# Recurrent Graph Tensor Networks: A Low-Complexity Framework for Modelling High-Dimensional Multi-Way Sequences

Yao Lei Xu, Danilo P. Mandic
*Department of Electrical and Electronic Engineering*
*Imperial College London*
London, United Kingdom
{yao.xu15, d.mandic}@imperial.ac.uk

***Abstract*—Recurrent Neural Networks (RNNs) are among the most successful machine learning models for sequence modelling, but tend to suffer from an exponential increase in the number of parameters when dealing with large multidimensional data. To this end, we develop a multi-linear graph filter framework for approximating the modelling of hidden states in RNNs, which is embedded in a tensor network architecture to improve modelling power and reduce parameter complexity, resulting in a novel Recurrent Graph Tensor Network (RGTN). The proposed framework is validated through several multi-way sequence modelling tasks and benchmarked against traditional RNNs. By virtue of the domain aware information processing of graph filters and the expressive power of tensor networks, we show that the proposed RGTN is capable of not only outperforming standard RNNs, but also mitigating the Curse of Dimensionality associated with traditional RNNs, demonstrating superior properties in terms of performance and complexity.**

***Index Terms*—Recurrent Graph Tensor Networks, Tensor Networks, Tensor Decomposition, Graph Neural Networks, Recurrent Neural Networks.**

## I. Introduction

Graphs and tensors have found numerous applications in deep learning systems. In this context, graph based methods have been used to generalize classical convolutional neural networks to irregular data domains, with graph neural networks achieving state-of-the-art results in a number of applications [1]. On the other hand, tensor methods have been used to relax the computational complexity of neural networks [2], as well as to alleviate their notorious "black-box" nature [3], [4]. These promising results have also highlighted a void in literature regarding the combination of both techniques in order to solve deep learning challenges, especially in the area of sequence modelling. To this end, we introduce a novel Recurrent Graph Tensor Network (RGTN) framework for multi-way time-series modelling, which enhances the sequence modelling ability of Recurrent Neural Networks (RNNs) [5] through tensor- and graph-theoretic concepts.

The field of Graph Data Analytics (GDA) generalizes traditional signal processing concepts to irregular domains [6]–[8], which are naturally represented as graphs. Developments in GDA have led to a range of spatial and spectral based techniques that generalize the notion of frequency and locality to irregular data, allowing for the processing of signals while taking into account the underlying data domain [9]. Several concepts developed in GDA have found applications in deep learning, where graph filters can be implemented across multiple graph neural network layers to incorporate graph topology information [1].

Tensors are multi-linear generalization of vectors and matrices to multi-way arrays, which allows for a richer representation by not limiting the data to the classical "flat-view" matrix approaches [10]. Recent developments in tensor manipulation have led to Tensor Decomposition (TD) techniques that can represent high dimensional tensors through a contracting network of smaller core tensors. Such TD techniques can be used to compress the number of parameters needed to represent high-dimensional data, and have already found applications in deep learning. Notably, it has been shown that TD techniques, such as the Tensor-Train Decomposition (TTD) [11], can be used to compress neural networks considerably while maintaining comparable performance [2], [12], [13].

However, despite promising results achieved in both individual fields, the full potential arising from the combination of graphs, tensors, and neural networks is yet to be explored, especially in the area of sequence modelling. To this end, we set out to investigate the extent to which a careful domain consideration of tensors and graphs can improve the complexity and performance of RNNs, by leveraging the theoretical frameworks underpinning graph machine learning and tensor networks. More specifically, we establish a novel structure for the modelling of RNN hidden states through a multi-linear graph filter embedded in a tensor network architecture, leading to a novel *Recurrent Graph Tensor Network* (RGTN) framework. The so derived RGTN exploits both the ability of graphs to process data defined on irregular time-domains and the expressive power of tensor decomposition, resulting in a new class of expressive models with drastically lower complexity compared to standard RNNs. Our experimental results confirm the superiority of the proposed RGTN models, demonstrating desirable properties in terms of both performance and complexity across several multi-way sequence modelling tasks.

## II. Theoretical Background

### A. Spatial Graph Filters

A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is defined by a set of $N$ vertices (or nodes) $v_n \subset \mathcal{V}$ for $n = 1, \ldots, N$, and a set of edges connecting the $n^{th}$ and $m^{th}$ vertices $e_{nm} = (v_n, v_m) \in \mathcal{E}$, for $n = 1, \ldots, N$ and $m = 1, \ldots, N$. A signal on a given graph is defined by a vector $\mathbf{f} \in \mathbb{R}^N$ such that $\mathbf{f} : \mathcal{V} \to \mathbb{R}$, which associates a signal value to every node on the graph [6].

A given graph can be fully described in terms of its weighted adjacency matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$, such that $a_{nm} > 0$ if $e_{nm} \in \mathcal{E}$, and $a_{nm} = 0$ if $e_{nm} \notin \mathcal{E}$. The normalized weighted adjacency matrix is defined as $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D} \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix such that $d_{nn} = \sum_m a_{nm}$ [6]. The weighted adjacency matrix can be used as a shift operator to filter a set of $M$ signals on a graph with $N$ vertices, $\mathbf{X} \in \mathbb{R}^{N \times M}$, as $\mathbf{Y} = \sum_{k=0}^{K-1} \alpha_k \mathbf{A}^k \mathbf{X}$. Such a spatial graph filter represents a linear combination of vertex-shifted graph signals, which captures graph information at a local level [7].

### B. Tensors and Tensor Networks

An order-$N$ tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, represents an $N$-way array with $N$ modes, where the $n^{th}$ mode is of size $I_n$, for $n = 1, 2, \ldots, N$. Special instances 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 respectively tensors of order-2, 1, and 0. The $(i_1, i_2, \ldots, i_N)$ entry of a tensor is denoted by $x_{i_1 i_2 \cdots i_N} \in \mathbb{R}$. A matrix can be *reshaped* into a tensor through a process known as *tensorization* [10], denoted by the operator $\text{ten}(\cdot)$. A tensor can also be reshaped into a vector through the *vectorization* process, denoted by the operator $\text{vec}(\cdot)$. The tensor indices in this paper are grouped according to the Little-Endian convention [14].

An $(m, n)$-contraction [10], denoted by $\times_n^m$, between an $N$-th order tensor, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$, and an $M$-th order tensor, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_m \times \cdots \times J_M}$, with equal dimensions $I_n = J_m$, yields a tensor of order $(N + M - 2)$, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$, with entries defined as:

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

The contraction, $\mathbf{A} \times_2^1 \mathbf{B}$, denotes the standard matrix multiplication between $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{B} \in \mathbb{R}^{J_1 \times J_2}$, where $I_2 = J_1$.

A (left) Kronecker product between two tensors, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$, denoted by $\otimes$, yields a tensor of the same order, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_N J_N}$, with entries $c_{\overline{i_1 j_1}, \ldots, \overline{i_N j_N}} = a_{i_1 \ldots i_N} b_{j_1 \ldots j_N}$, where $\overline{i_n j_n} = j_n + (i_n - 1) J_n$ [10]. For the special case of matrices $\mathbf{A} \in \mathbb{R}^{I_1 \times I_2}$ and $\mathbf{B} \in \mathbb{R}^{J_1 \times J_2}$, the Kronecker product yields a block-matrix:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{i_1 i_2}\mathbf{B} & \cdots & a_{i_1 I_2}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{I_1 i_2}\mathbf{B} & \cdots & a_{I_1 I_2}\mathbf{B} \end{bmatrix} \tag{2}$$

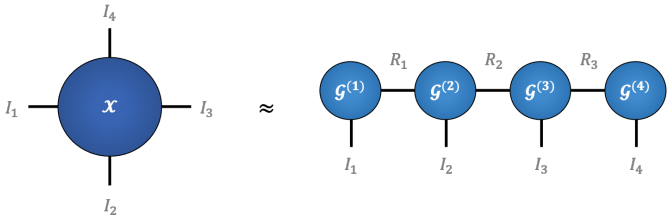


Fig. 1. Tensor Network diagram of Tensor-Train Decomposition for an order-4 tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times I_4}$, according to (3). The dimensionality of the tensors are denoted in gray letters.

A Tensor Network (TN) admits a graphical representation of tensor contractions, whereby each tensor is represented as a node, while the number of edges that extend from that node corresponds to the tensor order [15]. An edge connecting two nodes represents a linear contraction over modes of equal dimensions between the connected tensors.

Special instances of tensor networks include Tensor Decomposition (TD) networks. Such TD methods approximate high-order, large-dimensional tensors via contractions of smaller core tensors, which reduces the computational complexity drastically while preserving the data structure [15], [16]. For instance, the Tensor-Train (TT) decomposition [17] [11] is a highly efficient TD method that can decompose a large order-$N$ tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, into $N$ smaller core tensors, $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$, as:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}}^{(1)} \times_2^1 \boldsymbol{\mathcal{G}}^{(2)} \times_3^1 \boldsymbol{\mathcal{G}}^{(3)} \times_3^1 \cdots \times_3^1 \boldsymbol{\mathcal{G}}^{(N)} \tag{3}$$

where the set of $R_n$ for $n = 0, \ldots, N$ and $R_0 = R_N = 1$ is referred to as the *TT-rank*. By virtue of TT, the number of entries in the original tensor is drastically reduced from an exponential $\prod_{n=1}^{N} I_n$ to a linear $\sum_{n=1}^{N} R_{n-1} I_n R_n$ in the dimensions $I_n$, which is highly efficient for high $N$ and low TT-rank. An illustration of TT decomposition in TN notation is provided in Figure 1.

### C. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [5] [18] are among the most successful deep learning tools for sequence modelling. A standard RNN layer captures time-varying dependencies by processing hidden states, $\mathbf{h}_t \in \mathbb{R}^M$, at time $t$ through *feedback* (or *recurrent*) weights as:

$$\mathbf{h}_t = \sigma_h(\mathbf{W}^{(h)}\mathbf{h}_{t-1} + \mathbf{W}^{(x)}\mathbf{x}_t + \mathbf{b}^{(h)}) \tag{4}$$

where $\mathbf{h}_{t-1} \in \mathbb{R}^M$ is the hidden state vector from the previous time-step, $\mathbf{x}_t \in \mathbb{R}^N$ is the input features vector at time $t$, $\mathbf{W}^{(h)} \in \mathbb{R}^{M \times M}$ is the feedback matrix, $\mathbf{W}^{(x)} \in \mathbb{R}^{M \times N}$ is the input weight matrix, $\mathbf{b}^{(h)} \in \mathbb{R}^M$ is an optional bias vector, and $\sigma_h(\cdot)$ is an optional element-wise activation function.

Finally, after extracting the hidden states, these can be passed through additional weight matrices to generate outputs, $\mathbf{y}_t \in \mathbb{R}^P$ at time $t$, in the form:

$$\mathbf{y}_t = \sigma_y(\mathbf{W}^{(y)}\mathbf{h}_t + \mathbf{b}^{(y)}) \tag{5}$$

where $\mathbf{W}^{(y)} \in \mathbb{R}^{P \times M}$ is the output weight matrix, $\mathbf{h}_t$ is the hidden state at time $t$, $\mathbf{b}^{(y)}$ is an optional bias vector, and $\sigma_y(\cdot)$ is an optional element-wise activation function.

## III. RECURRENT GRAPH TENSOR NETWORKS

### A. General Recurrent Graph Tensor Networks

Consider the RNN forward pass in (4) without the optional bias vector and activation function:

$$\mathbf{h}_t = \mathbf{W}^{(h)}\mathbf{h}_{t-1} + \mathbf{W}^{(x)}\mathbf{x}_t \tag{6}$$

Denote $\hat{\mathbf{x}}_t = \mathbf{W}^{(x)}\mathbf{x}_t \in \mathbb{R}^M$, for $t = 1, \ldots, \tau$ time-steps; then (6) can be written in a block-matrix form:

$$\begin{bmatrix} \mathbf{h}_\tau \\ \mathbf{h}_{\tau-1} \\ \vdots \\ \mathbf{h}_1 \end{bmatrix} = \begin{bmatrix} (\mathbf{W}^{(h)})^0 & (\mathbf{W}^{(h)})^1 & \cdots & (\mathbf{W}^{(h)})^{\tau-1} \\ 0 & (\mathbf{W}^{(h)})^0 & \cdots & (\mathbf{W}^{(h)})^{\tau-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & (\mathbf{W}^{(h)})^0 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_\tau \\ \hat{\mathbf{x}}_{\tau-1} \\ \vdots \\ \hat{\mathbf{x}}_1 \end{bmatrix} \tag{7}$$

We now define: (i) $\mathbf{X} \in \mathbb{R}^{\tau \times N}$, as the input matrix generated by stacking row-vectors, $\mathbf{x}_t$, over $\tau$ successive time-steps; (ii) $\hat{\mathbf{X}} \in \mathbb{R}^{\tau \times M}$, as $\hat{\mathbf{X}} = \mathbf{X} \times_2^2 \mathbf{W}^{(x)}$; (iii) $\mathbf{H} \in \mathbb{R}^{\tau \times M}$, as the matrix generated by stacking hidden state vectors, $\mathbf{h}_t$, as row-vectors over $\tau$ time-steps; and (iv) $\mathbf{R} \in \mathbb{R}^{\tau M \times \tau M}$, as the block matrix composed by the powers of $\mathbf{W}^{(h)}$ from (7). This allows (7) to be expressed compactly as:

$$\mathrm{vec}(\mathbf{H}) = \mathbf{R} \times_2^1 \mathrm{vec}(\hat{\mathbf{X}}) \tag{8}$$

Without loss of generality, we shall further restrict the feedback matrix, $\mathbf{W}^{(h)}$, to be a scaled idempotent matrix, that is $\mathbf{W}^{(h)} = c\mathbf{W}^{(r)}$, where $c$ is a positive scaling constant strictly less than 1, and $\mathbf{W}^{(r)}$ is an idempotent matrix that models how information propagates between successive time-steps. For this setup, the feedback matrix has the property $(\mathbf{W}^{(h)})^p = c^p\mathbf{W}^{(r)}$, for $p > 0$. This allows the block matrix $\mathbf{R}$ to be decomposed as:

$$\mathbf{R} = \mathbf{I} + \mathbf{A} \otimes \mathbf{W}^{(r)} \tag{9}$$

where $\mathbf{A} \in \mathbb{R}^{\tau \times \tau}$ contains the constants $c^p$, as:

$$\mathbf{A} = \begin{bmatrix} 0 & c^1 & \cdots & c^{\tau-1} \\ 0 & 0 & \cdots & c^{\tau-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \tag{10}$$

Note that the matrix, $\mathbf{A}$, can be interpreted as the weighted graph adjacency matrix connecting $\tau$ successive time-steps as vertices (nodes). This also justifies its triangular (directed) nature, since only past information can influence future states but not vice-versa.

We now denote, $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{\tau \times M \times \tau \times M}$, as the 4-th order tensorization of $\mathbf{R}$, that is $\boldsymbol{\mathcal{R}} = \mathrm{ten}(\mathbf{I} + \mathbf{A} \otimes \mathbf{W}^{(r)})$; this allows us to simplify the expression in (8) via a double tensor contraction, and express the general Recurrent Graph Tensor Network filtering operation in its complete form as:

$$\mathbf{H} = \boldsymbol{\mathcal{R}} \times_{3,4}^{1,2} \mathbf{X} \times_2^2 \mathbf{W}^{(x)} \tag{11}$$

The proposed filtering operation in (11) can be used to extract features from time-series data, $\mathbf{X}$, in a neural network. We will refer to such neural network models as general Recurrent Graph Tensor Networks (gRGTN).

### B. Simplified Recurrent Graph Tensor Networks

To establish a link between the proposed RGTN filtering operation and classical spatial graph filters, we shall now consider a special case of equation (11).

Consider a special case where $\mathbf{W}^{(r)} = \mathbf{I}$. This implies that $\mathbf{W}^{(h)} = c\mathbf{I}$, which simplifies the hidden state evolution in (6) as $\mathbf{h}_t = c\mathbf{h}_{t-1} + \hat{\mathbf{x}}_t$. This corresponds a simplified system model where the past information is propagated to the future with a scaling constant of $c$. This simplifies (11) as:

$$\begin{aligned} \mathbf{H} &= \boldsymbol{\mathcal{R}} \times_{3,4}^{1,2} \mathbf{X} \times_2^2 \mathbf{W}^{(x)} \\ &= \mathrm{ten}(\mathbf{I} + \mathbf{A} \otimes \mathbf{W}^{(r)}) \times_{3,4}^{1,2} (\mathbf{X} \times_2^2 \mathbf{W}^{(x)}) \\ &= \mathrm{ten}(\mathbf{I} + \mathbf{A} \otimes \mathbf{I}) \times_{3,4}^{1,2} (\mathbf{X} \times_2^2 \mathbf{W}^{(x)}) \\ &= (\mathbf{I} + \mathbf{A}) \times_2^1 (\mathbf{X} \times_2^2 \mathbf{W}^{(x)}) \\ &= (\mathbf{I} + \mathbf{A}) \times_2^1 \hat{\mathbf{X}} \end{aligned} \tag{12}$$

Notice that (12) is equivalent to $\mathbf{H} = \sum_{k=0}^{K-1} \alpha_k \mathbf{A}^k \hat{\mathbf{X}}$, which is precisely a spatial graph filter as discussed in Section II-A, where $K = 2$, $\alpha_k = 1$, and $\mathbf{A}$ is the weighted graph adjacency matrix that enforces the directed flow of time. We will refer to neural networks employing equation (12) for feature extraction as simplified Recurrent Graph Tensor Networks (sRGTN).

### C. Tensor Network Formulation

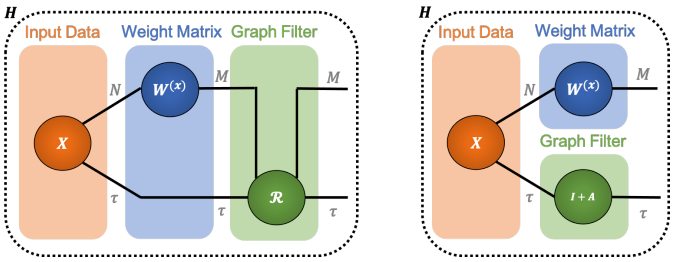


Fig. 2. Tensor Network (TN) diagram of the gRGTN filtering operation (left) according to (11), and the sRGTN filtering operation (right) according to (12). The nodes of the TN diagram represent different tensors, while the edges represent tensor contractions over common dimensions between tensors. The dimensions of different tensors are denoted in gray letters.

Consider the gRGTN filtering operation in (11). The multi-linear nature of the tensor $\boldsymbol{\mathcal{R}}$ and the associated double tensor contraction naturally admits a Tensor Network (TN) representation, as shown in Figure 2 (left). Similarly, the sRGTN filtering in (12) also admits a TN representation with a simpler topology, as shown in Figure 2 (right). This allows the hidden state modelling operation to benefit from the enhanced expressive power of tensors, which are not limited to the standard "flat-view" matrix methods [15], [16].

By integrating the concept of graph filtering in a TN framework, we can easily design network architectures for processing time-series data of any modalities, as well as leverage on the power of tensor decomposition to boost its expressive power while maintaining low complexity. For illustration, Figure 3 shows TN models designed to process multi-way time series data as order-3 input tensors (i.e. the

TABLE I
EXPERIMENT DATA MODALITIES

| | Mode 1 Physical Mode | Dimension | Mode 2 Physical Mode | Dimension | Mode 3 Physical Mode | Dimension |
|---|---|---|---|---|---|---|
| Air Quality Forecasting | Time | 6 | Site | 12 | Air Quality Features | 27 |
| Temperature Forecasting | Time | 6 | City | 14 | Temperature Features | 4 |
| House Price Forecasting | Time | 6 | House Type | 4 | Price Index Features | 2 |
| Activity Recognition | Time | 24 | Sensor | 3 | Measurement Features | 3 |

time-series features are indexed along a time-mode and an additional physical mode), which uses appropriate Tensor-Train (TT) networks to process filtered multi-way time-series data.

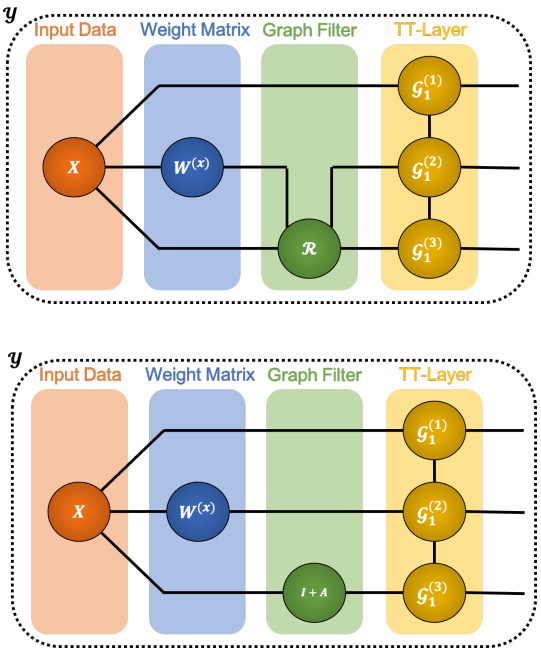


Fig. 3. A gRGTN model (top) and a sRGTN model (bottom) designed to handle order-3 input time-series tensors. In addition to the proposed filtering operations, it leverages on the power of Tensor-Train (TT) decomposition networks (in yellow) to achieve high expressive power at low parameter complexity, which is inherently compatible with the multi-way nature of the RGTN framework.

*Remark 1:* The double tensor contraction with $\mathcal{R}$ in gRGTN implies a stronger coupling of features with the underlying time-domain represented in graph form, thus yielding enhanced expressive power over the decoupled contractions in sRGTN.

*Remark 2:* The Tensor-Train layers in Figure 3 can be interpreted as tensorized fully-connected neural network layers compressed via Tensor-Train decomposition, which drastically reduces the number of parameters required to achieve the same expressive power [2], [12].

## IV. EXPERIMENTS

### A. Datasets

To validate the expressive power of the proposed gRGTN and sRGTN models, we verified their performance in a number of multi-way time-series modelling tasks, including:

1) *Beijing Multi-Site Air Quality Forecasting* [19]. This dataset consists of various air quality measurements obtained across 12 different sites in China recorded at an hourly rate. The learning task for this dataset is to forecast the air quality level across all 12 sites in the next hour.
2) *Global Land Temperature Forecasting* [20]. This dataset consists of monthly temperature recordings obtained across multiple cities around the world. The learning task for this dataset is to forecast the average temperature across 14 major cities in India during the next month.
3) *Liverpool House Price Forecasting* [21]. This dataset consists of monthly price indices across 4 different types of houses in Liverpool, United Kingdom. The learning task for this dataset is to forecast the price indices of different house types in the next month.
4) *Multi-Sensor Activity Recognition* [22]. This dataset consists of multi-sensor measurements of human bodies when performing different physical activities. The learning task for this dataset is to classify the physical activity from the multi-sensor measurements.

All of the considered data are multi-modal time-series tensors of order-3. The exact modalities of the input data tensors are summarized in Table I.

### B. Benchmark Models and Metrics

We compared the performance of the proposed gRGTN and sRGTN models against standard RNN, GRU, and LSTM based neural networks. For comparable results, all models have the exact same model architecture, hidden units, activation functions, and training method, with the only differences being: (i) the feature extraction layer used, which can be based on gRGTN, sRGTN, RNN, GRU, or LSTM, and (ii) the fully-connected dense layers, which are replaced by the equivalent TT networks for gRGTN and sRGTN as shown in Figure 3 [2]. For more details, please refer to the full experiment code provided on GitHub[1].

We compared the considered models across the proposed experiments both in terms of performance and complexity. In terms of performance metrics, we used out-of-sample Mean Absolute Error (MAE) for the regressions tasks related to datasets (1), (2), and (3), and classification accuracy for the classification task related to dataset (4). In terms of complexity, we compare the number of trainable parameters needed to achieve the same model specifications.

[1]The code is available on www.github.com/gylx/RGTN

TABLE II
PERFORMANCE AND COMPLEXITY OF THE CONSIDERED MODELS.

| Test Set Score | gRGTN | sRGTN | RNN | GRU | LSTM |
|---|---|---|---|---|---|
| Air Quality Forecasting (MAE) | **0.01598** | 0.01742 | 0.01872 | 0.01706 | 0.01652 |
| Temperature Forecasting (MAE) | 0.20959 | 0.27491 | 0.19905 | 0.18050 | **0.17744** |
| House Price Forecasting (MAE) | 0.72946 | 0.76768 | **0.71195** | 0.74081 | 0.73463 |
| Activity Classification (Accuracy) | **79.883%** | 78.740% | 50.731% | 79.398% | 78.629% |

| Number of Trainable Parameters | gRGTN | sRGTN | RNN | GRU | LSTM |
|---|---|---|---|---|---|
| Air Quality Forecasting | 556 | **492** | 2844 | 8196 | 10836 |
| Temperature Forecasting | 406 | **342** | 718 | 1782 | 2278 |
| House Price Forecasting | 220 | **156** | 244 | 540 | 652 |
| Activity Classification | 301 | **237** | 261 | 573 | 693 |

### C. Experiment Results

The experiment results are summarized in Table II. The top table shows the test set performance for three regression tasks (measured in MAE) and one classification task (measured in accuracy) achieved by the considered models. The bottom table shows the corresponding number of trainable parameters needed for each task.

By virtue of its graph and tensor structure, the proposed gRGTN model achieved the best performance overall, obtaining the highest score for 2 out of 4 datasets, while using drastically less number of trainable parameters compared to standard RNN, GRU, and LSTM models. On the other hand, the sRGTN model achieved the lowest parameter complexity due to its approximation assumption of $\mathbf{W}^{(r)} = \mathbf{I}$, but at the cost of marginally reduced performance.

## V. CONCLUSION

We have introduced a novel Recurrent Graph Tensor Network (RGTN) framework for modelling time-series data, by combining the expressive power of tensor networks with the ability of graphs to account for the structure underlying time-series data. Experiment results have verified the desirable properties of the proposed RGTN framework, which outperformed standard RNN, GRU, and LSTM models across multiple time-series modelling tasks, and at a drastically reduced parameter complexity.

## REFERENCES


[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[2] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 442–450.

[3] N. Cohen, O. Sharir, and A. Shashua, "On the expressive power of deep learning: A tensor analysis," in *Proceedings of The Conference on Learning Theory*, 2016, pp. 698–728.

[4] G. G. Calvi, A. Moniri, M. Mahfouz, Q. Zhao, and D. P. Mandic, "Compression and interpretability of deep neural networks via tucker tensor layer: From first principles to tensor valued back-propagation," *arXiv preprint arXiv:1903.06133*, 2019.

[5] D. P. Mandic and J. Chambers, *Recurrent neural networks for prediction: Learning algorithms, architectures and stability*. John Wiley & Sons, Inc., 2001.

[6] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, and T. Constantinides, "Data analytics on graphs. Part I: Graphs and spectra on graphs," *Foundations and Trends in Machine Learning*, vol. 13, no. 1, pp. 1–157, 2020.

[7] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, and A. G. Constantinides, "Data analytics on graphs. Part II: Signals on graphs," *Foundations and Trends in Machine Learning*, vol. 13, no. 2–3, pp. 158–331, 2020.

[8] L. Stankovic, D. Mandic, M. Dakovic, M. Brajovic, B. Scalzo, S. Li, and A. G. Constantinides, "Data analytics on graphs. Part III: Machine learning on graphs, from graph topology to applications," *Foundations and Trends in Machine Learning*, vol. 13, no. 4, pp. 332–530, 2020.

[9] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[10] A. Cichocki, "Era of big data processing: A new approach via tensor networks and tensor decompositions," *ArXiv e-prints*, Mar. 2014.

[11] I. V. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[12] Y. Yang, D. Krompass, and V. Tresp, "Tensor-train recurrent neural networks for video classification," in *Proceedings of International Conference on Machine Learning*. PMLR, 2017, pp. 3891–3900.

[13] R. Yu, S. Zheng, A. Anandkumar, and Y. Yue, "Long-term forecasting using tensor-train RNNs," *Arxiv*, 2017.

[14] S. Dolgov and D. Savostyanov, "Alternating minimal energy methods for linear systems in higher dimensions," *SIAM Journal on Scientific Computing*, vol. 36, no. 5, pp. A2248–A2271, 2014.

[15] A. Cichocki, N. Lee, I. Oseledets, A. Phan, Q. Zhao, D. P. Mandic *et al.*, "Tensor networks for dimensionality reduction and large-scale optimization. part 1: Low-rank tensor decompositions," *Foundations and Trends® in Machine Learning*, vol. 9, no. 4-5, pp. 249–429, 2016.

[16] A. Cichocki, A. Phan, Q. Zhao, N. Lee, I. Oseledets, M. Sugiyama, D. P. Mandic *et al.*, "Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives," *Foundations and Trends® in Machine Learning*, vol. 9, no. 6, pp. 431–673, 2017.

[17] I. V. Oseledets and E. E. Tyrtyshnikov, "Breaking the curse of dimensionality, or how to use SVD in many dimensions," *SIAM Journal on Scientific Computing*, vol. 31, no. 5, pp. 3744–3759, 2009.

[18] Y. Khalifa, D. P. Mandic, and E. Sejdic, "The role of hidden Markov models and recurrent neural networks in event detection and localization for biomedical signals: Theory and application," *Information Fusion, in print*, 2020.

[19] S. Zhang, B. Guo, A. Dong, J. He, Z. Xu, and S. X. Chen, "Cautionary tales on air-quality improvement in Beijing," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 473, no. 2205, p. 20170457, 2017.

[20] R. Rohde, R. A. Muller, R. Jacobsen, E. Muller, S. Perlmutter, A. Rosenfeld, J. Wurtele, D. Groom, and C. Wickham, "A new estimate of the average earth surface land temperature spanning 1753 to 2011," *Geoinfor Geostat: An Overview*, vol. 7, p. 2, 2013.

[21] "Reports for the UK House Price Index (UK HPI) for England, Scotland, Wales and Northern Ireland." https://www.gov.uk/government/collections/uk-house-price-index-reports.

[22] F. Palumbo, C. Gallicchio, R. Pucci, and A. Micheli, "Human activity recognition using multisensor data fusion based on reservoir computing," *Journal of Ambient Intelligence and Smart Environments*, vol. 8, no. 2, pp. 87–107, 2016.