Graph and tensor-train recurrent neural networks for high-dimensional models of limit order books

Jacobo Roa-Vicens
Machine Learning Centre of Excellence
JPMorgan Chase & Co.
London, UK
jacobo.roavicens@jpmorgan.com

Yao Lei Xu
Department of Electrical and Electronic Engineering
Imperial College London
London, UK
yao.xu15@imperial.ac.uk

Ricardo Silva
Department of Statistical Science
University College London
London, UK
ricardo.silva@ucl.ac.uk

Danilo Mandic
Department of Electrical and Electronic Engineering
Imperial College London
London, UK
d.mandic@imperial.ac.uk

ABSTRACT

Recurrent neural networks (RNNs) have proven to be particularly effective for the paradigms of learning and modelling time series. However, sequential data of high dimensions are considerably more difficult and computationally expensive to model, as the number of parameters required to train the RNN grows exponentially with data dimensionality. This is also the case with time series from limit order books, the electronic registries where prices of securities are formed in public markets. To this end, tensorization of neural networks provides an efficient method to reduce the number of model parameters, and has been applied successfully to high-dimensional series such as video sequences and financial time series, for example, using tensor-train RNNs (TTRNNs). However, such TTRNNs suffer from a number of shortcomings, including: (i) model sensitivity to the ordering of core tensor contractions; (ii) training sensitivity to weight initialization; and (iii) exploding or vanishing gradient problems due to the recurrent propagation through the tensor-train topology. Recent studies showed that embedding a multi-linear graph filter to model RNN states (Recurrent Graph Tensor Network, RGTN) provides enhanced flexibility and expressive power to tensor networks, while mitigating the shortcomings of TTRNNs. In this paper, we demonstrate the advantages arising from the use of graph filters to model limit order book sequences of high dimension as compared with the state-of-the-art benchmarks. It is shown that the combination of the graph module (to mitigate problematic gradients) with the radial structure (to make the tensor network architecture flexible) results in substantial improvements in output variance, training time and number of parameters required, without any sacrifice in accuracy.

ACM ICAIF'22, November 2–4, New York, NY

ACM ISBN 978-1-4503-9376-8/22/11…$15.00
https://doi.org/10.1145/3533271.3561710

CCS CONCEPTS

• **Applied computing** → **Mathematics and statistics**.

KEYWORDS

Tensor neural networks, graph networks, machine learning, limit order books, market simulation.

ACM Reference Format:
Jacobo Roa-Vicens, Yao Lei Xu, Ricardo Silva, and Danilo Mandic. 2022. Graph and tensor-train recurrent neural networks for high-dimensional models of limit order books. In *3rd ACM International Conference on AI in Finance, November 2–4, 2022. New York, NY*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3533271.3561710

1 INTRODUCTION

Recurrent Neural Networks (RNNs) have become a *de facto* standard for the processing of sequential data; their power stems from the natural way in which they estimate hidden states that generate sequences of observations. Advanced architectures, such as GRUs and LSTMs, deal more efficiently with some issues experienced with RNNs [4], including the problem of the vanishing gradients and unknown gap lengths, thus improving their learning ability over long sequences. However, RNNs in general become particularly difficult to train on sequential data of high dimensions, as the number of parameters required to train the network increases exponentially with the dimension of the data: the so-called *Curse of Dimensionality*.

Limit order books (LOB) are electronic registries where prices of publicly traded securities are formed, and therefore their dynamics are of great interest in learning and modelling financial data. Such time series data consist on numerous levels of ranked prices and sizes that are offered to buyers (offers) or sellers (bids), and are therefore of inherently high dimension. Modelling LOBs is particularly attractive when learning the dynamics of supply, demand and price formation in public markets, but their high dimensionality poses serious computational challenges to RNNs.

The curse of dimensionality in RNNs can be alleviated by factorising a tensorized version of the weight matrix into its core

Opinions expressed in this paper are those of the authors, and do not necessarily reflect views of JP Morgan.

tensors [8]; this technique has been applied to reduce the number of parameters both in fully connected neural networks for compressing the original models while preserving the expressive power of the network [5], and in recurrent neural networks in the context of video classification [14] and modelling financial series [11]. In particular, both the input and recurrent weight matrices of the RNN benefit from such tensorization.

However, TTRNNs are known to suffer from several shortcomings, including: (i) due to the sequential arrangement of tensor cores in the tensor-train (TT), the TTRNN is highly dependent on the ordering of the input dimensions, whereby the permutation of input dimensions can result in models with varying modelling power [3] [15]; (ii) the TTRNN is highly sensitive to the weight initialization due to the multiplicative nature of the TT, resulting in possibly large variance in model training depending on the initial weights [3] [6] [10] ; (iii) the multiplicative nature of the TT, coupled with the recurrent calculation of gradients in the TTRNN, exacerbates the problem of exploding or vanishing gradients, leading to highly unstable training [9] [10].

On the other hand, graph neural networks have proven their ability to generalize and learn data generation processes of irregular nature, by modelling explicitly dependencies in the underlying data structure. In particular, Graph Tensor Networks expand such results to multi-modal data structures, thus enabling the modelling of multi-way data on irregular domains [12]. The graph component can also be used as an alternative approach to model the propagation of information over time [13]. Recurrent Graph Tensor Networks [12] [13] (RGTNs) provide an alternative to TTRNNs for modelling financial time-series of high dimensions, offering several advantages: (i) the tensor network topology is radial in nature instead of sequential, which makes it independent from the the ordering of the input dimensions; (ii) the RGTN is more robust to weight initialization, since it has no multiplicative cross-core interactions; (iii) the propagation of information over time is modelled through an effective tensor contraction with an appropriate time-graph filter, which is free from the gradient problems associated with RNNs due to the recurrent gradient back-propagation. These advantages are validated in the experiments section.

In this paper, we explore the extent to which these desirable characteristics of RGTNs can be exploited to model limit order book data sequences, a particularly important class of high-dimensional data of irregular, multi-modal nature, that lays at the core of price formation dynamics in public financial markets.

2 BACKGROUND

Advanced versions of recurrent neural networks, such as LSTMs and GRUs, have enhanced the ability of deep learning models to learn increasingly complex sequences in many practical applications [14]. These are able to mitigate the limitations of standard RNNs such as vanishing gradients, and capture richer representations of underlying short and long term components in a signal.

On the other hand, practical applications of RNNs (processing of language or digital video) tend to require modelling data of high dimensions, where RNNs suffer exponential increases of the number of trainable parameters required for each kernel. The tensorization of each kernel allows for substantial mitigation of such explosion, further enriched by the inclusion of graph components to treat the time dimension as follows below.

2.1 Tensors and Graph Tensor Networks

An order-N tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, is a multidimensional array, where I_n is the size of its n-th dimension (or mode), $n = 1, \ldots, N$. Special cases of tensors include matrices, $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$, vectors, $\mathbf{x} \in \mathbb{R}^{I_1}$, and scalars, $x \in \mathbb{R}$, which are tensors of order 2, 1, and 0.

Contraction: Given an order-N tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ and an order-M tensor $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_m \times \cdots \times J_M}$, with equal dimensions $I_n = J_m$, their (m, n) tensor contraction [1], denoted by the operator $\times_n^m$, results in a smaller order-$(N + M - 2)$ tensor, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$, with the entries defined as

$$c_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N,j_1,\ldots,j_{m-1},j_{m+1},\ldots,j_M} = \sum_{i_n=1}^{I_n} a_{i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} b_{j_1,\ldots,j_{m-1},i_n,j_{m+1},\ldots,j_M} \tag{1}$$

Tensor Networks: The notation associated with tensor operations can quickly become cumbersome due to the presence of multi-dimensional indices. To this end, we employ the Tensor Network (TN) notation [2] to represent the tensor operations diagrammatically throughout this paper. A tensor in a TN diagram is represented as a node, where the number of edges that extends from that node corresponds to its tensor order. An edge connecting two nodes represents a contraction between two tensors over modes of equal dimensions. Figure 1 illustrates basic tensor notations while Figure 2 illustrates the tensor contraction operation in TN notation.

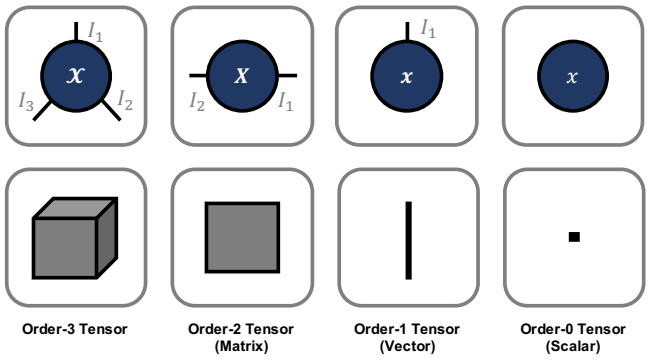

Figure 1: Tensor network representation of scalars, $x \in \mathbb{R}$, vectors, $\mathbf{x} \in \mathbb{R}^{I_1}$, matrices, $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$, and order-3 tensors, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. The number of outgoing edges correspond to the tensor order.

Tensor-Train Neural Networks: Tensor Decomposition (TD) techniques factorize large multi-dimensional tensors through a series of contractions between smaller core tensors. This reduces the computational burden associated with large tensor operations while preserving the underlying multi-dimensional structure, which effectively bypasses the *Curse of Dimensionality*. In particular, the Tensor-Train decomposition (TTD) [8], is a highly efficient TD method that decomposes a large order-$2N$ tensor, $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$,

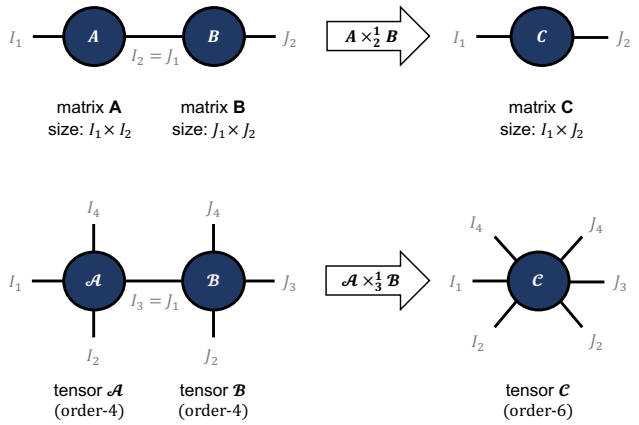

Figure 2: Tensor network representation of a matrix multiplication operation (top) and a tensor contraction operations (bottom).

into smaller contracting core tensors, $\mathcal{G}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$, as $\mathcal{W} = \mathcal{G}^{(1)} \times_4^1 \mathcal{G}^{(2)} \times_4^1 \cdots \times_4^1 \mathcal{G}^{(N)}$, where the set of R_n for $n = 0, \ldots, N$ with $R_0 = R_N = 1$ is known as the *TT-rank*. A Tensorized Neural Network [5] is obtained by reshaping the large weight matrix in Neural Networks, $\mathbf{W}^{I \times J}$, into a tensor $\mathcal{W} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$, where $I = I_1 I_2 \cdots I_N$ and $J = J_1 J_2 \cdots J_N$. In turn, such a tensor can be decomposed and stored in a low-rank TTD format.

This reduces the space complexity of the weight matrix from an exponential one, $\prod_{n=1}^{N} I_n J_n$, to a linear one, $\sum_{n=1}^{N} R_{n-1} I_n J_n R_n$ (in tensor dimensions) with a compression ratio [14] of:

$$\frac{\sum_{n=1}^{N} R_{n-1} I_n J_n R_n}{\prod_{n=1}^{N} I_n J_n} \tag{2}$$

In this way, the forward pass is effectively tensorized from a standard matrix-by-vector multiplication, $\mathbf{y} = \mathbf{W}\mathbf{x}$, to a series of tensor contractions:

$$\begin{aligned} \mathcal{Y} &= \mathcal{X} \times_{2,3,4,\cdots,N+1}^{1,3,5,\cdots,2N-1} \mathcal{W} \\ &= \mathcal{X} \times_{2,3,4,\cdots,N+1}^{1,3,5,\cdots,2N-1} \left(\mathcal{G}_1 \times_4^1 \mathcal{G}_2 \times_4^1 \cdots \times_4^1 \mathcal{G}_N \right) \\ &= \mathcal{X} \times_2^2 \mathcal{G}_1 \times_{2,N+3}^{2,1} \mathcal{G}_2 \times_{2,N+3}^{2,1} \cdots \times_{2,N+3}^{2,1} \mathcal{G}_N \end{aligned} \tag{3}$$

Figure 3 (left) illustrates the TT Neural Network (TTNN) forward pass in TN notation. The authors in [14] extended this idea to Recurrent Neural Networks, where the same notion of tensorization was applied to both the input and recurrent weight matrices.

Graph Tensor Networks: Given a multi-dimensional time-series tensor, $\mathcal{X} \in \mathbb{R}^{L \times I_1 \times \cdots \times I_N}$, where features of dimensionality $I_1 \times \cdots \times I_N$ are indexed along L time-steps, a Recurrent Graph Tensor Network [12] [13] computes the forward pass, $\mathcal{Z} = (\mathbf{I} + \mathbf{A}) \times_2^1 \mathcal{X}$, to extract time-series features, $\mathcal{Z} \in \mathbb{R}^{L \times I_1 \times \cdots \times I_N}$, through a time-domain graph adjacency matrix, $\mathbf{A} \in \mathbb{R}^{L \times L}$.

Specifically, the time-domain graph treats each of the L time-steps as a node of the graph, where the directed edges connecting the time-steps determine the direction and the magnitude of information between two points in time. The contraction of the input tensor with the adjacency matrix, $\mathbf{A} \in \mathbb{R}^{L \times L}$, along the time dimension can therefore be interpreted as a directed propagation of time-series information over time [13]. This propagation takes place simultaneously across all timestamps, removing the need for sequential processing of each individual timestamp.

Finally, the remaining feature dimensions, $I_1 \times \cdots \times I_N$, can be processed by a series of tensor contractions with trainable cores, $\mathbf{G}^{(n)} \in \mathbb{R}^{I_n \times J_n}$, resulting in the final forward pass:

$$\begin{aligned} \mathcal{Y} &= \left((\mathbf{I} + \mathbf{A}) \times_2^1 \mathcal{X} \right) \times_2^1 \mathbf{G}^{(1)} \times_3^1 \mathbf{G}^{(2)} \times_4^1 \cdots \times_{N+1}^1 \mathbf{G}^{(N)} \\ &= \mathcal{Z} \times_2^1 \mathbf{G}^{(1)} \times_3^1 \mathbf{G}^{(2)} \times_4^1 \cdots \times_{N+1}^1 \mathbf{G}^{(N)} \end{aligned} \tag{4}$$

The tensor network diagram for the above forward pass is shown in Figure 3 (right). The space complexity of RGTN is $\sum_{n=1}^{N} I_n J_n$, resulting in a compression ratio of $\dfrac{\sum_{n=1}^{N} I_n J_n}{\prod_{n=1}^{N} I_n J_n}$.

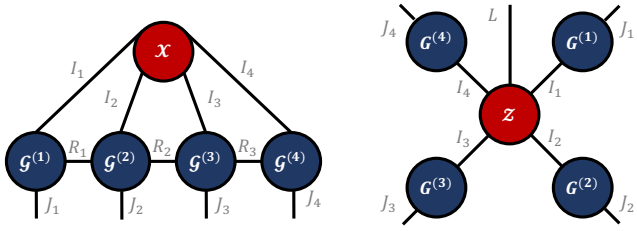

Figure 3: TN diagram of the TTNN forward pass (left) according to Eq. (3) and the TN diagram of the GTN forward pass (right) according to Eq. (4). Notice how the TTNN cores, $\mathcal{G}^{(n)}$, are sequentially arranged, while the GTN cores, $\mathbf{G}^{(n)}$, are radially arranged. This implies that the order in the sequence of tensor dimensions matters in the TTNN case but not in the GTN case.

2.2 Limit Order Book Time Series

Limit order books (LOB) play a central role in the formation of prices for financial securities in exchanges globally. These systems gather limit orders to buy or sell certain prices and sizes of securities from large numbers of dealers and investors, centralizing the orders in a public, electronic registry to match bids and offers in an automatic, transparent process. This function makes the understanding of order book dynamics especially interesting to model. The aggregation of prices and sizes for buy and sell orders across various levels results in non-stationary time series of high dimensions and multimodal nature, which poses a challenge for standard machine learning models. In addition, scaling up to higher dimensions implies an exponential increase in the number of parameters required, besides the associated increase of computational complexity.

3 EXPERIMENTS

The purpose of our experiments is to investigate the added value of including graph components in the tensorized RNN (producing an RGTN) when modelling the time series of limit order book data. Series of LOB states tend to be of high dimension, given the various levels of depth from the best bid (order to buy) and best ask

Size	Price (USD)	Total (USD)
420.00	$ 19.3300	$ 8,119
280.00	$ 19.3200	$ 5,410
300.00	$ 19.3100	$ 5,793
295.00	$ 19.3000	$ 5,694
210.00	$ 19.2900	$ 4,051
Current Price:	$ 19.2892	
150.00	$ 19.2800	$ 2,892
290.00	$ 19.2700	$ 5,588
110.00	$ 19.2600	$ 2,119
300.00	$ 19.2500	$ 5,775
275.00	$ 19.2400	$ 5,291

Figure 4: Limit Order Book: Five levels of bid and ask prices and sizes.

(order to sell) in the book, with every level containing price and size for bid and ask. We consider a LOB of $L = 10$ levels, whose input dimension is therefore 40 (bid/ask prices and bid/ask sizes for each level). Finally, as customary in modelling sequential data with recurrent neural networks, to build the training sample for each timestep with the N last data points available we use a look-back window of $N = 20$, resulting in an input dimension of $D = 800$, where the added value of the tensor-train structure becomes self-evident.

LOB level	Ask price	Ask size	Bid price	Bid size
1	585.94	200	585.33	18
...	...	...	...	...
10	587.90	500	584.27	300

Table 1: Sample LOB state structure with 40 dimensions for a given timestamp of AAPL stock on June 26th, 2012. Source: `lobsterdata.com` [7]

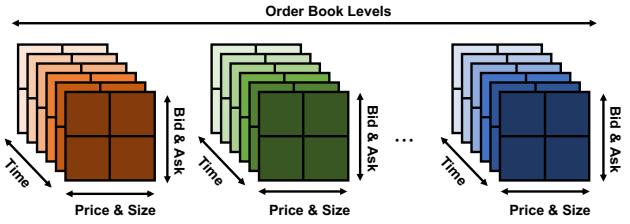

Figure 5: Tensor structure of the order book. The order book data of a stock at a given point in time contains the buy side (bid) and sell side (ask) information about pricing and the size. The l-th best bid and ask available are contained within the l-th level of the order book. By considering the order book information across L levels over T time steps, we obtain order-4 data tensors of [time] × [price and size] × [bid and ask] × [order book levels].

In order to evaluate the performance of each model on each stock over 40 dimensions, we considered a specific statistic to summarize the state of the order book at each timestamp: the Volume Weighted Average Price (VWAP) derived from the order book of each stock, as defined in equation (5). This quantity defines the average price for each stock weighted against the order sizes available for each price across the order book, thus summarizing the state of the supply and demand liquidities. Considering L levels of the order book, the VWAP at time t is defined as:

$$VWAP_L(t) = \frac{\sum_{l=1}^{L} p_l(t) V_l(t)}{\sum_{l=1}^{L} V_l(t)} \tag{5}$$

where $p_l(t)$ and $V_l(t)$ are the price and the size available at l-th depth at time t. In particular, we tested the ability of each model (RGTN, TTRNN and GRU) to learn the VWAP of each order book across 5 different stocks independently.

3.1 Data

We trained and tested our models on limit order book sample data supplied by LOBSTER, an online limit order book data tool which provides easy-to-use, high-quality limit order book data for academic research purposes [7] (`lobsterdata.com`).

The data consist of two types of input files:

- **Messages** containing the arrival of specific events that update the state of the order book, such as new orders, modifications, cancellations, etc. and their respective price, size and side (buy or sell). Timestamps are defined per seconds after midnight, down to microsecond precision. Sizes are number of shares, and prices represent USD prices times 10,000. Sides take either buy (+1) or sell (-1) values.

Timestamp	OrderID	Size	Price	Side	Stock
34200.0042	16113575	18	5853300	+1	AAPL
34200.0255	16120480	10	5859200	-1	AAPL
...	...	...	...	...	...
57599.9131	287150868	48	5776100	-1	AAPL

Table 2: Sample of LOB messages for AAPL

- **Order books** contain the evolution of the state of the order book for up to $L = 10$ levels of prices and sizes, updated per the events that arrive through the order messages.

In particular, it contains LOB data series for five stocks (Apple, Amazon, Google, Intel, Microsoft) during the trading session of June 26th, 2012 which consists of the registry of order messages for the day, and the sequence of LOB states that derives from such order messages up to 10 levels of bids and offers, respectively:

The LOB states consist of asynchronous time series, updated upon arrival of new orders with random distributions in time. For each stock series, we compute the value of VWAP following Eq. (5) for every second, sampling the state of the order book on that timestamp. Data from messages containing price and size orders is implicit in the LOB states series as the former imply the changes in the latter.

Stock	Ticker	Messages	Time span
Apple	AAPL	400,391	9:30:00.00 - 15:59:59.91
Amazon	AMZN	269,748	9:30:00.02 - 15:59:59.96
Google	GOOG	147,916	9:30:00.02 - 15:59:59.87
Intel	INTC	624,040	9:30:00.01 - 15:59:59.95
Microsoft	MSFT	668,765	9:30:00.01 - 15:59:59.91

Table 3: Order inputs for each stock considered.

3.2 Models and training

The experiments compared three models:

- **Gated Recurrent Unit**, the mainstream GRU class of recurrent neural network, as implemented in Keras TensorFlow 2, serves as benchmark of performance with respect to the two tensor networks.
- **Tensor-train Recurrent Neural Network** (TTRNN), a recurrent neural network structured to process a tensor input through kernels defined by the sequential contraction of core tensors. This allows for a substantial reduction in the number of parameters required.
- **Recurrent Graph Tensor Network** (RGTN), a tensor network including a time-domain graph adjacency matrix for the extraction of time-series features in the forward pass, as explained in Section 2.1.

Each of the three neural models was trained independently over 100 epochs for each of the five stocks listed in Table 3, resulting in 15 experiments. The training data split consists of 70% of the LOB data (of which 20% was used for validation), and the remaining 30% to test the outputs of the trained models. Each experiment was then repeated 30 times with different random seeds. Weights were initialized with the default Xavier method in `tf.keras`. Table 4 summarizes the dimensional structure of each model.

Layer	Input Shape	Output Shape	Parameters
GRU	20×40	40	9840
Dense	40	1	41

GRU Model summary.

Layer	Input Shape	Output Shape	Parameters
TTRNN	$20 \times 1 \times 10 \times 2 \times 2$	$1 \times 1 \times 10 \times 2 \times 2$	852
Flatten	$1 \times 1 \times 10 \times 2 \times 2$	40	0
Dense	40	1	41

TTRNN Model summary.

Layer	Input Shape	Output Shape	Parameters
RGTN	$20 \times 1 \times 10 \times 2 \times 2$	$1 \times 1 \times 10 \times 2 \times 2$	109
Flatten	$1 \times 1 \times 10 \times 2 \times 2$	40	0
Dense	40	1	41

RGTN Model summary.

Table 4: Model summaries for 40-dimensional LOB states times 20-timestep lookback, with number of trainable parameters and scalar output dimension for VWAP.

The performance was evaluated through various metrics, such as the mean standard error (MSE) of the output, its standard deviation, the average training time and the number of trainable parameters required for each model and stock considered. Results demonstrate a consistent superior performance of our proposed model across the considered five cases for each of these metrics. Next sections develop the details of the data used, each of the models, the success metrics and their conclusions.

4 ANALYSIS OF EXPERIMENTAL RESULTS

The outcomes of our experiments were consistent across the five stocks studied with respect to the advantages of employing the RGTN model structure. Table 5 summarizes the results for all the metrics considered in each case. The experimental results obtained were as follows:

- **Mean Standard Error**. Both tensor-based models outperformed the GRU with three- to four-fold improvements in the MSE when modelling the VWAP; the MSE for the RGTN was also slightly better than TTRNN across the five stocks. The MSE for each experiment was the average MSE obtained across the 30 independent repetitions of the experiment with different random seeds.
- **Variance**. The variance of the test-MSE across the 30 repetitions of each experiment is the variable that shows the greatest improvement when using the RGTN instead of the TTRNN, with improvements of around one order of magnitude of standard deviations with respect to the TTRNN model on average, and far superior to those of the GRU across all the stocks considered. Figure 6 displays the MSE distributions compared across all tensor models.
- **Training time**. The TTRNN took the longest to train, with well above 4h in all cases. While the GRU nearly halves that time, the RGTN concluded its training within 40 minutes in all cases, implying between five- and nine-fold training time improvements.
- **Trainable parameters**. Reducing the number of trainable parameters is the core motivation for the use of tensor networks in high-dimensional problems, and the TTRNN structures required in our experiments barely 9% the number of trainable parameters of the GRU counterpart. Including the graph component results in a further improvement, where the RGTN required around 17% of the trainable parameters of the TTRNN (1.52% of those of the GRU).
- **Gradients**. As discussed in previous sections, a shortcoming of TTRNNs is the phenomenon of exploding or vanishing gradients. We observed various such cases while training the TTRNNs for these experiments, while none of the RGTN cases exhibited such problematic gradients. This reinforces further the case for the use of RGTNs instead of TTRNNs for data and experiments of this nature. See Figure 7 for details of the gradient evolution for each experiment.

5 CONCLUSIONS

The experiments presented in this paper illustrate the advantages of combining a graph component with tensorized recurrent neural

networks (RGTN) when modelling time series of limit order books, which are inherently high-dimensional.

The results obtained with respect to the advantages of employing RGTNs are consistent across the five independent experiments, each conducted by training and testing the same model structure with the time series of a different underlying stock and its associated VWAP series. In all the cases where the RGTN is used, we find a substantial improvement in terms of variance of MSE, training time, and number of trainable parameters required, as well as improvements in the MSE results, when compared to the alternative GRU and TTRNN models.

Model	MSE	Training time	Parameters
GRU	1.4950 ± 0.0648	2:33:13	9881
TTRNN	0.4045 ± 0.0033	4:30:49	893
RGTN	0.3976 ± 0.0003	0:36:03	150

Results for AAPL

Model	MSE	Training time	Parameters
GRU	1.7814 ± 0.0845	2:32:12	9881
TTRNN	0.4720 ± 0.0021	4:33:03	893
RGTN	0.4686 ± 0.0002	0:37:41	150

Results for AMZN

Model	MSE	Training time	Parameters
GRU	1.5374 ± 0.0850	2:32:47	9881
TTRNN	0.3590 ± 0.0026	4:31:17	893
RGTN	0.3521 ± 0.0004	0:35:46	150

Results for GOOG

Model	MSE	Training time	Parameters
GRU	0.5703 ± 0.1297	2:33:40	9881
TTRNN	0.1834 ± 0.0014	4:30:38	893
RGTN	0.1801 ± 0.0002	0:36:41	150

Results for INTC

Model	MSE	Training time	Parameters
GRU	1.2437 ± 0.1899	2:34:08	9881
TTRNN	0.3972 ± 0.0093	4:31:09	893
RGTN	0.3819 ± 0.0005	0:36:26	150

Results for MSFT

Table 5: Summary of experimental results across models and stocks: MSE within a standard deviation, training time, and number of trainable parameters.

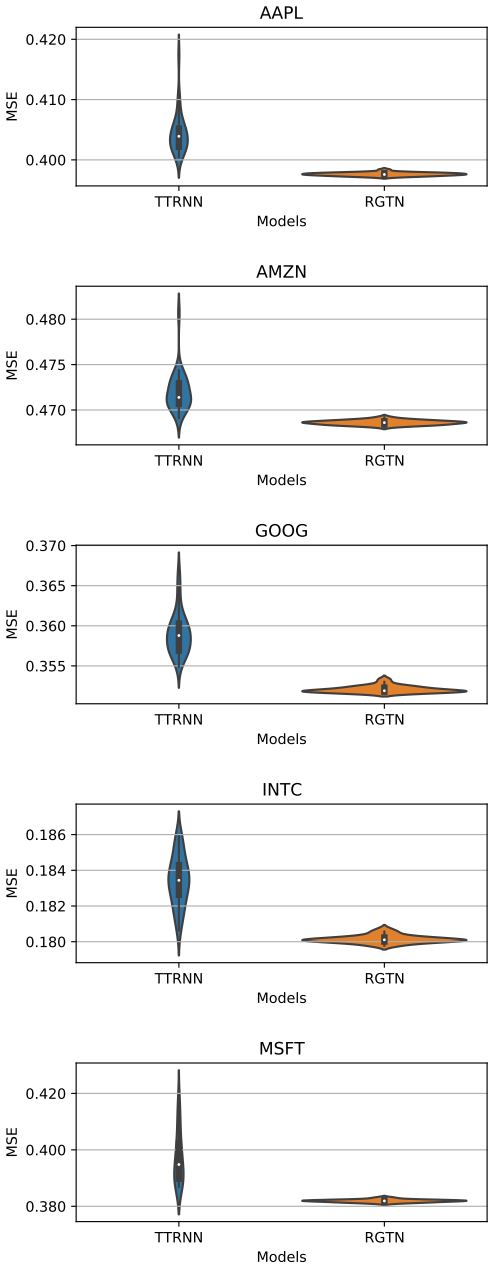

Figure 6: Mean squared errors and their variances. The RGTN models provide lower mean squared errors and smaller output variances than TTRNN for each of the five stocks considered. Both outperform significantly the GRU benchmark model.

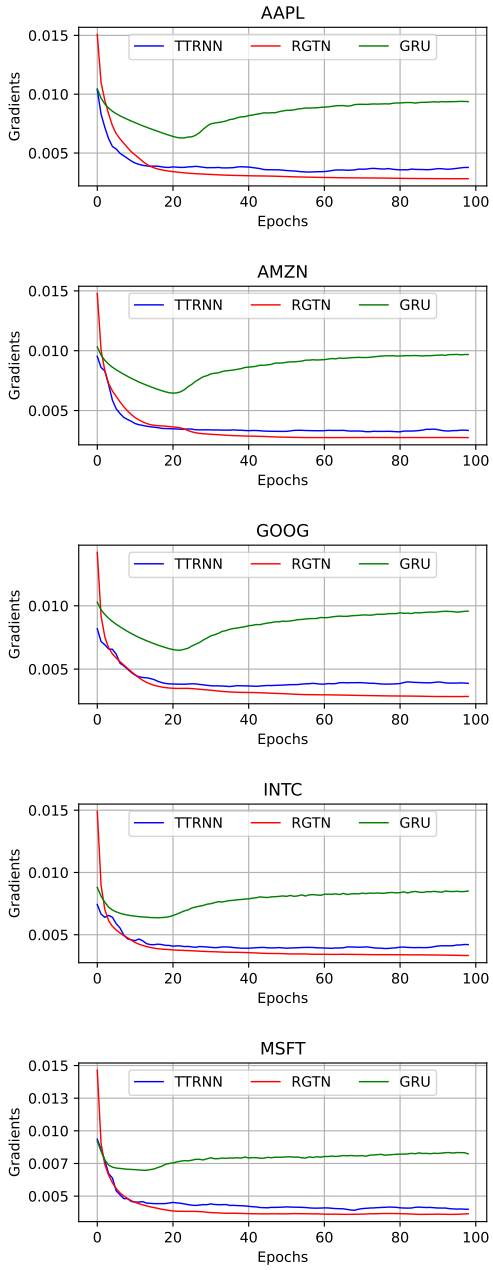

Figure 7: Gradients. Evolution of gradients of each model considered across training epochs. Tensorized models display better behavior, with fastest convergence of gradients in each instance of RGTN experiments.

REFERENCES

[1] Andrzej Cichocki. 2013. Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions. In *Proceedings of the International Workshop on Smart Info-Media Systems in Asia.*

[2] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P. Mandic. 2016. Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning* 9, 4-5 (2016), 249–429.

[3] Alexandros Haliassos, Kriton Konstantinidis, and Danilo P. Mandic. 2021. Supervised Learning for Nonsequential Data: A Canonical Polyadic Decomposition Approach. *IEEE Transactions on Neural Networks and Learning Systems* (2021), 1–15. https://doi.org/10.1109/TNNLS.2021.3069399

[4] Danilo P. Mandic and Jonathan A. Chambers. 2001. *Recurrent neural networks for prediction: Learning algorithms, architectures and stability.* John Wiley, Inc.

[5] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry P. Vetrov. 2015. Tensorizing Neural Networks. *CoRR* abs/1509.06569 (2015). arXiv:1509.06569 http://arxiv.org/abs/1509.06569

[6] Alexander Novikov, Mikhail Trofimov, and Ivan Oseledets. 2018. Exponential machines. *Bulletin of the Polish Academy of Sciences: Technical Sciences* (2018), 789–797.

[7] Frischedaten UG. Chair of Econometrics; School of Business and Economics; Humboldt University zu Berlin. 2012. LOBSTER Limit Order Book System; Academic data. *https://lobsterdata.com* (2012).

[8] Ivan Oseledets. 2011. Tensor-train decomposition. *SIAM J. Sci. Comput.* 33, 5 (2011), 2295–2317. https://doi.org/10.1137/090752286

[9] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. PMLR, 1310–1318.

[10] Jinhui Wang, Chase Roberts, Guifre Vidal, and Stefan Leichenauer. 2020. Anomaly detection with tensor networks. *arXiv preprint arXiv:2006.02516* (2020).

[11] Yao Lei Xu, Giuseppe Giovanni Calvi, and Danilo P. Mandic. 2021. Tensor-Train Recurrent Neural Networks for Interpretable Multi-Way Financial Forecasting. In *International Joint Conference on Neural Networks, IJCNN 2021, Shenzhen, China, July 18-22, 2021*. IEEE, 1–5. https://doi.org/10.1109/IJCNN52387.2021.9534120

[12] Yao Lei Xu, Kriton Konstantinidis, and Danilo P. Mandic. 2020. Multi-Graph Tensor Networks. In *First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS 2020) (NeurIPS 2020).*

[13] Yao Lei Xu and Danilo P. Mandic. 2021. Recurrent Graph Tensor Networks: A Low-Complexity Framework for Modelling High-Dimensional Multi-Way Sequence. *29th European Signal Processing Conference (EUSIPCO)* abs/2009.08727 (2021). https://arxiv.org/abs/2009.08727

[14] Yinchong Yang, Denis Krompass, and Volker Tresp. 2017. Tensor-Train Recurrent Neural Networks for Video Classification. *CoRR* abs/1707.01786 (2017). arXiv:1707.01786 http://arxiv.org/abs/1707.01786

[15] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. 2016. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535* (2016).