A Tensorized Spectral Attention Mechanism for Efficient Natural Language Processing

Yao Lei Xu, Kriton Konstantinidis, Shengxi Li, Danilo P. Mandic
Department of Electrical and Electronic Engineering
Imperial College London
{yao.xu15, k.konstantinidis19, shengxi.li17, d.mandic}@imperial.ac.uk

Abstract

The attention mechanism is at the core of state-of-the-art 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 sequence of tokens embedded in vector spaces, resulting in exceedingly high parameter complexity for practical applications. To this end, we introduce a novel *Tensorized Spectral Attention* (TSA) mechanism, which leverages on the Graph Tensor Network (GTN) framework to efficiently process tensorized token embeddings via attention based spectral graph filters. By virtue of multi-linear algebra, such tensorized token embeddings are shown to effectively bypass the Curse of Dimensionality, reducing the parameter complexity of the attention mechanism from exponential to linear in the weight matrix dimensions. Furthermore, the graph formulation of the attention domain enables the processing of tensorized embeddings through spectral graph convolution filters, which further increases its expressive power. The benefits of the TSA are demonstrated through five benchmark NLP experiments, where the proposed mechanism is shown to achieve better or comparable results against traditional attention models, while incurring drastically lower parameter complexity.

1 Introduction

The attention mechanism has become the *de-facto* tool of choice for Natural Language Processing (NLP) tasks, owing to its ability to capture context dependent relationships in data and thus focus on the most relevant part of a sequence [1]. Despite being at the core of many state-of-the-art language models [2–4], the attention mechanism is known to suffer from the Curse of Dimensionality, whereby the parameter complexity increases exponentially with the size of the associated weight matrices [5]. To alleviate the complexity issues in attention models, we introduce the Tensorized Spectral Attention (TSA) mechanism. The proposed TSA builds on a recent Graph Tensor Network (GTN) framework [6] [7], which drastically reduces the complexity costs through Tensor Decomposition (TD) techniques, while at the same time boosting the expressive power of the attention operation via spectral graph convolutions.

Contributions: The contributions of this work are threefold. First, we generalize the attention mechanism to make it possible to process token embeddings in a higher-order tensor space, as opposed to the traditional vector space embeddings. By virtue of its multi-modal structure, such a tensorized embedding benefits from the power of multi-linear algebra and the associated tensor decomposition algorithms, hence bypassing the Curse of Dimensionality aspect associated with traditional NLP methods. Second, we introduce a novel formulation of the attention mechanism as a graph filter operation, which enables the processing of tensorized embeddings through spectral graph

Second Workshop on Quantum Tensor Networks in Machine Learning, 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia

convolution filters. Third, a unified GTN framework is presented that integrates the advantages of both tensors and graphs in the context of attention, resulting in a highly expressive attention model with drastically low complexity costs.

Related Work: Tensor-Train Decomposition (TTD) [8] has been used to compress embedding layers for NLP tasks with large vocabularies in [9], although it has not been employed to compress the attention mechanism directly. The Khatri-Rao product was employed to develop a tensor based attention with lower complexity in [10], but it focuses on dimension-wise attention instead of token-wise attention, and is limited to only a low-dimensional order-3 tensor for processing vector embeddings. Block-Term Tensor Decomposition [11] has also been used to propose a multi-linear attention mechanism in [5], but it was still limited to a low-dimensional order-3 tensor for processing vector embeddings, and could only approximate the classical attention mechanism if the embedding dimension is equal to the length of the sequence. The attention mechanism has been applied to graph neural networks both in the spatial and spectral domains [12–14], but the attention operation was used in a masking context with a priori defined graph edges, and the application was limited to semi-supervised learning. Overall, to the best of our knowledge, our work is the first to extend the attention mechanism to process tensor embeddings of arbitrarily high dimensions, by leveraging on the expressive power of tensor networks and spectral graph convolutions.

Paper Organization: In Section 2, we introduce the theoretical background necessary for this work. A graph formulation of the spectral attention mechanism is next introduced in Section 3, which is then generalised to the tensor domain in Section 4. All of the derived concepts are then combined in Section 5. We finally validate the proposed framework via extensive experiments in Section 6, with practical limitations discussed in Section 7 and conclusions in Section 8.

2 Preliminaries

2.1 Tensor Algebra

Nomenclature: A tensor of order N, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ (denoted by a calligraphic font), is a multi-dimensional array, where I_n is the size of its n-th mode, for $n = 1, \ldots, N$. Special cases of tensors include matrices (denoted by bold capital letters, e.g., $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$) as order-2 tensors, vectors (denoted by bold lower-case letters, e.g., $\mathbf{x} \in \mathbb{R}^{I_1}$) as order-1 tensors ($N = 1$), and scalars (denoted by lower-case letters, e.g., $x \in \mathbb{R}$) as order-0 tensors. A specific entry of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is given by $x_{i_1,\ldots,i_N} \in \mathbb{R}$. The tensor indices in this paper follow the Little-Endian convention [15].

Kronecker Product: 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 $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_N J_N}$, of the same order, 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 $\overline{i_n j_n} = j_n + (i_n - 1)J_n$ [16].

Hadamard Product: The Hadamard (element-wise) product between two tensors of the same dimensionality, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, denoted by $\circledast$, yields a tensor, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, where $c_{i_1,\ldots,i_N} = a_{i_1,\ldots,i_N} b_{i_1,\ldots,i_N}$.

Matricization and Tensorization: The mode-n matricization of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, denoted by $\mathsf{mat}(\,)$, reshapes the multidimensional array into a matrix $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$ with $(x_{(n)})_{i_n, \overline{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N}} = x_{i_1,\ldots,i_N}$. The inverse process, *Tensorization*, is denoted by $\mathsf{ten}(\,)$.

Tensor Contraction: An (m, n)-contraction [16], denoted by $\overset{m}{n}$, 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 a third order-$(N + M - 2)$ tensor, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}$, where

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

Tensor Decompositions: Tensor Decomposition (TD) methods approximate high-order, large-dimensional tensors via contractions of smaller core tensors, therefore drastically reducing the computational complexity for tensor operations while preserving the data structure [17]. We here consider the Matrix Product Operator (MPO) definition of the Tensor-Train decomposition (TTD) [8], a highly efficient TD method that can decompose a large order-$2N$ tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$, into smaller contracting core tensors, $\boldsymbol{\mathcal{G}}^{(n)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$, as $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{G}}^{(1)} \overset{1}{4} \boldsymbol{\mathcal{G}}^{(2)} \overset{1}{4} \cdots \overset{1}{4} \boldsymbol{\mathcal{G}}^{(N)}$, 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 [18, 19]. Finally, Quantized TTD (QTTD) achieves *super-compression* if $I_n = J_n = 2$ for all n, and $R_n = 2$ for $n = 1, \ldots, N \quad 1$.

2.2 Spectral Graph Neural Networks

Graph Matrices: A graph, $G = \{V, E\}$, is defined by a set of L vertices (or nodes) $v_l \quad V$ for $l = 1, \ldots, L$, and a set of edges connecting the $l_1{}^{th}$ and $l_2{}^{th}$ vertex, $e_{l_1,l_2} = (v_{l_1}, v_{l_2}) \in E$, for $l_1 = 1, \ldots, L$ and $l_2 = 1, \ldots, L$. A graph can be fully described by the weighted adjacency matrix, $\mathbf{A} \in \mathbb{R}^{L \quad L}$, such that $a_{l_1,l_2} > 0$ if $e_{l_1,l_2} \in E$, and $a_{l_1,l_2} = 0$ if $e_{l_1,l_2} \notin E$. Alternatively, a graph can be described in terms of its Laplacian matrix, $\mathbf{L} \in \mathbb{R}^{L \quad L}$, defined as $\mathbf{L} = \mathbf{D} \quad \mathbf{A}$, where $\mathbf{D} \in \mathbb{R}^{L \quad L}$ is the diagonal degree matrix such that $d_{l_1,l_1} = \sum_{l_2} a_{l_1,l_2}$ [20]. If the edge weights are not already given in a problem setting, they can be defined based on a pair-wise similarity function, $g(\,)$, 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 [21].

Graph Fourier Transform: If the underlying graph is undirected, then the adjacency matrix admits an eigen-decomposition as $\mathbf{A} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$, where $\mathbf{\Phi}$ is the matrix of orthonormal eigenvectors, and $\mathbf{\Lambda}$ is the diagonal matrix of real-valued eigenvalues. In spectral graph theory, the eigenvectors (the columns of $\mathbf{\Phi}$) are regarded as the graph Fourier bases of a given graph domain, whereby the corresponding eigenvalues in $\mathbf{\Lambda}$ represent the associated frequencies [20]. More specifically, given a graph signal, $\mathbf{x} \in \mathbb{R}^L$, where a scalar value is associated with each of the L vertices, its graph Fourier transform can be computed as $\hat{\mathbf{x}} = \mathbf{\Phi}^T\mathbf{x}$, while the inverse Fourier transform can be computed as $\mathbf{x} = \mathbf{\Phi}\hat{\mathbf{x}}$ [22].

Remark 1. The Graph Fourier Transform can be defined both in terms of the adjacency matrix and the Laplacian matrix, albeit with the opposite eigenvalue ordering. For instance, the smoothest eigenvector (lowest frequency) is associated with the largest eigenvalue for the adjacency matrix case, but it is associated with the smallest eigenvalue in the Laplacian matrix case [22].

Spectral Graph Convolution: A Spectral Graph Neural Network learns a diagonal matrix of spectral multipliers, $\{\mathbf{\Gamma}_p\}$, for $p = 1, \ldots, P$, to extract features from a graph signal, $\mathbf{x}$, via the spectral convolution operation, $\mathbf{y} = \epsilon(\sum_{p=1}^{P} \mathbf{\Phi}\mathbf{\Gamma}_p\mathbf{\Phi}^T\mathbf{x})$ [23], where $\epsilon(\,)$ is an optional activation function. This follows from the convolution theorem where the spatial convolution of two signals correspond to their product in the Fourier domain.

2.3 Dot-Product Attention

Query, Key, Value: Given an input time-series matrix, $\mathbf{X} \in \mathbb{R}^{L \quad I}$, where I features are indexed along L time-steps, the dot-product attention [1] computes: (i) $\mathbf{Q} \in \mathbb{R}^{L \quad J}$, a query-representation of the input data generated by the transform, $\mathbf{Q} = \mathbf{X}\mathbf{W}^{(q)}$, where $\mathbf{W}^{(q)} \in \mathbb{R}^{I \quad J}$ is a trainable query-weight matrix; (ii) $\mathbf{K} \in \mathbb{R}^{L \quad J}$, a key-representation of the input data generated by the transform, $\mathbf{K} = \mathbf{X}\mathbf{W}^{(k)}$, where $\mathbf{W}^{(k)} \in \mathbb{R}^{I \quad J}$ is a trainable key-weight matrix; (iii) $\mathbf{V} \in \mathbb{R}^{L \quad J}$, a value-representation of the input data generated by the linear transform, $\mathbf{V} = \mathbf{X}\mathbf{W}^{(v)}$, where $\mathbf{W}^{(v)} \in \mathbb{R}^{I \quad J}$ is a trainable value-weight matrix.

Attention Coefficients: Given $\mathbf{Q}$ and $\mathbf{K}$, the self-attention coefficient matrix, $\mathbf{\Theta} \in \mathbb{R}^{L \quad L}$, is computed as $\mathbf{\Theta} = \sigma(\frac{1}{\sqrt{d_k}}\mathbf{Q}\mathbf{K}^T)$, where $\sigma(\,)$ is a $softmax$ activation function and $\frac{1}{\sqrt{d_k}}$ is a scaling factor. More specifically, the attention coefficient between the l_1-th and l_2-th time-step, θ_{l_1,l_2}, is defined as the inner product between the l_1-th query vector, $\mathbf{q}_{l_1}$, and l_2-th key vector, $\mathbf{k}_{l_2}$, as $\theta_{l_1,l_2} = \sigma(\frac{1}{\sqrt{d_k}}\mathbf{q}_{l_1}^T\mathbf{k}_{l_2})$.

Multi-Head Attention: Given $\mathbf{\Theta}$, the attention mechanism then generates the final output, $\mathbf{Y} \in \mathbb{R}^{L \quad J}$, 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, $\{\mathbf{Y}_1, \ldots, \mathbf{Y}_H\}$, that are then concatenated and combined to produce the final output, $\mathbf{Y}$.

2.4 Graph Tensor Networks

General Recurrent Graph Tensor Network: Consider a sequence learning problem where the input data, $\mathbf{X} \in \mathbb{R}^{L \quad J}$, is a time-series matrix with J features indexed along L time-steps. For a so

defined input matrix, a Recurrent Graph Tensor Network (RGTN) [7] extracts time-series features, $\mathbf{Y} \in \mathbb{R}^{L \times J}$, by applying a tensor contraction based forward pass, $\mathbf{Y} = \mathsf{ten}(\mathbf{I} + \mathbf{A} \otimes \mathbf{P}) \times_{3,4}^{1,2} \mathbf{X}$, where $\mathbf{I} \in \mathbb{R}^{LJ \times LJ}$ is an identity matrix, $\mathbf{P} \in \mathbb{R}^{J \times J}$ is an idempotent matrix modelling the propagation of information between successive time-steps, and $\mathbf{A} \in \mathbb{R}^{L \times L}$ is a time-domain graph adjacency matrix. More specifically, the time-adjacency matrix, $\mathbf{A}$, 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 with 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.

Fast Recurrent Graph Tensor Network: A low complexity version of RGTN was also proposed in [7] by approximating $\mathbf{P} \approx \mathbf{I}$, which simplifies the RGTN forward pass to $\mathbf{Y} = (\mathbf{I} + \mathbf{A})^{\frac{1}{2}} \mathbf{X}$. For multi-modal input features, the RGTN forward pass can be easily extended to tensor form, $\mathcal{Y} = (\mathbf{I} + \mathbf{A})^{\frac{1}{2}} \times_1 \mathcal{X}$, where $\mathcal{X} \in \mathbb{R}^{L \times J_1 \times \cdots \times J_N}$ is a time-series tensor of order-$(N+1)$, which contains features of dimensionality $J_1 \times \cdots \times J_N$ for each of the L time-steps.

Bi-Directional Graph Tensor Network: The time-adjacency matrix, $\mathbf{A}$, was originally developed to model sequential data. However, more generally, it can be replaced by other adjacency matrices depending on the demands of an application [6]. For instance, 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 $\mathbf{\Omega} = \frac{1}{2}(\mathbf{A}^T + \mathbf{A})$.

3 Spectral Attention

3.1 Time-Domain Attention Graph

We shall now introduce the notion of the time-domain attention graph. 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 $\theta_{l_1,l_2} = \epsilon(\rho \frac{1}{\sqrt{d_k}} \mathbf{k}_{l_1}^T \mathbf{k}_{l_2})$ for $l_1 \neq l_2$ and $\theta_{l_1,l_2} = 0$ otherwise, with $\epsilon(\,)$ as a $relu$ activation function, $\rho \frac{1}{\sqrt{d_k}}$ 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 $\mathbf{W}^{(k)}$, as discussed in Section 2.2.

Remark 2. The choice of a $relu$ activation to generate the adjacency matrix, $\mathbf{\Theta}$, serves two purposes: (i) the rectification promotes sparsity, which effectively reduces the number of connections between the nodes in the graph domain, as the weights are reduced to zero; (ii) the resulting entries are strictly non-negative, which is an usual assumption for weighted graph adjacency matrices.

Due to its dot product formulation, $\mathbf{\Theta}$ is context dependent, but it is ignorant to the time-series structure of the input data, which contains important sequence ordering information. To this end, we incorporate the bi-directional time graph adjacency matrix, $\mathbf{\Omega}$, as discussed in Section 2.4, 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 3. Unlike classical attention, the time-domain attention adjacency matrix, $\mathbf{\Psi}$, 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, $y_{l_1} = \sum_{l_2 \in \mathcal{N}_{l_1}} \psi_{l_1,l_2} = \theta_{l_1,l_2} \omega_{l_1,l_2} x_{l_2}$. 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, θ_{l_1,l_2}, and the time proximity, ω_{l_1,l_2}, between tokens at time-steps l_1 and l_2.

3.2 Spectral Attention Filtering

By construction, $\mathbf{\Psi}$ is a symmetric graph adjacency matrix representing an undirected graph, which admits an orthonormal eigen-decomposition of $\mathbf{\Psi} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T$, as discussed in Section 2.2. Generally,

we can learn a filter, $\mathbf{\Gamma}$, in the graph Fourier domain to perform spectral convolution on a graph signal as $\mathbf{y} = \mathbf{\Phi}\mathbf{\Gamma}\mathbf{\Phi}^T\mathbf{x}$. To reduce the risk of over-fitting, we can restrict the class of graph filters to be a polynomial function of the eigenvalues of order $(K \quad 1)$ to ensure smooth spectral multipliers [23], that is $\mathbf{\Gamma} = f_\alpha(\mathbf{\Lambda})$, such that $f_\alpha(\mathbf{\Lambda}) = \sum_{k=0}^{K \quad 1} \alpha_k \mathbf{\Lambda}^k$, where $f\alpha_k g$ is the set of filter coefficients. This simplifies the spectral convolution to

$$
\begin{aligned}
\mathbf{y} &= \mathbf{\Phi}\mathbf{\Gamma}\mathbf{\Phi}^T\mathbf{x} \\
&= \mathbf{\Phi} f_\alpha(\mathbf{\Lambda})\mathbf{\Phi}^T\mathbf{x} \\
&= \mathbf{\Phi}(\alpha_0\mathbf{\Lambda}^0 + \alpha_1\mathbf{\Lambda}^1 + \quad + \alpha_{K \quad 1}\mathbf{\Lambda}^{K \quad 1})\mathbf{\Phi}^T\mathbf{x} \\
&= (\alpha_0\mathbf{\Phi}\mathbf{\Lambda}^0\mathbf{\Phi}^T + \alpha_1\mathbf{\Phi}\mathbf{\Lambda}^1\mathbf{\Phi}^T + \quad + \alpha_{K \quad 1}\mathbf{\Phi}\mathbf{\Lambda}^{K \quad 1}\mathbf{\Phi}^T)\mathbf{x} \\
&= (\alpha_0\mathbf{\Psi}^0 + \alpha_1\mathbf{\Psi}^1 + \quad + \alpha_{K \quad 1}\mathbf{\Psi}^{K \quad 1})\mathbf{x} \\
&= f_\alpha(\mathbf{\Psi})\mathbf{x}
\end{aligned}
\tag{1}
$$

To reduce the computational and parameter complexity of the proposed attention filtering operation, $\mathbf{y} = f_\alpha(\mathbf{\Psi})\mathbf{x}$, we can restrict the order of the graph filter to $K = 2$, which simplifies (1) to $\mathbf{y} = (\alpha_0\mathbf{I} + \alpha_1\mathbf{\Psi})\mathbf{x}$. Furthermore, because $\mathbf{\Psi}$ is inherently trainable, the coefficient α_1 can be absorbed in $\mathbf{\Psi}$ and α_0 can be set to 1, which simplifies the attention filtering to $\mathbf{y} = (\mathbf{I} + \mathbf{\Psi})\mathbf{x}$.

Remark 4. Notice that the low complexity filtering operation, $\mathbf{y} = (\mathbf{I} + \mathbf{\Psi})\mathbf{x}$, is a special case of the RGTN forward pass discussed in Section 2.4, $\mathcal{Y} = (\mathbf{I} + \mathbf{\Psi}) \quad \frac{1}{2}\mathcal{X}$, which generalizes the graph filtering operation to process higher-order tensors.

Remark 5. The full order-$(K \quad 1)$ filtering operation, $\mathbf{y} = f_\alpha(\mathbf{\Psi})\mathbf{x}$, operates in a $(K \quad 1)$-hop neighbourhood in terms of node (or time-step) connections. The simplified attention filtering, $\mathbf{y} = (\mathbf{I} + \mathbf{\Psi})\mathbf{x}$, operates in a 1-hop neighbourhood. However, within the proposed model, we can reach $(K \quad 1)$-hop neighbours by stacking $(K \quad 1)$ layers of simplified attention filtering [24].

Finally, similar to the classical multi-head attention mechanism, we can repeat the attention filtering operation H times, using a set of attention filters, $f\mathbf{\Psi}_1, \mathbf{\Psi}_2, \ldots, \mathbf{\Psi}_H g$, to generate H different filtered signals that can be then combined together to generate the final output.

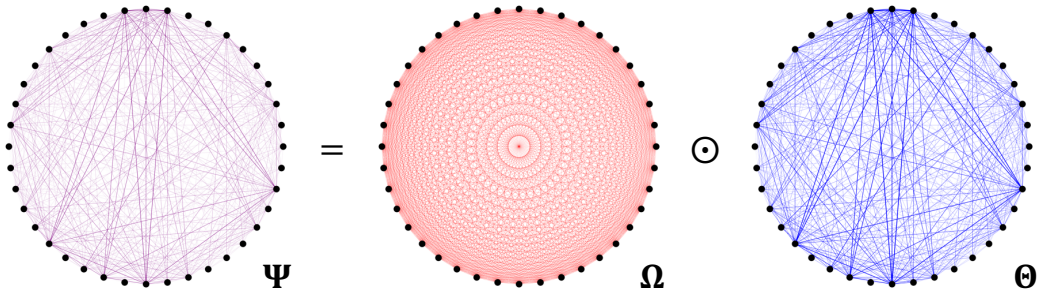

Figure 1: *Principle of the time-domain attention graph (in purple), computed from the bi-directional time-graph (in red), and the symmetric attention coefficient graph (in blue), as a Hadamard product of their adjacency matrices,* $\mathbf{\Psi} = \mathbf{\Omega} \quad \mathbf{\Theta}$. *The graph vertices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

4 Tensorizing Attention

4.1 Tensorizing Input Data

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, $\mathbf{X} \; 2 \; \mathsf{R}^{L \quad I}$. However, due to the "flat-view" nature of matrix methods, performing attention on $\mathbf{X}$ incurs expensive matrix multiplications. This results in high complexity costs, especially when working with large embedding dimensions.

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 benefiting from the ability of Graph Tensor Networks to operate with high-dimensional tensors at low-complexity. By formulating the problem in the tensor space, we can leverage on the power of tensor networks to drastically reduce the complexity needed to achieve high expressive power.

4.2 Tensorizing Large Dimensional Matrix Multiplications

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 $I = I_1 \cdots I_N$ and $J = J_1 \cdots J_N$. 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 $\mathcal{W} \in \mathbb{R}^{I_1 \times J_1 \times I_2 \times J_2 \times \cdots \times I_N \times J_N}$. In turn, this allows us to re-write the large-dimensional matrix multiplication, which is ubiquitous to the attention mechanism, as a higher order tensor contraction

$$\mathcal{Y} = \mathcal{X} \begin{smallmatrix} 1,3,5,\ldots,2N-1 \\ 2,3,4,\ldots,N+1 \end{smallmatrix} \mathcal{W} \tag{2}$$

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

$$\mathcal{W} \approx \mathcal{G}_1 \begin{smallmatrix} 1 \\ 4 \end{smallmatrix} \mathcal{G}_2 \begin{smallmatrix} 1 \\ 4 \end{smallmatrix} \mathcal{G}_3 \begin{smallmatrix} 1 \\ 4 \end{smallmatrix} \cdots \begin{smallmatrix} 1 \\ 4 \end{smallmatrix} \mathcal{G}_N \tag{3}$$

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

By storing $\mathcal{W}$ in the TT format, we can express the large-scale contraction in (2), as a series of smaller-scale tensor contractions

$$\begin{aligned} \mathcal{Y} &= \mathcal{X} \begin{smallmatrix} 1,3,5,\ldots,2N-1 \\ 2,3,4,\ldots,N+1 \end{smallmatrix} \Big(\mathcal{G}_1 \begin{smallmatrix} 1 \\ 4 \end{smallmatrix} \mathcal{G}_2 \begin{smallmatrix} 1 \\ 4 \end{smallmatrix} \mathcal{G}_3 \begin{smallmatrix} 1 \\ 4 \end{smallmatrix} \cdots \begin{smallmatrix} 1 \\ 4 \end{smallmatrix} \mathcal{G}_N\Big) \\ &= \mathcal{X} \begin{smallmatrix} 2 \\ 2 \end{smallmatrix} \mathcal{G}_1 \begin{smallmatrix} 2,1 \\ 2,N+3 \end{smallmatrix} \mathcal{G}_2 \begin{smallmatrix} 2,1 \\ 2,N+3 \end{smallmatrix} \cdots \begin{smallmatrix} 2,1 \\ 2,N+3 \end{smallmatrix} \mathcal{G}_N \end{aligned} \tag{4}$$

To further reduce the parameter complexity, we can employ the notion of Quantized Tensor-Train Decomposition (QTTD), by: (i) setting the dimensions 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 $n = 1, \ldots, N-1$. This effectively *super-compresses* the parameter complexity from an exponential $\prod_{n=1}^{N} I_n J_n = 4^N$ to a linear $\sum_{n=1}^{N} R_{n-1} I_n J_n R_n = 16(N-1)$.

Remark 7. The use of the compressed TT format in (4) to replace all dense matrix multiplications in the attention mechanism makes it possible to drastically reduce the parameter complexity from an exponential one to a linear one in the tensor order N.

4.3 Tensorizing Attention Coefficients

To perform spectral attention filtering in the tensor space, we need to compute the attention coefficient matrix, $\mathbf{\Theta}$, from the tensorized input. To achieve this, 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 (4), 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 also equivalent to computing $\mathbf{\Theta} = \epsilon(\frac{1}{\sqrt{d_k}} \mathcal{K} \begin{smallmatrix} 2, \ldots, N+1 \\ 2, \ldots, N+1 \end{smallmatrix} \mathcal{K})$ and then setting its diagonal elements to zero.

5 Putting Everything Together

The Graph Tensor Network (GTN) based Tensorized Spectral Attention (TSA) is summarised in Algorithm 1, which can be repeated for H different heads to perform multi-head TSA. In practice, the condition $I_n = J_n = 2$ is set for Algorithm 1 to *super-compress* the parameter complexity costs.

Algorithm 1: Tensorized Spectral Attention (TSA) Algorithm

Input : Data matrix $\mathbf{X} \in \mathbb{R}^{L \times I}$ and time-domain adjacency matrix $\mathbf{\Omega} \in \mathbb{R}^{L \times L}$
Output : Feature matrix $\mathbf{Y} \in \mathbb{R}^{L \times J}$
Initialize : Initialize $R_n = 2$ for $n = 1, \ldots, N$ and $R_0 = R_N = 1$
Initialize $I_1, \ldots, I_N$ such that $\prod_{n=1}^{N} I_n = I$
Initialize $J_1, \ldots, J_N$ such that $\prod_{n=1}^{N} J_n = J$
Initialize trainable key TT-cores $\mathcal{G}_n^{(k)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$ for $n = 1, \ldots, N$
Initialize trainable value TT-cores $\mathcal{G}_n^{(v)} \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$ for $n = 1, \ldots, N$

$\mathcal{X} \quad \text{ten}(\mathbf{X})$ // Tensorization from $\mathbf{X} \in \mathbb{R}^{L \times I}$ to $\mathcal{X} \in \mathbb{R}^{L \times I_1 \times \cdots \times I_N}$
$\mathcal{K} \quad \mathcal{X} \times_{2} \mathcal{G}_1^{(k)} \times_{2,N+3} \mathcal{G}_2^{(k)} \times_{2,N+3} \cdots \times_{2,N+3} \mathcal{G}_N^{(k)}$ // Key tensor
$\mathcal{V} \quad \mathcal{X} \times_{2} \mathcal{G}_1^{(v)} \times_{2,N+3} \mathcal{G}_2^{(v)} \times_{2,N+3} \cdots \times_{2,N+3} \mathcal{G}_N^{(v)}$ // Value tensor
$\mathbf{\Theta} \quad \epsilon(\frac{1}{\sqrt{d_k}} \mathcal{K} \times_{2,\ldots,N+1} \mathcal{K})$ // Attention coefficient matrix
$\mathbf{\Theta} \quad \mathbf{\Theta} - diag(\mathbf{\Theta})$ // Remove diagonal
$\mathbf{\Psi} \quad \mathbf{\Omega} \odot \mathbf{\Theta}$ // Time-domain attention adjacency matrix
$\mathcal{Y} \quad (\mathbf{I} + \mathbf{\Psi}) \times_{2} \frac{1}{2} \mathcal{V}$ // Spectral attention filtering of the tensor embeddings
$\mathbf{Y} \quad \text{mat}(\mathcal{Y})$ // Matricization from $\mathcal{Y} \in \mathbb{R}^{L \times J_1 \times \cdots \times J_N}$ to $\mathbf{Y} \in \mathbb{R}^{L \times J}$

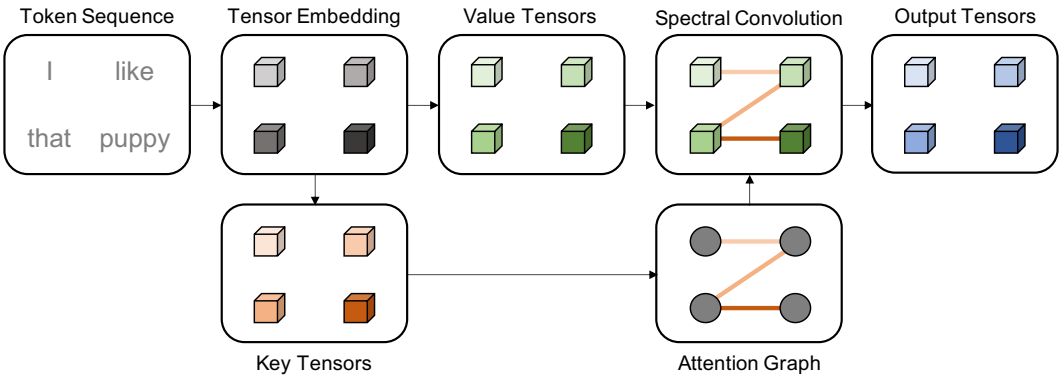

Figure 2: *Principle of the tensorized spectral attention mechanism. A sequence of tokens are embedded in the tensor space (in gray). The sequence of tensor embeddings are then mapped to generate the key tensors, $\mathcal{K}$ (in orange), and the value tensors, $\mathcal{V}$ (in green). The key tensors are then used to generate the adjacency matrix of the attention graph, $\mathbf{\Psi}$, which is used to perform spectral graph convolution on value tensors to generate the output tensors, $\mathcal{Y}$ (in blue).*

6 Experimental Results

The performance of the proposed TSA was evaluated in terms of its parameter complexity, expressive power, and performance on various Natural Language Processing (NLP) tasks. Experimental results confirm the superiority of the proposed model against standard attention mechanisms, achieving a better or comparable performance at a drastically lower complexity across five different datasets. The full experiment code is available at `www.github.com/gylx/Tensorized-Spectral-Attention`.

Experiment Setting: To benchmark the performance of the proposed TSA against classical attention mechanisms, we compared the training and testing accuracy of the considered models at different complexity levels across five NLP benchmark datasets. More specifically, we compared the achieved accuracy scores by varying the word embedding dimension in the attention layer while keeping all other parameters constant, which resulted in models of varying expressive power and complexity. This allowed us to analyse the performance-complexity trade-offs of the considered models. Finally, the experiments were repeated over 5 independent trials for each benchmark dataset.

Metrics: To evaluate the performance of the proposed model, we considered: (i) Training Accuracy (TrA) as a measure of expressive power, (ii) Testing Accuracy (TeA) as a measure of performance, and (iii) Number of Parameters (NP) as a measure of model complexity.

Datasets: Five benchmark NLP classification datasets were considered for the evaluation of the TSA model: (i) Reuters Newswire[1] (RN), (ii) 20 Newsgroups[2] (NG), (iii) Drugs Review[3] (DR), (iv) Research Articles[4] (RA), and (v) Spooky Authors Identification[5] (SA). All datasets consist of text data that were pre-processed as tokenized sequences of 200 words by considering the 20,000 most common words in the dataset. A train-test split of 60%-40% was used for all datasets.

Baselines: We benchmarked the proposed Tensorized Spectral Attention (TSA) against Luong-style Dot-Product Attention (DPA) and Bahdanau-style Additive Attention (ADA) based models. For a fair comparison, all considered models have the exact same training setting and model architecture, with the sole exception being the attention mechanism used. More specifically, we implemented a seven-layer architecture consisting of: (i) word embedding layer, (ii) multi-head ($H = 2$) attention layer (based either on the TSA, DPA, or ADA), (iii) global average pooling layer, (iv) drop-out layer with 10% drop-out rate, (v) dense layer with 20 hidden units and $relu$ activation, (vi) drop-out layer with 10% drop-out rate, and (vii) dense layer with $Softmax$ activation. The models were trained using the categorical cross-entropy loss function with an *Adam* optimizer.

As discussed in Section 1, we did not compare the proposed TSA against the TensorCoder in [10] due to its dimension-wise attention formulation, and the Multi-Linear Attention in [5] due to its constraint on the embedding dimension, which has to be at least equal to the sequence length for a fair comparison.

Implementation: All models were implemented using TensorFlow 2.4.0 and trained on a 2.4 GHz 8-Core Intel Core i9 CPU with 32GB RAM. The experiments took approximately 2 days to complete. For additional details, please refer to the full code at the provided link.

Results: The experiment results are illustrated in Figure 3. Across varying levels of word embedding dimensionality, the proposed TSA is shown to achieve better or comparable expressive power (measured in training accuracy) and performance (measured in testing accuracy), while incurring drastically lower complexity costs (measured in the number of parameters). Table 6 summarises the results at a low complexity level of around 300 trainable parameters, which further highlights the strong performance-to-complexity characteristics of the proposed TSA, as it achieved remarkably better performances compared to standard attention mechanisms.

Table 1: *Expressive power measured in training accuracy (TrA, top table), and performance measured in testing accuracy (TeA, bottom table) of the considered models at a low complexity level (measured in number of parameters, NP, of around 300) across five datasets (RN, NG, DR, RA, SA).*

Metric	Model	RN	NG	DR	RA	SA
TrA (%)	ADA	57.5 ± 2.7	48.7 ± 12.9	77.0 ± 1.9	76.0 ± 4.9	93.9 ± 1.2
	DPA	55.7 ± 4.4	46.0 ± 10.0	74.7 ± 4.3	80.3 ± 2.4	92.6 ± 2.3
	TSA	$\mathbf{85.3 \pm 1.0}$	$\mathbf{96.9 \pm 0.5}$	$\mathbf{96.3 \pm 0.2}$	$\mathbf{90.2 \pm 3.0}$	$\mathbf{97.4 \pm 0.4}$
TeA (%)	ADA	58.4 ± 3.0	49.0 ± 13.1	77.6 ± 2.8	71.9 ± 3.2	$\mathbf{81.3 \pm 0.7}$
	DPA	58.2 ± 3.6	49.0 ± 9.0	75.7 ± 4.5	71.8 ± 2.4	81.3 ± 1.6
	TSA	$\mathbf{73.9 \pm 0.5}$	$\mathbf{82.8 \pm 1.0}$	$\mathbf{87.7 \pm 0.4}$	$\mathbf{72.7 \pm 0.9}$	80.6 ± 0.8

Remark 8. As shown in Table 1, at a low complexity level (using around 300 trainable parameters), the proposed TSA framework achieved substantially better results than the classical attention model, which generated poor performance due to the excessively small embedding dimension ($I = 6$) needed to satisfy the low complexity requirement.

[1] Available at https://trec.nist.gov/data/reuters/reuters.html

[2] Available at http://people.csail.mit.edu/jrennie/20Newsgroups

[3] Available at https://archive.ics.uci.edu/ml/datasets/Drug+Review+Dataset+%28Drugs.com%29

[4] Available at https://www.kaggle.com/blessondensil294/topic-modeling-for-research-articles

[5] Available at https://www.kaggle.com/c/spooky-author-identification

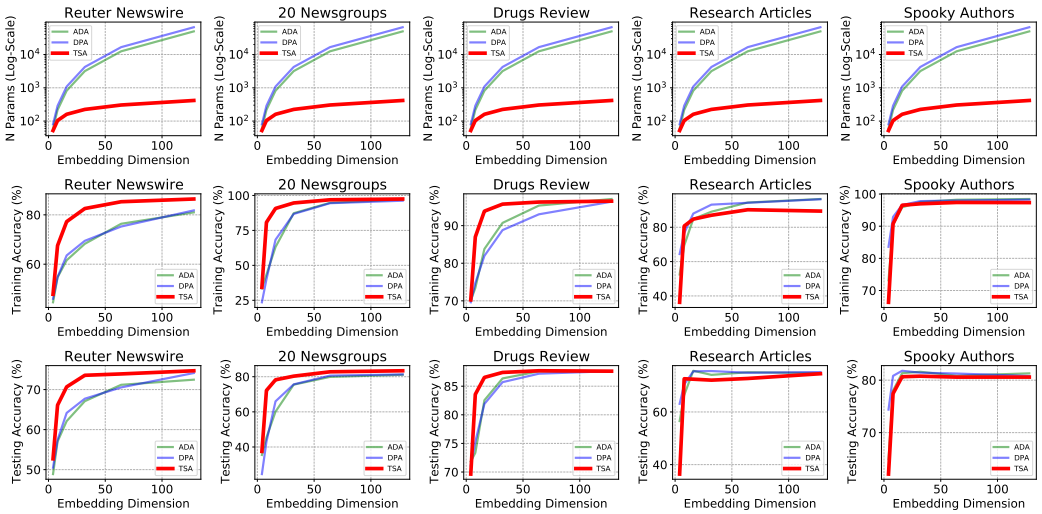

Figure 3: *Performance comparison of the proposed Tensorized Spectral Attention (TSA), Dot-Product Attention (DPA), and Additive Attention (ADA) models. The figure columns, from left to right, correspond to the five considered datasets. The figure rows, from top to bottom, correspond to: (i) embedding dimension vs number of parameters (complexity), (ii) embedding dimension vs training accuracy (expressive power), and (iii) embedding dimension vs testing accuracy (performance).*

7 Limitations and Future Work

The proposed QTTD approach (TTD with rank 2) achieves *super-compression* by reducing the parameter complexity optimally, but it does not guarantee optimal computational complexity. An important future research direction thus concerns the determination of TTD rank for optimal computational complexity.

8 Conclusion

We have introduced a novel Tensorized Spectral Attention (TSA) mechanism for Natural Language Processing (NLP). By virtue of its conjoint tensor and graph formulation, the proposed framework has been shown to achieve high expressive power at a drastically lower parameter complexity compared to traditional attention models. Experimental results have verified the desirable properties of the proposed TSA, with better or comparable performance achieved across five NLP benchmark tasks at varying levels of complexity. The performance gain has been demonstrated to be especially fruitful in low complexity cases.

Broader Impact

The proposed TSA exploits the notion of graph spectral convolution and tensor decomposition to achieve high expressive power at a low complexity. An immediate impact of our work, as illustrated through the considered experiments, is the relaxation of computational burden for training large-scale attention-based models (e.g. transformers), resulting in lighter neural language models suitable for practical training and deployment. The broader impact, as well as the impact arising from model failure and data bias, is identical to those of classical attention models in general. Finally, we have not identified anyone that can be put at disadvantage from this work, nor any negative societal impact.

Acknowledgments and Disclosure of Funding

Yao Lei Xu and Kriton Konstantinidis are supported by EPSRC scholarships, Shengxi Li is supported by the Lee Family scholarship at Imperial College London.

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)*, page 5998–6008, 2017.

[2] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training. *OpenAI*, 2018. URL `http://openai-assets.s3.amazonaws.com/research-covers/language-unsupervised/language_understanding_paper.pdf`.

[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*, 32:2232–2242, 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*, 33(5):2295–2317, 2011.

[9] 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*, pages 4847–4860, 2020.

[10] 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.

[11] L. De Lathauwer. Decompositions of a higher-order tensor in block terms—part ii: Definitions and uniqueness. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1033–1066, 2008.

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

[13] 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.

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

[15] S.V. Dolgov and D.V. Savostyanov. Alternating minimal energy methods for linear systems in higher dimensions. *SIAM Journal on Scientific Computing*, 36(5):A2248–A2271, 2014.

[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*, March 2014.

[17] 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*, 9(4-5):249–429, 2016.

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

[19] 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.

[20] 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*, 13(1):1–157, 2020.

[21] 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*, 13(4):332–530, 2020.

[22] 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*, 13(2–3):158–331, 2020.

[23] M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[24] 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*, pages 1–21, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]