Multi-Horizon Forecasting for Limit Order Books: Novel Deep Learning Approaches and Hardware Acceleration using Intelligent Processing Units

Zihao Zhang, Stefan Zohren
Oxford-Man Institute of Quantitative Finance,
University of Oxford

May 21, 2021

Abstract

We design multi-horizon forecasting models for limit order book (LOB) data by using deep learning techniques. Unlike standard structures where a single prediction is made, we adopt encoder-decoder models with sequence-to-sequence and Attention mechanisms, to generate a forecasting path. Our methods achieve comparable performance to state-of-art algorithms at short prediction horizons. Importantly, they outperform when generating predictions over long horizons by leveraging the multi-horizon setup. Given that encoder-decoder models rely on recurrent neural layers, they generally suffer from a slow training processes. To remedy this, we experiment with utilising novel hardware, so-called Intelligent Processing Units (IPUs) produced by Graphcore. IPUs are specifically designed for machine intelligence workload with the aim to speed up the computation process. We show that in our setup this leads to significantly faster training times when compared to training models with GPUs.

1 Introduction

Limit order books (LOBs), as the canonical example of high-frequency financial microstructure data, have received tremendous popularity in recent academic studies. At any time stamp, a LOB is a record of all outstanding limit orders (passive orders) for a financial instrument at an exchange. It is sorted into different levels based on the prices of the submitted orders. The LOB has two sides, representing buy and sell orders – also referred to as bid and ask. Each level of a LOB indicates the total available volume (number of shares) at the price of that level. Those detailed records of price and volume information provide us with a picture of the short-term supply and demand relationship. From this, we can compute quantities such as order imbalances (Chordia et al., 2002), which help to understand the dynamics of high-frequency microstructure data.

Recent works by Tsantekidis et al. (2017a); Tran et al. (2018); Zhang et al. (2019a); Briola et al. (2020) demonstrate that LOBs have strong predictive to forecast price moves at short time intervals. Their findings have inspired a range of extensions including high-frequency trading models (Briola et al., 2021). However, the aforementioned works are formulated as a standard supervised learning tasks where a price at a single future point in time is predicted. Given that financial time-series are notoriously stochastic with a low signal-to-noise ratio (Gould et al., 2013), a single prediction imposes limitations to describe the future evolution of market movements. Naturally, multi-horizon forecasting (predicting multiple steps into future) is desirable, since we can obtain a forecasting path which can be used for trading decision making or risk management.

In this work, we design multi-horizon forecasting models for LOB data with deep learning techniques (Goodfellow et al., 2016). Inspired by the machine translation problems (Bahdanau et al., 2014) from

Natural language processing (NLP), we apply sequence-to-sequence (Seq2Seq) (Sutskever et al., 2014; Cho et al., 2014) and Attention (Luong et al., 2015) models to generate multi-horizon forecasts. We adopt the deep network architecture from Zhang et al. (2019a) and engineer the output layers to produce a forecasting path. We test our method on the popular, publicly available LOB dataset (FI-2010 Ntakaris et al. (2018)). The experiments show that our model delivers competitive results when compared to state-of-the-art models for single step, short horizon forecasts. Furthermore, in our setting a single network is capable of predicting multi-steps into future, avoiding the limitations of a single point estimation. Interstingly, our method delivers superior results for predicting over long horizons as short-term predictions contribute to future estimation through an autoregressive structure.

The dominant Seq2Seq and Attention models are based on complex recurrent neural layers that include an encoder and a decoder. Such recurrent structures lead to substantially slow training processes even when employing GPUs for acceleration. This often poses challenges which need to be overcome. The work of Vaswani et al. (2017), for example, proposes Transformers to allow parallel training of attention mechanisms by using fully connected layers. In this work, we utilise a different form of hardware acceleration., so-called Intelligence Processing Units (IPUs). IPUs developed by Graphcore (Graphcore, 4 28) are a novel massively parallel processor. They can be used to accelerate the training process, offering an alternative solution to deal with this bottleneck. We compare the computation efficiency between GPUs and IPUs in our setting by benchmarking with a wide variety of state-of-the-art network architectures. The results indicate that IPUs are multiple times faster than GPUs. This significant improvement in computation is not necessarily restricted to the training process but could also lead to speedups in a wide range of applications within existing algorithms, for example, to reduce latency in market-making strategies.

The remainder of this paper is structured as follows: In Section 2, we include a literature review to discuss the development of deep learning algorithms on LOBs and review multi-horizon forecasting models. Section 3 gives a short introduction to IPUs and Section 4 discusses our proposed network architectures. We then describe our experiments and present the results in Section 5. We conclude in Section 6 by summarising our findings and proposing potential research problems.

2 Literature Review

Deep learning models have been heavily used for prediction tasks on LOB data, where Tsantekidis et al. (2017a,b); Sirignano and Cont (2019); Zhang et al. (2019a) helped to build the foundation in this area. Subsequently, a wide range of extensions have been proposed to improve predictive performance, including Bayesian deep networks (Zhang et al., 2018), Quantile regression (Zhang et al., 2019b), Transformers (Wallbridge, 2020) and usages of more granular market by order data (Zhang et al., 2021). In addition, LOB data has been studied in the context of reinforcement learning (Wei et al., 2019), market-making (Sadighian, 2019), cryptocurries (Jha et al., 2020) and portfolio optimisation (Sangadiev et al., 2020). However, to the best of our knowledge, we have not found any existing work that studys multi-horizon forecasts for LOB data and we aim to fill this gap in the literature.

In terms of the multi-horizon forecasting models, Taieb et al. (2010); Marcellino et al. (2006) introduced traditional econometric approaches. In this work, we focus on recent deep learning techniques, mainly Seq2Seq (Sutskever et al., 2014; Cho et al., 2014) and attention-based (Luong et al., 2015; Fan et al., 2019) approaches. A typical Seq2Seq model contains an encoder to summarise past time-series information and a decoder to combine hidden states with future known inputs to generate predictions. However, the Seq2Seq model only utilises the last hidden state from an encoder to make estimations, thus making it incapable of processing inputs with long sequences. Attention was proposed to assign a proper weight to each hidden state from the encoder to solve this limitation. We study both methods on LOB data and adapt the network architecture in Zhang et al. (2019a) to propose an end-to-end framework for generating multi-step predictions.

Despite the popularity of Seq2Seq and Attention models, the recurrent nature of their structure imposes bottlenecks for training. This potentially limits their use cases on high-frequency microstructure data as modern electronic exchanges can generate billions of observations in a single day, making the training of such models on large and complex LOB datasets infeasible even with multiple GPUs. Here we experiment with IPUs for hardware acceleration. Graphcore introduced their IPUs as a novel massively parallel processor for training deep learning models (IPU, 4 28). The work in Jia et al.

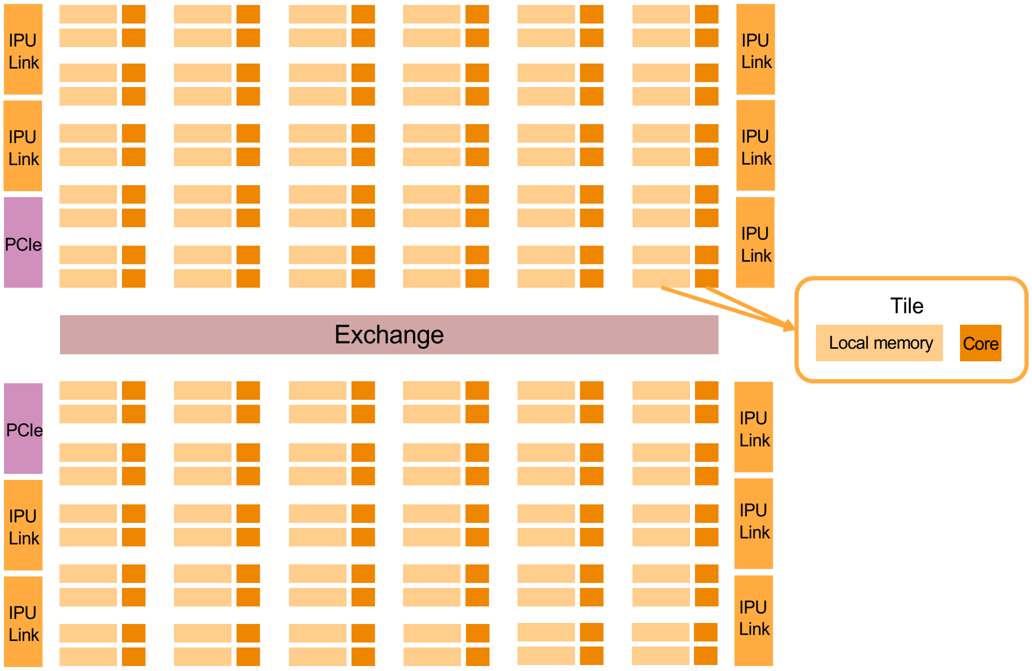

Figure 1: Simplified illustration of an IPU processor. Each processor has four main components: The tile, exchange, link and PCIe.

(2019) conducted a thorough investigation between GPUs and IPUs, observing massive speed-ups in training. We test the computational power of GPUs and IPUs on the state-of-art network architectures for LOB data and our findings are in line with Jia et al. (2019). We utilise Graphcore's Poplar graph framework software (Poplar, 4 28) which allows for seamless integration of IPUs with TensorFlow, Keras (Abadi et al., 2016) and PyTorch (Paszke et al., 2019), requiring minimum emendation from existing repositories written for GPUs.

3 Intelligence Processing Unit (IPU)

Graphcore has designed IPUs specifically for machine intelligence problems and some of the computing architectures differ radically from common hardware such as CPUs and GPUs. In this section, we present a brief introduction to IPUs and discuss some differences among these architectures. For a complete and in-depth comparison, interested readers are referred to Jia et al. (2019).

The cornerstone of an IPU-based system is the IPU processor with the aim of achieving efficient execution of fine-grained operations across a relatively large number of parallel threads. In general, each IPU processor contains four components: IPU-tile, IPU-exchange, IPU-link and PCIe. For each processor, there are 1216 tiles and each tile consists of one computing core and 256 kB of local memory. These tiles are interconnected by the IPU-exchange which allows for low-latency and high-bandwidth communication. In addition, each IPU contains ten IPU-link interfaces, which is a Graphcore proprietary interconnect that enables low latency, high-throughput communication between IPU processors. IPU-links enable transfers between remote tiles as efficient as between local tiles and they are key to the IPU's computational scalability. Besides that, each IPU contains two PCIe links for communication with CPU-based hosts. We illustrate the IPU architecture with a simplified diagram in Figure 1.

The architecture of IPUs differs significantly from CPUs and GPUs that are commonly used for training machine learning algorithms. In general, CPUs excel at single-thread performance as they offer complex cores in relatively small counts. However, even with the vectorisation of data, CPUs are incomparable with GPUs in aggregate floating-point arithmetic on large and complex workloads. GPUs, on the other hand, have architecturally simpler cores than CPUs but do not offer branch speculation or hardware prefetching. The typical arrangment of GPUs are grouped into clusters, so

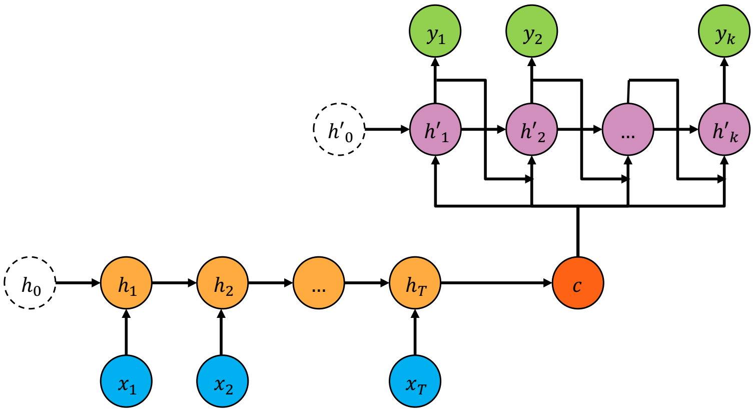

Figure 2: A schematic depiction of the Seq2Seq network architecture.

all cores in a cluster execute the same instruction at any point in time. Because of this architecture, GPUs are proficient at regular, dense, numerical, data-flow-dominated workloads and tend to be more energy efficient than CPUs.

The IPU's approach of accelerating computation is through shared memory, which is distinct from the other hardware. An IPU offers small and distributed memories that are locally coupled to each other, therefore, IPU cores pay no penalty when their control flows diverge or when the addresses of their memory accesses diverge. Such a structure allows cores to access data from local memory at a fixed cost that is independent of access patterns, making IPUs more efficient than GPUs when executing workloads with irregular or random data access patterns as long as the workloads can be fitted in IPU memory.

In this work, we compare the training efficiency between GPUs and IPUs for state-of-art deep networks on LOB data and the results indeed suggest that IPUs deliver superior computation performance.

4 Multi-Horizon Forecasting Models

This section introduces deep learning architectures for multi-horizon forecasting models, in particular Seq2Seq and Attention models. In essence, both of these architectures consist of three components: an encoder, a context vector and a decoder. We write a given input of any time-series as $\boldsymbol{x}_{1:T} = (x_1, x_2, \cdots, x_T) \in \mathbb{R}^{T\times m}$, where T represents the length of the input sequence and x_t indicates m features at each time stamp t. Similarly, a typical output is $\boldsymbol{y}_{1:k} = (y_1, y_2, \cdots, y_k) \in \mathbb{R}^{k\times n}$ where k is the furthest prediction point, and it is essentially a multi-input and multi-output setup.

An encoder steps through the input time steps to extract meaningful features. The resulting context vector encapsulates the resulting sequence into a vector for integrating information. Finally, a decoder reads from the context vector and steps through the output time step to generate multi-step predictions. The fundamental difference between the Seq2Seq and Attention model is the construction of the context vector. The Seq2Seq model only takes the last hidden state from the encoder to form the context vector, whereas the Attention model utilises the information from all hidden states in the encoder.

4.1 Sequence to Sequence Learning (Seq2Seq)

In this work, we employ the Seq2Seq architecture in Cho et al. (2014) in the context of multi-horizon forecasting models for LOBs. Overall, the encoder contains a recurrent neural network (RNN) that operates on a given input $\boldsymbol{x}_{1:T} = (x_1, x_2, \cdots, x_T)$. At each time step t of the encoder, the hidden state $\boldsymbol{h}_t$ is

$$\text{Encoder:} \quad \boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, x_t), \tag{1}$$

where f is a nonlinear function. The choice of f can vary, ranging from a simple logistic sigmoid function to a more complex LSTM (Hochreiter and Schmidhuber, 1997). The encoder reads through a given input and the last hidden state summarises the whole sequence. The last hidden state $\boldsymbol{c}$ is the "bridge" between the encoder and decoder, also known as the context vector. At time step t of the decoder, the hidden state $\boldsymbol{h}'_t$ is

$$\text{Decoder:} \quad \boldsymbol{h}'_t = f(\boldsymbol{h}'_{t-1}, y_{t-1}, \boldsymbol{c}), \tag{2}$$

and the distribution for output y_t is

$$P(y_t|y_{t-1}, y_{t-2}, \cdots, y_1, \boldsymbol{c}) = g(\boldsymbol{h}'_t, \boldsymbol{c}). \tag{3}$$

Here f and g are nonlinear functions and g needs to produce valid probabilities, which in our case is done through a softmax activation function. Figure 2 illustrates the structure of a standard Seq2Seq network. Seq2Seq models work well for inputs with small sequences, but suffers when the length of the sequence increases as it is difficult to summarise the entire input into a single hidden state represented by the context vector. Models tend to forget the earlier parts of the input and results often deteriorate as the size of the sequence increases.

4.2 Attention

The Attention model (Luong et al., 2015) is an evolution of the Seq2Seq model, developed in order to deal with inputs of long sequences. The core idea is to allow the decoder to selectively access hidden states of the encoder during decoding. We can build a different context vector for every time step of the decoder as a function of the previous hidden state and of all the hidden states in the encoder. Similar to the Seq2Seq model, the hidden state $\boldsymbol{h}_t$ of the encoder is

$$\text{Encoder:} \quad \boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, x_t), \tag{4}$$

where f is a nonlinear function. For the context vector $\boldsymbol{c}_t$ at time stamp t of the decoder, one defines

$$\begin{aligned} &\text{Context vector:} && \boldsymbol{c}_t = \sum_{i=1}^{T} \alpha_{t,i} h_i, \\ &\text{Attention weight:} && \alpha_{t,i} = \frac{exp(e(\boldsymbol{h}'_{t-1}, \boldsymbol{h}_i))}{\sum_{j=1}^{T} exp(e(\boldsymbol{h}'_{t-1}, \boldsymbol{h}_j))}, \end{aligned} \tag{5}$$

where $e(\boldsymbol{h}'_{t-1}, \boldsymbol{h}_i)$ is called the score derived from the previous hidden state $\boldsymbol{h}'_{t-1}$ of the decoder and the hidden state $\boldsymbol{h}_i$ of the encoder. In Luong et al. (2015), there are three alternatives to calculate the score

$$e(\boldsymbol{h}'_{t-1}, \boldsymbol{h}_i) = \begin{cases} \boldsymbol{h}_i^{\mathrm{T}} \boldsymbol{h}'_{t-1} & \text{dot,} \\ \boldsymbol{h}_i^{\mathrm{T}} \boldsymbol{W}_a \boldsymbol{h}'_{t-1} & \text{general,} \\ tanh(\boldsymbol{W}_a[\boldsymbol{h}_i^{\mathrm{T}}; \boldsymbol{h}'_{t-1}]) & \text{concatenate.} \end{cases} \tag{6}$$

We can then pass the context vector $\boldsymbol{c}_t$ to the decoder to calculate the probability distribution of the next possible output

$$\begin{aligned} \text{Decoder:} \quad \boldsymbol{h}'_t &= f(\boldsymbol{h}'_{t-1}, y_{t-1}, \boldsymbol{c}_t), \\ P(y_t|y_{t-1}, y_{t-2}, \cdots, y_1, \boldsymbol{c}_t) &= g(\boldsymbol{h}'_t, \boldsymbol{c}_t), \end{aligned} \tag{7}$$

where $\boldsymbol{h}'_t$ is the hidden state of the decoder at time t and g is a softmax activation function. We illustrate the Attention mechanism in Figure 3.

4.3 Network Architecture

LOBs are complex dynamic objects of high dimensionality. Furthermore, LOB data, just like any other financial time-series, is notoriously non-stationary and of low signal-to-noise ratio (Gould et al., 2013). Since the encoder reads through an input to extract meaningful information, we adapt the modern deep network (DeepLOB) designed specifically for limit order books in Zhang et al. (2019a) as the encoder, extracting representative features from raw LOB data.

Here we give a brief introduction to DeepLOB – the exact network architecture can be found in Zhang et al. (2019a). Overall, DeepLOB comprises three building blocks: a convolutional block with multiple convolutional layers (CNNs), an Inception Module and a LSTM layer. The usage of

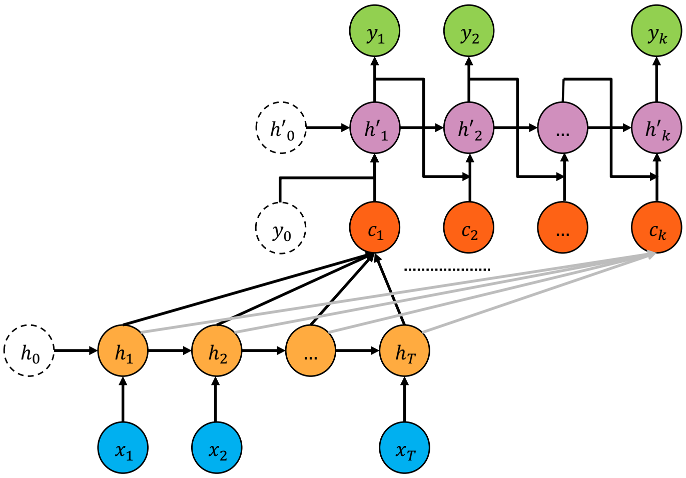

Figure 3: An schematic depiction of the Attention mechanism.

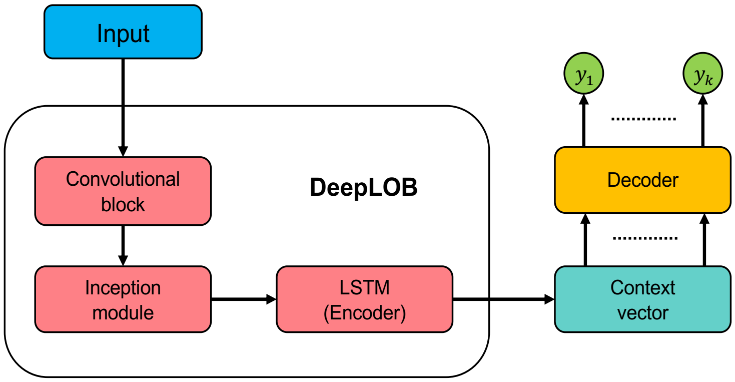

Figure 4: Model architecture schematic: A combination of DeepLOB and an encoder-decoder structure.

the CNNs and Inception Module is to automate the process of feature extraction. The convolutional layer possesses nice properties like smoothing and parameter sharing. The latter is important to deal with the low signal-to-noise ratio. The usage of the convolution smoothes input signals, similar to an auto-regressive model, and the Inception Module consists of multiple convolutional layers in parallel, each with a different kernel size, to infer local interactions over different time horizons. In the end, a LSTM layer is used to capture dynamic temporal behaviour among the resulting feature maps and the whole structure takes "space-time image" of the LOB as inputs.

Since DeepLOB is a hybrid model that combines CNNs and LSTMs, we can directly take it as our encoder model. We illustrate the resulting model architecture in Figure 4. For the decoder, we experiment with both Seq2Seq and Attention models to generate multi-horizon forecasts. An interesting byproduct of using Attention models is that the attention weights can be used to understand the importance of input features.

5 Experiments

5.1 Descriptions of Datasets

The FI-2010 dataset (Ntakaris et al., 2018) is the first publicly available benchmark dataset of high-frequency limit order book data. Many previous algorithms are tested on this dataset and we use it to build a fair comparison between our method and state-of-art algorithms. The FI-2010 dataset consists of LOB updates for five stocks from the Nasdaq Nordic stock exchange for a time period of 10 consecutive days. It contains 10 levels for both ask and bid of a LOB and each level has information for price and available volume at that price level. A classification setup is formulated in which we have three classes of labels: pricing going up, staying stationary or going down. Also, it studied five prediction horizons at $k = (10, 20, 30, 50, 100)$ in "tick time", i.e. consecutive LOB updates. As argued in multiple place, tick time, similar to volume time, is a natural time to consider for financial instruments.

The exact procedure of data normalisation and label formation can be found in Ntakaris et al. (2018) and we follow the setting originated from the works (Tsantekidis et al., 2017a, 2020) to evaluate our network architecture, in which the first 7 days are used as the train data and the last 3 days as the test data. We split the last 20% observations from the train set as the validation set to optimise hyperparameters. Each input contains the most recent 50 updates and each update includes information for both ask and bid of a LOB, therefore, a single input $\boldsymbol{x}_{1:T} \in \mathbb{R}^{T \times m}$ has the dimension of $(50, 40)$. We feed inputs to the designed network where the encoder extracts representative features and the decoder generates multi-horizon forecasts. In this work, our model can directly forecast all 5 prediction horizons in contrast to standard methods where a separate model is needed for each prediction horizon.

5.2 Training Configuration

As described above, we adapt DeepLOB (Zhang et al., 2019a) as our encoder and further details of the architecture and hyperparameters can be found in their work. In terms of the decoder, we use a single LSTM with 64 units for both Seq2Seq and Attention, denoted as DeepLOB-Seq2Seq and DeepLOB-Attention respectively. We include a wide variety of benchmark algorithms in the experiment, including a support vector machine (SVM (Tsantekidis et al., 2017b)), a multi-layer perceptron (MLP (Tsantekidis et al., 2017b)), a convolutional network (CNN-I (Tsantekidis et al., 2017a)), a LSTM ((Tsantekidis et al., 2017b)), a variant convolutional network (CNN-II (Tsantekidis et al., 2020)), as well as an Attention-augmented-Bilinear-Network with one hidden layer (B(TABL) (Tran et al., 2018)) and two hidden layers (C(TABL) (Tran et al., 2018)). Note that these benchmark algorithms produce a single-point estimation and the authors did not test on all prediction horizons available for the FI-2010 dataset.

The categorical cross-entropy loss is our objective function, and we use four evaluation metrics: Accuracy, Precision, Recall and F1-score. Since the labels in FI-2010 dataset are not well balanced, Ntakaris et al. (2018) suggested to focus on F1 score as the main evaluation metric and Kolmogorov-Smirnov (Massey Jr, 1951) tests are used to check how results are statistically different. The code is available at GitHub [1].

5.3 Experimental Results

Table 5 summarises the results for all models studied at each prediction horizon and all results are statistically different in terms of the Kolmogorov-Smirnov test. We observe that both DeepLOB-Seq2Seq and DeepLOB-Attention deliver comparable results to the state-of-art benchmark algorithms. Specifically, with shorter prediction horizons ($k = 10, 20, 30, 50$), the performance gap between DeepLOB and our methods is very small. However, both our new multi-horizon forecasting models achieve superior results for predicting a long horizon ($k = 100$), with DeepLOB-Attention being the best. The architecture of a decoder allows short-term predictions to be fed into the next estimation and our results indicate that this autoregressive structure helps with longer prediction horizons through the iterative estimation procedure.

The normalised confusion matrices for DeepLOB and our methods are presented in Figure 6. We use these plots to understand how models perform at predicting each class label. In general, all three methods achieve better accuracy for predicting stationary labels at short prediction horizons but the performance deteriorates as the horizon increases. Interestingly, all three models are getting better at predicting up and down labels with an increase in prediction horizon. Price movements are more significant helping to distinguish up or down labels from stationary labels. As a result, the models can do better at detecting larger price moves.

[1] `https://github.com/zcakhaa/Multi-Horizon-Forecasting-for-Limit-Order-Books`

Table 5: Experiment results for the FI-2010 dataset.

Model	**Accuracy** %	**Precision** %	**Recall** %	**F1** %
Prediction Horizon k = 10				
SVM (Tsantekidis et al., 2017b)	-	39.62	44.92	35.88
MLP (Tsantekidis et al., 2017b)	-	47.81	60.78	48.27
CNN-I (Tsantekidis et al., 2017a)	-	50.98	65.54	55.21
LSTM (Tsantekidis et al., 2017b)	-	60.77	75.92	66.33
CNN-II (Tsantekidis et al., 2020)	-	56.00	45.00	44.00
B(TABL) (Tran et al., 2018)	78.91	68.04	71.21	69.20
C(TABL) (Tran et al., 2018)	84.70	76.95	78.44	77.63
DeepLOB (Zhang et al., 2019a)	84.47	84.00	84.47	83.40
DeepLOB-Seq2Seq	82.58	81.65	82.58	81.51
DeepLOB-Attention	83.28	82.50	83.28	82.37
Prediction Horizon k = 20				
SVM (Tsantekidis et al., 2017b)	-	45.08	47.77	43.20
MLP (Tsantekidis et al., 2017b)	-	51.33	65.20	51.12
CNN-I (Tsantekidis et al., 2017a)	-	54.79	67.38	59.17
LSTM (Tsantekidis et al., 2017b)	-	59.60	70.52	62.37
CNN-II (Tsantekidis et al., 2020)	-	-	-	-
B(TABL) (Tran et al., 2018)	70.80	63.14	62.25	62.22
C(TABL) (Tran et al., 2018)	73.74	67.18	66.94	66.93
DeepLOB (Zhang et al., 2019a)	74.85	74.06	74.85	72.82
DeepLOB-Seq2Seq	74.38	73.12	74.38	72.99
DeepLOB-Attention	75.25	74.31	75.25	73.73
Prediction Horizon k = 30				
CNN-I (Tsantekidis et al., 2017a)	67.98	66.52	67.98	65.72
DeepLOB (Zhang et al., 2019a)	76.36	76.00	76.36	75.33
DeepLOB-Seq2Seq	76.41	75.86	76.41	75.75
DeepLOB-Attention	77.59	77.32	77.59	76.94
Prediction Horizon k = 50				
SVM (Tsantekidis et al., 2017b)	-	46.05	60.30	49.42
MLP (Tsantekidis et al., 2017b)	-	55.21	67.14	55.95
CNN-I (Tsantekidis et al., 2017a)	-	55.58	67.12	59.44
LSTM (Tsantekidis et al., 2017b)	-	60.03	68.58	61.43
CNN-II (Tsantekidis et al., 2020)	-	56.00	47.00	47.00
B(TABL) (Tran et al., 2018)	75.58	74.58	73.09	73.64
C(TABL) (Tran et al., 2018)	79.87	79.05	77.04	78.44
DeepLOB (Zhang et al., 2019a)	80.51	80.38	80.51	80.35
DeepLOB-Seq2Seq	78.10	77.96	78.10	77.99
DeepLOB-Attention	79.49	79.51	79.49	79.38
Prediction Horizon k = 100				
CNN-I (Tsantekidis et al., 2017a)	64.87	65.51	64.87	65.05
DeepLOB (Zhang et al., 2019a)	76.72	76.85	76.72	76.76
DeepLOB-Seq2Seq	79.09	79.31	79.09	79.16
DeepLOB-Attention	81.45	81.62	81.45	81.49

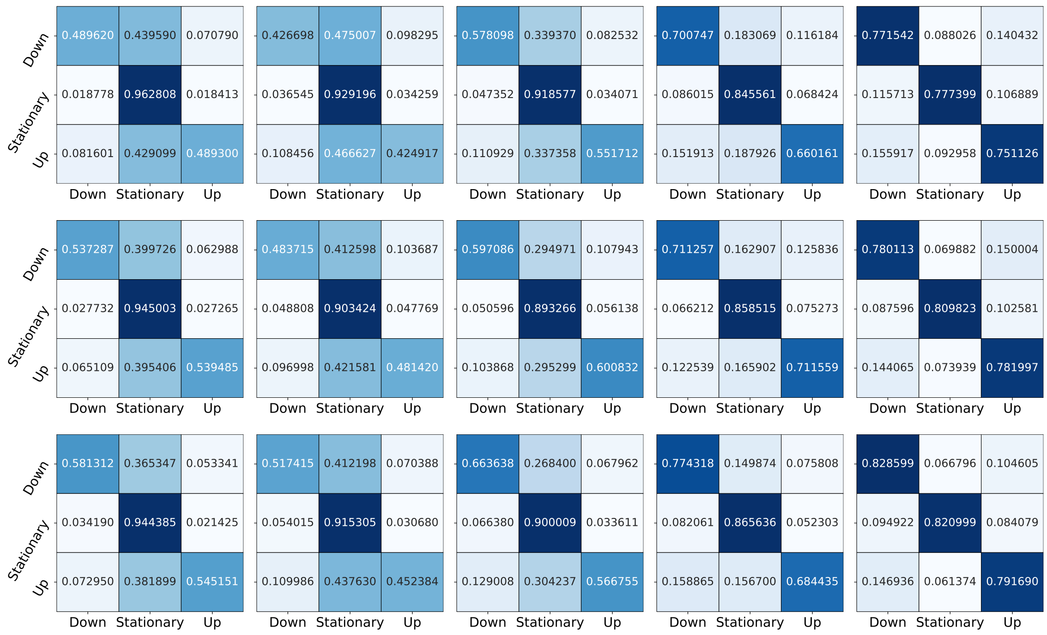

Figure 6: Normalised confusion matrices for DeepLOB (**TOP**), DeepLOB-Seq2Seq (**Middle**) and DeepLOB-Attention (**Bottom**). From the left to right, the prediction horizon (k) equals to 10, 20, 30, 50 and 100.

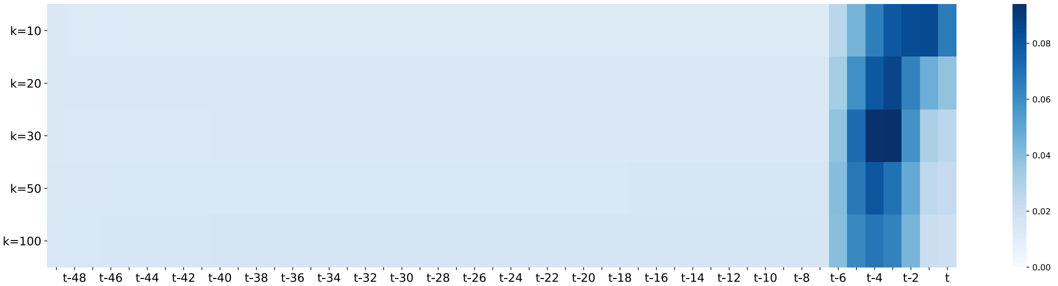

Figure 7: Attention weights for features from the encoder of DeepLOB-Attention.

When comparing our two networks with each other, we observe that DeepLOB-Attention delivers better results than DeepLOB-Seq2Seq. However, it seems that the Seq2Seq model does not suffer from long input sequences of LOB data. One possible explanation is that the markets are sufficiently efficient so that the current LOB observation contains most available information and features distant from current time stamp contain minimal predictive information. To verify this statement, we use the attention weights from DeepLOB-Attention to understand feature importance at different time steps from the encoder. Figure 7 shows the plot of attention weights. We observe that the weights are largest for most recent observations with any features further in the past being inactive. This helps to explain why the Seq2Seq model does not suffer from long input sequences as past information is less relevant in this case.

5.4 Comparison between IPU and GPU

Anther important focus of this work is to utilise novel IPU hardware for the training process of our models. In particular, we want to benchmark training times using IPUs against equivalent times when using GPUs. In our experiment, we compare a single GPU (NVIDIA GeForce RTX 2080) to an IPU unit. We test, DeepLOB, DeepLOB-Seq2Seq, DeepLOB-Attention and other three networks separately on the GPU and IPU. The model training lasts for 200 epochs and we report the average training time per epoch in Table 8. The IPU achieves superior performance in training speed and, in

Table 8: Average training time (per epoch) comparison between IPU and GPU.

Model	**Training Time** (in sec.)		**# of parameters**
	GPU	IPU	
MLP (Tsantekidis et al., 2017b)	19	4	256515
CNN-I (Tsantekidis et al., 2017a)	36	4	17635
LSTM (Tsantekidis et al., 2017b)	83	31	60099
DeepLOB (Zhang et al., 2019a)	96	23	105347
DeepLOB-Seq	215	38	176419
DeepLOB-Attention	270	41	177699

particular, it only takes about 15% of the corresponding GPU wall-clock time to train an encoder-decoder model. The improvement in training speed is outstanding and it offers an alternative solution to deal with the slow training of an encoder-decoder network instead of using Transformers.

6 Conclusion

In this work we design multi-horizon forecasting models for limit order book (LOB) data by using deep learning techniques. We adapt encoder-decoder models, with Seq2Seq and Attention models as decoders, to generate forecast paths over multiple time steps. An encoder reads through the raw LOB data to extract representative features and a decoder steps through the output time step to generate multi-step forecasts. Our experiments suggest that our method delivers superior results compared to state-of-art algorithms. This is due to the iterative nature the decoder delivers which yields better predictive performance over long horizons as short-term estimates are fed into next prediction through an autoregressive structure.

Encoder-decoder models rely on complex recurrent neural layers that often suffer from slow training processes. We address this problem by using a novel hardware IPU developed by Graphcore which is specifically designed for machine intelligence workload. We conduct a comparison between GPUs and IPUs to benchmark their training speed on modern deep neural networks for LOB data. We observe that IPUs leads to an acceleration that is significantly faster than common GPUs. Such speed-ups in training time could open up a wide variety of applications, for example, application of online learning or reinforcement learning in the context of market-making, as such a high-frequency trading strategy has strict requirements on communication latency. It would be interesting to deploy IPUs to such setups and test their computational efficiency. Also, we can apply the encoder-decoder structure to a Reinforcement Learning framework as studied in Zhang et al. (2020b,a).

Acknowledgements

The authors would like to thank Graphcore for making their hardware available for this study. They would also like to thank Alex Titterton and Alex Tsyplikhin from Graphcore for their valuable advise and help on hardware specific implementation details. Furthermore, the authors would like to thank the members of Machine Learning Research Group at the University of Oxford for their useful comments. Financial support from the Oxford-Man Institute of Quantitative Finance and Man Group is kindly acknowledged.

References

Abadi, M., A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*.

Bahdanau, D., K. Cho, and Y. Bengio (2014). Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Briola, A., J. Turiel, and T. Aste (2020). Deep learning modeling of the limit order book: A comparative perspective. *Available at SSRN 3714230*.

Briola, A., J. Turiel, R. Marcaccioli, and T. Aste (2021). Deep reinforcement learning for active high frequency trading. *arXiv:2101.07107*.

Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv:1406.1078*.

Chordia, T., R. Roll, and A. Subrahmanyam (2002). Order imbalance, liquidity, and market returns. *Journal of Financial economics 65*(1), 111–130.

Fan, C., Y. Zhang, Y. Pan, X. Li, C. Zhang, R. Yuan, D. Wu, W. Wang, J. Pei, and H. Huang (2019). Multi-horizon time series forecasting with temporal attention learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2527–2535.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gould, M. D., M. A. Porter, S. Williams, M. McDonald, D. J. Fenn, and S. D. Howison (2013). Limit order books. *Quantitative Finance 13*(11), 1709–1742.

Graphcore (Accessed: 2021-04-28). Graphcore. `https://www.graphcore.ai`.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

IPU (Accessed: 2021-04-28). Graphcore ipu. `https://www.graphcore.ai/products/ipu`.

Jha, R., M. De Paepe, S. Holt, J. West, and S. Ng (2020). Deep learning for digital asset limit order books. *arXiv:2010.01241*.

Jia, Z., B. Tillman, M. Maggioni, and D. P. Scarpazza (2019). Dissecting the graphcore IPU architecture via microbenchmarking. *arXiv:1912.03413*.

Luong, M.-T., H. Pham, and C. D. Manning (2015). Effective approaches to attention-based neural machine translation. *arXiv:1508.04025*.

Marcellino, M., J. H. Stock, and M. W. Watson (2006). A comparison of direct and iterated multistep AR methods for forecasting macroeconomic time series. *Journal of econometrics 135*(1-2), 499–526.

Massey Jr, F. J. (1951). The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American statistical Association 46*(253), 68–78.

Ntakaris, A., M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis (2018). Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting 37*(8), 852–866.

Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. (2019). PyTorch: An imperative style, high-performance deep learning library. *arXiv:1912.01703*.

Poplar (Accessed: 2021-04-28). Graphcore poplar. `https://www.graphcore.ai/products/poplar`.

Sadighian, J. (2019). Deep reinforcement learning in cryptocurrency market making. *arXiv:1911.08647*.

Sangadiev, A., R. Rivera-Castro, K. Stepanov, A. Poddubny, K. Bubenchikov, N. Bekezin, P. Pilyugina, and E. Burnaev (2020). DeepFolio: Convolutional neural networks for portfolios with limit order book data. *arXiv:2008.12152*.

Sirignano, J. and R. Cont (2019). Universal features of price formation in financial markets: Perspectives from deep learning. *Quantitative Finance 19*(9), 1449–1459.

Sutskever, I., O. Vinyals, and Q. V. Le (2014). Sequence to sequence learning with neural networks. *arXiv:1409.3215*.

Taieb, S. B., A. Sorjamaa, and G. Bontempi (2010). Multiple-output modeling for multi-step-ahead time series forecasting. *Neurocomputing 73*(10-12), 1950–1957.

Tran, D. T., A. Iosifidis, J. Kanniainen, and M. Gabbouj (2018). Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE transactions on neural networks and learning systems*.

Tsantekidis, A., N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis (2017a). Forecasting stock prices from the limit order book using convolutional neural networks. In *Business Informatics (CBI), 2017 IEEE 19th Conference on*, Volume 1, pp. 7–12. IEEE.

Tsantekidis, A., N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis (2017b). Using deep learning to detect price change indications in financial markets. In *Signal Processing Conference (EUSIPCO), 2017 25th European*, pp. 2511–2515. IEEE.

Tsantekidis, A., N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis (2020). Using deep learning for price prediction by exploiting stationary limit order book features. *Applied Soft Computing 93*, 106401.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). Attention is all you need. *arXiv:1706.03762*.

Wallbridge, J. (2020). Transformers for limit order books. *arXiv:2003.00130*.

Wei, H., Y. Wang, L. Mangu, and K. Decker (2019). Model-based reinforcement learning for predictions and control for limit order books. *arXiv:1910.03743*.

Zhang, Z., B. Lim, and S. Zohren (2021). Deep learning for market by order data. *arXiv:2102.08811*.

Zhang, Z., S. Zohren, and S. Roberts (2018). BDLOB: Bayesian deep convolutional neural networks for limit order books. *Third workshop on Bayesian Deep Learning (NeurIPS 2018)*.

Zhang, Z., S. Zohren, and S. Roberts (2019a). DeepLOB: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing 67*(11), 3001–3012.

Zhang, Z., S. Zohren, and S. Roberts (2019b). Extending deep learning models for limit order books to quantile regression. *Proceedings of Time Series Workshop of the 36 th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019*.

Zhang, Z., S. Zohren, and S. Roberts (2020a). Deep learning for portfolio optimization. *The Journal of Financial Data Science 2*(4), 8–20.

Zhang, Z., S. Zohren, and S. Roberts (2020b). Deep reinforcement learning for trading. *The Journal of Financial Data Science 2*(2), 25–40.