF under this setting therefore tests whether prospective supervision during pretraining improves the transferability of the frozen representation. The comparison is performed across the same downstream tasks, security group, and target-data budgets used in the main experiments.

### 4.6 Evaluation protocol

Each task is evaluated using $M_\tau = \mathrm{MacroF1}$ score, which is our primary measure of data efficiency. Whenever relevant, we report performance separately for each target-data budget and security split rather than averaging across tasks with different output spaces or metrics. The principal comparisons are paired: competing model variants are trained and evaluated using the same target-data windows and test observations.

Because adjacent LOB messages are strongly dependent, the input sequences $\mathbf{X}$ are spaced with a stride equal to the input sequence length $L = 512$ messages to avoid overlap between any two observations. The same stride is used across training, validation, and test data.

## 5 Results

We report Macro-F1 separately for each downstream task and summarize effects across tasks using paired differences between model variants on matched task–budget–asset configurations. The $\mathcal{A}_{\mathrm{main}}$ versus $\mathcal{A}_{\mathrm{held}}$ comparison is necessarily unpaired across assets. Overall, pretraining consistently improves low-data adaptation, the benefit transfers to assets excluded from pretraining, and the pretrained representation remains effective when reused with the base frozen. Detailed numerical results are reported in Appendix A.7.

### 5.1 Pretraining improves low-data adaptation

Figure 1 compares pretrained–frozen LOBRE with the architecturally identical randomly initialized model across four downstream tasks, four target-data budgets, and six main assets. Pretraining improves Macro-F1 in 90 of 96 paired configurations (93.8%), with a mean gain of 0.092. The advantage is present across the full experimental grid, with task–budget gains ranging from 0.038 to 0.165. The benefit is already substantial with only one labeled asset–day: averaged across tasks, pretraining improves Macro-F1 by 0.100 at $B = 1$. It also persists as more target data become available, with an average gain of 0.093 at $B = 15$. Thus, the pretrained representation provides a consistent advantage over learning the same architecture entirely from limited downstream data.

### 5.2 Pretraining transfers to assets excluded from pretraining

We next test whether the benefit extends to entirely unseen assets. Across the six held-out assets, four downstream tasks, and four target-data budgets, pretrained–frozen LOBRE outperforms random initialization in 88 of 96 paired configurations (91.7%), with a mean Macro-F1 gain of 0.097 (Figure 2b). The mean gain is positive for every task–budget combination and is comparable to the

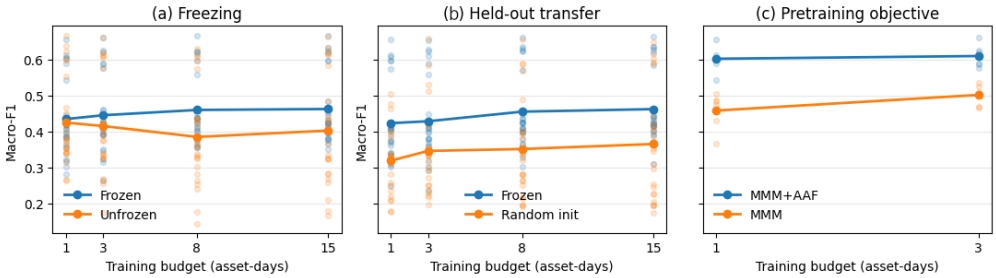



Figure 2: Transfer and adaptation experiments. (a) Pretrained–frozen versus pretrained–unfrozen LOBRE across all tasks and target-data budgets on the main assets. (b) Pretrained–frozen versus randomly initialized LOBRE on assets excluded from pretraining, across all tasks and budgets. (c) MMM-only versus MMM+AAF pretraining for mid-price prediction at $B \in \{1, 3\}$. Lines or markers show means across the corresponding assets, with individual-asset results shown as faint points.

0.092 gain on the main assets, indicating that the benefit of pretraining does not materially diminish when the target asset was absent from the pretraining corpus.

Absolute performance provides complementary evidence. Averaged across the 16 task–budget combinations, pretrained–frozen LOBRE scores only 0.008 Macro-F1 higher on the main assets than on the held-out assets, and the difference is approximately zero at $B = 15$. These results show that the learned representation transfers effectively to new assets using limited target-specific supervision.

### 5.3 The pretrained representation remains effective when frozen

Having established transfer across assets, we ask whether exploiting the pretrained representation requires updating the encoder. Across the four tasks, four budgets, and six main assets, pretrained–frozen LOBRE exceeds pretrained–unfrozen LOBRE in 72 of 96 paired configurations (75.0%), with a mean Macro-F1 difference of 0.044 and a median difference of 0.019 (Figure 2a).

The effect varies by task: the mean frozen advantage is 0.061 for $\tau_{\text{freq}}$, 0.057 for $\tau_{\text{cross}}$, and 0.046 for $\tau_{\text{ofi}}$, while the difference for $\tau_{\text{midprice}}$ is only 0.010 and individual comparisons are mixed. The larger mean than median reflects several configurations in which full fine-tuning degrades substantially. Overall, the predictive information learned during pretraining is readily accessible through the frozen pooled representation, allowing the shared encoder to be reused through lightweight task-specific heads without requiring end-to-end adaptation.

### 5.4 AAF improves frozen low-data transfer

We next isolate whether AAF contributes to the usefulness of the pooled representation. For mid-price prediction on the main assets at $B \in \{1, 3\}$, the MMM+AAF checkpoint consistently outperforms the otherwise identical MMM-only checkpoint when the pretrained base is frozen (Figure 2c). Macro-F1 increases from 0.458 to 0.603 at $B = 1$ and from 0.502 to 0.610 at $B = 3$, corresponding to gains of 0.144 and 0.108, respectively. Because the checkpoints otherwise share the architecture, pretraining corpus, and downstream adaptation procedure, this ablation supports the intended role of AAF: prospective sequence-level supervision makes the pooled representation more useful for downstream forecasting than masked-message reconstruction alone.

### 5.5 Comparison with task-specific LOB models

Finally, we place these controlled transfer results in the context of established task-specific LOB models. Table 2 compares pretrained–frozen LOBRE with DeepLOB (Zhang et al., 2019) and TABL (Tran et al., 2019) across all four downstream tasks at $B = 1$ and $B = 15$. LOBRE achieves the highest Macro-F1 in all eight task–budget comparisons. Over the stronger baseline in each cell, LOBRE improves Macro-F1 by 0.017–0.099 (mean 0.059). Because the models differ in architecture and input, this comparison does not isolate the effect of pretraining, which is established by the controlled comparisons above.

Table 2: Comparison with task-specific LOB models on the main assets. Values are Macro-F1 averaged across the six assets at the smallest and largest target-data budgets. Best results are shown in bold.

| Task | B | DeepLOB | TABL | LOBRE |
|---|---|---|---|---|
| $\tau_{\text{midprice}}$ | 1 | 0.446 | 0.503 | **0.603** |
| | 15 | 0.477 | 0.569 | **0.623** |
| $\tau_{\text{cross}}$ | 1 | 0.235 | 0.332 | **0.381** |
| | 15 | 0.244 | 0.348 | **0.415** |
| $\tau_{\text{ofi}}$ | 1 | 0.222 | 0.329 | **0.346** |
| | 15 | 0.274 | 0.341 | **0.385** |
| $\tau_{\text{freq}}$ | 1 | 0.238 | 0.328 | **0.408** |
| | 15 | 0.284 | 0.372 | **0.429** |

## 6 Discussion and limitations

The results support the central premise of LOBRE: self-supervised message-level pretraining can learn representations that transfer across forecasting objectives rather than being specialized to a single supervised target. The controlled comparison with randomly initialized LOBRE attributes this benefit to pretraining rather than the downstream architecture alone, while the similar gains on securities excluded from pretraining provide evidence of cross-security transfer. Together, these findings suggest that common structure in message-level market dynamics can be learned once and reused for new forecasting problems with limited target-specific supervision.

The AAF ablation further indicates that the pretraining objective matters for this transferability. Whereas MMM reconstructs information within the observed sequence, AAF directly trains the pooled representation against subsequent market activity. Its improvement over MMM alone is therefore consistent with the intended role of prospective supervision, although the present ablation is restricted to mid-price prediction at $B \in \{1, 3\}$. Keeping the pretrained base frozen also outperforms full fine-tuning on average under the evaluated procedure, supporting head-only adaptation as an effective way to reuse the shared encoder. This comparison concerns complete adaptation procedures, however, rather than freezing in isolation; different optimization or regularization choices may improve full fine-tuning.

Several limitations bound the generality of these conclusions. Pretraining and evaluation cover twelve equities from one exchange over a limited historical period, and the held-out experiment jointly introduces asset and temporal distribution shift. Moreover, all four downstream objectives are short-horizon classification tasks, while the AAF ablation and baseline evaluation cover only subsets of the full experimental grid. Transfer to the infrequent market regimes that motivate this work, and across exchanges, asset classes, objectives, and pretraining scales, therefore remains to be established.

## 7 Conclusion

We introduced LOBRE, a message-level transformer encoder pretrained with Masked Message Modeling and Aggregate Activity Forecasting for reusable limit order book representations. Pretraining improves low-data forecasting across four downstream tasks, and this benefit persists on securities excluded from pretraining. The frozen pretrained representation also performs competitively with, and on average better than, full fine-tuning under the evaluated procedure, while outperforming DeepLOB and TABL across the benchmark settings. Together with the AAF ablation, these results support a pretrain-once, adapt-many approach in which a shared message-level representation is reused for new LOB forecasting problems with limited target-specific data.

## AI USE STATEMENT

In this study, we have used generative AI tools to support background research and identify previous related work in the domain. We have also used it for proofreading and polishing our writing to improve the readability of this work. All text produced by AI is manually reviewed and re-written if necessary for correctness.

We have not used generative AI tools for developing conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, cleaning or reformatting datasets, supporting qualitative or thematic data analysis, or interpreting the results. AI-assistance in generating synthetic datasets, developing theoretical models, formulating mathematical claims, proving mathematical claims, writing proofs, assisting with translation are not applicable to this work.

We take responsibility for the full final content of this work.

## REPRODUCIBILITY STATEMENT

We provide detailed specifications of the preprocessing and representation construction, training procedures and hyperparameters, model architectures, dataset composition, downstream-task definitions, and evaluation setup in Appendix A.1–A.6. Supplementary material additionally contains a runnable smoke-test implementation illustrating the end-to-end pipeline, including model architecture, data preprocessing, pretraining with MMM and AAF, checkpoint selection, and frozen-base downstream adaptation. The smoke test uses fabricated data since we do not have the license to distribute the original data, and deliberately reduced models and training schedules. It is intended to make the computational procedure executable and inspectable, rather than to reproduce the numerical results reported in the paper. The reported experiments use the full configurations and data described in the paper and appendix.

## REFERENCES

Leonardo Berti and Gjergji Kasneci. Tlob: A novel transformer model with dual attention for price trend prediction with limit order book data, 2025. URL https://arxiv.org/abs/2502.15757.

Antonio Briola, Silvia Bartolucci, and Tomaso Aste. Hlob–information persistence and structure in limit order books. *Expert Systems with Applications*, 266:126078, 2025. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2024.126078. URL https://www.sciencedirect.com/science/article/pii/S0957417424029452.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/das24c.html.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186, 2019.

Jiaxiang Dong, Haixu Wu, Yuxuan Wang, Yun-Zhong Qiu, Li Zhang, Jianmin Wang, and Mingsheng Long. TimeSiam: A pre-training framework for siamese time-series modeling. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11412–11436. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/dong24e.html.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In Ruslan Salakhutdinov, Zico

Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16115–16152. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/goswami24a.html.

Ruihong Huang and Tomas Polak. Lobster: Limit order book reconstruction system. *SSRN Electronic Journal*, 12 2011. doi: 10.2139/ssrn.1977207.

Hao Li, Bowen Deng, Chang Xu, ZhiYuan Feng, Viktor Schlegel, Yu-Hao Huang, Yizheng Sun, Jingyuan Sun, Kailai Yang, Yiyao Yu, and Jiang Bian. Mira: Medical time series foundation model for real-world health data. In *Advances in Neural Information Processing Systems*, volume 38, pp. 98657–98685, 2025a.

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. *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=Yqk7EyT52H. OpenReview.

Yuanzhe Li, Yue Wu, Muyao Zhong, Shengcai Liu, and Peng Yang. Simlob: Learning representations of limit order book for financial market simulation. *IEEE Transactions on Artificial Intelligence*, 7(6):3429–3444, 2026. doi: 10.1109/TAI.2025.3640135.

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. *Proceedings of the Fourth ACM International Conference on AI in Finance*, pp. 91–99, 2023.

Peer Nagy, Sascha Yves Frey, Kang Li, Bidipta Sarkar, Svitlana Vyetrenko, Stefan Zohren, Ani Calinescu, and Jakob Nicolaus Foerster. LOB-bench: Benchmarking generative AI for finance - an application to limit order book data. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 45437–45460. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/nagy25a.html.

Nikolaos Passalis, Juho Kanniainen, Moncef Gabbouj, Alexandros Iosifidis, and Anastasios Tefas. Forecasting financial time series using robust deep adaptive input normalization. *Journal of Signal Processing Systems*, 93(10):1235–1251, 2021.

Yu Shi, Zongliang Fu, Shuo Chen, Bohan Zhao, Wei Xu, Changshui Zhang, and Jian Li. Kronos: A foundation model for the language of financial markets, 2025. URL https://arxiv.org/abs/2508.02739.

Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning. *Quantitative Finance*, 19(9):1449–1459, 2019. ISSN 1469-7688. doi: 10.1080/14697688.2019.1622295. Publisher Copyright: © 2019, © 2019 Informa UK Limited, trading as Taylor & Francis Group.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

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, 2019. doi: 10.1109/TNNLS.2018.2869225.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

Yue Xiao, Carmine Ventre, Yuhan Wang, Haochen Li, Yuxi Huan, and Buhong Liu. Lit: limit order book transformer. *Frontiers in Artificial Intelligence*, Volume 8 - 2025, 2025. ISSN 2624-8212. doi: 10.3389/frai.2025.1616485. URL https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2025.1616485.

Yanzhi Zhang, Yu Ma, Yilin Cheng, Jian Li, and Yitong Duan. M3: A state-event generative foundation model for market microstructure dynamics, 2026. URL https://arxiv.org/abs/2608.19227.

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. doi: 10.1109/TSP.2019.2907260.

Muyao Zhong, Yushi Lin, and Peng Yang. Representation learning of limit order book: A comprehensive study and benchmarking, 2025. URL https://arxiv.org/abs/2505.02139.

# A Appendix

## A.1 Pre-processing

Pre-processing is performed independently for each asset–day, so neither input windows nor inter-arrival times cross daily file boundaries. Each Nasdaq message is aligned with its post-message ten-level LOB snapshot. We retain regular market hours, identified by Nasdaq system-event markers, and event types $a_i \in \{1, \dots, 5\}$ corresponding to submission, partial cancellation, deletion, visible execution, and hidden execution. Events affecting levels deeper than ten are discarded. All differences below are computed between consecutive retained messages.

LOBERT represents each retained event using a categorical message token together with continuous price, volume, and inter-arrival-time features. We use the side convention $s_i = +1$ for bids and $s_i = -1$ for asks. Because hidden executions do not provide a reliable side, their side is inferred from the preceding midpoint. Let $q_{i-1}^{\text{same}}$ and $q_{i-1}^{\text{opp}}$ denote the previous best quote on the message side and opposing side, respectively. With tick size $\tau = 100$ integer price units, we define

$$d_i^{\text{same}} = s_i \frac{q_{i-1}^{\text{same}} - p_i}{\tau}, \qquad d_i^{\text{opp}} = s_i \frac{q_{i-1}^{\text{opp}} - p_i}{\tau}.$$

The categorical price component bins $d_i^{\text{same}}$ using lower bounds

$$(-\infty, -1, 0, 1, 2, 3, 5, 10, +\infty),$$

whereas the continuous price feature uses the opposing-quote distance,

$$\widetilde{p}_i = \frac{\log(\min\{25, \max\{1, d_i^{\text{opp}}\}\})}{\log 25}.$$

Thus, the categorical and continuous price representations deliberately use different reference quotes.

Volume is similarly represented through both categorical and continuous features. The categorical volume bins use lower bounds $(0, 10, 25, 40, 100, +\infty)$ shares, together with a binary indicator for volumes exactly equal to one of the finite bin boundaries. The continuous volume is linear below 100 shares and increasingly compressed above this point using a geometrically saturating transformation with decay factor $0.99$, normalization constant 200, and clipping at 1000 shares. This preserves resolution among typical small orders while limiting the influence of extreme volumes.

Inter-arrival time is computed after filtering as $\Delta t_i = t_i - t_{i-1}$, with $\Delta t_1 = 0$. In nanoseconds, let $d_i = \text{round}(m\Delta t_i)$, where $m$ converts timestamp units to nanoseconds. With $\ell = 10$ and $u = 5 \times 10^8$, the continuous time feature is

$$\widetilde{\Delta t}_i = \frac{\log(\min\{u, \max\{\ell, d_i\}\}) - \log \ell}{\log u - \log \ell}.$$

The categorical token separately records $\mathbf{1}_{d_i=0}$.

The categorical token additionally records whether a cancellation, deletion, or visible execution fully consumes the corresponding displayed price level, determined by comparing the message with the preceding and current LOB snapshots. The complete categorical identity is constructed in the fixed order

$$(a_i, s_i, \mathrm{bin}_v, \mathbf{1}_{\mathrm{round}}, \mathrm{bin}_p, \mathbf{1}_{\Delta t=0}, \mathbf{1}_{\text{level consumed}}).$$

For visible executions the categorical price bin is set to zero, while for hidden executions the categorical round-volume indicator is fixed to one; these canonicalizations affect only token identity and not the continuous features.

The vocabulary is fixed before model training. Message combinations with occurrence above $0.01\%$ in the training data corpus $\mathcal{A}_{\mathrm{main}} \times \mathcal{T}_{\mathrm{train}}$ are retained, producing 665 message tokens. Together with UNKNOWN and MASK, the complete vocabulary contains 667 entries. Any unrecognized or sufficiently rare combination is mapped to UNKNOWN, while MASK is introduced only during masked-message pretraining.

The final input associated with message $i$ is therefore

$$\mathbf{x}_i = \left(c_i, \widetilde{p}_i, \widetilde{v}_i, \widetilde{\Delta t}_i\right),$$

where $c_i$ is the categorical message token.

### A.2 TRAINING DETAILS

**General setup.** During training, samples are shuffled and the required classes are balanced by randomly downsampling each class to the frequency of the least common class; validation and test sets retain their natural class distributions. Models are trained with AdamW using batch size 32, weight decay $0.01$, and gradient clipping at global norm 1.0. Biases and normalization parameters are excluded from weight decay. The learning rate is linearly warmed up for 10,000 optimizer updates in pretraining and 1,000 optimizer updates in fine-tuning from $5 \times 10^{-6}$ to $5 \times 10^{-4}$ using a single cosine decay to $5 \times 10^{-5}$. Training runs for at most 50 epochs and stops after 10 epochs without improvement in validation loss. The checkpoint with the lowest validation loss is used for final test evaluation.

**Masked Message Modeling.** For MMM, each message position is independently selected with probability $0.20$. Among selected positions, approximately 10% are replaced by a random message, 9% are left unchanged, and the remaining 81% are explicitly masked. Random replacement changes the categorical token and samples each continuous component from $\mathcal{U}(0, 1)$, while explicit masking replaces the token by MASK and sets the continuous price, volume, and inter-arrival-time features to $0.5$. The same corruption decision is applied jointly to all components of a message, and corruption is resampled for every training batch.

The model predicts the original categorical token and continuous price, volume, and inter-arrival time at selected positions. Token prediction uses cross-entropy and the continuous outputs use mean squared error. The implemented MMM loss is

$$\mathcal{L}_{\mathrm{MMM}} = \frac{1}{4}\left(L_c + 15L_v + 45L_p + 25L_t\right),$$

where $L_c$ is token cross-entropy and $L_v$, $L_p$, and $L_t$ are the volume, price, and inter-arrival-time regression losses. The multipliers are selected to balance the regression losses approximately equally while giving the token loss the biggest impact. Unselected positions do not contribute to the objective.

**Aggregate Activity Forecasting.** AAF trains the pooled sequence representation to predict aggregate market activity at future offsets of 25, 50, and 100 messages. At each offset, the target contains eight quantities: signed net message volume on the bid and ask sides for two book-depth regions, changes in the best bid and ask, elapsed time, and the number of visible executions. Volume and count targets are normalized relative to statistics of the input window and adjusted for forecast horizon; quote movements are normalized by the mean input spread.

Each target $y$ is compressed using

$$q(y) = \mathrm{sign}(y)\left(1 - \exp(-|y|/2)\right),$$

and the forecast head uses a corresponding $\tanh$ output. AAF is optimized using Smooth L1 loss over valid target values and multiplied with $\lambda_f = 100$ to place it in the same range as the token loss $L_c$,

$$\mathcal{L}_{\mathrm{AAF}} = \lambda_f \ \mathrm{SmoothL1}\left(\widehat{\mathbf{Y}}_{\mathrm{forecast}}, \mathbf{Y}_{\mathrm{forecast}}\right).$$

The complete pretraining objective is

$$\mathcal{L}_{\mathrm{pre}} = \mathcal{L}_{\mathrm{MMM}} + \mathcal{L}_{\mathrm{AAF}}.$$

For the MMM-only ablation, the AAF head and loss term are omitted while all other pretraining settings are unchanged.

**Initialization and downstream adaptation.** Fresh LOBERT models use truncated-normal weight initialization with standard deviation $0.02$, zero biases, unit LayerNorm scales, and dropout probability $0.1$. For downstream adaptation, matching pretrained parameters are transferred to the new model while task-specific prediction heads are newly initialized.

In full fine-tuning, both the transferred LOBERT base and the downstream head are optimized. In frozen-base adaptation, gradients are disabled for the entire pretrained base, including the message embedding, transformer encoder, and sequence pooler, and the base is kept in evaluation mode so that its dropout is disabled. Only the task-specific prediction head is optimized.

### A.3 Model details

**Architecture overview.** LOBERT consists of a transferable transformer base followed by a lightweight task-specific classification head. The configuration used in the experiments has hidden dimension $H = 256$, four transformer layers, four attention heads, and feed-forward dimension 512. The model processes sequences of 512 messages using a vocabulary of 667 categorical message tokens together with continuous price and volume features. Absolute positional embeddings and LOB-snapshot inputs are disabled; temporal position is represented through continuous RoPE using scaled cumulative time delta. Dropout is 0.1 throughout the model.

#### A.3.1 Message embedding

For message $i$, the categorical token $c_i$ is mapped to a learned embedding and added to a linear projection of the scaled volume and relative-price features:

$$\mathbf{e}_i = \mathrm{LayerNorm}\left(\mathbf{E}[c_i] + \mathbf{W}_c \begin{pmatrix} \widetilde{v}_i \\ \widetilde{p}_i \end{pmatrix} + \mathbf{b}_c\right).$$

The resulting 256-dimensional representation is followed by dropout before entering the encoder. Inter-arrival time is not embedded as a content feature; instead, it determines temporal position inside self-attention.

#### A.3.2 Continuous-time transformer encoder

LOBERT uses four bidirectional transformer encoder layers with standard multi-head scaled dot-product attention and GELU feed-forward blocks. To represent irregular event timing, standard integer-position RoPE is replaced by scaled and cumulative continuous-time RoPE. For message $i$, the temporal coordinate is the cumulative scaled inter-arrival time

$$T_i = \sum_{j=1}^{i} \widetilde{\Delta t}_j.$$

These coordinates determine the rotations applied to the query and key vectors inside each attention head. Consequently, attention receives information about elapsed market time rather than only the number of intervening messages.

Each encoder layer contains four attention heads of dimension 64, followed by a position-wise feed-forward network $256 \to 512 \to 256$. Residual connections, LayerNorm, and dropout are applied in both sublayers. The output of every encoder layer is retained for subsequent sequence pooling.

### A.3.3 MULTI-LAYER SEQUENCE POOLING

Rather than relying only on the final transformer layer, LOBERT learns a scalar mixture of all four layer outputs. Let $\mathbf{H}^{(\ell)}$ denote the sequence representation produced by layer $\ell$. The mixed representation is

$$\widetilde{\mathbf{H}} = \gamma \sum_{\ell=1}^{4} w_\ell \mathbf{H}^{(\ell)}, \qquad w_\ell = \frac{\exp(u_\ell)}{\sum_{k=1}^{4} \exp(u_k)},$$

where $u_\ell$ and $\gamma$ are learned parameters. The mixture is initialized as an equal weighting of the encoder layers.

A learned attention pooler then reduces the mixed sequence to a single vector

$$\mathbf{z} = \sum_{i=1}^{L} \alpha_i \widetilde{\mathbf{h}}_i, \qquad \sum_{i=1}^{L} \alpha_i = 1,$$

where the attention weights $\alpha_i$ are produced using a learned global query. The scalar mixture and attention pooler are part of the transferable LOBERT base and are therefore trained during pretraining and reused during downstream adaptation.

### A.3.4 DOWNSTREAM CLASSIFICATION HEAD

For a task with $C$ classes, the pooled representation is passed through a small residual feed-forward block,

$$\mathbf{r} = \mathrm{LayerNorm}\left(\mathbf{z} + \mathbf{W}_o \,\mathrm{GELU}(\mathbf{W}_i \mathbf{z} + \mathbf{b}_i) + \mathbf{b}_o\right),$$

followed by dropout and a linear projection $\mathbb{R}^{256} \to \mathbb{R}^C$. Classification is optimized with cross-entropy loss.

The first-mid-price-move task uses $C = 2$, while spread crossing, order-flow imbalance, and message frequency use $C = 3$. This output dimension is the only architectural difference between their downstream LOBERT models.

**Transferable and task-specific parameters.** The shared LOBERT base contains approximately $2.35$ million parameters. The downstream classification head adds approximately $0.13$ million parameters, giving a total model size of approximately $2.48$ million parameters. In frozen-base adaptation, only the task-specific head is optimized; in full fine-tuning, both the shared base and downstream head are updated.

## A.4 DATA DETAILS

**Dataset composition.** After preprocessing, the dataset contains 12 assets (6 main and 6 held-out) observed over 94 trading days, giving 1,128 asset–days and 1.988 billion retained messages. An asset–day contains a median of 1,411,340 messages (range 20,912–6,167,595). The training, validation, and test partitions contain 75, 9, and 10 trading days, respectively, corresponding to 1.521, 0.197, and 0.270 billion messages. The retained message stream consists of submission 48.3%, partial cancellation 0.2%, deletion 43.1%, visible execution 6.0%, and hidden execution 2.3%. Among messages with an assigned side, 49.5% are bid-side and 50.5% are ask-side.

## A.5 DEEPLOB AND TABL BASELINE CONFIGURATIONS

Both baselines operate on reconstructed limit-order-book snapshots rather than LOBERT's message representation. Each input consists of 512 consecutive snapshots with 40 features per snapshot, corresponding to price and volume at the ten best ask and bid levels. For a task with $C$ classes, both models output $C$ logits and are trained end-to-end with cross-entropy loss.

Unlike LOBERT, neither baseline uses self-supervised pretraining or a reusable frozen representation. Both models are initialized independently for each downstream task and trained with all parameters unfrozen. Their batching, optimization, validation, and early-stopping procedures otherwise follow the common training protocol described in Appendix A.2.

Table 3: Dataset composition by asset. Message counts are computed after the preprocessing and filtering described in Appendix A.1.

| Asset | Set | Train | Val. | Test | Messages (M) | Median/day (M) |
|---|---|---|---|---|---|---|
| AAPL | Main | 75 | 9 | 10 | 149.7 | 1.46 |
| AMD | Main | 75 | 9 | 10 | 126.5 | 1.23 |
| AMZN | Main | 75 | 9 | 10 | 171.3 | 1.62 |
| ASML | Main | 75 | 9 | 10 | 52.7 | 0.53 |
| GOOG | Main | 75 | 9 | 10 | 248.8 | 2.56 |
| PLTR | Main | 75 | 9 | 10 | 180.7 | 1.75 |
| INTC | Held-out | 75 | 9 | 10 | 137.5 | 1.33 |
| META | Held-out | 75 | 9 | 10 | 51.7 | 0.48 |
| MSFT | Held-out | 75 | 9 | 10 | 83.1 | 0.85 |
| NFLX | Held-out | 75 | 9 | 10 | 85.5 | 0.96 |
| NVDA | Held-out | 75 | 9 | 10 | 456.9 | 5.10 |
| TSLA | Held-out | 75 | 9 | 10 | 243.6 | 2.54 |

Table 4: Natural downstream label distributions after non-overlapping 512-message sampling. Valid/day reports the median number of labeled observations per asset–day with the minimum and maximum in brackets. Training classes are subsequently balanced by downsampling during model training; validation and test retain the natural distributions shown here.

| Task | Split | Valid obs. | Valid/day | Classes (%) |
|---|---|---|---|---|
| message_frequency | Train | 2,969,461 | 2,635 [40–12,046] | 0 30.5 / 1 36.7 / 2 32.8 |
| message_frequency | Validation | 384,857 | 2,823 [647–11,898] | 0 30.7 / 1 36.9 / 2 32.5 |
| message_frequency | Test | 527,940 | 3,925 [526–11,683] | 0 29.9 / 1 38.1 / 2 32.0 |
| midprice_first | Train | 2,969,461 | 2,635 [40–12,046] | 0 49.9 / 1 50.1 |
| midprice_first | Validation | 384,857 | 2,823 [647–11,898] | 0 50.1 / 1 49.9 |
| midprice_first | Test | 527,940 | 3,925 [526–11,683] | 0 50.1 / 1 49.9 |
| order_flow_imbalance | Train | 2,969,461 | 2,635 [40–12,046] | 0 31.6 / 1 36.8 / 2 31.6 |
| order_flow_imbalance | Validation | 384,857 | 2,823 [647–11,898] | 0 31.3 / 1 37.4 / 2 31.3 |
| order_flow_imbalance | Test | 527,940 | 3,925 [526–11,683] | 0 31.0 / 1 38.1 / 2 30.8 |
| spread_cross_horizon | Train | 2,969,461 | 2,635 [40–12,046] | 0 41.7 / 1 17.1 / 2 41.3 |
| spread_cross_horizon | Validation | 384,857 | 2,823 [647–11,898] | 0 41.6 / 1 17.4 / 2 41.1 |
| spread_cross_horizon | Test | 527,940 | 3,925 [526–11,683] | 0 42.5 / 1 15.9 / 2 41.6 |

**DeepLOB.** DeepLOB follows the convolutional–recurrent architecture of Zhang et al. (2019). The $512\times40$ snapshot sequence is first normalized and processed by a convolutional feature extractor that progressively combines information across book levels and neighboring time steps. An inception-style module then applies multiple temporal receptive fields in parallel, producing a sequence of 192-dimensional features. These features are passed to a single-layer LSTM with hidden dimension 64, and the final recurrent state is mapped to the $C$ output classes. The configuration used here contains approximately $0.14$ million trainable parameters and does not use dropout.

**TABL.** TABL follows the bilinear attention architecture of Tran et al. (2019). The $512\times40$ input is first processed by the model's bilinear input-normalization module and subsequently by two bilinear transformation layers that reduce the temporal and feature dimensions. A final temporal-attention bilinear layer produces the $C$ class logits. ReLU activations are used between the intermediate transformations. The configuration used here contains approximately $0.09$ million trainable parameters and does not use dropout.

Table 5: Effective downstream training-set sizes under each target-data budget. For each task and budget, values are summarized across target assets. Counts are median [minimum–maximum]. Raw denotes all valid non-overlapping labeled observations in the selected asset–days; balanced denotes the number remaining after downsampling every class to the least frequent class.

| Task | B | Raw observations | Balanced observations | Retained (%) |
|---|---|---|---|---|
| message_frequency | 1 | 2,172 [847–9,414] | 2,018 [498–9,009] | 90.5 |
| message_frequency | 3 | 7,530 [2,760–31,969] | 7,168 [1,878–29,862] | 90.6 |
| message_frequency | 8 | 21,020 [7,159–80,645] | 19,928 [4,377–74,838] | 91.9 |
| message_frequency | 15 | 35,876 [12,138–134,454] | 32,080 [7,278–124,617] | 92.4 |
| midprice_first | 1 | 2,172 [847–9,414] | 2,114 [778–9,322] | 97.9 |
| midprice_first | 3 | 7,530 [2,760–31,969] | 7,510 [2,702–31,882] | 99.1 |
| midprice_first | 8 | 21,020 [7,159–80,645] | 20,828 [7,158–80,630] | 99.5 |
| midprice_first | 15 | 35,876 [12,138–134,454] | 35,529 [12,122–134,284] | 99.5 |
| order_flow_imbalance | 1 | 2,172 [847–9,414] | 1,910 [213–8,256] | 88.6 |
| order_flow_imbalance | 3 | 7,530 [2,760–31,969] | 6,891 [804–28,101] | 90.3 |
| order_flow_imbalance | 8 | 21,020 [7,159–80,645] | 18,442 [1,866–72,477] | 91.7 |
| order_flow_imbalance | 15 | 35,876 [12,138–134,454] | 32,966 [3,429–122,436] | 94.7 |
| spread_cross_horizon | 1 | 2,171 [847–9,413] | 543 [180–6,072] | 23.8 |
| spread_cross_horizon | 3 | 7,530 [2,759–31,968] | 2,038 [639–16,926] | 25.0 |
| spread_cross_horizon | 8 | 21,018 [7,155–80,642] | 5,222 [1,692–41,511] | 24.7 |
| spread_cross_horizon | 15 | 35,871 [12,129–134,449] | 11,056 [3,162–66,879] | 29.3 |

### A.6 Downstream task details

All downstream labels are constructed independently within each security–day from market activity occurring strictly after the final message of the input sequence. Horizons below count retained messages unless otherwise stated. For the price-based tasks, the daily tick size $\tau$ is inferred as the smallest nonzero change in either the best bid or best ask. Let $a_i$ and $b_i$ denote the best ask and bid after message $i$, and define the midpoint in ticks as

$$m_i = \frac{a_i + b_i}{2\tau}.$$

#### A.6.1 First mid-price move

The mid-price task predicts the direction of the first subsequent midpoint move of at least half a tick. Starting from $m_i$, we define

$$y_i = \begin{cases} 0, & \text{if } m_{i+h} \le m_i - 0.5 \text{ is reached first,} \\ 1, & \text{if } m_{i+h} \ge m_i + 0.5 \text{ is reached first.} \end{cases}$$

The search is limited to approximately $10^4$ subsequent messages. Samples for which neither threshold is reached within the available horizon are excluded. The resulting binary classes correspond to down and up. Because a one-tick movement in either best quote changes the midpoint by half a tick, this target captures the direction of the next observable midpoint movement while allowing its realization horizon to vary.

#### A.6.2 Spread crossing

The spread-crossing task predicts whether either side of the future book moves beyond the opposite quote observed at the end of the input. Within the next 500 messages, a bearish crossing occurs when

$$\frac{a_{i+h}}{\tau} \le \frac{b_i}{\tau} - 1,$$

and a bullish crossing occurs when

$$\frac{b_{i+h}}{\tau} \ge \frac{a_i}{\tau} + 1.$$

The target is

$$y_i = \begin{cases} 0, & \text{if a bearish crossing occurs first,} \\ 2, & \text{if a bullish crossing occurs first,} \\ 1, & \text{if neither occurs within 500 messages.} \end{cases}$$

Samples without sufficient future observations to determine the 500-message outcome are excluded.

### A.6.3 Order-flow imbalance

The order-flow imbalance task measures future signed order flow near the top of the book. Beginning after message $i$, we collect the next 100 eligible events, where eligible events are submissions, cancellations, deletions, and visible executions affecting levels 1–3.

For event $j$, define signed volume

$$q_j = \begin{cases} +v_j, & a_j = 1 \quad \text{(submission)}, \\ -v_j, & a_j \in \{2,3,4\} \quad \text{(cancellation, deletion, or execution)}. \end{cases}$$

Using $s_j = +1$ for bids and $s_j = -1$ for asks, let

$$B_i = \sum_{j \in \mathcal{H}_i} q_j \mathbf{1}[s_j = +1], \qquad A_i = \sum_{j \in \mathcal{H}_i} q_j \mathbf{1}[s_j = -1],$$

and

$$\bar{v}_i = \frac{1}{|\mathcal{H}_i|} \sum_{j \in \mathcal{H}_i} |v_j|,$$

where $\mathcal{H}_i$ contains the 100 eligible future events. The label is

$$y_i = \begin{cases} 2, & B_i \geq 3A_i \text{ and } B_i - A_i \geq 10\bar{v}_i, \\ 0, & A_i \geq 3B_i \text{ and } A_i - B_i \geq 10\bar{v}_i, \\ 1, & \text{otherwise.} \end{cases}$$

Classes 0, 1, and 2 therefore represent ask-side dominance, neutral flow, and bid-side dominance, respectively. The two conditions require both relative dominance and a difference large relative to typical message volume.

### A.6.4 Message frequency

The message-frequency task compares recent and subsequent message arrival rates. Let

$$\ell_j = \log(1 + \Delta t_j),$$

where $\Delta t_j$ is the inter-arrival time in nanoseconds. We compare the mean log inter-arrival time over the preceding 500 messages,

$$\bar{\ell}_i^{\text{past}} = \frac{1}{500} \sum_{j=i-499}^{i} \ell_j,$$

with that over the next 100 messages,

$$\bar{\ell}_i^{\text{future}} = \frac{1}{100} \sum_{j=i+1}^{i+100} \ell_j.$$

Defining

$$\delta_i = \bar{\ell}_i^{\text{future}} - \bar{\ell}_i^{\text{past}}, \qquad \kappa = \log(1.75),$$

the target is

$$y_i = \begin{cases} 0, & \delta_i \leq -\kappa \quad \text{(faster)}, \\ 2, & \delta_i \geq \kappa \quad \text{(slower)}, \\ 1, & |\delta_i| < \kappa \quad \text{(no substantial change)}. \end{cases}$$

Thus, the directional classes correspond approximately to at least a $1.75\times$ multiplicative change in the geometric mean of $1 + \Delta t$. The logarithmic formulation makes proportional increases and decreases comparable while remaining well defined for simultaneous messages.

### A.7 RESULT DETAILS

This section reports the numerical results underlying the aggregate comparisons in Section 5. Unless otherwise stated, values are Macro-F1 scores evaluated on the held-out test period. Differences between model variants are computed within matched task–budget–security configurations before aggregation. Positive differences therefore indicate an advantage for the first model named in each comparison.

**Frozen versus full fine-tuning.** Across the 96 paired task–budget–security configurations on the main securities, pretrained–frozen LOBERT exceeds pretrained–unfrozen LOBERT by 0.044 Macro-F1 on average (median 0.019). The frozen variant obtains the higher score in 72/96 configurations (75.0%).

Table 6: Pretrained–frozen versus pretrained–unfrozen LOBERT by downstream task on the main securities. Scores and differences are averaged over all target-data budgets and securities. Positive differences favor frozen adaptation.

| Task | Frozen | Unfrozen | $\Delta$ |
|---|---|---|---|
| $\tau_{\text{midprice}}$ | 0.613 | 0.602 | 0.010 |
| $\tau_{\text{cross}}$ | 0.408 | 0.350 | 0.057 |
| $\tau_{\text{ofi}}$ | 0.366 | 0.320 | 0.046 |
| $\tau_{\text{freq}}$ | 0.417 | 0.356 | 0.061 |

Table 7: Pretrained–frozen versus pretrained–unfrozen LOBERT by target-data budget on the main securities. The final four columns summarize the paired frozen-minus-unfrozen differences across tasks and securities.

| B | Frozen | Unfrozen | Mean $\Delta$ | Median $\Delta$ | Min. $\Delta$ | Max. $\Delta$ |
|---|---|---|---|---|---|---|
| 1 | 0.435 | 0.425 | 0.010 | 0.002 | -0.047 | 0.077 |
| 3 | 0.445 | 0.415 | 0.030 | 0.025 | -0.031 | 0.095 |
| 8 | 0.460 | 0.385 | 0.075 | 0.041 | -0.048 | 0.335 |
| 15 | 0.463 | 0.403 | 0.060 | 0.034 | -0.061 | 0.256 |

Table 8: Fraction of main securities for which pretrained–frozen LOBERT outperforms pretrained–unfrozen LOBERT for each downstream task and target-data budget.

| Task | $\mathbf{B = 1}$ | $\mathbf{B = 3}$ | $\mathbf{B = 8}$ | $\mathbf{B = 15}$ |
|---|---|---|---|---|
| $\tau_{\text{midprice}}$ | 16.7% | 33.3% | 50.0% | 66.7% |
| $\tau_{\text{cross}}$ | 66.7% | 83.3% | 100.0% | 100.0% |
| $\tau_{\text{ofi}}$ | 66.7% | 100.0% | 100.0% | 83.3% |
| $\tau_{\text{freq}}$ | 100.0% | 83.3% | 66.7% | 83.3% |

**Absolute performance on held-out securities.** Averaged across the 16 task–budget cells, pretrained–frozen LOBERT scores 0.008 Macro-F1 higher on the main securities than on securities excluded from pretraining. This comparison is unpaired across securities.

**Pretraining benefit on held-out securities.** Across the 96 paired held-out-security configurations, pretrained–frozen LOBERT improves Macro-F1 over random initialization by 0.097 on average and obtains the higher score in 88/96 configurations (91.7%).

**Comparison with task-specific baselines.** Across the 8 evaluated task–budget cells, pretrained–frozen LOBERT exceeds the stronger of DeepLOB and TABL by 0.059 Macro-F1 on average and achieves the highest score in 8/8 cells.

**Pretraining benefit on main securities.** Across the 96 paired main-security configurations, pretrained–frozen LOBERT improves Macro-F1 over random initialization by 0.092 on average and obtains the higher score in 90/96 configurations (93.8%).

**Comparison of the pretraining effect across security sets.** The mean paired pretraining gain is 0.092 Macro-F1 on the main securities and 0.097 on the held-out securities, a difference of -0.004. Pretrained–frozen LOBERT wins 90/96 (93.8%) paired configurations on the main securities and 88/96 (91.7%) on the held-out securities.

Table 9: Individual main-security configurations with the largest frozen-minus-unfrozen differences in either direction. Negative values indicate configurations in which full fine-tuning performs better.

| Subset | Task | B | Asset | Frozen | Unfrozen | Δ |
|---|---|---|---|---|---|---|
| Unfrozen helps most | $\tau_{\text{ofi}}$ | 15 | ASML | 0.422 | 0.483 | -0.061 |
| Unfrozen helps most | $\tau_{\text{midprice}}$ | 8 | ASML | 0.559 | 0.606 | -0.048 |
| Unfrozen helps most | $\tau_{\text{ofi}}$ | 1 | ASML | 0.420 | 0.467 | -0.047 |
| Unfrozen helps most | $\tau_{\text{freq}}$ | 15 | ASML | 0.398 | 0.440 | -0.041 |
| Unfrozen helps most | $\tau_{\text{freq}}$ | 8 | AAPL | 0.388 | 0.427 | -0.039 |
| Unfrozen helps most | $\tau_{\text{cross}}$ | 1 | AMD | 0.302 | 0.339 | -0.037 |
| Unfrozen helps most | $\tau_{\text{freq}}$ | 8 | ASML | 0.404 | 0.438 | -0.034 |
| Unfrozen helps most | $\tau_{\text{midprice}}$ | 3 | ASML | 0.578 | 0.609 | -0.031 |
| Unfrozen helps most | $\tau_{\text{ofi}}$ | 1 | PLTR | 0.317 | 0.343 | -0.026 |
| Unfrozen helps most | $\tau_{\text{midprice}}$ | 1 | PLTR | 0.613 | 0.638 | -0.026 |
| Frozen helps most | $\tau_{\text{midprice}}$ | 8 | GOOG | 0.668 | 0.333 | 0.335 |
| Frozen helps most | $\tau_{\text{freq}}$ | 8 | GOOG | 0.439 | 0.144 | 0.294 |
| Frozen helps most | $\tau_{\text{freq}}$ | 15 | AMZN | 0.423 | 0.168 | 0.256 |
| Frozen helps most | $\tau_{\text{freq}}$ | 15 | PLTR | 0.425 | 0.180 | 0.244 |
| Frozen helps most | $\tau_{\text{cross}}$ | 15 | AAPL | 0.418 | 0.210 | 0.207 |
| Frozen helps most | $\tau_{\text{ofi}}$ | 8 | PLTR | 0.356 | 0.176 | 0.180 |
| Frozen helps most | $\tau_{\text{freq}}$ | 8 | PLTR | 0.438 | 0.263 | 0.176 |
| Frozen helps most | $\tau_{\text{cross}}$ | 8 | PLTR | 0.401 | 0.241 | 0.159 |
| Frozen helps most | $\tau_{\text{cross}}$ | 8 | AAPL | 0.414 | 0.283 | 0.131 |
| Frozen helps most | $\tau_{\text{ofi}}$ | 15 | PLTR | 0.381 | 0.258 | 0.123 |

Table 10: Absolute pretrained–frozen LOBERT performance on main and held-out securities. Values are Macro-F1 averaged across the securities in each group. Positive differences indicate higher performance on main securities.

| Task | B | Main | Held-out | Main–held |
|---|---|---|---|---|
| $\tau_{\text{midprice}}$ | 1 | 0.603 | 0.607 | -0.005 |
| $\tau_{\text{midprice}}$ | 3 | 0.610 | 0.611 | -0.001 |
| $\tau_{\text{midprice}}$ | 8 | 0.614 | 0.624 | -0.010 |
| $\tau_{\text{midprice}}$ | 15 | 0.623 | 0.628 | -0.005 |
| $\tau_{\text{cross}}$ | 1 | 0.381 | 0.374 | 0.008 |
| $\tau_{\text{cross}}$ | 3 | 0.411 | 0.381 | 0.030 |
| $\tau_{\text{cross}}$ | 8 | 0.424 | 0.395 | 0.029 |
| $\tau_{\text{cross}}$ | 15 | 0.415 | 0.413 | 0.002 |
| $\tau_{\text{ofi}}$ | 1 | 0.346 | 0.321 | 0.026 |
| $\tau_{\text{ofi}}$ | 3 | 0.356 | 0.320 | 0.036 |
| $\tau_{\text{ofi}}$ | 8 | 0.378 | 0.374 | 0.004 |
| $\tau_{\text{ofi}}$ | 15 | 0.385 | 0.376 | 0.009 |
| $\tau_{\text{freq}}$ | 1 | 0.408 | 0.392 | 0.017 |
| $\tau_{\text{freq}}$ | 3 | 0.405 | 0.403 | 0.002 |
| $\tau_{\text{freq}}$ | 8 | 0.424 | 0.428 | -0.003 |
| $\tau_{\text{freq}}$ | 15 | 0.429 | 0.433 | -0.004 |

Table 11: Mean main-minus-held-out Macro-F1 difference for pretrained–frozen LOBERT by target-data budget, averaged across downstream tasks.

| B | Mean main–held |
|---|---|
| 1 | 0.011 |
| 3 | 0.017 |
| 8 | 0.005 |
| 15 | 0.000 |

Table 12: Pretrained–frozen LOBERT Macro-F1 for each security excluded from self-supervised pretraining. Each row reports one security–task pair across the four target-data budgets.

| Asset | Task | $\mathbf{B = 1}$ | $\mathbf{B = 3}$ | $\mathbf{B = 8}$ | $\mathbf{B = 15}$ |
|---|---|---|---|---|---|
| INTC | $\tau_{\text{midprice}}$ | 0.607 | 0.624 | 0.630 | 0.628 |
| INTC | $\tau_{\text{cross}}$ | 0.375 | 0.382 | 0.411 | 0.418 |
| INTC | $\tau_{\text{ofi}}$ | 0.335 | 0.235 | 0.380 | 0.370 |
| INTC | $\tau_{\text{freq}}$ | 0.307 | 0.396 | 0.426 | 0.433 |
| META | $\tau_{\text{midprice}}$ | 0.573 | 0.560 | 0.573 | 0.585 |
| META | $\tau_{\text{cross}}$ | 0.341 | 0.412 | 0.425 | 0.426 |
| META | $\tau_{\text{ofi}}$ | 0.249 | 0.344 | 0.407 | 0.415 |
| META | $\tau_{\text{freq}}$ | 0.404 | 0.425 | 0.396 | 0.417 |
| MSFT | $\tau_{\text{midprice}}$ | 0.598 | 0.610 | 0.625 | 0.635 |
| MSFT | $\tau_{\text{cross}}$ | 0.368 | 0.380 | 0.399 | 0.422 |
| MSFT | $\tau_{\text{ofi}}$ | 0.322 | 0.369 | 0.327 | 0.309 |
| MSFT | $\tau_{\text{freq}}$ | 0.408 | 0.378 | 0.435 | 0.442 |
| NFLX | $\tau_{\text{midprice}}$ | 0.598 | 0.590 | 0.623 | 0.619 |
| NFLX | $\tau_{\text{cross}}$ | 0.411 | 0.409 | 0.423 | 0.416 |
| NFLX | $\tau_{\text{ofi}}$ | 0.375 | 0.361 | 0.367 | 0.389 |
| NFLX | $\tau_{\text{freq}}$ | 0.402 | 0.359 | 0.424 | 0.423 |
| NVDA | $\tau_{\text{midprice}}$ | 0.612 | 0.629 | 0.635 | 0.637 |
| NVDA | $\tau_{\text{cross}}$ | 0.360 | 0.303 | 0.302 | 0.394 |
| NVDA | $\tau_{\text{ofi}}$ | 0.303 | 0.320 | 0.365 | 0.372 |
| NVDA | $\tau_{\text{freq}}$ | 0.411 | 0.449 | 0.449 | 0.441 |
| TSLA | $\tau_{\text{midprice}}$ | 0.657 | 0.653 | 0.660 | 0.663 |
| TSLA | $\tau_{\text{cross}}$ | 0.388 | 0.399 | 0.410 | 0.402 |
| TSLA | $\tau_{\text{ofi}}$ | 0.341 | 0.293 | 0.401 | 0.401 |
| TSLA | $\tau_{\text{freq}}$ | 0.418 | 0.408 | 0.437 | 0.440 |

Table 13: Mean Macro-F1 gain from self-supervised pretraining on securities excluded from the pretraining corpus. Each value is the mean paired difference between pretrained–frozen and randomly initialized LOBERT across the held-out securities.

| Task | $\mathbf{B = 1}$ | $\mathbf{B = 3}$ | $\mathbf{B = 8}$ | $\mathbf{B = 15}$ |
|---|---|---|---|---|
| $\tau_{\text{midprice}}$ | 0.181 | 0.117 | 0.087 | 0.045 |
| $\tau_{\text{cross}}$ | 0.099 | 0.117 | 0.108 | 0.156 |
| $\tau_{\text{ofi}}$ | 0.075 | 0.052 | 0.138 | 0.133 |
| $\tau_{\text{freq}}$ | 0.062 | 0.044 | 0.083 | 0.054 |

Table 14: Detailed comparison with the task-specific DeepLOB and TABL baselines on the main securities. Scores are Macro-F1 averaged across the six main securities. Best baseline denotes the larger of the DeepLOB and TABL scores within each task–budget cell, and the margin is LOBERT minus this value.

| Task | B | DeepLOB | TABL | LOBERT | Best baseline | Margin |
|---|---|---|---|---|---|---|
| $\tau_{\text{midprice}}$ | 1 | 0.446 | 0.503 | 0.603 | 0.503 | 0.099 |
| $\tau_{\text{midprice}}$ | 15 | 0.477 | 0.569 | 0.623 | 0.569 | 0.054 |
| $\tau_{\text{cross}}$ | 1 | 0.235 | 0.332 | 0.381 | 0.332 | 0.049 |
| $\tau_{\text{cross}}$ | 15 | 0.244 | 0.348 | 0.415 | 0.348 | 0.067 |
| $\tau_{\text{ofi}}$ | 1 | 0.222 | 0.329 | 0.346 | 0.329 | 0.017 |
| $\tau_{\text{ofi}}$ | 15 | 0.274 | 0.341 | 0.385 | 0.341 | 0.044 |
| $\tau_{\text{freq}}$ | 1 | 0.238 | 0.328 | 0.408 | 0.328 | 0.080 |
| $\tau_{\text{freq}}$ | 15 | 0.284 | 0.372 | 0.429 | 0.372 | 0.057 |

Table 15: Mean Macro-F1 gain from self-supervised pretraining on the main securities. Each value is the mean paired difference between pretrained–frozen and randomly initialized LOBERT across the six main securities.

| **Task** | $\mathbf{B = 1}$ | $\mathbf{B = 3}$ | $\mathbf{B = 8}$ | $\mathbf{B = 15}$ |
|---|---|---|---|---|
| $\tau_{\text{midprice}}$ | 0.165 | 0.113 | 0.101 | 0.093 |
| $\tau_{\text{cross}}$ | 0.082 | 0.109 | 0.128 | 0.119 |
| $\tau_{\text{ofi}}$ | 0.087 | 0.095 | 0.063 | 0.107 |
| $\tau_{\text{freq}}$ | 0.067 | 0.061 | 0.038 | 0.053 |