TRANSFORMERS VERSUS LSTMS FOR ELECTRONIC TRADING

A PREPRINT

Paul Bilokon
Department of Computing
Imperial College London
South Kensington Campus
London SW7 2AZ
paul.bilokon@imperial.ac.uk

Yitao Qiu
Department of Computing
Imperial College London
South Kensington Campus
London SW7 2AZ
yitao.qiu21@imperial.ac.uk

September 21, 2023

ABSTRACT

With the rapid development of artificial intelligence, long short term memory (LSTM), one kind of recurrent neural network (RNN), has been widely applied in time series prediction.

Like RNN, Transformer is designed to handle the sequential data. As Transformer achieved great success in Natural Language Processing (NLP), researchers got interested in Transformer's performance on time series prediction, and plenty of Transformer-based solutions on long time series forecasting have come out recently. However, when it comes to financial time series prediction, LSTM is still a dominant architecture. Therefore, the question this study wants to answer is: whether the Transformer-based model can be applied in financial time series prediction and beat LSTM.

To answer this question, various LSTM-based and Transformer-based models are compared on multiple financial prediction tasks based on high-frequency limit order book data. A new LSTM-based model called DLSTM is built and new architecture for the Transformer-based model is designed to adapt for financial prediction. The experiment result reflects that the Transformer-based model only has the limited advantage in absolute price sequence prediction. The LSTM-based models show better and more robust performance on difference sequence prediction, such as price difference and price movement.

1 Introduction

Financial time series prediction is a significant task in investing and market-making activities. The Efficient Market Hypothesis proposed by Eugene [1] states that all the information of the asset's inner value has already precisely and completely reflected on the asset price, and it is impossible to beat the market by financial prediction. However, whether the market is efficient is questionable because technical analysis [2] believes the financial market is the physical movement of price (or features derived from prices). The price information can be interpreted by waves and patterns that can repeat themselves, where it is possible to make profitable buy or sell decisions in advance [3, 4]. During the prediction, challenging factors are noise and volatile features because price information is generally non-linear and non-stationary [5]. Lots of models are proposed to solve the financial time series problem. A typical linear model for regression is Auto-Regressive Integrated Moving average (ARIMA) [6] and its variations, which requires domain expertise to handcraft features. With the development of machine learning, Artificial Neural Networks (ANN) raises great interest because of their capability to extract more abstract features from data and find a hidden non-linear relationship without assumptions or human expertise by adding more parameters. Long short-term memory (LSTM), which is a special recurrent neural network (RNN) architecture that has been proven successful in the application of sequential data, is widely applied to handwriting recognition [7] and speech recognition [8]. Like RNN, the Transformer [9] is also used to handle the sequential data. Compared to LSTM, the Transformer does not need to handle the sequence data in order, which instead confers the meaning of the sequence by the Self-attention mechanism.

Applying LSTM and Transformer for financial time series prediction is a popular trend nowadays. Depending on the historical financial data, researchers usually make predictions for the future numerical prices, price difference, return or future price movement (rise, stationary, fall). Although LSTM and Transformer are applied in different aspects for this problem. There are mainly two research directions:

1) Make predictions based on high-frequency Limit Order Book (LOB) data and its derived features, such as Volume Order Imbalance (VOI) and Trade Flow Imbalance (TFI) [10–14].

2) Make predictions based on OHLC (Open, High, Low, Close) data and its derived financial indices, such as Relative Strength Index (RSI) and Moving average convergence divergence (MACD) [15–25].

Since 2017, the Transformer has been increasingly used for Natural Language Processing (NLP) problems. It produces more impressive results than RNN, such as machine translation [26] and speech applications [27], replacing RNN models such as LSTM in NLP tasks. Recently, a surge of Transformer-based solutions for less explored long time series forecasting problem has appeared [28]. However, as for the financial time

series prediction, LSTM remains the dominant architecture.

Whether Transformer-based methods can be the right solution for financial time series forecasting is a problem worth investigating. Therefore, this paper is going to compare the performance of different Transformer-based and LSTM-based methods on financial time series prediction problems based on LOB data and attempt to adapt new models based on Transformer and LSTM. The contributions of this study are summed up as follows:

1. Systematically compare Transformer-based and LSTM-based methods in different financial prediction tasks based on high frequency LOB data collected from Binance Exchange. Tasks include (1) mid-price prediction, (2) mid-price difference prediction and (3) mid-price movement prediction.
2. For the first and second tasks, comparisons are all conducted on previous LSTM-based and Transformer-based methods. In the first task, the Transformer-based method has around $10\% - 25\%$ prediction error less than the LSTM-based method, but the prediction result quality is insufficient for trading. In the second task, the LSTM-based method performs better than the Transformer-based method, where its highest out-of-sample R^2 reaches around 11.5%.
3. The most significant contribution of this study is in the last task, mid-price movement prediction. A new LSTM-based model named DLSTM is developed for this task, which combines LSTM and the time series decomposition method. This model achieves 63.73% to 73.31% accuracy and shows strong profitability and robustness in simulated trading, outperforming previous LSTM and Transformer-based methods. In addition, the architecture of previous Transformer-based methods is also changed in order to adapt movement prediction task.

The later parts of this study are structured as follows: The background and related work are introduced in Section 2. Section 3 describes the formulation of three financial prediction tasks. Section 4 explains the details of previous Transformer-based and LSTM-based methods used for comparison. Section 5 is the analysis of experiment results. Please note that the details of the newly developed DLSTM model and the architecture changes of Transformer models are explained in Section 5.3 in Section 5. The study is organized in this way to provide readers with a better understanding of the relationship among three financial prediction tasks.

The **Source Code** of this study is available at: `https://github.com/772435284/transformers_versus_lstms_for_electronic_trading`

Acknowledgements We would like to express our gratitude to Zhipeng Wang for constructive comments and suggestions.

2 Background and Related work

Time series prediction has been applied in different financial activities. For example, the trader or market maker predicts the future price or the price movement of the assets so that he/she can design trading/market making strategies based on the prediction results to make profit from it. This part examines papers related to time series prediction using LSTM and Transformer. Most of them are related to financial time series prediction.

2.1 LSTM-based Time Series Prediction Solutions

LSTM has been widely utilized in financial time series prediction depending on OHLC data and its derived financial indices. Many works [15–19] present predicting stock prices as successful by LSTM. Bidirectional LSTM (BiLSTM) is applied to increase the prediction performance [20]. With the rise of NLP, Sequence-to-Sequence Model (S2S) [29] is proposed and applied in machine translation and question answering. It is now also applied in financial time series prediction by combining LSTM structure and attention mechanisms contributing to higher performance [21, 22].

LSTM has successfully forecasted high-frequency data depending on the large datasets extracted from the limit order book (LOB). In terms of making predictions upon order book data, Convolution Neural Network (CNN) and LSTM are both in favour of research and sometimes they are combined. Sirignano et al. [10] trained a universal model using LSTM to predict the LOB price movement by data from all stocks, which outperforms linear and non-linear models trained on a specific asset. Zhang et al. [11] combine CNN and LSTM to form a new deep neural network architecture called DeepLOB to predict future stock price movements in LOB data outperforming architecture only containing LSTM. Zhang et al. [12] also combine the DeepLOB architecture with Seq2Seq and Attention model to forecast multi-horizon future price movements in one forward procedure, which reduces the training effort and achieves better performance in long horizon prediction than DeepLOB. According to Tsantekidis et al. [13], the structure of CNN-LSTM architecture is able to outperform other models on LOB price movement prediction because of its more stable behaviour. Kolm et al. [14] use the OFI feature derived from the LOB to predict the future min-price return, where the CNN-LSTM structure still achieves the best performance.

2.2 Transformer-based Time Series Prediction Solutions

As Transformer makes a great contribution to NLP [30], many advanced models are proposed, such as BERT and GPT-3, which now have more influence in time series prediction. As the canonical Self-attention mechanism has $O(L^2)$ time and memory complexity, many modifications have been made to the Transformer in order to adapt to the time series prediction problem to process the long sequence efficiently. There are

many alternative Transformer models for long time series forecasting problem have been developed recently [28]: LogTrans[31], Reformer [32], Informer [33], Autoformer [34], Pyraformer [35] and the recent FEDformer [36].

Theses Transformer models mentioned above are mainly evaluated on non-financial datasets, such as electricity consumption, traffic usage, and solar energy dataset. They achieve a considerable performance increase in accuracy over the LSTM model. Some works start applying Transformers for financial time series prediction. Hu [23] uses a Temporal Fusion Transformer with support vector regression (SVR) and LSTM to predict stock price. Sridhar et al. [24] predict the Dogecoin price through Transformer, which has superior performance compared to LSTM. Sonkiya et al. [25] do sentiment analysis by BERT to generate the sentiment score, which is then combined with other financial indices and passed into the Generative Adversarial Network (GAN) for stock price prediction. These works [23–25] use Transformer-based methods to make predictions based on OHLC data and the research on applying Transformer for LOB prediction is limited.

Overall, LSTM is applied broadly in financial time series prediction and has been tested on various datasets, while the Transformer is limited. Therefore, this study wants to apply these Transformer-based models to a wider region in financial time series prediction to compare their performance to LSTM.

3 Financial Time Series Prediction Tasks Formulation

This study compares LSTM-based and Transformer-based methods among three financial prediction tasks based on LOB data. In this section, the basic concept of LOB will be first introduced, and then the formulation of three financial prediction tasks will be explained in detail. Three tasks are listed below:

- Task 1: LOB Mid-Price Prediction
- Task 2: LOB Mid-Price Difference Prediction
- Task 3: LOB Mid-Price Movement Prediction

3.1 Limit Order Book

With computers and the Internet, most financial products such as stocks, forex and cryptocurrency are traded on the electronic market nowadays. Two types of orders exist in the electronic market: limit order and market order. According to Gould et al. [37], a limit order is the order to execute buy or sell direction at a specific price, where the orders can be succeeded, overdue, or cancelled and are recorded by the LOB. There are bid limit orders and ask limit orders used to buy and sell products by the trader or sell and buy products by the

market marker. The highest bid price the buyers are ready to buy is referred to as the best bid price, and the lowest ask price the sellers are ready to sell is called the best ask price. The average of these two prices is called the mid-price, which reflects the current value of the financial product. The difference between them is the spread, which the market marker can usually make a profit. The illustration of the LOB is shown in Fig 1. Another type of order is the market order, where the trader can immediately buy/sell the product at the

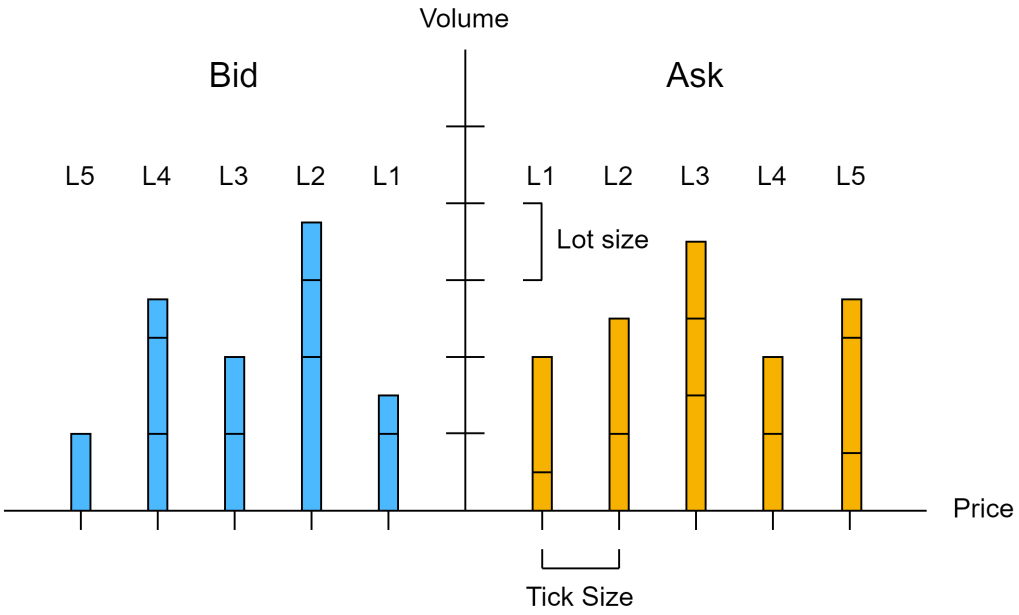

Figure 1: An illustration of LOB based on Zhang et al. [11] and Kolm et al. [14].

best price. There exists a matching mechanism in the LOB. Most exchanges adopt the price and time priority matching mechanism. The limit orders will be first executed in order with a better price. If two orders have the same execution price, then the order that comes first in time will be executed first, following the first in first out (FIFO) principle.

3.2 Task 1: LOB Mid-Price Prediction

The first task is to predict the LOB Mid-Price Prediction, which is to compare the ability to predict absolute price values similar to non-financial datasets in previous works [31, 33–36]. The definition of time series prediction is given below and shown in Figure 2:

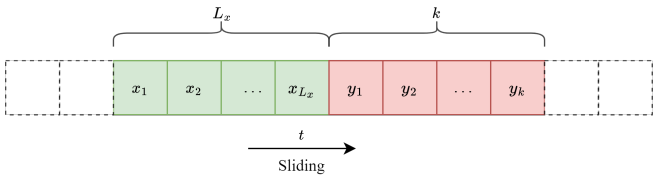

Figure 2: The illustration of time series prediction.

First, define a sliding window size L_x for the past data. The input data at each time step t is defined as:

$$X_t = \{x_1, x_2, \ldots, x_{L_x}\}_t \tag{1}$$

Then define a prediction window size k, where the goal is to predict the information in future $L_x + k$ steps. It will be the single-step prediction when $k = 1$ and be multi-horizon prediction when $k > 1$. Then the output at time step t is defined as:

$$Y_t = \{y_1, y_2, \ldots, y_k\}_t \tag{2}$$

The next step is to define the x_t and y_t in the input and output for mid-price prediction. Assume the market depth is 10. For a limit bid order at time t, the bid price is denoted as $p_{i,t}^{bid}$ and the volume is $v_{i,t}^{bid}$, where i is the market depth. Same for the limit ask order, ask price is $p_{i,t}^{ask}$ and volume is $v_{i,t}^{ask}$. Then the LOB data at time t is defined as:

$$x_t = \left[p_{i,t}^{ask}, v_{i,t}^{ask}, p_{i,t}^{bid}, v_{i,t}^{bid}\right]_{i=1}^{n=10} \in R^{40} \tag{3}$$

The past mid-price will be added to LOB data as input, and the mid-price is represented as:

$$p_t^{\text{mid}} = \frac{p_{1,t}^{ask} + p_{1,t}^{bid}}{2} \tag{4}$$

Finally, the x_t will be:

$$x_t = \left[p_{i,t}^{ask}, v_{i,t}^{ask}, p_{i,t}^{bid}, v_{i,t}^{bid}, p_t^{\text{mid}}\right]_{i=1}^{n=10} \in R^{41} \tag{5}$$

The target is to predict the future mid-price, so $y_t = p_t^{\text{mid}}$.

3.3 Task 2: LOB Mid-Price Difference Prediction

The second task is to predict the mid-price change, which is the the difference of two mid-prices in different time step. Trading strategies can be designed if the price change becomes negative or positive. The input of this task is the same as the mid-price prediction, as described in Equation 3. The target is to regress the future difference between current mid-price p_t^{mid} and the future mid-price $p_{t+\tau}^{\text{mid}}$:

$$d_{t+\tau} = p_{t+\tau}^{\text{mid}} - p_t^{\text{mid}} \tag{6}$$

Like the mid-price prediction, a prediction window size is defined as k, then the output of this task in each timestamp t is represented as:

$$Y_t = \{d_{t+1}, d_{t+2}, \ldots, d_{t+k}\}_t \tag{7}$$

3.4 Task 3: LOB Mid-Price Movement Prediction

According to Ruppert [5], the absolute price information is generally non-stationary, while the price change information, such as price difference and return and approximately stationary. The the mid-price difference is a difficult target for deep learning methods to predict because it is hard to extract meaningful pattern

from it, although it helps design trading strategies. An example of non-stationary and stationary sequence is shown in Figure 3.

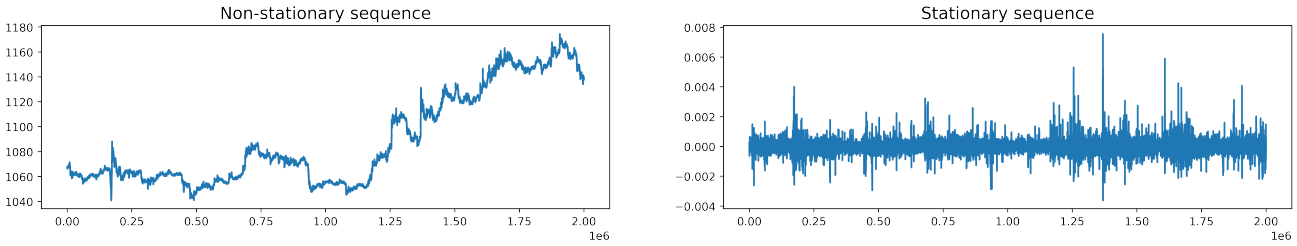

Figure 3: An example of non-stationary sequence vs stationary sequence.

For this reason, an easier classification task for predicting mid-price movement is introduced here. To train a model to predict mid-price movement, the first step is to create price movement labels for each timestamp. This study follows the smoothing labelling method from Tsantekidis et al. [38] and Zhang et al. [11]: Use m^- to represent the average of the last k mid-price and m^+ to represent the average of the next k mid-price:

$$m^-(t) = \frac{1}{k}\sum_{i=0}^{k} p_{t-k}^{mid} \tag{8}$$

$$m^+(t) = \frac{1}{k}\sum_{i=1}^{k} p_{t+k}^{mid} \tag{9}$$

k is set to $20, 30, 50, 100$ in this study following previous work of Zhang et al. [11].

And then, define a percentage change l_t to decide the price change direction.

$$l_t = \frac{m^+(t) - m^-(t)}{m^-(t)} \tag{10}$$

The label is dependent on the value of l_t. A threshold δ is set to decide the corresponding label. There are three labels for the price movement:

$$\text{label} = \begin{cases} 0(\text{ fall }), \text{ when } l_t > \delta \\ 1(\text{ stationary }), \text{ when } -\delta \leq l_t \leq \delta \\ 2(\text{ rise }), \text{ when } l_t < -\delta \end{cases} \tag{11}$$

An example of labelling for horizon 100 is shown in Figure 4. Assume there is an input in Equation 3 at

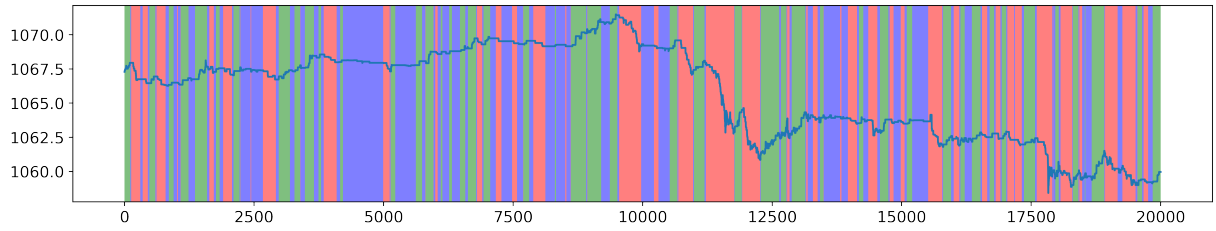

Figure 4: An example of labelling on horizon 100 on ETH-USDT dataset with fixed threshold δ. The green colour represents the rise signal. Purple represents the price is stationary and red colour means the price fall.

timestamp t, predicting mid-price movement is a one-step ahead prediction, which is to predict the mid-price movement in timestamp $t+1$.

4 Methodology

4.1 LSTM

LSTM was introduced by Hochreiter et al. [39] is one of the RNNs with structural adaptability in time series data input. Although the traditional RNN has the capacity to store data, but it suffers from the exploding gradient problem and vanishing gradient problem. Exploding/vanishing gradient means the gradient that is used to update the neural networks increases/decreases exponentially, which makes the neural network untrainable [40]. Therefore, RNN is not successful in studying long-time series relations [41]. LSTM neural network utilizes the coordination of three gates to keep long-term dependency and short-term memory. According to Gers et al. [42], the three gates that LSTM utilizes are 1) forget gate, 2) input gate 3) output gate. The structure of an LSTM cell is shown in Figure 5. The calculations of the LSTM are as follows [39]:

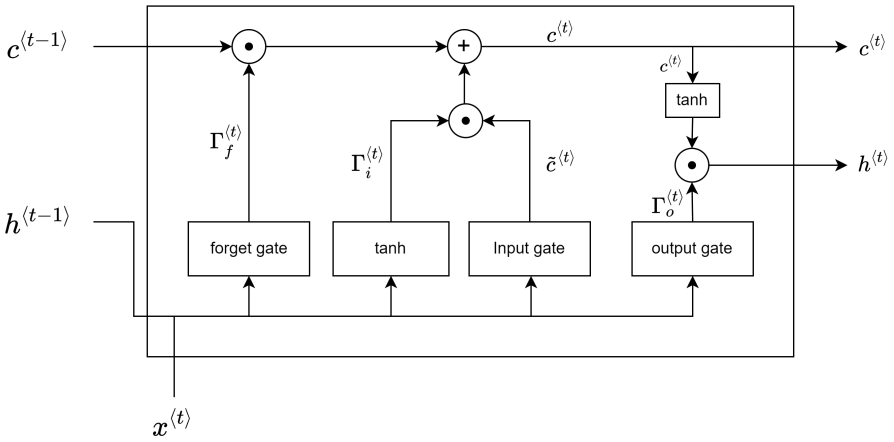

Figure 5: LSTM Structure based on Graves [43] and Fisher [19].

First, the LSTM need to decide to forget some information from the cell state, which is done by the forget gate. The forget gate has its sigmoid function σ. Then the function of forget gate is:

$$\Gamma_f^{\langle t\rangle} = \sigma\left(W_f\left[h^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_f\right) \tag{12}$$

Where the W_f is the weight of the last hidden state and input, b_f is the bias of the hidden state.

The next step is to design what information the neural cell should remember. To update the information, the input gate should coordinate with a $tanh$ layer containing a vector of new candidate values $\tilde{c}^{\langle t\rangle}$. The calculation is as follows:

$$\Gamma_i^{\langle t\rangle} = \sigma\left(W_i\left[h^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_i\right) \tag{13}$$

$$\tilde{c}^{\langle t\rangle} = \tanh\left(W_c\left[h^{\langle t-1\rangle}, x^{\langle t\rangle}\right] + b_c\right) \tag{14}$$

And then, with the previous steps of calculations, the cell state can be updated:

$$c^{\langle t \rangle} = \Gamma_f^{\langle t \rangle} \circ c^{\langle t-1 \rangle} + \Gamma_i^{\langle t \rangle} \circ \tilde{c}^{\langle t \rangle} \tag{15}$$

Lastly, calculate the result from the output gate to get the new hidden state:

$$\Gamma_o^{\langle t \rangle} = \sigma \left(W_o \left[h^{\langle t-1 \rangle}, x^{\langle t \rangle} \right] + b_o \right) \tag{16}$$

$$h^{\langle t-1 \rangle} = \Gamma_o^{\langle t \rangle} \circ \tanh \left(\tilde{c}^{\langle t \rangle} \right) \tag{17}$$

The output of LSTM can be every hidden state or the final hidden state, which depends on the application. In the implementation, this hidden state will be fed into a multi-layer perceptron (MLP), also known as feedforward-backwards propagation neural network (FFBPN). The output of this layer will pass through an activation function to generate the final output. Usually, the final hidden state will be utilized in financial time series prediction, which produces absolute price prediction or price movement prediction.

4.1.1 Alternative LSTM-based Models

Besides the canonical LSTM, three more LSTM based-models are chosen for comparison to Transformer-based models. They are DeepLOB [11], DeepLOB-Seq2Seq [12] and DeepLOB-Attention [12] created by Zhang et al. The architecture of these three models are shown in Figure 6 and 7. Here the structures of these three models are briefly explained:

DeepLOB [11] DeepLOB's architecture consists of three main components: Convolutional Blocks, an Inception Module and an LSTM layer.

A. Convolutional Blocks The LOB inputs mentioned in Equation 3 are fed into the convolutional blocks that contain multiple convolutional layers, where the first and second convolutional blocks are more important than the third one. The first convolutional block has a layer with filter size of (1×2) and stride of (1×2). At each order book level, this layer summarise the price and volume information $\left\{p^{(i)}, v^{(i)}\right\}$. For the second convolutional block, it has a layer with the same filter size and stride as the first one, but it is a feature mapping for the micro-price defined by [44]:

$$p^{\text{micro}} = I p_i^{ask} + (1 - I) p_i^{\text{bid}} \tag{18}$$

$$I = \frac{v_i^{bid}}{v_i^{ask} + v_i^{bid}} \tag{19}$$

I is called the imbalance. Then the last convolutional block integrates the feature information from the previous two layers. The whole convolutional blocks work as a feature extractor.

B. Inception Module the Inception Module employs the time series decomposition method. The input is decomposed by two 1×1 convolutions and one max-pooling layer into three lower-dimensional representations. Then these representations pass through convolution layers with 32 channels to be merged together.

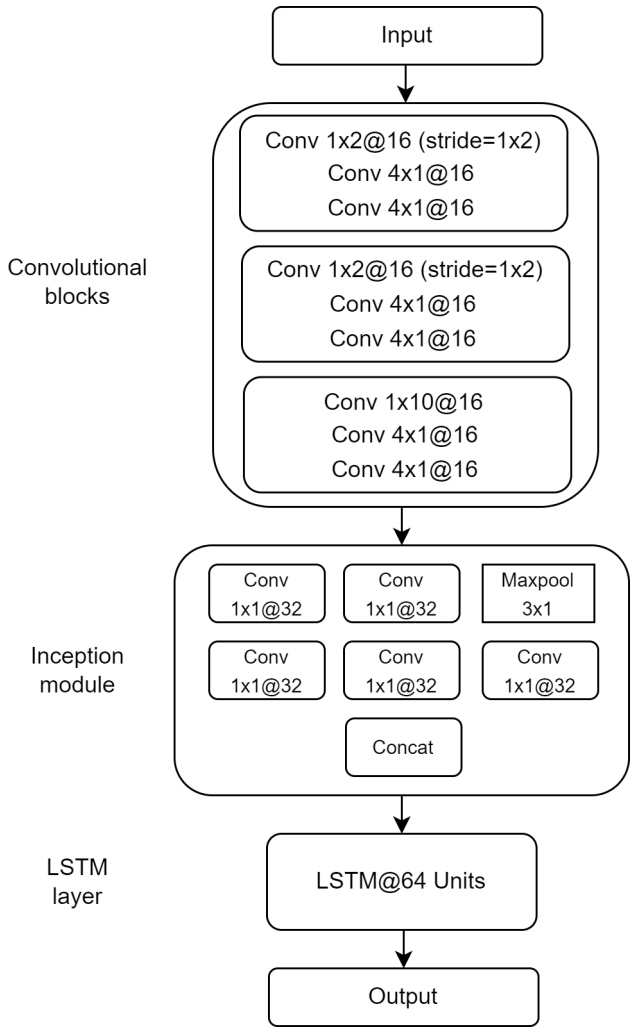

Figure 6: DeepLOB architecture sourced from Zhang et al. [11]. "Conv 1×2 @16" means there is 16 filters of size 1×2 in this convolutional layer.

This decomposition method improves the prediction accuracy.

C. LSTM Layer Finally, the extracted features are inputted into one LSTM layer to capture the underlying pattern and dependencies. The last output layer can be a SoftMax layer or a linear layer, which depends on the specific tasks.

DeepLOB-Seq2Seq [12] To generate multi-horizon predictions, Zhang et al. developed DeepLOB-Seq2Seq. The main idea is to feed the output of the Inception Module into a Seq2Seq architecture to do iterated multi-step (IMS) prediction. Seq2Seq [45] architecture contains encoder and decoder constructed by recurrent neural network (RNN). Assume the sequence input is $X_T = (x_1, x_2, \cdots, x_T)$, the encoder will output a

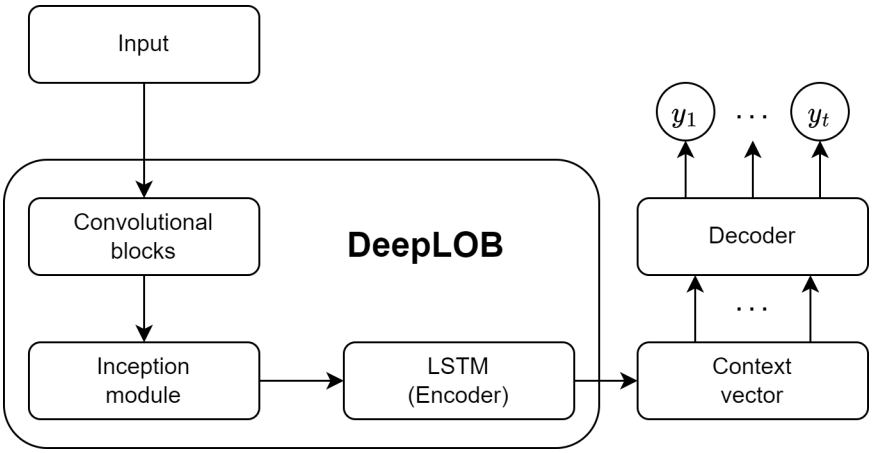

Figure 7: DeepLOB-Seq2Seq and DeepLOB-Attention architecture sourced from Zhang et al. [12].

hidden state at each timestamp t:

$$h_t = f\left(h_{t-1}, x_t\right) \tag{20}$$

After obtaining the hidden states from the encoder, a context vector c has to be constructed from these hidden states. The last hidden state or the mean of all hidden states can be taken as a context vector. Context vector work as a "bridge" between the encoder and decoder, where the context vector is utilized to initialize the decoder, and the hidden state output of the decoder at each timestamp is:

$$d_t = f\left(d_{t-1}, y_{t-1}, c\right) \tag{21}$$

Then the distribution of the output y_t is:

$$P\left(y_t \mid y_{<t}, c\right) = g\left(d_t, c\right) \tag{22}$$

where the output of the decoder not only depends on the previous true value as input but is also conditioned on the context vector.

DeepLOB-Attention [12] With the same idea as the DeepLOB-Seq2Seq model, the difference of the DeepLOB-Attention model is changing the Seq2Seq architecture into Attention. The attention model [46] constructs the context vector differently instead of using the last hidden state or the mean of all hidden states. Same as Seq2Seq model, the encoder outputs hidden state h_t and decoder outputs hidden state d_t. The first step is to compute a similarity score between the hidden state d_t of the decoder and each encoder state h_i, where the similarity score is usually calculated by dot product:

$$s_i = h_i^T d_t \tag{23}$$

And then, normalize the similarity scores to obtain a weight distribution by softmax:

$$\{\alpha_1, \alpha_2, \ldots, \alpha_S\} = \operatorname{softmax}\left(\{s_1, s_2, \ldots, s_S\}\right) \tag{24}$$

Finally, generate the context vector from the attention weights:

$$c_t = \sum_{i=1}^{S} \alpha_i h_i \tag{25}$$

After that, the process of producing the output y_t is the same as the Seq2Seq model.

4.2 Transformer

The Transformer [9] has an encoder-decoder structure that relies on the Self-attention mechanism without relying on CNN and RNN. This architecture allows the Transformer to process the long sequence and has no problem with the vanishing gradient, which means it can model the dependency regardless of the length of the input/output. A series of components forms the Transformer: Multi-head Self-attention, positional-wise feed-forward neural network, layer-normalization, and residual connection. According to Vaswani et al. [9] and Farsani et al. [47], the architecture of the Transformer for financial time series prediction is shown in Fig.8. There is a slight difference between this transformer and the vanilla one used for NLP tasks. The word

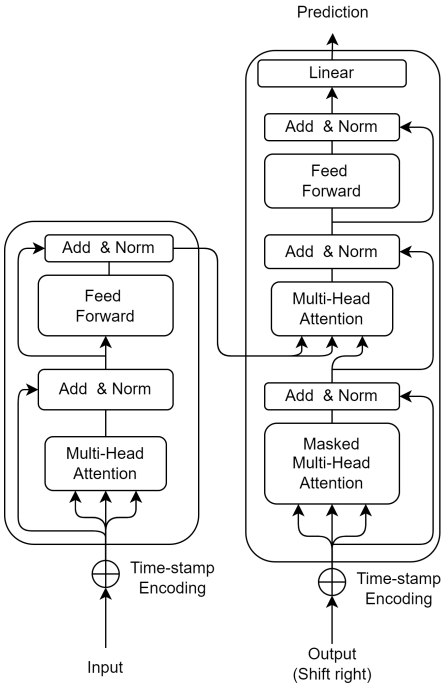

Figure 8: Transformer Structure based on Vaswani et al. [9] and Farsani et al. [47].

embedding process is omitted, and the financial time series is fed into the transformer using time-stamp encoding. The details of time-stamp encoding will be explained in Section 4.2.2.

4.2.1 Multi-head Self-attention Mechanism

Self-attention is a mechanism for finding the relevant vector in a sequence. The target of Self-attention is to compute the attention score and then extract information based on the score. According to Vaswani et al. [9], the calculations of scaled dot-production Self-attention are as follows: First, to compute the attention score, multiply the input vector I with different learnable weight matrices W^q and W^k to obtain the query and key:

$$Q = W^q I \tag{26}$$

$$K = W^k I \tag{27}$$

The next step is to calculate the attention matrix, where each element in it is an attention score:

$$A = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \tag{28}$$

Where d_k is the dimension of query and key.

Lastly, multiply the attention matrix with the values, and the final output can be obtained:

$$\text{Attention}\ (Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{29}$$

The output is the weighted sum of the relevance of different vectors in the sequence. In the implementation of the transformer, Multi-head Self-attention is used, which is beneficial for finding different types of relevance.

4.2.2 Learnable Time-stamp Encoding

Different from the fixed sinusoid encoding used in the vanilla transformer [9], timestamp encoding from the Informer [33] is more informative for time series data and a similar method is applied in Autoformer [34] and FEDformer [36]. The timestamp encoding method uses learnable embedding layers to produce positional information to add to the sequence, where timestamp information like a minute, hour, week, year and extra time stamps like event or holiday can be incorporated. To obtain the time-step encoding, the first step is to calculate fixed sinusoid encoding. Assume the input is $X_t = \left\{x_1, x_2, \ldots, x_{L_x} \mid x_i \in R^{d_x}\right\}_t$ at timestamp t , where L_x is the sliding window size and d_x is the model dimensionality, then the encoding is calculated as follows:

$$PE_{pos,2i} = \sin\left(\frac{pos}{(2L_x)^{\frac{2i}{d}}}\right) \tag{30}$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{(2L_x)^{\frac{2i}{d}}}\right) \tag{31}$$

And then, project the original input x_i^t into the model dimensionality vector u_i^t using convolutional filters. The next step is to use a learnable embedding layer SE_{pos} to incorporate the timestamp information. The structure of the timestamp embedding is shown in Fig.9.

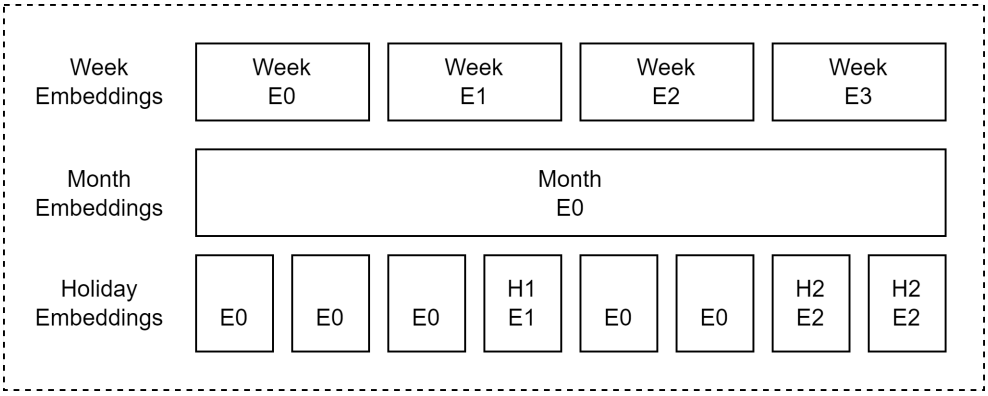

Figure 9: Time-stamp embeddings based on Zhou et al. [33].

Finally, add up all the calculation results above to get the final encoding:

$$X_{feed[i]}^{t} = \alpha u_i^t + PE_{(L_x \times (t-1)+i)} + \sum_p \left[SE_{L_x(t-1)+i}\right]_p \tag{32}$$

Where $i \in \{1, \ldots, L_x\}$ and α is the parameter balancing the ratio of scalar projection and embeddings.

4.2.3 Alternative Transformer-based Models

As mentioned in Section 2, several new Transformer-based models [31–34, 36] are dedicated for long time series forecasting. However, they have not been tested on financial time series data. In this study, they are chosen as the alternative models to compare with the vanilla Transformer and LSTM. These models and their relationships are briefly summarized as follows (see Figure 10):

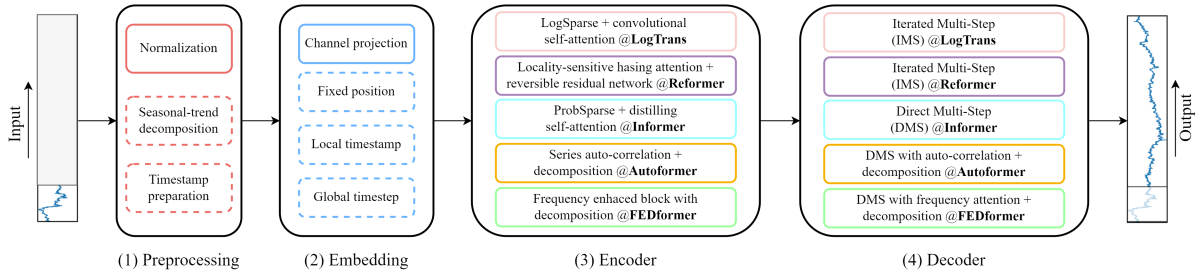

Figure 10: Architecture designs for alternative Transformer models sourced from Zeng et al.[48]. In step (1) and (2), the operations inside the solid box are required, while those in the dashed box can be applied optionally.

LogTrans Li et al. [31] put forward LogTrans with convolutional Self-attention generating queries and keys in the Self-attention layer. This work makes the convolutional layer widely used in the attention module in the later studies [33, 34, 36]. It uses a Logsparse mask to reduce the time complexity from $O(L^2)$ to $O(L \log(L))$.

Reformer Reformer [32] changes the time complexity of the Transformer to $O(L\log(L))$ as well by using sensitive hashing instead of dot-product in the calculate of attention. It also replaces the residual connection in the vanilla Transformer with a reversible residual connection, which makes the Transformer more efficient.

Informer Although Reformer and LogTrans reduced the time complexity to $O(L\log(L))$, the memory consumption is still the same, so the efficiency gain is not high. Also, LogTrans and Reformer use iterated multi-step prediction (IMS), generating a single prediction in each timestamp and using that prediction iteratively to obtain multi-step prediction [49], which suffers from error accumulation. Informer [33] proposed by Zhou et al. employs a ProbSparse Self-attention mechanism to achieve $O(L\log(L))$ time and memory complexity. They propose an innovative generative style decoder to make direct multi-step(DMS) prediction, which is to generate the multi-step prediction at once in one forward procedure [50]. This method speed up the long-term forecast compared to the LogTrans and Reformer. The DMS forecast method and learnable timestamp encoding are applied in the Autoformer [34] and FEDformer [36].

Autoformer The optimization of Transformer on time series prediction is a trade-off between efficiency and information utilization. Depending on the structure of Informer, Autoformer [34] reduced the time complexity with an Auto-Correlation mechanism rather than making Self-attention sparse in LogTrans and Informer, which can preserve the information well and measure the sub-series dependency. Time series decomposition is a method commonly used in time series analysis to deconstruct the time series into several components [51, 52]. The underlying temporal pattern can be revealed from these components to make the time series more predictable [53]. Autoformer first embeds the time series decomposition as an inner neural block to derive the trend-cyclical component from the input sequence and the seasonal component from the difference between the trend-cyclical component and the input sequence. This new decomposition architecture can deconstruct time series to use the series periodicity to update attention.

FEDformer Based on the decomposition architecture of Autoformer, Zhou et al. [36] builds the FEDformer to use Fourier transform and Wavelet transform, which are in the frequency domain as a new decomposition method to reach linear complexity $O(L)$. .

5 Experimentation Result and Evaluation

5.1 Comparison of LOB Mid-Price Prediction

The first task to compare transformer versus LSTM is the mid-price prediction. In this task, predicting the absolute value of the mid-price is similar to the previous work's [31, 33–36] experiment on non-financial datasets.

5.1.1 Experiment Setting for LOB Mid-Price Prediction

Dataset All the experiments are based on cryptocurrency LOB data, which are collected in real-time from Binance Exchange using cryptofeed [54] WebSocket API and saved to the database using kdb+tick triplet [55]. In this experiment, one-day LOB data of product BTC-USDT (Bitcoin-U.S. dollar tether) on $2022.07.15.$ containing 863397 ticks is utilized. The time interval between each ticks is not evenly spaced. The time interval is 0.1 second on average. The first 70% data is used to construct the training set, and the rest 10% and 20% of data are used for validation and testing. The reason why only using one day of LOB data is that it is enough to train for a task prediction for absolute value without overfitting according to previous works' experiments on non-financial datasets.

Models For the comparison purpose, I choose canonical LSTM and vanilla Transformers along with four Transformer-based models: FEDformer [36], Autoformer [34], Informer [33] and Reformer [32]. For the implementation of Transformer-based models, they are taken from open-source code repositories [56, 57]. The implementation of vanilla Transformer [9], Reformer [32], Informer [33] and Autoformer [34] are from the Autoformer repository [56]. The implementation of FEDformer [36] is from its own repository [57].

Training setting The dataset is normalized by the z-score normalization method. The validation set and the test set are normalized by the mean and standard deviation of the training set. All the models are trained for 10 epochs using the Adaptive Momentum Estimation optimizer and L2 loss with early stopping. The batch size is 32, and the initial learning rate is 1e-4. All models are implemented by Pytorch [58] and trained on a single NVIDIA RTX A5000 GPU with 24 GB memory.

5.1.2 Result and Analysis for LOB Mid-Price Prediction

Models	FEDformer		Autoformer		Informer		Reformer		Transformer		LSTM	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
96	**0.0793**	**0.179**	0.0926	0.201	1.411	0.543	2.186	0.619	2.836	0.696	0.104	0.204
192	**0.155**	**0.257**	0.176	0.279	1.782	0.749	1.842	0.824	2.799	0.832	0.195	0.287
336	**0.274**	**0.348**	0.319	0.376	2.080	0.830	9.218	1.947	1.456	0.665	0.315	0.369
720	**0.608**	**0.514**	0.643	0.539	2.808	1.093	72.57	6.824	4.306	1.297	0.771	0.587

Table 1: Mid price prediction result with different prediction lengths $k \in \{96, 192, 336, 720\}$ in test set. The input window size is set to 96 (MSE's unit is in 10^{-2} and MAE's unit is in 10^{-1}).

Quantitative result To evaluate the performance of different models, following the previous works [31, 33–36], the performance metrics consist of Mean Square Error (MSE) and Mean Absolute Error (MAE), representing the prediction error. Lower MSE or MAE indicates the model has less prediction error. MSE and MAE are calculated by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \widehat{Y}_i\right)^2 \tag{33}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_i - \hat{Y}_i\right| \tag{34}$$

where Y_i is the true value and $\widehat{Y}_i$ is the predicted value. n is the number of ticks.

The results of all models are shown in Table 1. From the table, these outcomes can be summarized:

1. Both FEDformer and Autoformer outperform LSTM and FEDformer has the best performance in all the prediction lengths. FEDformer and Autoformer give a large increase in performance in terms of MSE and MAE compared to LSTM. For FEDformer, it gives $24\%(0.104 \rightarrow 0.0793)$ MSE reduction on 96 prediction length and $21\%(0.771 \rightarrow 0.608)$ on 336 prediction length. For Autoformer, it gives $11\%(0.104 \rightarrow 0.0926)$ MSE reduction on 96 prediction length and $16\%(0.771 \rightarrow 0.643)$ MSE reduction on 336 prediction length. These results show that both Autoformer and FEDformer perform well in terms of MSE and MAE because of their low error and long-term robustness.

2. Although FEDformer and Autoformer's MSE and MAE are low on this task, LSTM is relatively not bad on mid-price prediction. LSTM outperforms the other three models: Informer, Reformer, and vanilla Transformer, which indicates that LSTM is robust in handling LOB data, while transformer-based models require lots of modification to perform well.

3. Vanilla Transformer model has worse performance on prediction lengths 96 and 192 and Reformer has worse performance on prediction lengths 336 and 720 because they suffered from error accumulation during the IMS prediction process. Informer's worse performance than LSTM is mainly due to its sparse version of attention, leading to information loss on the time series.

Qualitative results The prediction results of compared models on all the prediction horizons are shown in Figure 11. When the prediction horizon is 96, Autoformer and Reformer are able to generate a proper trend for the future mid-price, while other models generate almost a flat line as predictions. On the prediction horizon of 192, almost all the models' predictions plateau except the Reformer, but Reformer's result becomes more stochastic than the prediction horizon of 96. For larger prediction horizons 336 and 720, all the models can hardly predict a proper trend and Reformer's result in not plotted because it becomes too stochastic.

Based on the qualitative results above, although Autoformer and FEDformer outperform LSTM in terms of MSE and MAE, their actual prediction performance is far more inadequate for high-frequency trading. The analysis from Figure 11 is just "eyeballing" whether the model is good. In this case, another formal metric

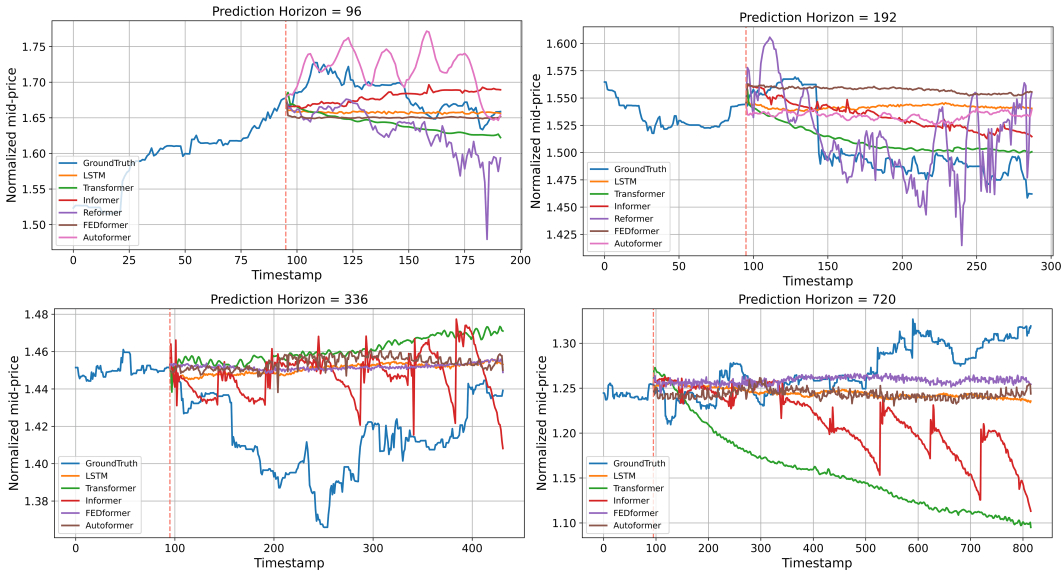

Figure 11: Illustration of normalized forecasting outputs with 96 input window size and $\{96, 192, 336, 720\}$ prediction lengths. Each timestamp is one tick. Reformer's results are not plotted in the lower two panels for better visualisation.

Models	Autoformer	FEDformer	Informer	Reformer	LSTM	Transformer
96	-0.753	-0.237	-43.811	-69.080	-0.946	-87.899
192	-0.596	-0.205	-25.281	-26.792	-0.644	-43.368
336	-1.032	-0.364	-20.123	-63.252	-0.414	-13.035
720	-0.521	-0.189	-7.760	-137.322	-0.589	-16.314

Table 2: Average of out of sample R^2 result with different prediction lengths $k \in \{96, 192, 336, 720\}$.

out of sample R^2 is added here to judge the prediction quality. According to Lewis-Beck [59], R^2 determine how well the prediction result Y can be explained by the input X, and higher R^2 indicates the model has a better fit for the predicted value. Out of sample R^2 is defined by:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} \left(Y_i - \widehat{Y}_i\right)^2}{\sum_{i=1}^{n} \left(Y_i - \bar{Y}\right)} \tag{35}$$

where $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$; $\widehat{Y}_i$ is the predicted value and Y_i is the ground truth.

Here the out-of-sample R^2 is calculated based on the price difference. Please note that the price difference here differs from the one mentioned in Section 3.3. The price difference here is calculated from the absolute price prediction result, while the one in Section 3.3 is the direct prediction target. The result of R^2 is shown in Table 2.

From the table, all the out-of-sample R^2 values are negative for all models. However, according to Lewis-Beck [59], R^2 at least needs to be larger than zero to indicate that input X can explain output Y. This indicates that predictions for absolute mid-price are useless for trading purposes. The metrics of MSE and MAE obscure the real quality of prediction results, highlighting the importance of R^2 calculated based

on the price difference.

To sum up, although Autoformer and FEDformer have lower MSE and MAE than LSTM, their prediction result is not helpful and practical for trading. The more sensible way is to directly use the price difference as the prediction target.

5.2 Comparison of LOB Mid-Price Diff Prediction

Both transformer-based models and LSTM are unable to generate the satisfactory result on the mid-price prediction, so this task turns to the mid-price difference prediction. The mid-price difference is a useful alpha-term structure in trading for traders and market makers [14].

5.2.1 Experiment Setting for LOB Mid-Price Diff Prediction

Dataset The dataset for this experiment is collected the same way as the last experiment, but the dataset size becomes larger to avoid overfitting. In this experiment, four days of LOB data for the product BTC-USDT from 2022.07.03 (inclusive) to 2022.07.06 (inclusive) is used, containing 3432211 ticks. The first 80% of data is used as a training set, and the rest 20% is split in half for validation and testing.
Models and Limitations Five models are being compared in this experiment: canonical LSTM [39], vanilla transformer [9], CNN-LSTM (DeepLOB [11] model used for regression), Informer [33] and Reformer [32]. State-of-the-art FEDformer and Autoformer are not compared in this task because their time decomposition structure is limited in handling price difference series. As mentioned in Section 3.4, the price difference is approximately stationary. The time decomposition method is useful for the non-stationary time series, such as the mid-price series, where it can extract meaningful patterns. In contrast, the price difference series is approximately stationary, and little meaningful information can be extracted from it. Therefore, FEDformer and Autoformer can only produce a poor result in this task and will not be compared below.
Training settings the training setting is the same as the last experiment.

5.2.2 Result and analysis for LOB Mid-Price Diff Prediction

Following the previous works [14], out of sample R^2 is the evaluation metric for this task. The performance of all the models is shown in Figure 12. The canonical LSTM achieves the best performance among all models, which reaches the highest R^2 around 11.5% in forecast length 5 to 15. For CNN-LSTM, it has comparable performance to LSTM. On the other hand, Informer, Reformer and Transformer have worse R^2 than LSTM, but their R^2 trend is similar. In short, for the price difference prediction task, LSTM-based models is more stable and more robust than Transformer-based models. This result is in expectation because Reformer, Informer and Transformer already have worse performance than LSTM in mid-price prediction task because of their shortcomings. At the same time, the state-of-art FEDformer and Autoformer cannot be

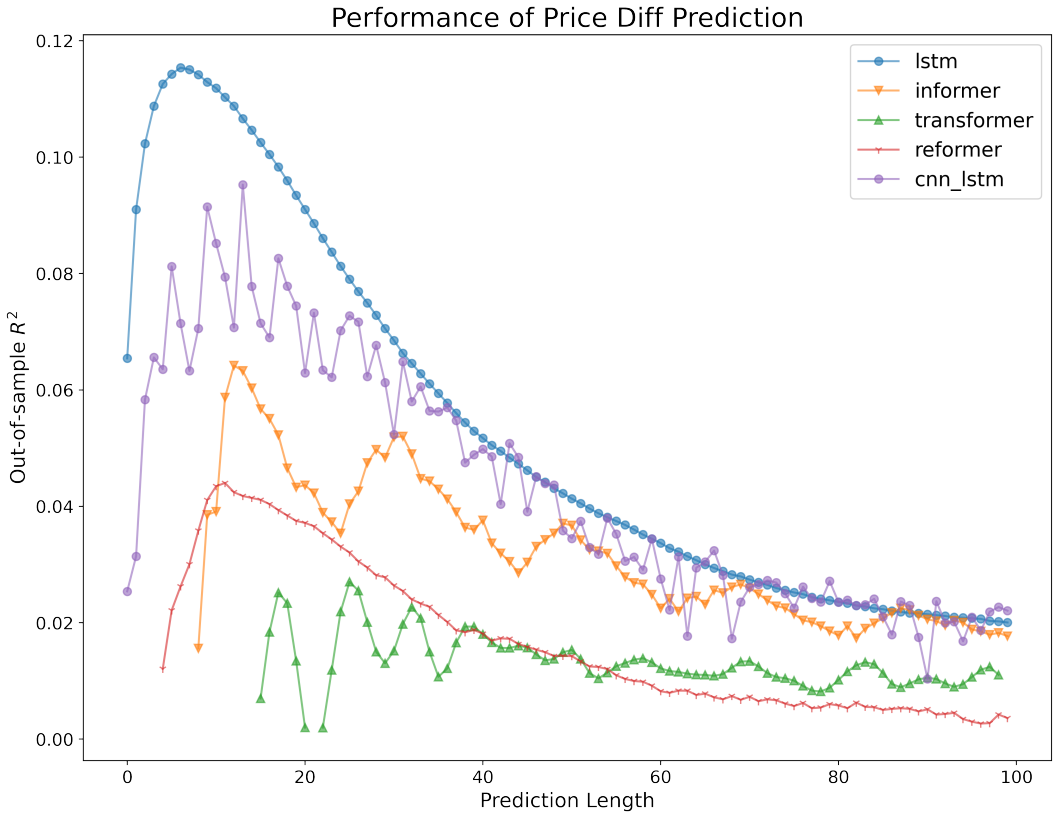

Figure 12: Performance of price difference prediction with input window size 100 and prediction length 100. Negative data points are not plotted for ease of visualization.

applied because of their limitation. In order to let these state-of-the-art transformer-based models make a meaningful prediction, a new structure is designed in the next part, and it is applied to the price movement prediction task.

5.3 Comparison of LOB Mid-Price Movement Prediction

5.3.1 Innovative Architecture on Transformer-based Methods

The alternative Transformer-based models mentioned in Section 4.2.3 mainly focus on the long time series forecasting problem, which is a regression task. For the LOB data, it is easy to adapt these models for the mid-price prediction and mid-price difference prediction because both are regression problems. For the mid-price movement prediction task, the model needs to produce a classification result for the future, and there are few existing Transformer models specialized available for this task. In contrast, most of the Transformer-based models are designed for classification tasks without forecasting, such as sentiment analysis, spam detection and pos-tagging in NLP. In this case, adapting the existing Transformer-based models to do price movement forecasting is necessary. I first adapt Transformer-based models in price movement forecasting and want to facilitate transformer development in this specific task. The new architecture of the transformer-based

model is shown in Figure 13. The details are explained as follows:

Predicting the next mid-price movement based on the past price and volume information is an one step ahead of prediction. A straightforward method to adapt the transformer-based model is to pass the next predicted

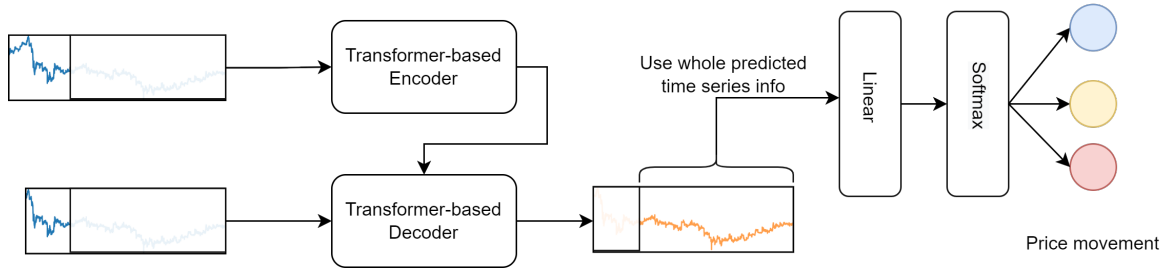

Figure 13: New architecture of transformer-based model for LOB mid-price movement prediction.

mid-price into a softmax activation. However, this method performs poorly because it only considers the past mid-price information and ignores future ones. It is worth noting that in the labelling process in Section 3.4, previous and next k mid-price information are utilized. In this case, I adapt the existing transformer-based models to feed the whole predicted mid-price sequence into a linear layer and finally pass through a softmax activation function to generate price movement output. This adaptation will benefit those transformer-based models using the DMS forecasting method because they have fewer errors in the long-time series prediction process.

5.3.2 DLSTM: Innovation on LSTM-based Methods

Inspired by the Dlinear model [48] and Autoformer, combining the merits of time decomposition with the LSTM, a new model named DLSTM is designed. DLSTM is designed based on these three observations: Firstly, the time series decomposition method is capable of increasing the performance, especially embedding this process by neural blocks in previous works [11, 34, 36]. Secondly, LSTM is a robust and simple model for multiple forecasting tasks. Thirdly, Dlinear beats other Transformer-based models in some long time series forecasting tasks thanks to the time series decomposition method and DMS prediction. However, predicting price movement is one step ahead prediction, where the model will not suffer from the error accumulation effect. In this case, it is sensible to replace linear with LSTM, because LSTM is a model well-known better than linear in handing time series.

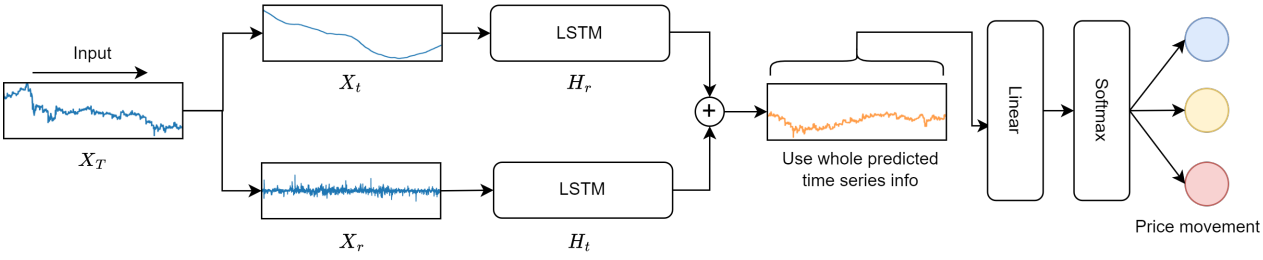

Figure 14: Architecture of DLSTM

The architecture of DLSTM is shown in Figure 14. The main difference between Dlinear and DLSTM is that the LSTM layers replace the Linear layer. According to the time decomposition method introduced in Autoformer [34], in the prediction process, assume there is a time series $X_T = (x_1, x_2, \ldots, x_T)$, first decompose it into Trend series by the moving average:

$$X_t = AvgPool(Padding(X_T)) \tag{36}$$

where $AvgPool(\cdot)$ is the average pooling operation and the $Padding(\cdot)$ is to fix the input length.
And then the Remainder series is calculated by $X_r = X_T - X_t$. After that, these two series are inputted into two LSTM layers. Finally, the hidden states H_t and H_r produced by two LSTM layers will be added together and then pass through a linear and softmax activation to generate the final price movement result.

5.3.3 Setting for LOB Mid-Price Movement Prediction

Dataset In this experiment, the largest dataset among three tasks is utilized to avoid over-fitting and test the model's robustness. The whole dataset contains 12 days of LOB data of product ETH-USDT (Ethereum-U.S. dollar tether) from 2022.07.03 (inclusive) to 2022.07.14 (inclusive), containing 10255144 ticks. The training and testing data are taken from the first six days and the last three days, and the left data are used for validation. The test set is also used for the simple trading simulation.
Models Thanks to the innovative structure mentioned in Section 5.3.1, most of the transformer-based models can be adapted and applied in this task for comparison, which are: Vanilla Transformer [9], Reformer [32], Informer [33], Autoformer [34], FEDformer [36]. On the other hand, all the LSTM-based models are compared in this task as well, which are: canonical LSTM [39], DLSTM, DeepLOB [11], DeepLOB-Seq2Seq [12], DeepLOB-Attention [12]. Besides these models, a simple MLP model is built as a baseline. The implementation of the Transformer-based models are based on the code repository mentioned above. The implementation of DeepLOB [11], DeepLOB-Seq2Seq [12], DeepLOB-Attention [12] are based on two repositories [60, 61]. For the DLSTM, it is inspired by code of Dlinear[48] model from its repository [62].
Training settings The batch size for training is set to 64 and the loss function is changed to Crossentropy loss. Other training settings are the same as the last experiment.

5.3.4 Result and analysis for LOB Mid-Price Movement Prediction

The performance of models is evaluated by classification metrics: accuracy and the mean of precision, recall and F1-score. Result are shown in Table 3 and Table 4.

A few outcomes can be observed from the result:
1. DLSTM outperforms all the previous LSTM-based and Transformer-based models. It achieves the highest accuracy, precision, recall and F1 score in all the prediction horizons. This result shows that the time series decomposition structure originating from Autoformer can effectively handle time series, especially when combined with a simple LSTM model. DLSTM is making one step ahead prediction for the mid-price

Model	Accuracy	Precision	Recall	F1
Prediction Horizon k = 20				
MLP	61.58	61.70	61.58	61.47
LSTM	62.77	62.91	62.77	62.78
DeepLOB	70.29	70.58	70.30	70.24
DeepLOB-Seq2Seq	70.40	70.79	70.42	70.37
DeepLOB-Attention	70.04	70.26	70.03	70.01
Autoformer	68.89	68.99	68.89	68.91
FEDformer	65.37	65.70	65.37	65.20
Informer	68.71	68.82	68.72	68.71
Reformer	68.01	68.26	68.00	67.95
Transformer	67.80	67.99	67.81	67.77
DLSTM	**73.10**	**74.01**	**73.11**	**73.11**
Prediction Horizon k = 30				
MLP	59.19	59.30	58.70	58.48
LSTM	60.64	60.47	60.45	60.45
DeepLOB	67.23	67.26	67.17	67.15
DeepLOB-Seq2Seq	67.56	67.73	67.53	67.49
DeepLOB-Attention	67.21	67.39	66.98	66.96
Autoformer	67.93	67.86	67.77	67.77
FEDformer	66.57	66.44	66.05	65.83
Informer	65.41	65.33	65.14	65.13
Reformer	64.28	64.31	64.08	64.06
Transformer	64.25	64.16	64.13	64.13
DLSTM	**70.61**	**70.83**	**70.63**	**70.59**

Table 3: Experiment results of Mid Price Movement for prediction horizons 20 and 30. **Red Bold** represents the best result and blue underline represents the second best result.

movement, so it will not suffer from error accumulation from the DMS prediction process.

2. DeepLOB-Attention model has the second best result in horizon 50 and 100 (excluding accuracy). DeepLOB-Seq2Seq has the second best result for prediction horizon 20. This result indicates that the encode-decoder structure and attention mechanism can contribute to the prediction performance because the autoregressive process can correlate the mid-price movement from different prediction horizons.

3. The DeepLOB-Attention and DeepLOB-Seq2Seq performance is comparable to DeepLOB but better than DeepLOB, especially in the long prediction horizon. This result accords with the result in the previous paper [12], which proves the correctness of the result.

4. The Autoformer gets the second-best result in prediction horizon 30. Although it is not the best model, it still means that Autoformer is usable for the time series prediction, and its time decomposition structure is adequate. The shortcoming of Autoformer and the latest FEDformer is that they are huge models compared to LSTM, and they need to be fine-tuned to work well in a specific task. In contrast, LSTM-based models' sizes are much smaller and do not need much hyper-parameters tuning. More analysis of the efficiency will be discussed in the Section 5.3.7.

Model	Accuracy	Precision	Recall	F1
Prediction Horizon k = 50				
MLP	55.65	55.71	55.62	54.98
LSTM	62.77	62.91	62.77	62.78
DeepLOB	63.32	63.69	63.32	63.37
DeepLOB-Seq2Seq	63.62	64.04	63.61	63.59
DeepLOB-Attention	64.05	64.19	64.04	63.94
Autoformer	60.17	60.64	60.12	58.40
FEDformer	63.46	63.44	63.42	62.52
Informer	61.76	61.64	61.74	61.55
Reformer	60.43	60.79	60.42	60.37
Transformer	59.51	59.78	59.51	59.46
DLSTM	**67.45**	**67.96**	**67.45**	**67.59**
Prediction Horizon k = 100				
MLP	57.03	56.03	56.36	56.01
LSTM	53.49	52.83	52.82	52.36
DeepLOB	58.12	58.50	57.92	57.86
DeepLOB-Seq2Seq	58.30	58.43	57.93	57.77
DeepLOB-Attention	59.16	58.59	58.65	58.50
Autoformer	59.18	58.34	58.40	57.83
FEDformer	57.97	56.97	56.62	54.14
Informer	56.11	56.15	55.85	55.81
Reformer	54.92	54.47	54.53	54.47
Transformer	55.42	55.04	54.92	54.72
DLSTM	**63.73**	**63.02**	**63.18**	**63.05**

Table 4: Experiment results of Mid Price Movement for prediction horizons 50 and 100.**Red Bold** represents the best result and blue underline represents the second best result.

To summarize, combing the results of this task and previous tasks, LSTM-based models generally have their advantage in financial time series for their robustness and good compatibility. Although the Transformer-based model is large and complicated to tune and requires a long training time, they are still usable for the forecasting task. Furthermore, the research of Transformer-based method in time series prediction is meaningful because the time decomposition method from Autoformer contributes back to the original LSTM model.

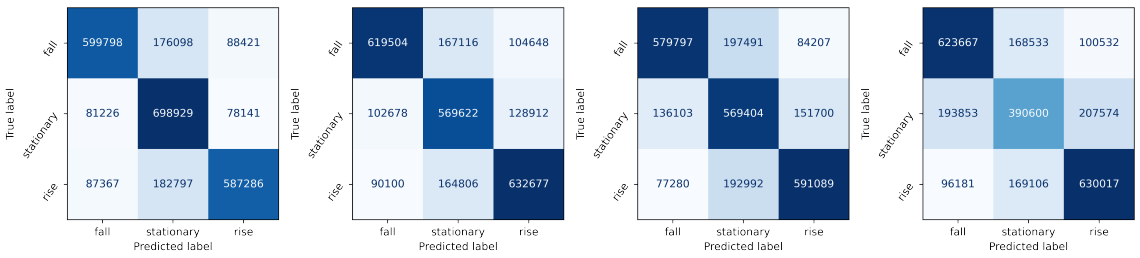

Figure 15: Confusion matrix of DLSTM model with prediction horizon 20, 30, 50, 100.

5.3.5 Simple Trading Simulation without transaction cost

In order to show the models and their predictions are practical and useful in trading, a simple trading simulation (backtesting) is designed. Three models with good classification metrics performance are chosen for comparison: DLSTM, DeepLOB [11], Autoformer [34]. Canonical LSTM [39] and Vanilla Transformer [9] are used as baselines. The three-day test set is used for this trading simulation. To make a fair comparison among models, the trading simulation follows the simple setup in the previous work [11]: The number of shares pre-trade μ (volume) is set to one. At each timestamp, the model will predict the price movement (0: fall, 1: stationary, 2: rise) as a trading signal. When the prediction is 2, enter the long position, and the position is held until it encounters 0. The same rule is applied to the short position when the prediction is 0, and only one direction of position can exist in this simulation trading. A delay is set between the prediction and the order execution to simulate the high-frequency trading latency. For example, assume the model generates a prediction 2 at time t, μ shares will be bought at time $t + 5$.

Forecast Horizon	Prediction Horizon = 20		Prediction Horizon = 30		Prediction Horizon =50		Prediction Horizon=100	
Model	CPR	SR	CPR	SR	CPR	SR	CPR	SR
LSTM	**15.396**	51.489	12.458	41.411	**8.484**	**28.817**	**4.914**	**20.941**
DLSTM	14.966	46.949	12.634	37.432	6.194	22.027	3.215	16.346
DeepLOB	13.859	56.094	**12.789**	**42.567**	5.726	21.014	2.646	14.992
Transformer	14.553	**59.995**	12.737	41.044	6.896	28.147	2.859	16.981
Autoformer	9.942	32.688	8.617	30.576	8.214	25.882	3.620	17.765

Table 5: Cumulative price returns and annualized sharpe ratio of different models.

Several assumptions are made for the simulation trading:

1) Since the trading product is in cryptocurrency exchange, the trading volume is considerable sufficient in the market, which means the simulated trades will not have a market impact.

2) The focus of this part of the experiment is to show the practicality of the prediction result and make relative comparisons among models instead of inventing a fully developed high-frequency trading strategy. Industrial HFT trading strategies usually require the combination of different prediction signals and precise entry exit rules [11]. For simplicity, the order is assumed to be executed at the mid-price without transaction cost.

As displayed in Table 5 and Figure 16, each model's profitability is presented. The performance of simulated trading is evaluated by cumulative price return (CPR) and the Annualized Sharpe Ratio (SR). The CPR is formulated by:

$$CPR = \sum_{1}^{t} s * \mu * \left(p_{\text{mid}}^{\text{holding },t} - p_{\text{mid}}^{\text{settlement },t} \right) \tag{37}$$

where s is the trading position, which is 1 for long position and -1 for short position. μ is the number of shares.

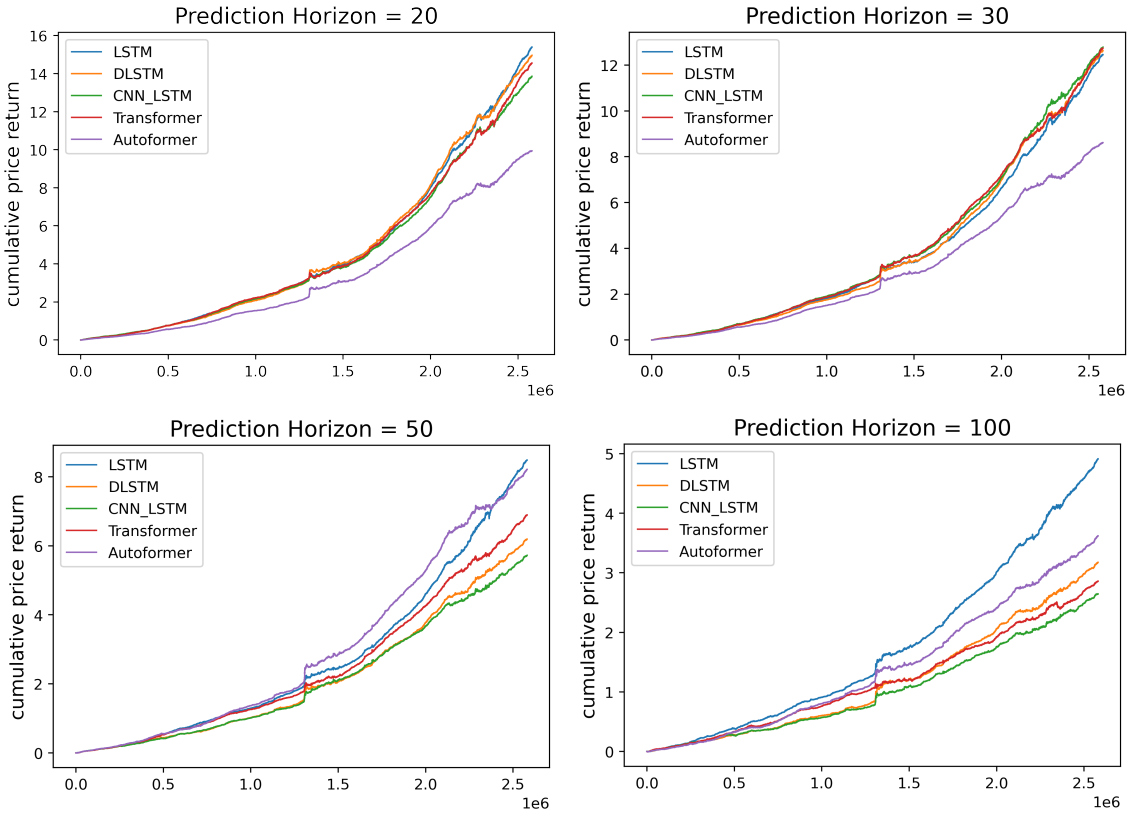

Figure 16: Cumulative return curve for different models in trading simulation.

And the Sharpe Ratio is calculated by:

$$SR = \sqrt{365} \times \frac{\text{Average } (\text{ daily } CPR)}{\text{StandardDeviation } (\text{ daily } CPR)} \tag{38}$$

The value of annualized SR is enormous because the assumptions mentioned above are not realistic for practical trading.

Based on the results, LSTM based-model's performance in simulated trading is generally better than Transformer-based model. The canonical LSTM model achieves highest CPR and SR in prediction horizon 20 and 30 and DeepLOB has the best performance in prediction horizon 50. For DLSTM, it has comparable performance to canonical LSTM and DeepLOB model. This result shows that the prediction result from LSTM-based models are robust and practical for trading. Autoformer's CPR is the lowest in prediction horizon 20 and 30. The state-of-the-art Autoformer sometimes has even worse performance than the vanilla Transformer in simulated trading, although it obtains a better classification metrics. To summarize, LSTM-based models are relatively the better models for electronic trading.

5.3.6 Simple Trading Simulation with transaction cost

In the real-world market, all operations, including buying or selling, need a commission fee, and sometimes the transaction cost might outweight the return. This section will introduce a hypothetical transaction cost

of 0.002% to further compare the robustness among different models. The results are shown in Table 6 and Figure 17.

Forecast Horizon	Prediction Horizon = 20		Prediction Horizon = 30		Prediction Horizon =50		Prediction Horizon=100	
Model	CPR	SR	CPR	SR	CPR	SR	CPR	SR
LSTM	2.102	15.160	1.767	12.429	1.596	11.536	0.778	6.014
DLSTM	**3.039**	**19.962**	**2.716**	**16.523**	**1.957**	**12.359**	**1.180**	**9.811**
DeepLOB	1.964	15.082	1.924	13.128	1.450	10.273	0.823	7.993
Transformer	1.860	13.894	1.561	10.917	1.047	6.612	0.118	-23.496
Autoformer	0.189	-8.704	0.873	5.118	-0.225	-9.193	-0.061	-14.835

Table 6: Cumulative price returns and annualized sharpe ratio of different models under 0.002% transaction cost.

From the table, DLSTM has the highest CPRs and SRs for all the prediction horizons, outperforming all other models. This shows DLSTM's strong profitability and robustness against the risk brought by the transaction cost. LSTM-based methods' performance is generally better than Transformer-based methods. Canonical LSTM and DeepLOB achieve the second-best CPRs and SRs in different prediction horizons. This indicates that the LSTM-based model's prediction results are more practical and effective in electronic trading. Interestingly, Transformer-based models' performance drops significantly under the transaction cost. The state-of-the-art Autoformer produces even less profit than vanilla Transformer, yielding negative CPRs and SRs in prediction horizon 50 and 100, although its prediction classification metrics is better than Transformer.

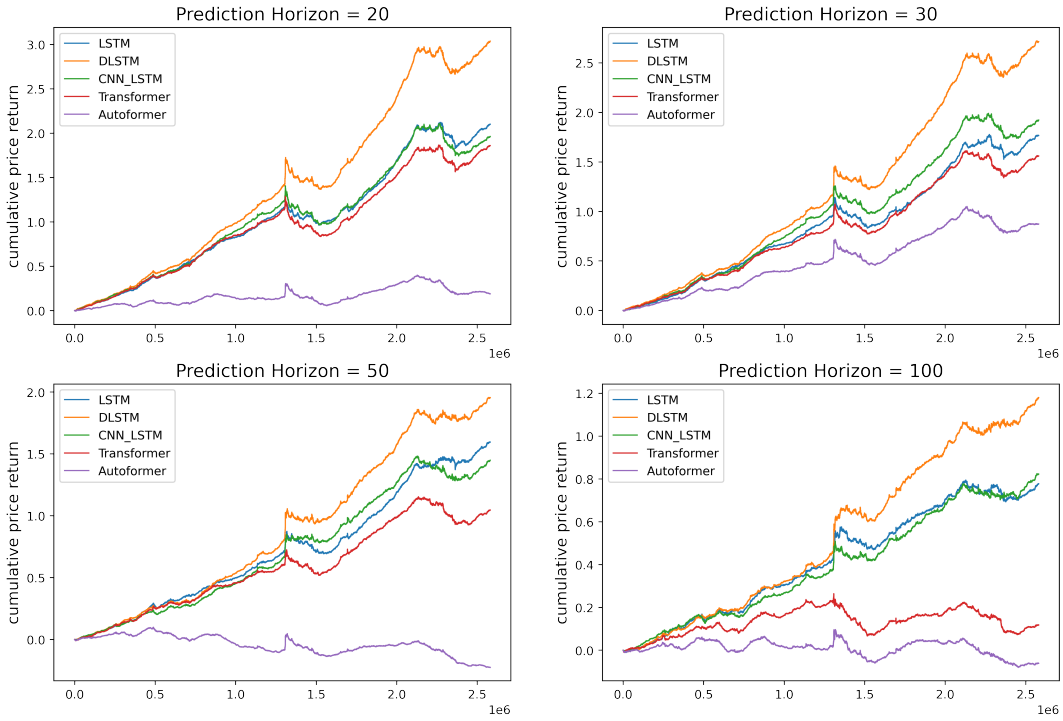

Figure 17: Cumulative return curve under 0.002% transaction cost.

To further investigate the impact of transaction cost on simulated trading. An experiment is extended to see how the CPR and SR change as the transaction cost increase. The experiment on CPR is done on the three-day test set from 2022.07.12 (inclusive) to 2022.07.14 (inclusive) as mentioned in Section 5.3.3. The experiment on SR has a longer backtesting period ranging from 2022.07.13 (inclusive) to 2022.07.24 (inclusive) because annualized Sharpe Ratio is calculated based on daily CPR, so using a longer backtesting period can produce a more accurate SR. The experiment results are shown in Figure 18 and Figure 19.

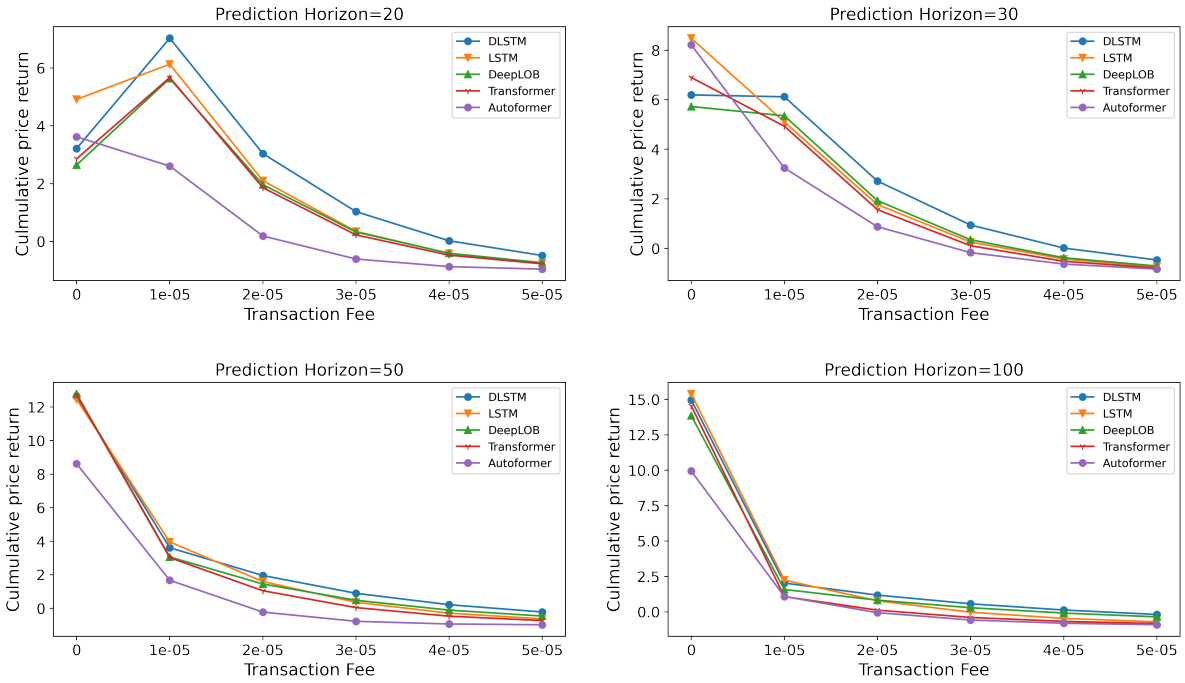

Figure 18: Cumulative price return change with increasing transaction cost (back testing period: 2022.07.12 (inclusive) to 2022.07.14 (inclusive)).

In terms of the CPR, all models' CPR decreases as the transaction cost increases. For the prediction horizon 20 and 30, DLSTM still outperforms other models when the transaction cost increases. All the models generate comparable CPR except the Autoformer in the prediction horizon 50 and 100. Autoformer's CPR is the lowest in all prediction horizons for most transaction cost settings. Regarding the SR, all models' SR decrease as the transaction cost increases. However, DLSTM maintains a higher SR than other models as the transaction cost increases. At the same time, for the Autoformer, its SR drops significantly and even becomes the lowest as the transaction cost increases. Overall, DLSTM keeps its profitability and robustness for different transaction costs, while the Transformer-based method's performance can be largely affected by the transaction cost. This result further indicates that the LSTM-based method is superior for electronic trading.

5.3.7 Efficiency Comparison on Transformers versus LSTM

The efficiency comparison of Transformer-based and LSTM-based models on price movement prediction is shown in Table 7. The comparison is separated into two parts, the left panel is the practical efficiency,

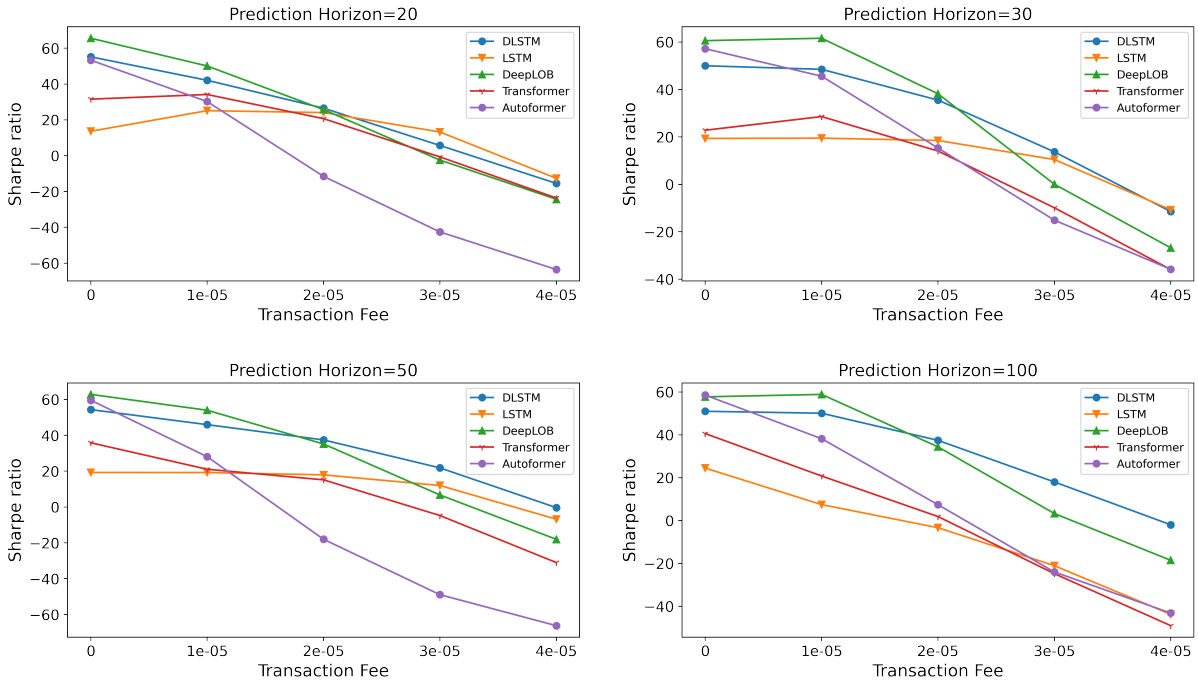

Figure 19: Sharpe ratio change with increasing transaction cost (back testings period: 2022.07.13 (inclusive) to 2022.07.24 (inclusive)).

Method	MACs	Parameter	Time	Memory	Time	Memory	Test Step
DLSTM	6.72 M	193.9 k	4.4ms	1404MiB	$O(L)$	$O(L)$	1
DeepLOB	36.42 M	143.91 k	6.3ms	2250MiB	$O(L)$	$O(L)$	1
Transformer	1.25 G	10.64 M	17.7ms	3534MiB	$O(L^2)$	$O(L^2)$	1
Reformer	1.17 G	5.84 M	23.6ms	4966MiB	$O(LlogL)$	$O(L^2)$	1
Informer	1.15 G	11.43 M	22.1ms	4361MiB	$O(LlogL)$	$O(LlogL)$	1
Autoformer	1.25 G	10.64M	75.2ms	5394MiB	$O(L)$	$O(L)$	1
FEDformer	1.25 G	16.47 M	38.3ms	3556MiB	$O(L)$	$O(L)$	1

Table 7: Efficiency comparison of Transformer-based and LSTM-based models on price movement prediction. MACs represent the number of Multiply-accumulate operations.

and the right is the theoretical efficiency. The latest transformer-based models have a focus on lowering the time and memory complexity. Autoformer and FEDformer claim to achieve $O(L)$ time and memory complexity in theory. However, their actual inference time and memory consumption are higher than the vanilla transformer models because of their complex design. For the training process, it usually takes more than 12 hours to train an Autoformer and FEDformer model, even with the cutting-edge GPU device (e.g., 24GB NVIDIA RTX 3090 GPU is used here), which is not efficient to retrain the model on new data. The researchers should reconsider the focus of the Transformer-based model on time series application. The time and memory complexity is not a big threshold for the vanilla Transformer, where its inference speed and memory consumption is acceptable depending on today's computing power.

The LSTM-based model has higher efficiency than the Transformer-based model for its low inference time and small model size, where its theoretical efficiency corresponds to its practical efficiency. This gives the LSTM-based model advantage in high-frequency trading, which requires fast execution speed. This also again emphasizes that LSTM-based models are the better model in electronic trading.

6 Conclusion and Future work

This study systematically compares LSTM-based and Transformer-based models among three financial time series prediction tasks based on cryptocurrency LOB data. The first task is to predict the LOB mid-price. FEDformer and Autoformer have less error than other models, and LSTM is still a strong model that surpasses Informer, Reformer and vanilla Transformer. Although the mid-price prediction error is low, the quality of the mid-price prediction result is far from sufficient for practical use in high-frequency trading. The second task is to predict LOB mid-price difference. LSTM-based methods show their robustness in time series prediction and perform better than Transformer-based models, which reach the highest 11.5% R^2 in around 10 prediction steps. State-of-the-art Autoformer and FEDformer are limited in this task because their time decomposition architecture can not handle the difference sequence. However, in a separate study [63], it was shown that custom transformer configurations can outperform the standard transformers.

The last task is to predict the LOB mid-price movement. New architecture for the Transformer-based model is designed for adapting the classification task. A new DLSTM model is proposed combining the merits of LSTM and time decomposition architecture from Autoformer. DLSTM outperforms all other models in classification metrics, and Autoformer shows comparable performance to LSTM-based models. A simple trading simulation is done to verify the practicality of the prediction. LSTM-based models have overall better performance than Transformer-based models and DLSTM model beats all other models under the transaction cost.

In conclusion, based on all the experiments on three different tasks, the Transformer-based model can only outperform LSTM-based models by a large margin in terms of the limited metrics for mid-price prediction. In comparison, the LSTM-based model is still dominant in the later two tasks, so LSTM-based models are generally the better model in financial time series prediction for electronic trading.

For future research, applying LSTM-based and Transformer-based models in Deep Reinforcement Learning (DRL) can be a proper direction. A complete high-frequency trading strategy usually requires the combination of different prediction signals and needs an experienced trader to control the take-profit and stop-loss.

In this case, using DRL to generate the optimal trading strategy directly can get us one step closer to the actual trading.

References

[1] Eugene F. Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of finance (New York)*, 25(2):383–, 1970. ISSN 0022-1082.

[2] John J. Murphy. *Study guide for Technical analysis of the financial markets : a comprehensive guide to trading methods and applications*. New York Institute of Finance, New York, 1999. ISBN 0735200653.

[3] Constance M. Brown. *Mastering elliott wave principle elementary concepts, wave patterns, and practice exercises*. Bloomberg financial series. Wiley, Hoboken, N.J, 1st edition edition, 2012. ISBN 1-280-67304-4.

[4] Carolyn Boroden. *Fibonacci trading: how to master the time and price advantage*. Mcgraw-hill New York, NY, 2008.

[5] David. Ruppert. *Statistics and Data Analysis for Financial Engineering with R examples*. Springer Texts in Statistics. Springer New York, New York, NY, 2nd ed. 2015. edition, 2015. ISBN 1-4939-2614-4.

[6] Adebiyi A. Ariyo, Adewumi O. Adewumi, and Charles K. Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112, 2014. doi: 10.1109/UKSim.2014.67.

[7] Victor Carbune, Pedro Gonnet, Thomas Deselaers, Henry A. Rowley, Alexander N. Daryin, Marcos Calvo, Li-Lun Wang, Daniel Keysers, Sandro Feuz, and Philippe Gervais. Fast multi-language lstm-based online handwriting recognition. *CoRR*, abs/1902.10525, 2019. URL `http://arxiv.org/abs/1902.10525`.

[8] Hagen Soltau, Hank Liao, and Hasim Sak. Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition, 2016. URL `https://arxiv.org/abs/1610.09975`.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL `http://arxiv.org/abs/1706.03762`.

[10] Justin Sirignano and Rama Cont. Universal features of price formation in financial markets: perspectives from deep learning, 2018. URL `https://arxiv.org/abs/1803.06917`.

[11] 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, jun 2019. doi: 10.1109/tsp.2019.2907260. URL `https://doi.org/10.1109%2Ftsp.2019.2907260`.

[12] Zihao Zhang and Stefan Zohren. Multi-horizon forecasting for limit order books: Novel deep learning approaches and hardware acceleration using intelligent processing units. *CoRR*, abs/2105.10430, 2021. URL `https://arxiv.org/abs/2105.10430`.

[13] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning for price prediction by exploiting stationary limit order book features, 2018. URL https://arxiv.org/abs/1810.09965.

[14] Petter N. Kolm, Jeremy D. Turiel, and Nicholas Westray. Deep order flow imbalance: Extracting alpha at multiple horizons from the limit order book. *Econometric Modeling: Capital Markets - Portfolio Theory eJournal*, 2021.

[15] Murtaza Roondiwala, Harshal Patel, and Shraddha Varma. Predicting stock prices using lstm. *International Journal of Science and Research (IJSR)*, 6, 04 2017. doi: 10.21275/ART20172755.

[16] Jian Cao, Zhi Li, and Jian Li. Financial time series forecasting model based on ceemdan and lstm. *Physica A: Statistical Mechanics and its Applications*, 519:127–139, 2019. ISSN 0378-4371. doi: https://doi.org/10.1016/j.physa.2018.11.061. URL https://www.sciencedirect.com/science/article/pii/S0378437118314985.

[17] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLOS ONE*, 12(7):1–24, 07 2017. doi: 10.1371/journal.pone.0180944. URL https://doi.org/10.1371/journal.pone.0180944.

[18] Sreelekshmy Selvin, R Vinayakumar, E. A Gopalakrishnan, Vijay Krishna Menon, and K. P. Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1643–1647, 2017. doi: 10.1109/ICACCI.2017.8126078.

[19] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, 270(2):654–669, 2018. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2017.11.054. URL https://www.sciencedirect.com/science/article/pii/S0377221717310652.

[20] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparative analysis of forecasting financial time series using arima, lstm, and bilstm. *CoRR*, abs/1911.09512, 2019. URL http://arxiv.org/abs/1911.09512.

[21] Sangyeon Kim and Myungjoo Kang. Financial series prediction using attention lstm, 2019. URL https://arxiv.org/abs/1902.10877.

[22] Xuan Zhang, Xun Liang, Aakas Zhiyuli, Shusen Zhang, Rui Xu, and Bo Wu. AT-LSTM: An attention-based LSTM model for financial time series prediction. *IOP Conference Series: Materials Science and Engineering*, 569(5):052037, jul 2019. doi: 10.1088/1757-899x/569/5/052037. URL https://doi.org/10.1088/1757-899x/569/5/052037.

[23] Xiaokang Hu. Stock price prediction based on temporal fusion transformer. In *2021 3rd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 60–66, 2021. doi: 10.1109/MLBDBI54094.2021.00019.

[24] Sashank Sridhar and Sowmya Sanagavarapu. Multi-head self-attention transformer for dogecoin price prediction. In *2021 14th International Conference on Human System Interaction (HSI)*, pages 1–6, 2021. doi: 10.1109/HSI52170.2021.9538640.

[25] Priyank Sonkiya, Vikas Bajpai, and Anukriti Bansal. Stock price prediction using bert and gan, 2021. URL https://arxiv.org/abs/2107.09055.

[26] Surafel M. Lakew, Mauro Cettolo, and Marcello Federico. A comparison of transformer and recurrent neural networks on multilingual neural machine translation, 2018. URL https://arxiv.org/abs/1806.06957.

[27] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. A comparative study on transformer vs RNN in speech applications. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, dec 2019. doi: 10.1109/asru46091.2019.9003750. URL https://doi.org/10.1109%2Fasru46091.2019.9003750.

[28] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: A survey, 2022. URL https://arxiv.org/abs/2202.07125.

[29] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014. URL https://arxiv.org/abs/1409.3215.

[30] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[31] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, 2019. URL https://arxiv.org/abs/1907.00235.

[32] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020. URL https://arxiv.org/abs/2001.04451.

[33] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting, 2020. URL https://arxiv.org/abs/2012.07436.

[34] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, 2021. URL https://arxiv.org/abs/2106.13008.

[35] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X. Liu, and Schahram Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=0EXmFzUn5I.

[36] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, 2022. URL https://arxiv.org/abs/2201.12740.

[37] Martin D. Gould, Mason A. Porter, Stacy Williams, Mark McDonald, Daniel J. Fenn, and Sam D. Howison. Limit order books, 2010. URL https://arxiv.org/abs/1012.0349.

[38] 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 01, pages 7–12, 2017. doi: 10.1109/CBI.2017.23.

[39] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8): 1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[40] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[41] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[42] F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999. doi: 10.1049/cp:19991218.

[43] Alex Graves. Generating sequences with recurrent neural networks, 2013. URL https://arxiv.org/abs/1308.0850.

[44] Roel Oomen and Jim Gatheral. Zero-intelligence realized variance estimation. *Finance and Stochastics*, 14:249–283, 04 2010. doi: 10.1007/s00780-009-0120-1.

[45] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

[46] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL http://arxiv.org/abs/1508.04025.

[47] R Farsani, Ehsan Pazouki, and Jecei Jecei. A transformer self-attention model for time series forecasting. *Journal of Electrical and Computer Engineering Innovations*, 9:1–10, 01 2021. doi: 10.22061/JECEI.2020.7426.391.

[48] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting?, 2022. URL https://arxiv.org/abs/2205.13504.

[49] Souhaib Ben Taieb and Rob J Hyndman. Recursive and direct multi-step forecasting: the best of both worlds. Monash Econometrics and Business Statistics Working Papers 19/12, Monash University, Department of Econometrics and Business Statistics, 2012. URL https://ideas.repec.org/p/msh/ebswps/2012-19.html.

[50] Guillaume Chevillon. Direct multi-step estimation and forecasting. *Journal of Economic Surveys*, 21(4):746–785, 2007. doi: https://doi.org/10.1111/j.1467-6419.2007.00518.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-6419.2007.00518.x.

[51] Robert B. Cleveland, William S. Cleveland, Jean E. McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6:3–73, 1990.

[52] James Douglas Hamilton. *Time Series Analysis*. Princeton University Press, Princeton, 1994. ISBN 0691042896.

[53] Rob J. Hyndman. *Forecasting : principles and practice*. OTexts, Melbourne, third edition. edition, 2021 - 2021. ISBN 9780987507136.

[54] Bryant Moscon. cryptofeed. https://github.com/bmoscon/cryptofeed, 2022.

[55] Jan Novotny, Paul A Bilokon, Aris Galiotos, and Frédéric Délèze. *Machine Learning and Big Data with kdb+/q*. John Wiley & Sons, 2019.

[56] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer. https://github.com/thuml/Autoformer, 2022.

[57] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer. https://github.com/MAZiqing/FEDformer, 2022.

[58] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL http://arxiv.org/abs/1912.01703.

[59] Michael S. Lewis-Beck. *Applied regression : an introduction*. Sage university papers series. Quantitative applications in the social sciences ; no. 07-022. Sage Publications, Beverly Hills, Calif, 1980. ISBN 0803914946.

[60] Zihao Zhang. Deeplob-deep-convolutional-neural-networks-for-limit-order-books. `https://github.com/zcakhaa/DeepLOB-Deep-Convolutional-Neural-Networks-for-Limit-Order-Books`, 2021.

[61] Zihao Zhang. Multi-horizon-forecasting-for-limit-order-books. `https://github.com/zcakhaa/Multi-Horizon-Forecasting-for-Limit-Order-Books`, 2021.

[62] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Ltsf-linear. `https://github.com/cure-lab/LTSF-Linear`, 2022.

[63] Fazl Barez, Paul Bilokon, Arthur Gervais, and Nikita Lisitsyn. Exploring the advantages of transformers for high-frequency trading. *SSRN Electronic Journal*, 2023. doi: 10.2139/ssrn.4364833.

A Labelling Details

As mentioned in Section 3.4, a threshold δ needs to be set to decide the corresponding labels. The choice of δ follows a simple rule, which is to make the labelling roughly balanced. The choice of δ for different prediction horizon on ETH-USDT dataset is shown in Table 8 and the distribution of labelling is shown in Figure 20.

Horizon	20	30	50	100
δ	0.17	0.3	0.6	0.92

Table 8: δ for different prediction horizons for ETH-USDT dataset.(units in 10^{-4})

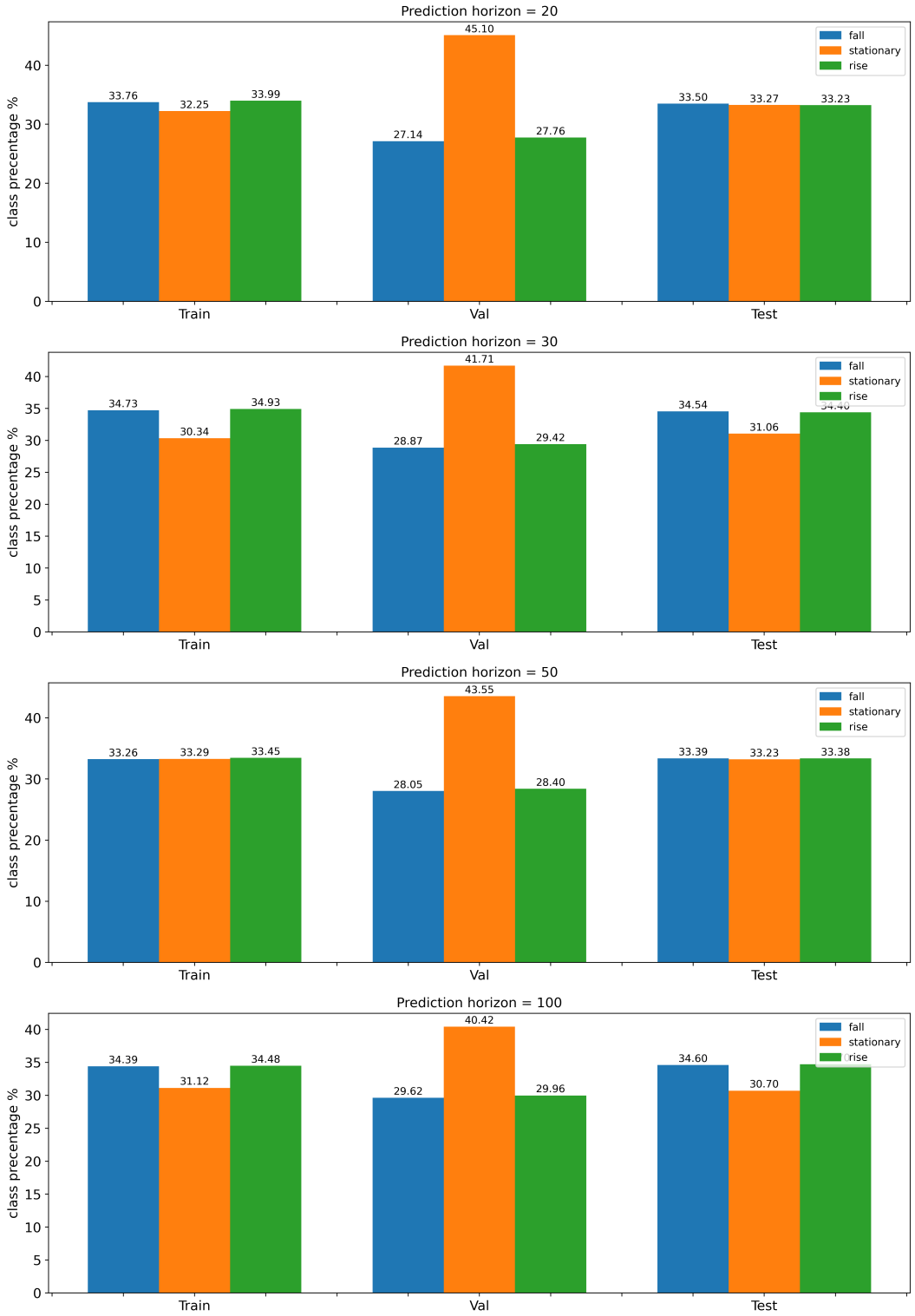

Figure 20: Labelling distribution for different prediction horizon in ETH-USDT dataset