Long Sequence Modeling with Attention Tensorization: From Sequence to Tensor Learning

Aosong Feng, Rex Ying and Leandros Tassiulas
Yale University
{aosong.feng, rex.ying, leandros.tassiulas}@yale.edu

Abstract

As the demand for processing extended textual data grows, the ability to handle long-range dependencies and maintain computational efficiency is more critical than ever. One of the key issues for long-sequence modeling using attention-based model is the mismatch between the limited-range modeling power of full attention and the long-range token dependency in the input sequence. In this work, we propose to scale up the attention receptive field by tensorizing long input sequences into compact tensor representations followed by attention on each transformed dimension. The resulting Tensorized Attention can be adopted as efficient transformer backbones to extend input context length with improved memory and time efficiency. We show that the proposed attention tensorization encodes token dependencies as a multi-hop attention process, and is equivalent to Kronecker decomposition of full attention. Extensive experiments show that tensorized attention can be used to adapt pretrained LLMs with improved efficiency. Notably, using customized Triton kernels, tensorization enables Llama-8B training under 32,768 context length and can steadily extrapolate to 128k length during inference with $11\times$ speedup (compared to full attention with FlashAttention-2).

1 Introduction

In the swiftly advancing field of natural language processing (NLP), large language models (LLMs) like GPT-4 and Llama have emerged as vital tools, demonstrating expertise in comprehending and producing human language. These complex scenarios often involve context lengths longer than those LLMs were trained on, posing significant challenges to Transformer-based architectures in handling long sequences. For example, LLMs are expected to read long paragraphs or books to answer questions while they are usually trained with a much smaller context length (like 8k for Llama-3 (Touvron et al., 2023)).

Scaling up LLMs for long sequences presents significant challenges related to limited attention windows during training and length extrapolation during inference. First, transformers usually adopt a bounded context window in the training phase due to quadratic computation costs (Vaswani et al., 2017; Tay et al., 2022). To mitigate these costs and expand the training context window, existing works usually adopt attention approximation methods like sparse, low-rank, and softmax-free attention. Despite their efficiency and scalability with relatively long sequences, these methods gain less popularity due to inefficient implementations and incompatibility with pre-trained LLMs widely used in community.

Given such open-sourced pretrained LLMs, researchers start to focus more on the length extrapolation challenge: leveraging the short-range modeling power acquired during pre-training to handle unseen long-range dependencies. Addressing this challenge requires methods that take into account both efficiency and effectiveness perspectives. One line of thought is to employ full attention with hardware-efficient implementations such as FlashAttention and quantization for efficiency, paired with positional extrapolation (Su et al., 2024) or interpolation (Chen et al., 2023a) for enhanced performance. While full attention captures all potential correlations, it significantly increases running time, and many correlations in long sequences are in fact unnecessary and distracting. The second line of approaches involves using segmented or sliding windows for efficiency, supplemented by additional recurrent memory modules (Bulatov et al., 2023; Wang et al., 2023) or global sinks (Xiao et al., 2023; Han et al., 2023) to integrate segments. However, it remains difficult for these additional modules to retain past or global information as windows move and update, posing a persistent challenge in processing long sequences effectively.

Proposed Work. To address these challenges in modeling long-range dependencies, we propose folding the one-dimensional long-range token interactions into a higher-order tensor, where each dimension can be effectively modeled with short-range interactions. As shown in Figure 1(a), given a limited context window, tensorization reduces the interaction distance between two tokens, converting previously out-of-window interactions into within-window interactions. Inspired by the compact representation of the input tensor, we generalize input at the attention layer from the conventional sequence to higher-order tensor format and propose a supporting tensorized attention mechanism that replaces traditional vector operations with corresponding tensor operations.

Unlike traditional attention approximations, tensorized attention is implemented with an efficient custom Triton kernel and can be adapted from pretrained LLMs through continued pretraining. Besides, the multi-dimensional encoding of token interactions provides tensorized attention with natural advantages for length extrapolation. Given finite attention window size or memory budget, tensorized attention captures correlations along different dimensions and incorporates hierarchical multi-hop interactions in the original sequence at multiple scales as shown in Figure 1(b), which exponentially extends the effective context length and avoids the time consumption of using full attention. Compared to segmented and sliding windows, tensorized attention hierarchically merges interactions from low to high levels and does not need additional global or recurrent modules to retrieve past information under exponential extrapolation speed.

We then integrates Tensorized Attentions into LLMs to improve the model scalability to longer-sequence inputs. Experiments with diverse NLP datasets show that tensorized attention improves the performance and efficiency of existing pretrained models on long-sequence tasks after continued pretraining. Notably, tensorized attention enables Llmama-3-8B training under 32,768 context length and can steadily extrapolate to 128k length during inference with 11× speedup (compared to full attention with FlashAttention-2).

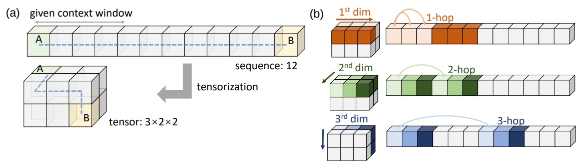

Figure 1: (a) The interaction distance from A to B is decreased from 12 in the sequence format to 5 in the tensor format and fits into context length. (b) Token interactions along each dimension are equivalent to multi-scale interaction in the original sequence.

2 Related Works

2.1 Context Window Extension

Efficient transformers have adopted various attention approximation including sparse and low-rank methods to extend the context length that can be processed by one attention layer. For example, sparse Transformer (Child et al., 2019), Longformer (Beltagy et al., 2020), Bigbird (Zaheer et al., 2020) and Diffusers (Feng et al., 2022) adopt combinations of random, window, and global sparse attention targeting the long-sequence scenario. Data-adaptive sparsity has also been considered in recent works including Reformer (Kitaev et al., 2020) BiFormer (Zhu et al., 2023b), DAT (Xia et al., 2022), and SparseViT (Chen et al., 2023b). The proposed tensorization is compatible with such sparisty masks. Low-rank attention is based on the assumption that the attention matrix has an intrinsic low-rank structure. Linformer (Wang et al., 2020), Performer (Katharopoulos et al., 2020), Linear Transformer (Katharopoulos et al., 2020), Synthesizer (Tay et al., 2021), and LRT (Winata et al., 2020) achieve efficient attention with linear complexity by projecting attention matrix to a lower dimension in the column vector space along the query or key dimension. However, these low-rank assumptions are mostly based on the vector space by directly reducing q,k,v sequence length, ignoring the hierarchical structure embedded in the attention matrix. Given the observed structure of attention matrices, we show that attention representations in the tensor space which is constructed by a set of hierarchically decomposed attention blocks can be better approximated with low-rank matrices.

2.2 Length Extrapolation

Given limited context length during training, length extrapolation aims to extend the model's comprehension to sequences beyond training context length. These extrapolation methods can be mainly summarized into three categories including position encoding variation, window based approach, and memory/global token augmented approach.

Position encoding improves the model's understanding of the sequential structure of input sequences. RoPE (Su et al., 2024) deploys distinct rotary matrices to calculate scores between keys and queries using relative position information. While widely adopted in Llama and PaLM, it has been shown limited in zero-shot extrapolation setting (Press et al., 2021). Position Interpolation is adopted by (Chen et al., 2023a), Yarn (Peng et al., 2023), and Pose (Zhu et al., 2023a), which interpolate the position indices within the pretrained context and improves RoPE extrapolation performance after finetuning. Our method adopts position encoding along each tensor dimension and can be combined with existing methods to extrapolate along a specific dimension.

Window-based attention adopts the limited context length to model the window attention and slide the window to extrapolate. For example, LM-Infitnite (Han et al., 2023) adopt a Λ shape attention to bound token interactions to nearby tokens within the window. Such window based approach are usually combined with recurrent memory like Transformer-XL (Dai et al., 2019), RMT (Bulatov et al., 2023), (Chevalier et al., 2023) or global/external memory like (Izacard and Grave, 2020), StreamingLLM (Xiao et al., 2023), (Wu et al., 2022). Our tensorization approach generalizes window attention to different tensor dimensions which can hierarchically capture multi-hop relationships. With tensorization, out-of-window token pairs along one dimension can be in-window along another dimension.

2.3 Tensor learning

Tensor or multidimensional array has been introduced into deep learning for various applications. Tensor representation of weights is commonly used to compress neural networks For example, (Novikov et al., 2015) considers a decomposed tensor format (TT) to represent the weight matrix in MLP which can take tensorized inputs. They are also adopted to explore efficient structure in CNN (Lebedev et al., 2014; Denton et al., 2014; Kossaifi et al., 2019; Phan et al., 2020) and RNN (Yang et al., 2017; Pan et al., 2019; Su et al., 2020). In transformers, (Ren et al., 2022) and (Pan et al., 2023) use tensor decomposition to compress the transformer weights. (Alman and Song, 2023) considers modeling high-order interactions by Kronecker decomposing key vector. The work that is most similar to ours is (Ma et al., 2019) which adopts Block-Term Tensor Decomposition to model the 3-way interactions between query, key, and value. Compared to these works about tensorizing the neural network weights for compression, we adopt tensorization as a method to efficiently compress the token interactions, corresponding to the hierarchical structure embedded in the attention matrix.

3 Attention Tensorization

In this section, we first compare the representation of input sequences in both vector and tensor space. To accommodate the tensorized input, we then introduce attention tensorization by changing conventional attention from the vector space to the corresponding tensor space. Finally, we show that low-rank approximation of attention can be more efficient in tensor space compared to vector space, both empirically and theoretically.

3.1 From Sequence to Tensor Learning

In the following, we denote a scalar (order-0) with the lowercase letter a, a vector (order-1) with the lowercase bold letter $\mathbf{a}$, a matrix (order-2) with uppercase bold letter $\mathbf{A}$, and a tensor (order>2) with calligraphic bold letter $\boldsymbol{\mathcal{A}}$. The order of input sequence is determined by the sequence length with feature dimensions omitted. $\otimes$ denotes the outer product, Kronecker product, and tensor product (defined below) when operated on vectors, matrices, and tensors respectively. We use vector space to denote the space spanned by a set of base vectors and tensor space to denote the tensor direct product of several vector spaces.

Attention in Vector Space. Given input sequence with n tokens $\mathbf{x} \in \mathbb{R}^{n \times d}$, query $\mathbf{q}$, key $\mathbf{k}$ and value $\mathbf{v}$ are defined by linear projections as $\mathbf{q} = \mathbf{x}\mathbf{W}_\mathbf{q}, \mathbf{k} = \mathbf{x}\mathbf{W}_\mathbf{k}, \mathbf{v} = \mathbf{x}\mathbf{W}_\mathbf{v}$. The n-by-n sparse attention matrix is then calculated as sampled scaled dot-product:

$$\mathbf{A} = \mathrm{softmax}\left(\frac{\mathbf{q}\mathbf{k}^\intercal}{\sqrt{d}} \circ \mathbf{M}\right), \mathbf{o} = \mathbf{A}\mathbf{v} \qquad (1)$$

where softmax denotes the row-wise softmax normalization, and $\mathbf{M}$ is the attention mask. $\mathbf{o}$ is the output vector which is the updated value vector.

Consider we have tensorized inputs $\boldsymbol{\mathcal{Q}}, \boldsymbol{\mathcal{K}}, \boldsymbol{\mathcal{V}} \in \mathbb{R}^{n_1 \times \dots n_m}$ $\prod_{i=1}^{m} n_i = n$, through order-m tensorization which is achieved by reshaping $\mathbf{q}, \mathbf{k}, \mathbf{v}$ (discussed in Appendix C.1). Our following tensor treatment is based on the principle that tensors interact with each other only through the same matched dimensions. In other words, to update the i-th dimension of value tensor $\boldsymbol{\mathcal{V}}$, we consider the interactions between the i-th dimension of $\boldsymbol{\mathcal{Q}}$ and $\boldsymbol{\mathcal{K}}$

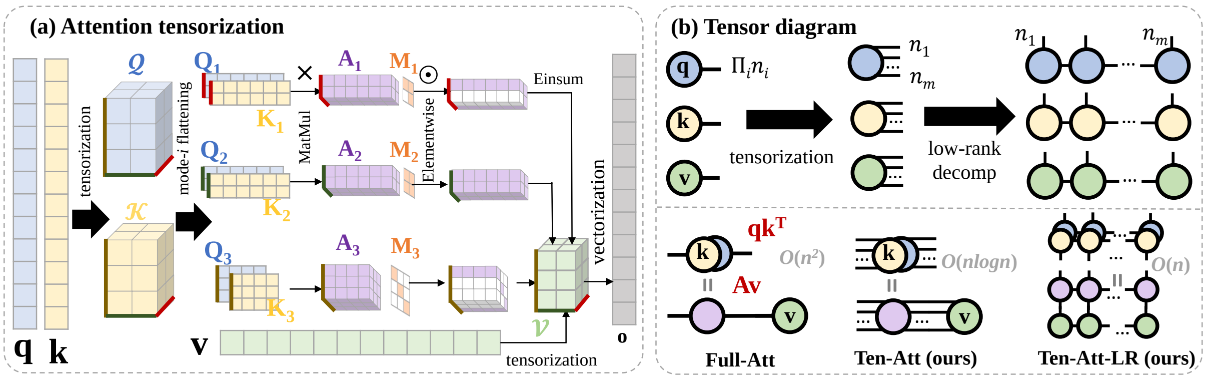

Figure 2: (a) Input sequences $\mathbf{q}, \mathbf{k}, \mathbf{v}$ are first tensorized into $\mathcal{Q}, \mathcal{K}, \mathcal{V}$. Each row in the middle represents the attention along one matching dimension of tensors, and all dimensions except the matching dimension of $\mathcal{Q}$ and $\mathcal{K}$ are flattened. The result from each row is used to sequentially update the value tensor $\mathcal{V}$. (b) Different types of attention processes can be visualized using a tensor diagram, where each circle represents data content and each edge represents a dimension.

while treating other dimensions as batch dimension. The update of $\mathcal{V}$ can then be achieved by sequential updating from the first to the last dimension (irrelevant to the order of updating).

3.2 Attention Tensorization

To model the tensor interaction between $\mathcal{Q}, \mathcal{K}, \mathcal{V}$, we propose attention tensorization which converts long-range interactions into short-range interactions along each dimension. Since the length along each dimension is much smaller than the entire sequence, such tensorized attention can work with much less context window budget. Besides, the hierarchical nature of the method enables exponential extrapolation to unseen long-sequences. Attention process in Equation 1 can be tensorized as

$$\mathcal{A} = \text{softmax}\left(\frac{\mathcal{Q} \otimes \mathcal{K}}{\sqrt{d}} \circ \mathcal{M}\right), \mathcal{O} = \mathcal{A} \times \mathcal{V}, \tag{2}$$

where $\mathcal{A}, \mathcal{M} \in \mathbb{R}^{n_1 \times n_1 \times \ldots n_m \times n_m}$, softmax is performed along every even dimension, $\circ$ is still elementwise production, and $\mathcal{O}$ is the output tensor. Following the aforementioned tensor interaction principle, $\otimes$ is the tensor (outer) product on the corresponding dimension, and can be written in Einsum notation as "$n_1..n_i..n_m, p_1..p_i..p_m \rightarrow n_1p_1..n_ip_i..n_mp_m$", or formally as $(\mathcal{Q} \otimes \mathcal{K})[.., j_{2i}, k_{2i+1}, ..] = \mathcal{Q}[.., j_i, ..]\mathcal{K}[.., k_i, ..]$ where j_i represents the j-th element along the i-th dimension. $\times$ is the tensor product between order-$2m$ and order-m tensors, written in Einsum as "$n_1p_1..n_ip_i..n_mp_m, p_1..p_i..p_m \rightarrow n_1..n_i..n_m$" or formally as $(\mathcal{A} \times \mathcal{V})[j_i] = \sum_{k_i=1}^{n_i} \mathcal{A}[.., j_{2i}, k_{2i+1}, ..]\mathcal{V}[.., k_i, ..]$. The above calculations can be easily visualized using a tensor diagram as shown in Figure 2 (b).

Sequential Updating $\mathcal{A} \times \mathcal{V}$. Because the value tensor updating can be sequential which is irrespective of the updating order, for efficient implementation, we adopt the sequential calculation of Equation 2 and model the i-th dimension of $\mathcal{Q}, \mathcal{K}, \mathcal{V}$ interaction as

$$\mathcal{A}_i = \text{softmax}\left(\frac{\mathcal{Q} \otimes_i \mathcal{K}}{\sqrt{d}} \circ \mathcal{M}_i\right), \mathcal{O} = \mathcal{A}_i \times_i \mathcal{V}, \tag{3}$$

where $\mathcal{A}_i, \mathcal{M}_i \in \mathbb{R}^{n_1 \times .. n_i \times n_i .. \times n_m}$, $\otimes_i$ and $\times_i$ are the tensor products confined to the i-th dimension with Einsum annotation as "$n_1..n_i..n_m, p_1..p_i..p_m \rightarrow n_1..n_ip_i..n_m$" and "$n_1..n_ip_i..n_m, p_1..p_i..p_m \rightarrow n_1..n_i..n_m$".

We then express the this sequential updating in the matrix multiplication format using mode-i flattening. Mode-i flattening reshapes a tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times .. \times n_m}$ into matrix $\mathbf{T}_i \in \mathbb{R}^{(n_1..n_{i-1}n_{i+1}..n_m) \times n_i}$ which can be interpreted as batching $n_1..n_{i-1}n_{i+1}..n_m$ mode-i vectors. We call the corresponding reverse process mode-i folding. Equation 3 can be equivalently rewritten as

$$\mathbf{A}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\intercal}{\sqrt{d}} \circ \mathbf{M}_i\right), \quad \mathbf{O} = \mathbf{A}_i\mathbf{V}_i, \mathcal{O} = \text{fold}_i(\mathbf{O}_i), \tag{4}$$

where fold_i represents mode-i folding, and it can also be shown that $\mathcal{A}_i = \text{fold}_i(\mathbf{A}_i)$.

We show how to use Equation 4 for sequential updates in Figure 2 (a). For the i-th dimension

Algorithm 1 Tensorized attention forward

```
Input: $\mathcal{Q}, \mathcal{K}, \mathcal{V} \in \mathbb{R}^{n_1 \times .. \times n_m}$
Initialize $\mathcal{O} = \mathcal{V}$.
for $i = 0$ to $m - 1$ do
    Mode-i flattening $\mathcal{Q}, \mathcal{K}$ into $\mathbf{Q}_i, \mathbf{K}_i$
    $\mathbf{A}_i = \text{softmax}(\mathbf{Q}_i \mathbf{K}_i^\intercal / \sqrt{d} \circ \mathbf{M}_i)$
    Mode-i flattening $\mathcal{O}$ into $\mathbf{O}_i$
    Value updates: $\mathbf{O}_i = \mathbf{A}_i \mathbf{O}_i$
    Mode-i folding $\mathbf{O}_i$ into $\mathcal{O}$
end for
Vectorize $\mathcal{O}$ to $\mathbf{o}$
Return $\mathbf{o}$
```

update, $\mathcal{Q}, \mathcal{K}$ are matricized by mode-i flattening. The resulting batched attention matrix is then used to update the i-th dimension of the value tensor. The computational complexity is decreased from $\mathcal{O}(n^2)$ to $\mathcal{O}(nlogn)$, and detailed complexity comparisons of each step are shown in Appendix C.2. We summarize the forward pass of tensorized attention in Algorithm 1. For efficient calculation, we adopt Triton (Tillet et al., 2019) to implement the forward and backward process with hardware awareness.

Tensorized Positional Encoding. To indicate the position of a token in tensor, we convert the original sequential position into the corresponding hierarchical tensor position. As shown in Figure 3, the position of a token is defined as a vector that records the relative sequential position along each dimension. Following pretrained Llama, we adopt ROPE to derive the corresponding positional encoding for sequential and tensorized position. Compared to sequential positioning, this hierarchical tensorized positioning enhances the extrapolation ability of LLMs. Extrapolating the positional encoding along a specific dimension can exponentially extend the entire sequence, while keeping positional encodings on other dimensions within range.

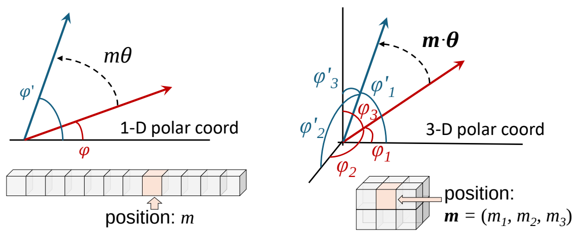

Figure 3: Comparison of token position in 1-D sequence and 3-D tensor under polar coordinates.

3.3 Efficient Approximation in Tensor Space

We focus on the properties of the attention matrix and show that the representation of attention in the tensor space can better capture the intrinsic hierarchical and low-rank structure embedded in data, compared to vector space. By comparing the low-rank approximation performance, we show that to recover the same amount of information from the attention matrix, using representation in the tensor space requires much fewer parameters than vector space.

Attention Rank in Vector Space. Given a 2-D matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with columns $[\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_n]$, the corresponding vector space is constructed as $\mathbf{a} = \sum_{j=1}^{n} \lambda_j \mathbf{c}_j$ with λ_j as a scalar. Alternatively, we take an equivalent definition of vector space as $\mathbf{A} = \sum_{j=1}^{n} \lambda_j \mathbf{c}_j \otimes \mathbf{e}_j$ where unit vector $\mathbf{e}_j$ has all 0 except 1 in j-th position. The rank r of the matrix $\mathbf{A}$ is then defined as the minimum n. Since $\{e_j\}$ are linearly independent, the basis of vector space is $\{\mathbf{c}_j \otimes \mathbf{e}_j\}_{j=1}^{r}$ or equivalently $\{\mathbf{c}_j\}_{j=1}^{r}$.

Attention Rank in Tensor Space. We then generalize the above formulations to higher-order attention tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times ... n_m}$ with order m. A rank-r tensor $\mathcal{T}$ can be decomposed into the sum of r rank-1 simple tensors using CP decomposition: $\mathcal{T} = \sum_{j=1}^{r} \lambda_j \mathbf{u}_{1,j} \otimes ... \otimes \mathbf{u}_{m,j} = \sum_{j=1}^{r} \lambda_j \otimes_{i=1}^{m} \mathbf{u}_{i,j}$, with basis $\{\otimes_{i=1}^{m} \mathbf{u}_{i,j}\}_{j=1}^{r}$. For n-by-n attention matrix $\mathbf{A}$ with corresponding tensor form $\mathcal{A} \in \mathbb{R}^{n1^2 .. \times n_m^2}$, we show that attention tensorization hierarchically encodes different levels of structural information in attention matrix into each dimension of $\mathcal{A}$. As shown in Figure 4 (a,b), vectors in the tensor basis of $\mathcal{A}$ induced by attention tensorization are formed by hierarchical Kronecker decomposition of the original attention matrix. More specifically, $\mathbf{A}$ is first decomposed to several block matrices through matrix Kronecker decomposition $\mathbf{A} = \sum_{j=1}^{r} \lambda_j \otimes_{i=1}^{m} \mathbf{U}_{i,j}$. Then $\mathbf{U}_{i,j} \in \mathbb{R}^{n_i \times n_i}$ is flattened to $\mathbf{u}_{i,j} \in \mathbb{R}^{n_i^2}$. The corresponding tensor space is then spanned by a set of (r linearly independent) base tensors $\otimes_{i=1}^{m} \mathbf{u}_{i,j}$ with the representation of $\mathcal{A}$ written as $\mathcal{A} = \sum_{j=1}^{r} \lambda_j \otimes_{i=1}^{m} \mathbf{u}_{i,j}$ with singular values λ_j.

Spectrum Analysis. We then analyze and compare the spectrum of the attention matrix in both vector and tensor space. Empirical attention statistics are obtained from pretrained RoBERTa with IMDB dataset. We consider the tensorization from 512 seuqnece to $\{32, 16\}$ and $\{8, 8, 8\}$ tensor and denote the resulting space as tensor-2 and tensor-

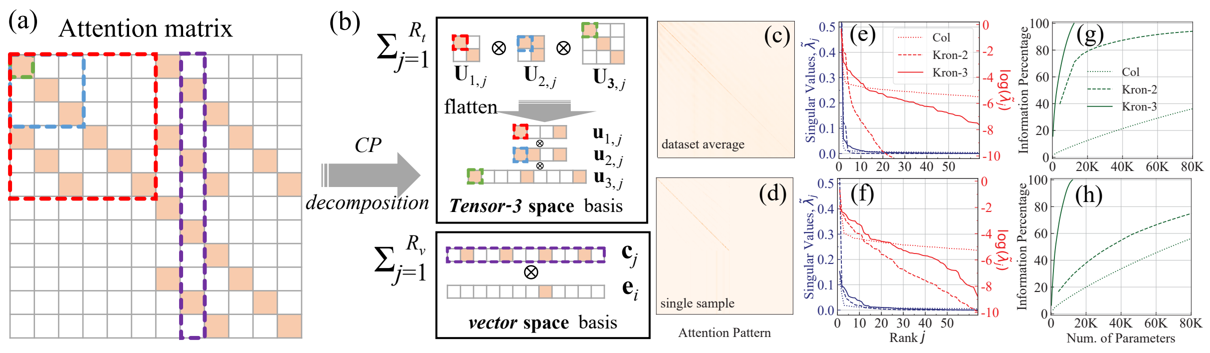

Figure 4: The attention or mask (a) can be decomposed to a set of columns or 2-D blocks which can be used to span vector or tensor space. (b) The total number of needed singular vectors to reconstruct the given example pattern $R_t = 1 < R_v = 12$, shows the advantage of using tensor space for diagonal and structured patterns. (c,d) CP decomposition of real attention patterns (layer 3, head 2) from (c) dataset average or (d) single-sample pattern. (e, f) Calculated spectrum as sorted normalized singular values $\tilde{\lambda}_i$. (g, h) Total percentage of information contained v.s. number of parameters needed for approximation.

3 space, respectively. To compare efficiency, we also calculate the number of parameters needed for approximation as the total number of singular vectors in use multiplied by the size of each singular vector $\otimes_{i=1}^{m} \mathbf{u}_{i,j}$. For example, approximating attention with a rank-3 vector in tensor-3 space needs $3 \times (8^2 + 8^2 + 8^2)$ parameters.

As shown in Figure 4 (e-h), singular values in tensor space decay faster than those in vector space, and thus require fewer singular vectors and parameters to recover the same amount of information of $\mathbf{A}$. For tensor space with different orders, $m = 2$ needs fewer singular vectors than $m = 3$ for approximation, but $m = 3$ requires much fewer parameters than $m = 2$ because the size of each singular vector is much smaller (e.g., $8^2 + 8^2 + 8^2 < 32^2 + 16^2$). We defer similar discussions of the vision model to Appendix B.1. Intuitively, if the attention matrix is more hierarchically structured with respect to base 2-D blocks, it can be more easily (with a lower rank) decomposed into a low-order tensor space. Conversely, if the matrix exhibits a more irregular and complex pattern, it may be necessary to decompose it into a higher-order tensor space. We give the following theorem to demonstrate low-rank attention properties in tensor space, with proof in Appendix B.2.

Theorem 3.1. *For attention matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and any column $\mathbf{y} \in \mathbb{R}^n$ of value vector along the feature dimension, there exists a low-rank matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{n \times n}$ with rank $\mathcal{O}(3^m log_{2m} n)$ defined in tensor-m space, such that $Pr\left(||\tilde{\mathbf{A}}\mathbf{y} - \mathbf{A}\mathbf{y}|| < \epsilon||\mathbf{A}\mathbf{y}||\right) > 1 - \mathcal{O}(1)$, for any ϵ.*

4 Experiments

In this section, we conduct various experiments to demonstrate that tensorized attention can serve as an efficient backbone for LLMs in long-sequence applications. We first discuss the continued pretraining process, followed by an evaluation of its performance on downstream tasks and its extrapolation capabilities. Additionally, we showcase the strong performance of tensorized attention in conventional NLP and time series models, as detailed in the Appendix D.

4.1 Efficient Language Modeling Backbone

We adopt three recent LLM backbones: OpenLlama-3B, Mistral-7B, and LLama-8B with context lengths 2,048, 8,192, and 8,192, respectively. Based on their pretrained checkpoints, we then adopt continued pretraining using (1) tensorized attention with tensorized positional encoding (denoted as "-Tens") and (2) full attention with position interpolation (PI) (denoted as "-Full") as baseline. We perform the pretraining on sampled RedPajama dataset (Computer, 2023) which contains 20% Arxiv 20% Book and 10% other data with data shorter than 2K words excluded. For tensorized attention, we set the tensor dimensions to be $\{32, 32, 32\}$ corresponding to the training context length 32,768, while for full attention we adopt FlashAttention-2 for efficient implementation. The continued pretraining is running for 20B tokens for both models with A6000 GPUs.

4.2 Pretraining Perplexity

We evaluate the continued pretraining performance using next-token prediction perplexity on Proof-pile test datasets (Rae et al., 2019). The perplexity and efficiency with different context lengths are shown in Figure 5. As depicted in the figure, the model perplexity benefits from increased context length from 4k to 32k. Compared to the standard full attention pretraining, tensorized attention can effectively extend the training context window to 32k while keeping low perplexity. FlashAttention-2 can make memory usage steady by fixing block size after reaching the threshold, at the cost of increasing inference time. Compared to FlashAttention, our kernel implementation for tensorized attention achieves consistently lower running times with similar memory usage, with the advantage becoming more pronounced for longer sequences. It should be noted that the memory usage for the tensorization kernel and FlashAttention are nearly identical due to the blockwise attention calculation.

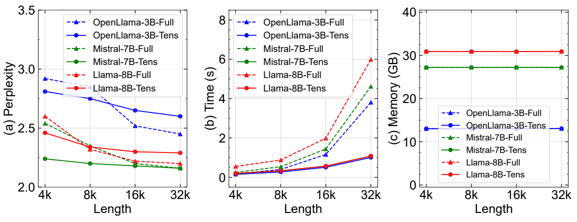

Figure 5: (a) Perplexity on Proof-pile test dataset after the continued pretraining. Comparison of GPU memory usage (b) and running time (c) efficiency for full and tensorized attention. The time and memory are calculated by averaging 50 forward passes and batch size 5.

4.3 Downstream Evaluation

We follow Llama to report the zero- and few-shot results on various tasks after the continued pretraining stage. Following the setting in Gao et al. (2023), we adopt tasks include HellaSwag (Zellers et al., 2019), SIQA (Sap et al., 2019), PIQA (Bisk et al., 2020), and WinoGrande (Sakaguchi et al., 2021) for common sense reasoning ability with zero-shot evaluations; NaturalQuestions (NQ) (Kwiatkowski et al., 2019) and TriviaQA (TQA) (Joshi et al., 2017) for world knowledge understanding with zero-shot evaluations; RACE-middle (RACE-m) and RACE-hard (RACE-h) (Lai et al., 2017) for reading comprehension capabilities with zero-shot evaluations; Massive multilingual language understanding (MMLU) (Hendrycks et al., 2020) with 5-shot evaluations. Table 1 indicates the pretrained tensorized attention outperforms both 3B, 7B and 8B benchmarks for world knowledge and reading tasks and achieves comparable results in common sense tasks. To further evaluate the model performance on tasks with longer sequences, we follow the evaluation protocol in LongBench (Bai et al., 2023) and show the task performance in Table 2. Tensorized attention consistently outperforms full attention in OpenLlama-3B with original context window 2,048 and achieves comparable results on Mistral-7B and Llama-8B with original context window 8,192, which shows the continued pertaining effectively extends the original context window using tensorized inputs. Besides, tensorization enjoys less running time compared to full attention benchmarks (implemented with FlashAttention-2). For example, tensorized attention achieves $0.61\times$ running time average across all tasks compared to full attention.

4.4 Length Extrapolation

LLM extrapolation techniques extend the model's comprehension to sequences beyond its initially observed length within the training context window. Besides position- and window-based extrapolation, tensorization provides a new perspective to extend the context length by extrapolating the tensor on each dimension instead of the original sequence. By increasing the context length by p tokens on the i-th order, the total effective context length will increase by $p \cdot \Pi_{j \neq i} n_j$, which amplifies small increment on one dimension and effectively extends the total sequence length. For example, for 32,768 tokens with $\{32, 32, 32\}$ tensorization, 1 token increment on the first dimension is equivalent to 1,024 token length expansion in total. We extrapolate from the last (lower) to the first (higher) dimension for hierarchically encoding unseen positions.

We show the extrapolation performance and efficiency of tensorization using the perplexity of the Proof-pile test dataset in Table 3 with spectrum in Figure 6. For comparison, we compare three widely used extrapolation methods: positional interpolation (PI) (Chen et al., 2023a), LM-Infinite (Inf) (Han et al., 2023) with attention distance 1,024, and YARN (Peng et al., 2023) with scale factor 32. While the original full attention model with positional encoding fails when surpassing the training context length, Inf, YARN, and Tens maintain performance from 32k to 128k, with Tens achieving consistently lower perplexity as sequence length increases, especially in the Llama-8B model. Additionally, the time improvement from attention ten-

Model	Common Sense				World Knowledge		Reading		MMLU
	HellaSwag	SIQA	WinoGrande	PIQA	NQ	TQA	RACE-m	RACE-h	
Llama-7B	76.5	49.0	69.2	79.0	12.5	56.6	41.2	39.5	46.1
Falcon-7B	76.3	49.1	67.1	80.5	14.6	50.9	42.3	37.2	30.5
Gemma-7B	80.3	50.7	72.0	80.9	15.2	57.5	43.6	38.3	63.8
StableLM-7B	71.7	44.1	69.1	79.8	9.1	46.4	41.2	38.9	45.6
OpenLlama-3B-Full	65.2	44.8	**63.5**	76.2	6.3	31.5	40.6	36.8	27.1
OpenLlama-3B-Tens	**69.4**	**45.3**	63.2	**78.0**	**7.2**	**32.2**	**42.3**	**37.7**	**27.4**
Mistral-7B-Full	79.5	47.0	**73.5**	81.1	15.5	58.2	45.2	39.8	62.5
Mistral-7B-Tens	**80.5**	**49.1**	73.2	**81.3**	**15.8**	**58.8**	**45.8**	**41.3**	**63.3**
Llama-8B-Full	**81.4**	49.5	73.8	**81.5**	16.0	57.8	44.8	40.3	65.4
Llama-8B-Tens	81.0	**50.3**	**74.6**	81.0	**16.3**	**58.2**	**45.2**	**42.6**	**66.2**

Table 1: Downstream tasks from different domains including common sense, world knowledge, and reading comprehension. All benchmark models are running with FlashAttention-2.

Model	Multi-DocQA		Summarization		Few-shot			Time
	HotpotQA	2Wiki	GovReport	QMSum	TREC	TriviaQA	SAMSum	
AvgLength	9,151	4,887	8,734	10,614	5,177	8,209	6,258	
OpenLlama-3B-Full	18.4	20.1	12.4	15.6	52.3	72.7	33.8	1×
OpenLlama-3B-Tens	**19.5**	**21.5**	**13.5**	**16.2**	**53.0**	**74.8**	**38.2**	0.68×
Mistral-7B-Full	34.9	29.9	24.2	22.2	**68.2**	**87.7**	41.3	1×
Mistral-7B-Tens	**35.5**	28.4	**24.8**	**23.5**	68.0	87.5	**44.3**	0.61×
Llama-8B-Full	**45.3**	**34.5**	30.7	**25.8**	71.1	86.1	42.5	1×
Llama-8B-Tens	45.2	33.8	**32.5**	24.9	**72.5**	**88.2**	**44.0**	0.73×

Table 2: Performance comparison of full and tensorized attention on LongBench.

sorization becomes more pronounced with increasing context length. Notably, tensorized attention reduces the running time by 0.09× at 128k length compared to full attention for Llama-3 model.

Length		16k	32k	64k	96k	128k
OpenLlama-3B	PI	2.52	2.45	-	-	-
	InfLM	2.43	2.50	2.58	2.55	2.60
	YARN	2.52	2.48	2.65	2.71	2.68
	Tens	2.65	2.60	2.50	2.53	2.57
	Time	0.44×	0.27×	0.14×	0.10×	0.08×
Mistral-7B	PI	2.20	2.16	-	-	-
	InfLM	2.43	2.40	2.41	2.42	2.41
	YARN	2.20	2.16	2.17	2.16	2.16
	Tens	2.18	2.16	2.15	2.15	2.16
	Time	0.40×	0.23×	0.13×	0.09×	0.07×
Llama-8B	PI	2.22	2.20	-	-	-
	InfLM	2.45	2.34	2.36	2.41	2.39
	YARN	2.22	2.20	2.27	2.29	2.28
	Tens	2.30	2.29	2.21	2.19	2.16
	Time	0.72×	0.25×	0.15×	0.11×	0.09×

Table 3: Extrapolation perplexity and efficiency with sequence length ranging from 16k to 128k. "-" denotes perplexity $>$ 10. Time usage is compared between tensorized and full attention.

5 Conclusion

In this paper, we explore attention tensorization by replacing vector/matrix operations with corresponding tensor operations. From an efficiency perspective, decomposing long sequences into individual dimensions allows the limited context length

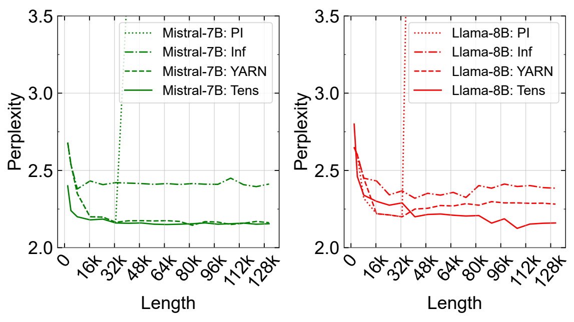

Figure 6: The perplexity spectrum up to length 128k with different extrapolation methods.

budget to be utilized more effectively for attention calculation along each dimension. From a performance perspective, attention tensorization naturally encodes multi-hop hierarchical token interactions, making attention more efficiently approximated compared to its representation in vector space. Extensive results on datasets from various domains demonstrate that tensorization can extend the training context length, enhancing both efficiency and length extrapolation performance.

6 Limitations

Due to limited computational resources and time, the proposed method has not been evaluated for LLM training from scratch settings. This work evaluates a wide range of open-domain LLMs but cannot be applied to proprietary APIs like GPT-4. The proposed method has not been evaluated with even longer sequences due to computational constraints. Attention tensorization is parallel to other sequence extrapolation methods, and further exploration is needed to combine different methods with tensorization to achieve better performance in future work.

References

Josh Alman and Zhao Song. 2023. How to capture higher-order correlations? generalizing matrix softmax attention to kronecker computation. *arXiv preprint arXiv:2310.04064*.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Iz Beltagy, Matthew E Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Aydar Bulatov, Yuri Kuratov, and Mikhail S Burtsev. 2023. Scaling transformer to 1m tokens and beyond with rmt. *arXiv preprint arXiv:2304.11062*.

Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. 2023a. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*.

Xuanyao Chen, Zhijian Liu, Haotian Tang, Li Yi, Hang Zhao, and Song Han. 2023b. Sparsevit: Revisiting activation sparsity for efficient high-resolution vision transformer. *arXiv preprint arXiv:2303.17605*.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. *arXiv preprint arXiv:2305.14788*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Together Computer. 2023. Redpajama: an open dataset for training large language models.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in neural information processing systems*, 27.

Aosong Feng, Irene Li, Yuang Jiang, and Rex Ying. 2022. Diffuser: Efficient transformers with multi-hop attention diffusion for long sequences. *arXiv preprint arXiv:2210.11794*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. 2023. A framework for few-shot language model evaluation.

Chi Han, Qifan Wang, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2023. Lm-infinite: Simple on-the-fly length generalization for large language models. *arXiv preprint arXiv:2308.16137*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

William B Johnson. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Isak Karlsson, Panagiotis Papapetrou, and Henrik Boström. 2016. Generalized random shapelet forests. *Data mining and knowledge discovery*, 30:1053–1085.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Jean Kossaifi, Adrian Bulat, Georgios Tzimiropoulos, and Maja Pantic. 2019. T-net: Parametrizing fully convolutional nets with a single high-order tensor. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7822–7831.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. 2014. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Xindian Ma, Peng Zhang, Shuai Zhang, Nan Duan, Yuexian Hou, Ming Zhou, and Dawei Song. 2019. A tensorized transformer for language modeling. *Advances in neural information processing systems*, 32.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2022. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. 2015. Tensorizing neural networks. *Advances in neural information processing systems*, 28.

Yu Pan, Jing Xu, Maolin Wang, Jinmian Ye, Fei Wang, Kun Bai, and Zenglin Xu. 2019. Compressing recurrent neural networks with tensor ring for action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4683–4690.

Yu Pan, Ye Yuan, Yichun Yin, Zenglin Xu, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Reusing pretrained models by multi-linear operators for efficient training. *arXiv preprint arXiv:2310.10699*.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. 2023. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*.

Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. 2020. Stable low-rank tensor decomposition for compression of convolutional neural network. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 522–539. Springer.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*.

Beheshteh Rakhshan and Guillaume Rabusseau. 2020. Tensorized random projections. In *International Conference on Artificial Intelligence and Statistics*, pages 3306–3316. PMLR.

Yuxin Ren, Benyou Wang, Lifeng Shang, Xin Jiang, and Qun Liu. 2022. Exploring extreme parameter compression for pre-trained language models. *arXiv preprint arXiv:2205.10036*.

Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. 2021. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. 2019. Socialiqa: Commonsense reasoning about social interactions. *arXiv preprint arXiv:1904.09728*.

Jiahao Su, Wonmin Byeon, Jean Kossaifi, Furong Huang, Jan Kautz, and Anima Anandkumar. 2020. Convolutional tensor-train lstm for spatio-temporal learning. *Advances in Neural Information Processing Systems*, 33:13714–13726.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. 2021. Synthesizer: Rethinking self-attention for transformer models. In *International conference on machine learning*, pages 10183–10192. PMLR.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28.

Philippe Tillet, Hsiang-Tsung Kung, and David Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. *arXiv preprint arXiv:2306.07174*.

Genta Indra Winata, Samuel Cahyawijaya, Zhaojiang Lin, Zihan Liu, and Pascale Fung. 2020. Lightweight and efficient end-to-end speech recognition using low-rank transformer. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6144–6148. IEEE.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.

Zhuofan Xia, Xuran Pan, Shiji Song, Li Erran Li, and Gao Huang. 2022. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4794–4803.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2023. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*.

Yinchong Yang, Denis Krompass, and Volker Tresp. 2017. Tensor-train recurrent neural networks for video classification. In *International Conference on Machine Learning*, pages 3891–3900. PMLR.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128.

Dawei Zhu, Nan Yang, Liang Wang, Yifan Song, Wenhao Wu, Furu Wei, and Sujian Li. 2023a. Pose: Efficient context window extension of llms via positional skip-wise training. *arXiv preprint arXiv:2309.10400*.

Lei Zhu, Xinjiang Wang, Zhanghan Ke, Wayne Zhang, and Rynson Lau. 2023b. Biformer: Vision transformer with bi-level routing attention. *arXiv preprint arXiv:2303.08810*.

A Attention Patterns in Transformers

In this section, we show the hierarchical and low-rank attention structure using empirical results. We show the attention patterns of the IMDB validation set using a trained RoBERTa model. As can be seen, shallow layers show more diagonal patterns while deeper layers have more block and column patterns (Figure 7). Different heads in the same layer have different patterns that attend to different aspects of data (Figure 8). Different samples will induce slightly different patterns to attend to sample-specific details (Figure 9).

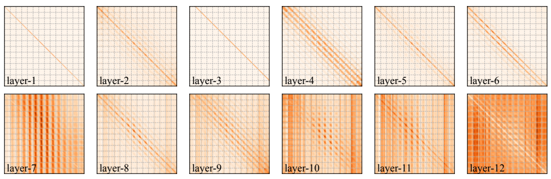

Figure 7: Averaged attention patterns at different layers with the 6th head.

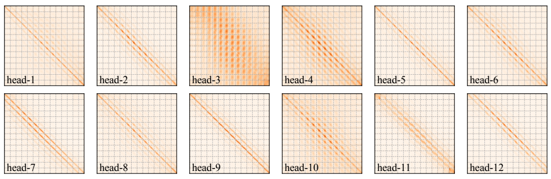

Figure 8: Averaged attention patterns at different heads at the 5th layer.

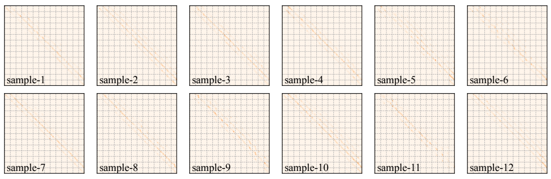

Figure 9: Attention patterns of different samples at the 3rd layer and 0th head.

B More Spectrum Analysis

B.1 Spectrum on ViT attention

We perform the low-rank approximation analysis on ViT-Base model with Imagenet-1K dataset, similar to the image model analysis in Section 3.3. As shown in Figure 10, compared to vector space, we can obtain approximations with lower rank and fewer parameters in the proposed tensor space.

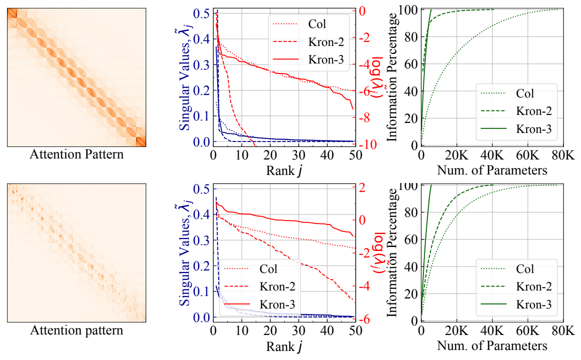

Figure 10: Singular value spectrum of averaged and single attention patterns with NLP dataset.

B.2 Theorem 3.1

The theorem shows that the required rank can decay exponentially with $m \geq 1$, and as we increase the Kronecker decomposition order m, the exponential decay will be faster. Since in practice, m is usually very small (usually 2 - 5) while n can be over thousands, the main factor in the rank complexity is from term $\log_{2m} n$, indicating the exponential contribution of m.

Proof. We follow (Wang et al., 2020) to prove the theorem using the distributional Johnson-Lindenstrauss (JL) lemma (Johnson, 1984). Defining the low-rank approximation matrix $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{T}^\intercal\mathbf{T}$, where $\mathbf{T} \in \mathbb{R}^{k\times n}$ and $k < n$. Then we get $\text{rank}(\tilde{\mathbf{A}}) \leq \text{rank}(\mathbf{T}) \leq k$.

Assuming tensor-m space is constructed from block sizes $\{n_i\}_{i=1}^m$ with $\prod_{i=1}^m n_i = n$. Then for i-th row vector $\mathbf{t}^{(i)} \in \mathbb{R}^n$ of $\mathbf{T}$ and $\mathbf{y}$, the corresponding representations in tensor-m space is $\boldsymbol{\mathcal{T}}^{(i)} \in \mathbb{R}^{n_1\times\ldots n_m}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{n_1\times\ldots n_m}$. We additionally assume $\boldsymbol{\mathcal{T}}^{(i)}$ has rank-r: $\boldsymbol{\mathcal{T}}^{(i)} = \sum_{j=1}^r \otimes_{h=1}^m \mathbf{t}_{h,j}^{(i)}$.

Next, we consider the transformation induced by $\mathbf{T}$ in the row-by-row form as

$$\mathbf{T}\mathbf{y} = \begin{bmatrix} \left\langle \mathbf{t}^{(1)}, \mathbf{y} \right\rangle \\ \ldots \\ \left\langle \mathbf{t}^{(k)}, \mathbf{y} \right\rangle \end{bmatrix}, \tag{5}$$

where each row can be calculated using mixed Kronecker-matrix(vector) product property as

$$\left\langle \mathbf{t}^{(i)}, \mathbf{y} \right\rangle = \left\langle \boldsymbol{\mathcal{T}}^{(i)}, \boldsymbol{\mathcal{Y}} \right\rangle \tag{6}$$

with the general inner product defined as $\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle = \sum_{i_1,\ldots,i_m} \boldsymbol{\mathcal{A}}_{i_1,\ldots,i_m} \boldsymbol{\mathcal{B}}_{i_1,\ldots,i_m}$.

JL properties of this transformation $\mathbf{T}\mathbf{y}$ can be then characterized by the following lemma:

Lemma B.1. *(Theorem 2 in (Rakhshan and Rabusseau, 2020)) Let* $\mathbf{T}$ *be* $k \times n$ *matrix with* *i.i.d entries from* $N(0, r^{-\frac{1}{m}})$*, for any* $\epsilon > 0, \delta > 0$*, if* $k > \epsilon^{-2}3^{m-1}(1 + 2/R)log_{2m}(N/\delta)$*, we have*

$$Pr(|\|\mathbf{Ty}\| - \|\mathbf{y}\|| > \epsilon\|\mathbf{y}\|) < \delta \qquad (7)$$

$$Pr(|\|\mathbf{x}^\intercal\mathbf{T}^\intercal\mathbf{Ty}\| - \|\mathbf{x}^\intercal\mathbf{y}\|| > \epsilon\|\mathbf{x}^\intercal\mathbf{y}\|) < 2\delta. \quad (8)$$

Therefore, we have

$$\begin{aligned} &Pr\left(\|\tilde{\mathbf{A}}\mathbf{y} - \mathbf{A}\mathbf{y}\| < \epsilon\|\mathbf{A}\mathbf{y}\|\right) \\ &= Pr\left(\|\mathbf{A}\mathbf{T}^\intercal\mathbf{T}\mathbf{y} - \mathbf{A}\mathbf{y}\| < \epsilon\|\mathbf{A}\mathbf{y}\|\right) \\ &\geq 1 - \\ &\sum_{i=1}^{n} Pr\left(|\|\mathbf{a}^{(i)\intercal}\mathbf{T}^\intercal\mathbf{T}\mathbf{y}\| - \|\mathbf{a}^{(i)\intercal}\mathbf{y}\|| > \epsilon\|\mathbf{a}^{(i)\intercal}\mathbf{y}\|\right) \\ &\geq 1 - 2n\delta. \end{aligned} \qquad (9)$$

Let $\delta = \frac{1}{n}$ and hence $\text{rank}(\tilde{\mathbf{A}}) = k = \mathcal{O}(3^m \log_{2m} n)$, the theorem follows.

C More Discussions of Attention Tensorization

C.1 Tensorization with Reshaping

As mentioned, to map an n-by-n attention matrix $\mathbf{A}$ into its representation $\boldsymbol{\mathcal{A}}$ in tensor space through tensorization, we simply need to reshape the attention matrix. We show a tensorization example in Figure 11 to better understand the construction of tensor space. The yellow row dimension in $\boldsymbol{\mathcal{A}}$ is from reshaping the yellow square block of $\mathbf{A}$. It can be seen that row, column, and height dimensions in $\boldsymbol{\mathcal{A}}$ represent low-, middle-, and high-level attention in $\mathbf{A}$ respectively.

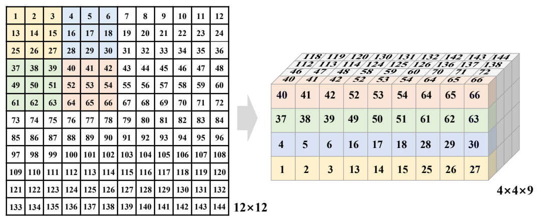

Figure 11: Tensorization from a 12×12 matrix to $4 \times 4 \times 9$ tensor in tensor space (constructed from order-3 Kronecker decomposition of the matrix).

C.2 Time Complexity

We compare the theoretical time complexity of the proposed tensorization mechanism at the three attention steps with other popular attention approximation methods as in Table 4.

Operations	$\mathbf{qk}^\intercal$	softmax	$\mathbf{Av}$
Full	$\mathcal{O}(n^2d)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2d)$
Sparse	$\mathcal{O}(pn^2d)$	$\mathcal{O}(pn^2)$	$\mathcal{O}(pn^2d)$
Linformer	$\mathcal{O}(nkd)$	$\mathcal{O}(nk)$	$\mathcal{O}(nkd)$
Performer	$\mathcal{O}(nkd)$	$\mathcal{O}(nk)$	$\mathcal{O}(nkd)$
Reformer	$\mathcal{O}(n_r(4n/n_c)^2d)$	$\mathcal{O}(n_r(4n/n_c)^2)$	$\mathcal{O}(n_rn(4n/n_c)^2d)$
Nystromformer	$\mathcal{O}(nkd + k^2d)$	$\mathcal{O}(nk + k^2)$	$\mathcal{O}(nk^2 + 2nkd + k^3)$
Tensorization	$\mathcal{O}(\sum_{i=1}^m p_in_i^2d)$	$\mathcal{O}(\sum_{i=1}^m p_in_i^2)$	$\mathcal{O}(pn\sum_{i=1}^m n_id)$

Table 4: Time complexity of different efficient attention mechanisms, with length dimension n, k, feature dimension d, sparsity p.

D Experiments

We present additional experimental results of tensorized attention across various datasets, including both fine-tuning and training from scratch settings.

D.1 Language Modeling with RoBERTa

Given pretrained full-attention model, tensorized attention can be directly adopted as an efficient method for finetuning, with improved training and inference time. We evaluate the finetuning performance of tensorized attention ith RoBERTa backbones on GLUE (Wang et al., 2018) benchmark. We follow the finetuning setting of RoBERTa on GLUE development as (Liu et al., 2019), and initialize the model with pretrianed RoBERTa on every task (without using MNLI results to initialize MRPC, RTE, and STS-B). Full parameter and sparse fituning with different attention masks (90% masked) are used as benchmarks. We use "Tens-n" to denote the n-th order tensorization. Median results for each task over five random seeds are reported in Table 5. The optimal tensorization order is different among tasks and determined by the input data type. On average, RoBERTa model adapted with tensorized attention shows on-par and better results compared to full-attention baselines while only a small fraction of the attention calculations is needed.

Dataset		MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
Full		**87.6**	$\underline{91.6}$	92.0	$\underline{94.6}$	$\underline{63.8}$	$\underline{91.2}$	**90.0**	80.2	$\underline{86.38}$
	Window	82.4	88.2	87.6	92.5	58.6	86.2	83.6	76.5	81.95
Sparse	Rand	79.8	85.8	84.9	91.5	54.3	83.3	82.8	73.4	79.48
	Top-k	80.3	88.0	88.2	92.4	59.8	86.5	86.2	79.2	82.58
Tens-1		82.5	89.5	89.5	93.1	59.5	87.8	86.0	78.7	83.33
Tens-2		$\underline{87.2}$	91.1	**92.4**	**94.8**	**64.1**	$\underline{91.2}$	$\underline{89.5}$	**86.4**	**87.09**
Tens-3		86.8	**91.8**	$\underline{92.2}$	94.5	62.6	90.9	86.2	$\underline{84.5}$	86.19

Table 5: Comparison of fine-tuning RoBERTa model using full, sparse, and tensorization on the GLUE development set (%). Underline values are second best, and bold values are the best.

D.2 Long Time Series Modeling

We evaluate the performance of attention tensorization in the time series domain under training from

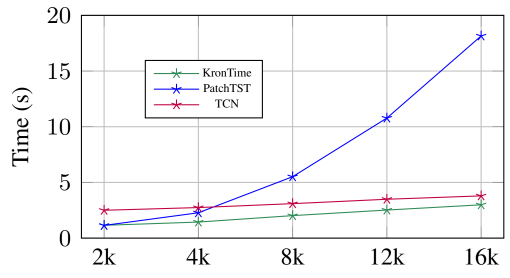

Figure 12: Comparison of running timewith different input lengths.

scratch setting. We choose SOTA time series transformers PatchTST (Nie et al., 2022) as the backbone, and replace the full attention in the attention layer with tensorization. The resulting Tens time series model is compared with SOTA baselines 1-NN (ED)(Ruiz et al., 2021), ResNet18(Ismail Fawaz et al., 2019), RandomShapelet(Karlsson et al., 2016), DLinear (Zeng et al., 2023), Temporal Convolutional Network (Bai et al., 2018), and PatchTST.

From the UEA Time Series Classification Archive, we choose the following fine-resolution time series datasets BinaryHeartbeat, EigenWorms, FaultDetectionA, CatsDogs, each has lengths greater than 10,000. For each dataset, we uses 80% of data as training data, 10% of data as validation data, and the rest 10% of data as testing data. The final test accuracy is obtained using the checkpoint of the lowest validation loss, with the early-stop patience epochs 20.

Model	BinaryHeartbeat	EigenWorms	FaultDetectionA	CatsDogs
1-NN (ED)	0.585	0.500	0.460	0.420
ResNet18	0.630	0.420	0.990	0.570
RandomShapelet	0.585	0.692	N/A	0.606
DLinear	0.658	0.423	0.532	0.516
TCN	**0.707**	**0.769**	0.986	0.575
PatchTST	**0.707**	0.692	0.991	0.810
Tens	**0.707**	**0.769**	**0.996**	**0.844**

Table 6: Time series classification performance comparison.

The classification result is shown in Table 6 with efficiency shown in Figure 12. Tensorization achieves the same or superior classification accuracy compared to SOTA models. The results demonstrate that tensorization achieves approximately $0.3\times$ running time compared to PatchTST at a length of 16k, and is comparable to the conventional convolution-based TCN. This advantage of improved running time from attention decomposition becomes more significant as the input length increases.

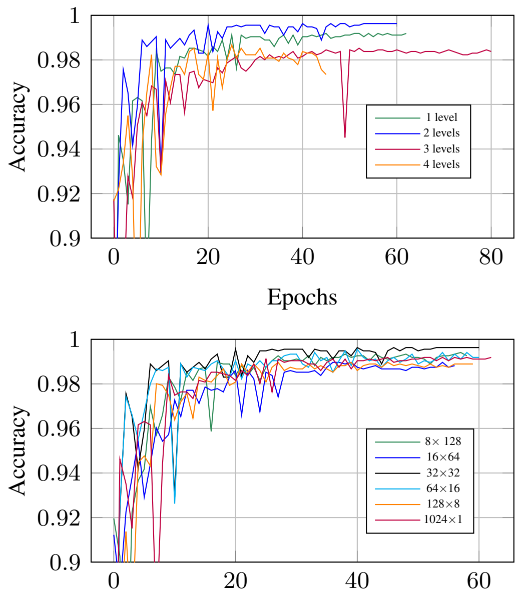

Figure 13: The validation accuracy with different tensorization strategies (upper: number of levels decomposed; lower: different tensorization with 2 levels) during the training phase.

We next demonstrate the influence of the tensorization strategies by comparing the training curves. We perform such ablation studies on FaultDetectionA dataset with 1024 input length after tokenized. We change the total number of orders (levels) of tensorization and the size of each level while keeping other model and training hyperparameters unchanged for fair comparisons. As shown in Figure 13(a), training with 2-level tensorization (16×16) converges to the higher validation accuracy with faster speed, compared to no tensorization (1-level tensorization), 3-level tensorization ($16 \times 16 \times 4$), and 4-level tensorization ($16 \times 4 \times 4 \times 4$). We then compare different tensorization strategies with 2 levels, and results in Figure 13(b) show that tensorizing the 1024 sequence into 32×32 achieves superior results compared to other tensorizations scheme for this dataset.