LOW-COMPLEXITY ATTENTION MODELLING VIA GRAPH TENSOR NETWORKS

Yao Lei Xu[1], *Kriton Konstantinidis*[1], *Shengxi Li*[1], *Ljubiša Stanković*[2], *Danilo P. Mandic*[1]

[1]Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, UK
[2]Faculty of Electrical Engineering, University of Montenegro, Podgorica, 81000, Montenegro
E-mails: {yao.xu15, k.konstantinidis19, shengxi.li17, d.mandic}@imperial.ac.uk, ljubisa@ucg.ac.me

ABSTRACT

The attention mechanism is at the core of modern Natural Language Processing (NLP) models, owing to its ability to focus on the most contextually relevant part of a sequence. However, current attention models rely on "flat-view" matrix methods to process tokens embedded in vector spaces; this results in exceedingly high parameter complexity which is prohibitive for practical applications. To this end, we introduce a novel *Tensorized Graph Attention* (TGA) mechanism, which leverages on the recent Graph Tensor Network (GTN) framework to efficiently process tensorized token embeddings via attention based graph filters. Such tensorized token embeddings are shown to effectively bypass the Curse of Dimensionality, reducing the parameter complexity of the attention mechanism from an exponential to a linear one in the embedding dimensions. The expressive power of the TGA framework is further enhanced by virtue of domain-aware graph convolution filters. Simulations across benchmark NLP paradigms verify the advantages of the proposed framework over existing attention models, at drastically lower parameter complexity.

Index Terms— Attention, Tensor Decomposition, Graph Neural Networks, Tensor-Train Decomposition, Compression.

1. INTRODUCTION

The attention mechanism has become the *de-facto* standard for Natural Language Processing (NLP) tasks, owing to its ability to capture context dependent relationships in data, focusing on the most relevant part of a sequence [1]. The success of attention models [2, 3, 4] has also highlighted that their parameter complexity increases exponentially with the size of the associated weight matrices [5], making it prohibitive for practical applications. To alleviate the critical complexity issues, we introduce the Tensorized Graph Attention (TGA) mechanism, which leverages on the recent Graph Tensor Network (GTN) framework [6] [7] to enable processing tensorized embeddings on graphs through highly efficient tensor networks.

Regarding the use of tensors in this context, the Tensor-Train Decomposition (TTD) [8, 9] has been used to compress embedding layers for NLP tasks with large vocabularies [10], although it has not been directly employed to compress the attention mechanism. The Khatri-Rao product was also employed to develop a tensor based attention with reduced complexity [11], but its focus on dimension-wise attention instead of token-wise attention and the use of a low-dimensional order-3 tensor pose serious limitations. The Block-Term Tensor Decomposition [12] has also been used within a multi-linear attention mechanism [5], with the same low-dimensional order-3 tensor limitation. The current spatial and spectral graph neural networks only use the attention operation for masking a-priori defined graph edges, and are limited to semi-supervised learning [13, 14, 15]. Overall, to the best of our knowledge, this is the first attempt to equip the attention mechanism with the ability to process tensor embeddings of arbitrarily high dimensions; this is achieved by leveraging on the expressive power of tensor networks and spectral graph convolutions.

The contributions of this work are as follows:

- We generalize the attention mechanism to process token embeddings in a higher-order tensor space, as opposed to currently used vector space embeddings.
- Such a tensorized embedding is shown to benefit from the power of multi-linear algebra and the associated tensor decomposition algorithms to bypass the Curse of Dimensionality associated with existing NLP methods.
- The attention mechanism is formulated as a context-aware graph filter, thus empowering the processing of tensorized embeddings via spectral graph convolutions.
- Such a unified GTN framework integrates the virtues of both tensors and graphs in the context of attention, resulting in a highly expressive attention model with drastically reduced computational costs.

The benefits of the proposed approach are demonstrated over a comprehensive set of benchmark NLP experiments.

2. PRELIMINARIES

Tensor Algebra. An order-N tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, is a multi-dimensional array, where I_n is the size of its n-th mode, $n = 1, \ldots, N$. Special cases of tensors include matrices as order-2 tensors, $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$, vectors as order-1 tensors, $\mathbf{x} \in \mathbb{R}^{I_1}$, and scalars as order-0 tensors, ($x \in \mathbb{R}$). The mode-n matricization of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, denoted by $\mathrm{mat}(\cdot)$, reshapes the multidimensional array into a matrix,

$\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$, while the inverse process, *Tensorization*, is denoted by $\text{ten}(\cdot)$. The Hadamard product, denoted by $\odot$, between two tensors, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, yields, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, where $c_{i_1,\ldots,i_N} = a_{i_1,\ldots,i_N} b_{i_1,\ldots,i_N}$. The (m,n)-contraction [16], denoted by $\times_n^m$, between 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 $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{J_1 \times \cdots \times J_m \times \cdots \times J_M}$, where $I_n = J_m$, yields an 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 entries defined as $c_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N,j_1,\ldots,j_{m-1},j_{m+1},\ldots,j_M} = \sum_{i_n=1}^{I_n} a_{i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} b_{j_1,\ldots,j_{m-1},i_n,j_{m+1},\ldots,j_M}$. Finally, the Matrix Product Operator (MPO) definition of the Tensor-Train decomposition (TTD) [8] can approximate a large order-$2N$ tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$, via N contracting core tensors, $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$, as $\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}}^{(1)} \times_4^1 \boldsymbol{\mathcal{G}}^{(2)} \times_4^1 \cdots \times_4^1 \boldsymbol{\mathcal{G}}^{(N)}$, where the set of R_n for $n = 0, \ldots, N$ with $R_0 = R_N = 1$ is referred to as the *TT-rank*. The compression properties of TTD can be applied to significantly compress neural networks while maintaining comparable performance [17, 18].

Graph Convolutional Networks. A graph, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, is defined by a set of L vertices (or nodes), $\mathcal{V}$, and a set of edges, $\mathcal{E}$, and fully described by its weighted adjacency matrix, $\mathbf{A} \in \mathbb{R}^{L \times L}$, such that $a_{l_1,l_2} > 0$ if there is an edge connecting the l_1-th and l_2-th node [19]. If the edge values are not given, they can be defined based on a pair-wise similarity function, $g(\cdot)$, such that $a_{l_1,l_2} = g(\mathbf{z}_{l_1}, \mathbf{z}_{l_2})$, where $\mathbf{z}_{l_1}$ and $\mathbf{z}_{l_2}$ are the attributes associated with the l_1-th and l_2-th vertices [20]. If the graph is undirected, that is $\mathbf{A}^T = \mathbf{A}$, then the operation, $\mathbf{y} = (\mathbf{I} + \mathbf{A})\mathbf{x}$, represents a spectral graph convolution operation over the graph signal, $\mathbf{x} \in \mathbb{R}^L$, which effectively acts as a low-pass filter over the irregular graph domain [21, 22, 23].

Graph Tensor Networks. Consider a multi-way sequence learning problem, where the input data, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{L \times I_1 \times \cdots \times I_N}$, is a time-series tensor with features of dimensionality, $I_1 \times \cdots \times I_N$, indexed along L consecutive time-steps. For the so defined input matrix, a Recurrent Graph Tensor Network [7, 6] extracts a time-series feature map, $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{L \times I_1 \times \cdots \times I_N}$, through a multi-linear graph filter operation, $\boldsymbol{\mathcal{Y}} = (\mathbf{I} + \mathbf{A}) \times_2^1 \boldsymbol{\mathcal{X}}$, based on the time-adjacency matrix, $\mathbf{A} \in \mathbb{R}^{L \times L}$. More specifically, $\mathbf{A}$ considers each of the L time-steps as a node of a time-domain graph, where the l_1-th and l_2-th time-steps are connected via a weight, a_{l_1,l_2}, such that $a_{l_1,l_2} = c^{l_2 - l_1}$ for $l_2 > l_1$, and $a_{l_1,l_2} = 0$ otherwise, where c is a damping constant strictly less than 1. The condition $l_2 > l_1$ ensures that $\mathbf{A}$ is an upper-triangular matrix to reflect the structured flow of information over time, as past information can influence the future states but not vice-versa. For time-series applications where the directed flow of information over time is not necessary, we can replace $\mathbf{A}$ with a bi-directional time-graph adjacency matrix, $\mathbf{\Omega} \in \mathbb{R}^{L \times L}$, defined as $\mathbf{\Omega} = \frac{1}{2}(\mathbf{A}^T + \mathbf{A})$.

Dot-Product Attention. Given an input time-series matrix, $\mathbf{X} \in \mathbb{R}^{L \times I}$, where I features are indexed along L time-steps, the dot-product attention [1] computes $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)} \in \mathbb{R}^{L \times J}$, $\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)} \in \mathbb{R}^{L \times J}$, and $\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)} \in \mathbb{R}^{L \times J}$, using the query, key, and value weight matrices, $\mathbf{W}^{(q)} \in \mathbb{R}^{I \times J}$, $\mathbf{W}^{(k)} \in \mathbb{R}^{I \times J}$, and $\mathbf{W}^{(v)} \in \mathbb{R}^{I \times J}$. Given $\mathbf{Q}$ and $\mathbf{K}$, the self-attention coefficient matrix, $\mathbf{\Theta} \in \mathbb{R}^{L \times L}$, is computed as $\mathbf{\Theta} = \sigma(\frac{1}{\sqrt{d_k}}\mathbf{Q}\mathbf{K}^T)$, where $\sigma(\cdot)$ is a $softmax$ activation function and $\frac{1}{\sqrt{d_k}}$ a scaling factor. The attention mechanism then generates the final output, $\mathbf{Y} \in \mathbb{R}^{L \times J}$, as $\mathbf{Y} = \mathbf{\Theta}\mathbf{V}$. For multi-head attention, this can be repeated H times using a set of attention coefficient matrices, $\{\mathbf{\Theta}_1, \ldots, \mathbf{\Theta}_H\}$, to generate H different outputs, $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_H\}$, which are then concatenated and combined to produce the overall output, $\mathbf{Y}$.

3. ATTENTION GRAPH FILTERS

Given an input time-series matrix, $\mathbf{X} \in \mathbb{R}^{L \times I}$, where I features are indexed along L time-steps, we compute: (i) $\mathbf{K} \in \mathbb{R}^{L \times J}$, a key-representation of the input data generated by the transform, $\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$, where $\mathbf{W}^{(k)} \in \mathbb{R}^{I \times J}$ is a trainable key-weight matrix; (ii) $\mathbf{V} \in \mathbb{R}^{L \times J}$, a value-representation of the input data generated by the transform, $\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$, where $\mathbf{W}^{(v)} \in \mathbb{R}^{I \times J}$ is a trainable value-weight matrix.

Given $\mathbf{K}$, the symmetric attention coefficient matrix, $\mathbf{\Theta} \in \mathbb{R}^{L \times L}$, is next computed, where the attention coefficient between the l_1-th and l_2-th time-step is defined as $\theta_{l_1,l_2} = \epsilon(\frac{1}{\sqrt{d_k}}\mathbf{k}_{l_1}^T \mathbf{k}_{l_2})$ for $l_1 \neq l_2$ and $\theta_{l_1,l_2} = 0$ otherwise, with $\epsilon(\cdot)$ as a $ReLU$ activation function, $\frac{1}{\sqrt{d_k}}$ as a scaling factor, and $\mathbf{k}_{l_1}$ and $\mathbf{k}_{l_2}$ as respectively the key-vectors at the l_1-th and l_2-th time-step. By viewing each of the L time-steps as nodes of a time-domain graph [7], we can interpret $\mathbf{\Theta}$ as an undirected graph adjacency matrix, where the edge weight between the l_1-th and l_2-th vertices, θ_{l_1,l_2}, is context dependent and generated by a pair-wise similarity function parameterized by $\mathbf{W}^{(k)}$.

Remark 1. The choice of a $ReLU$ activation to generate the adjacency matrix, $\mathbf{\Theta}$, serves two purposes: (i) rectification promotes sparsity, i.e., a reduction of the number of node connections in the graph domain, as some edges become disconnected; (ii) the remaining weights are strictly non-negative, a usual assumption for weighted graph adjacency matrices.

Due to its dot product formulation, $\mathbf{\Theta}$ is context dependent, but is agnostic to the time-series structure of the input data, that is, to sequence ordering. To this end, we incorporate the bi-directional time graph adjacency matrix, $\mathbf{\Omega}$, as discussed in Section 2, to define the adjacency matrix of the time-domain attention graph, $\mathbf{\Psi} \in \mathbb{R}^{L \times L}$, as $\mathbf{\Psi} = \mathbf{\Omega} \odot \mathbf{\Theta}$.

Remark 2. Unlike classical attention mechanisms, the time-domain attention adjacency matrix, $\mathbf{\Psi}$, within the proposed TGA mechanism, is both time- and context-aware. Indeed, given a signal $\mathbf{x} \in \mathbb{R}^L$, the operation, $\mathbf{y} = \mathbf{\Psi}\mathbf{x}$, can be expressed in an element-wise form, $y_{l_1} = \sum_{l_2 \in \mathcal{N}_{l_1}} \Theta_{l_1,l_2} \Omega_{l_1,l_2} x_{l_2}$. This represents a weighted sum of signals in the neighbourhood of the l_1-th time-step, $\mathcal{N}_{l_1}$, with weights proportional to both the contextual relevance,

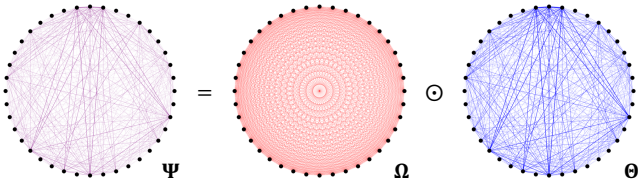

Fig. 1. *Principle of the time-domain attention graph (in purple), which is computed from the bi-directional time-graph (in red), and the symmetric attention coefficient graph (in blue), as a Hadamard product of their adjacency matrices,* $\mathbf{\Psi} = \mathbf{\Omega} \odot \mathbf{\Theta}$. *The graph vertices represent a sequence of tokens embedded in the tensor space, which are interconnected with edge weights proportional to their time-distance,* Ω_{l_1,l_2}, *and attention coefficient,* Θ_{l_1,l_2}, *for* $l_1 = 1, \ldots, L$ *and* $l_2 = 1, \ldots, L$.

Θ_{l_1,l_2}, and the time proximity, Ω_{l_1,l_2}, between tokens at the time-steps l_1 and l_2.

By construction, $\mathbf{\Psi}$ is a symmetric graph adjacency matrix representing an undirected graph. Therefore, the graph filter operation, $\mathbf{y} = (\mathbf{I} + \mathbf{\Psi})\mathbf{x}$, can be interpreted as a spectral graph convolution operation as discussed in Section 2, where the input sequences of tokens are processed via context- and time-dependent graph topologies. Finally, similar to the classical multi-head attention mechanism, we can repeat the attention filtering operation H times, using a set of attention filters, $\{\mathbf{\Psi}_1, \mathbf{\Psi}_2, \ldots, \mathbf{\Psi}_H\}$, to generate H different filtered signals that can be then combined to generate the overall output.

Remark 3. Notice that the filtering operation, $\mathbf{y} = (\mathbf{I} + \mathbf{\Psi})\mathbf{x}$, is a special case of the RGTN forward pass, $\mathcal{Y} = (\mathbf{I} + \mathbf{\Psi}) \times_2^1 \mathcal{X}$, which allows for the generalization of the graph filtering operation to process higher-order tensors [7].

4. TENSORIZING ATTENTION

Traditional NLP models process a sequence of tokens (e.g. words) of length, L, embedded in a vector space of dimension, I, resulting in an input matrix, $\mathbf{X} \in \mathbb{R}^{L \times I}$. However, due to the "flat-view" nature of matrix methods, performing attention on the matrix $\mathbf{X}$ incurs expensive matrix multiplications.

To alleviate the complexity costs of the attention mechanism, we embed the sequence of symbols in the tensor space to generate the input tensor, $\mathcal{X} \in \mathbb{R}^{L \times I_1 \times \cdots \times I_N}$, and thus benefit from the ability of Graph Tensor Networks to operate with high-dimensional tensors at low computational complexity. By formulating the problem in the tensor space, we next leverage on the power of tensor networks to drastically reduce the complexity needed to achieve high expressive power.

Consider a large-dimensional matrix multiplication of the form, $\mathbf{Y} = \mathbf{X}\mathbf{W}$, where the matrices are of dimensionality $\mathbf{Y} \in \mathbb{R}^{L \times J}$, $\mathbf{X} \in \mathbb{R}^{L \times I}$, and $\mathbf{W} \in \mathbb{R}^{I \times J}$. Without loss of generality, let the dimensionality of the considered matrices be factorizable as $I = I_1 \times \cdots \times I_N$ and $J = J_1 \times \cdots \times J_N$. This allows us to tensorize $\mathbf{X} \in \mathbb{R}^{L \times I}$ to $\mathcal{X} \in \mathbb{R}^{L \times I_1 \times \cdots \times I_N}$, $\mathbf{Y} \in \mathbb{R}^{L \times J}$ to $\mathcal{Y} \in \mathbb{R}^{L \times J_1 \times \cdots \times J_N}$, and $\mathbf{W} \in \mathbb{R}^{I \times J}$ to $\mathcal{W} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$. This allows us to re-write large-dimensional matrix multiplications, which are ubiquitous to the attention mechanism, as a higher order tensor contraction

$$\mathcal{Y} = \mathcal{X} \times_{2,3,4,\cdots,N+1}^{1,3,5,\cdots,2N-1} \mathcal{W} \tag{1}$$

By virtue of the above tensor structure, we can now apply Tensor-Train (TT) Decomposition to express the weight tensor, $\mathcal{W}$, in a low-rank Matrix-Product-Operator (MPO) form

$$\mathcal{W} \approx \mathcal{G}_1 \times_4^1 \mathcal{G}_2 \times_4^1 \mathcal{G}_3 \times_4^1 \cdots \times_4^1 \mathcal{G}_N \tag{2}$$

where $\mathcal{G}_n \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$ for $n = 1, \ldots, N$ are referred to as the TT-cores, and the set $\{R_0, R_1, \ldots, R_N\}$ is the TT-rank of the TT decomposition, where $R_0 = R_N = 1$.

Remark 4. In the attention mechanism context, instead of learning a large dimensional weight matrix, $\mathbf{W}$, we can directly learn the low-rank TT-cores, $\mathcal{G}_n$ for $n = 1, \ldots, N$, which reduces the parameter complexity from an exponential $\mathcal{O}\big(\prod_{n=1}^{N} I_n J_n\big) = \mathcal{O}\big(IJ\big)$ to a linear $\mathcal{O}\big(\sum_{n=1}^{N} R_{n-1} I_n J_n R_n\big)$ one in the dimensions I_n and J_n. This is highly efficient for a low TT-rank, R_n, and a large number of dimensions, N.

The TT format of $\mathcal{W}$ simplifies the large-scale contraction in (1) into a series of small-size tensor contractions

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

To achieve minimal parameter complexity, we employ the Quantized Tensor-Train Decomposition (QTTD), by: (i) setting I and J to be a power of 2, such that $I_n = J_n = 2$ for $n = 1, \ldots, N$, and (ii) setting the TT-rank $R_n = 2$ for $n = 1, \ldots, N-1$.

Remark 5. The use of the compressed TT format in (3), in conjunction with the QTTD, to replace all dense matrix multiplications in the attention mechanism makes it possible to drastically reduce the parameter complexity from an exponential one, $\prod_{n=1}^{N} I_n J_n = 4^N$, to a linear one, $\sum_{n=1}^{N} R_{n-1} I_n J_n R_n = 16(N-1)$, in the tensor order, N.

To perform attention graph filtering in the tensor domain, we need to compute the attention coefficient matrix, $\mathbf{\Theta}$, from the tensorized input. To this end, we can first compute the key-representation of the input data, $\mathcal{K} \in \mathbb{R}^{L \times J_1 \times \cdots \times J_N}$, by employing (3), and then compute the coefficients, θ_{l_1,l_2}, as $\theta_{l_1,l_2} = \epsilon(\frac{1}{\sqrt{d_k}} \langle \mathcal{K}_{l_1}, \mathcal{K}_{l_2} \rangle)$, where $\mathcal{K}_{l_1} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ and $\mathcal{K}_{l_2} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ are the slices of $\mathcal{K}$ at the l_1-th and l_2-th time-steps. This is equivalent to computing $\mathbf{\Theta} = \epsilon(\frac{1}{\sqrt{d_k}} \mathcal{K} \times_{2,\cdots,N+1}^{2,\cdots,N+1} \mathcal{K})$ and then setting its diagonal elements to zero.

The proposed Graph Tensor Network (GTN) based Tensorized Graph Attention (TGA) mechanism is summarised in Algorithm 1. The procedure can be repeated for H different heads to perform multi-head TGA.

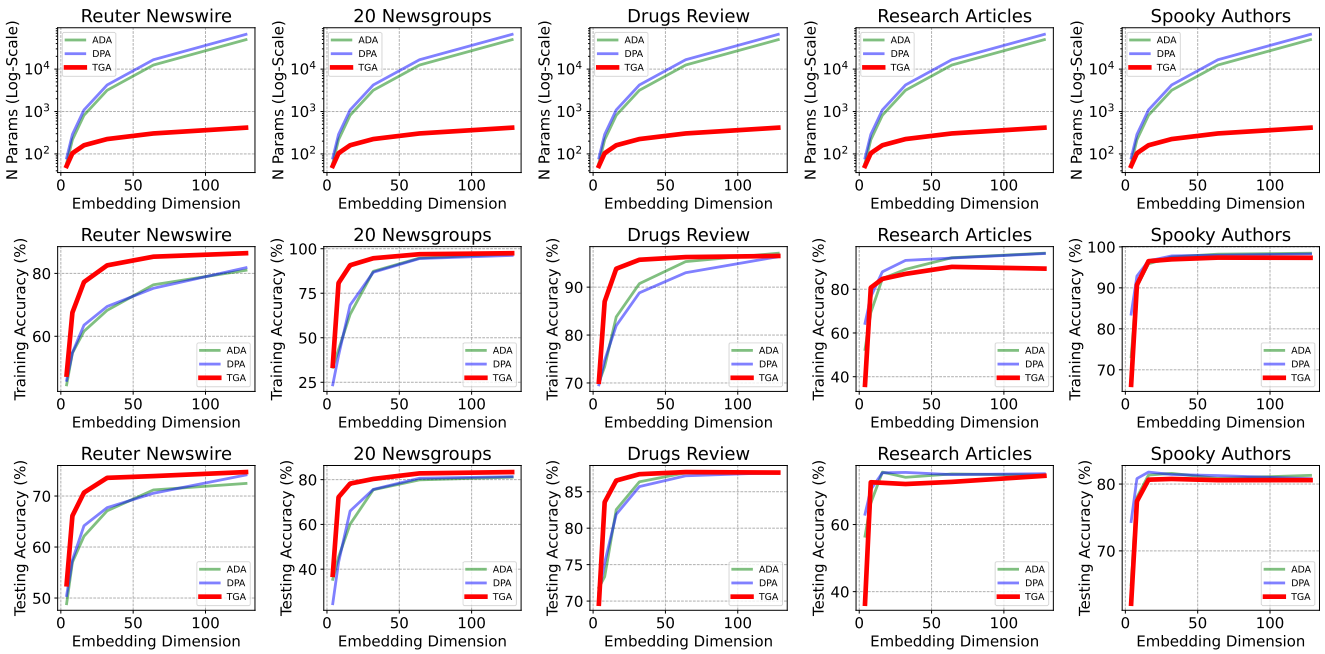

Fig. 2. *Performance comparison of the proposed Tensorized Graph Attention (TGA) mechanism, against the existing Dot-Product Attention (DPA), and Additive Attention (ADA) models. The columns of panels correspond to the five considered datasets. The rows of panels correspond to: (i) embedding dimension vs number of parameters (complexity), (ii) embedding dimension vs training accuracy (expressive power), and (iii) embedding dimension vs testing accuracy (performance).*

Algorithm 1: Tensorized Graph Attention (TGA)

Input : Tokens tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{L \times I_1 \times \cdots \times I_N}$ and time-domain adjacency matrix $\boldsymbol{\Omega} \in \mathbb{R}^{L \times L}$

Output : Feature tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{L \times J_1 \times \cdots \times J_N}$

Initialize : Initialize $R_n = 2$ for $n = 1, \ldots, N-1$

Initialize $I_1, \ldots, I_N$ s.t. $\prod_{n=1}^{N} I_n = I$

Initialize $J_1, \ldots, J_N$ s.t. $\prod_{n=1}^{N} J_n = J$

Initialize $\boldsymbol{\mathcal{G}}_n^{(k)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$

Initialize $\boldsymbol{\mathcal{G}}_n^{(v)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$

$\boldsymbol{\mathcal{K}} \leftarrow \boldsymbol{\mathcal{X}} \times_2^2 \boldsymbol{\mathcal{G}}_1^{(k)} \times_{2,N+3}^{2,1} \boldsymbol{\mathcal{G}}_2^{(k)} \times_{2,N+3}^{2,1} \cdots \times_{2,N+3}^{2,1} \boldsymbol{\mathcal{G}}_N^{(k)}$

$\boldsymbol{\mathcal{V}} \leftarrow \boldsymbol{\mathcal{X}} \times_2^2 \boldsymbol{\mathcal{G}}_1^{(v)} \times_{2,N+3}^{2,1} \boldsymbol{\mathcal{G}}_2^{(v)} \times_{2,N+3}^{2,1} \cdots \times_{2,N+3}^{2,1} \boldsymbol{\mathcal{G}}_N^{(v)}$

$\boldsymbol{\Theta} \leftarrow \epsilon(\frac{1}{\sqrt{d_k}} \boldsymbol{\mathcal{K}} \times_{2,\cdots,N+1}^{2,\cdots,N+1} \boldsymbol{\mathcal{K}})$

$\boldsymbol{\Theta} \leftarrow \boldsymbol{\Theta} - diag(\boldsymbol{\Theta})$

$\boldsymbol{\Psi} \leftarrow \boldsymbol{\Omega} \odot \boldsymbol{\Theta}$

$\boldsymbol{\mathcal{Y}} \leftarrow (\mathbf{I} + \boldsymbol{\Psi}) \times_2^1 \boldsymbol{\mathcal{V}}$

5. EXPERIMENTS

The performance of the proposed TGA mechanism was evaluated in terms of parameter complexity, expressive power, and performance over various Natural Language Processing (NLP) tasks. The full experiment code and data is available on GitHub[1], for reproducibility.

[1] www.github.com/gylx/Tensorized-Spectral-Attention

Figure 2 compares the training and testing accuracy of the considered models at varying complexity levels and across five NLP benchmark datasets. The achieved accuracy scores were compared by varying the word embedding dimension in the attention layer while keeping all other parameters fixed, which resulted in models of varying expressive power and complexity. This allowed us to analyse the performance-complexity trade-offs of the considered models in a comprehensive way, addressing the aspects of complexity, expressive power, and performance. The proposed TGA achieved better or comparable performance than the existing attention mechanisms, while incurring drastically lower complexity costs.

6. CONCLUSION

We have introduced a novel Tensorized Graph Attention (TGA) mechanism for Natural Language Processing (NLP), with the aim to alleviate the associated Curse of Dimensionality by exploiting the super-compression abilities of tensor algorithms and the domain-aware processing power of graph filters. By virtue of its conjoint tensor and graph formulation, the proposed TGA framework has been shown to achieve high expressive power at a drastically lower parameter complexity (linear in the tensor order) compared to traditional attention models (exponential in the tensor order). Experimental results have demonstrated the proposed TGA exhibiting on-par performance with the existing models across five NLP benchmark tasks at significantly lower complexity costs.

7. REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2017, pp. 5998–6008.

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," *OpenAI*, 2018.

[3] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and L. Kaiser, "Universal transformers," *arXiv preprint arXiv:1807.03819*, 2018.

[5] X. Ma, P. Zhang, S. Zhang, N. Duan, Y. Hou, M. Zhou, and D. Song, "A tensorized transformer for language modeling," *Proceedings of Advances in Neural Information Processing Systems*, vol. 32, pp. 2232–2242, 2019.

[6] Y. L. Xu, K. Konstantinidis, and D. P. Mandic, "Multi-graph tensor networks," in *the First Workshop on Quantum Tensor Networks in Machine Learning, 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[7] Y. L. Xu and D. P. Mandic, "Recurrent graph tensor networks: A low-complexity framework for modelling high-dimensional multi-way sequences," in *Proceedings of the 29th European Signal Processing Conference (EUSIPCO)*, 2021.

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

[9] I. Kisil, G. G. Calvi, K. Konstantinidis, Y. L. Xu, and D. P. Mandic, "Reducing computational complexity of tensor contractions via tensor-train networks," *arXiv preprint arXiv:2109.00626*, 2021.

[10] O. Hrinchuk, V. Khrulkov, L. Mirvakhabova, E. Orlova, and I. Oseledets, "Tensorized embedding layers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 4847–4860.

[11] S. Zhang, P. Zhang, X. Ma, J. Wei, Q. Liu, et al., "Tensorcoder: Dimension-wise attention via tensor representation for natural language modeling," *arXiv preprint arXiv:2008.01547*, 2020.

[12] L. De Lathauwer, "Decompositions of a higher-order tensor in block terms. Part II: Definitions and uniqueness," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1033–1066, 2008.

[13] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[14] H. Chang, Y. Rong, T. Xu, W. Huang, S. Sojoudi, J. Huang, and W. Zhu, "Spectral graph attention network," *arXiv preprint arXiv:2003.07450*, 2020.

[15] G. Wang, R. Ying, J. Huang, and J. Leskovec, "Direct multi-hop attention based graph neural network," *arXiv preprint arXiv:2009.14332*, 2020.

[16] A. Cichocki, "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*, 2013.

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

[18] Y. L. Xu, G. G. Calvi, and D. P. Mandic, "Tensor-train recurrent neural networks for interpretable multi-way financial forecasting," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–5.

[19] 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.

[20] 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.

[21] H. Nt and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," *arXiv preprint arXiv:1905.09550*, 2019.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017, pp. 1–14.

[23] 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.