Enhancing Neural Language Models: A Comprehensive Approach with Tensorized Transformer and Over-Parameterization

PRATHAM TANEJA[1], KESHAV CHANDRA [2], DAAMINI BATRA [3], AKSHITA GUPTA[4], RAHUL KUMAR[5], and BHAUMIK TYAGI[6]

[1]*Graduate Student, ADGITM, Delhi, India*
[2,3,4] *Undergraduate Student, (Information Technology), ADGITM, Delhi, India*
[5]*Graduate Student, The Heritage College, Kolkata, India*
[6]*Jr. Research Scientist, Delhi, India*

Abstract—This research paper introduces novel strategies to enhance the performance and efficiency of neural language models, addressing challenges in resource-limited settings and scalability. This research presents multi-linear attention with Block-Term Tensor Decomposition (BTD), a self-attention model leveraging tensor decomposition and parameters sharing. This approach achieves significant parameter compression while demonstrating improved performance on language modeling tasks. Comparative evaluations against traditional Transformer models underscore the effectiveness of multi-linear attention. TensorCoder employs a dimension-wise attention mechanism to address the quadratic complexity of the scaled dot-product attention in Transformers, making it suitable for long sequence tasks. The proposed approach is validated on masked language modeling and neural machine translation tasks, showcasing a substantial reduction in computational complexity while maintaining or surpassing performance compared to the original Transformer. This research also optimizes pre-trained language models (PLMs) through fine-tuning. To overcome computational challenges associated with large PLMs, the paper introduces a matrix product operator for over-parameterization during fine-tuning. Efficient decomposition methods factorize parameter matrices into higher-dimensional tensors, enabling the selection of important parameter matrices through static and dynamic strategies. Extensive experiments demonstrate that this approach significantly enhances the fine-tuning performance of small PLMs, enabling them to outperform larger counterparts with three times the parameters. This research opens avenues for efficiently scaling language models without compromising inference latency, showcasing the potential of over-parameterization in enhancing the applicability of large PLMs in real-world systems.

Keywords—Tensorized Transformer, Over-Parameterization, Language Model Optimization, Tensor Decomposition, Multi-linear Attention, Parameter Compression, Inference Latency

I. INTRODUCTION

In the realm of Natural Language Processing (NLP), leveraging neural language model pre-training has proven highly effective across various tasks. One prominent architecture, the Transformer, relies exclusively on attention mechanisms, diverging from traditional recurrent and convolutional networks. This approach has garnered considerable attention and serves as a cornerstone in numerous neural language models like BERT, GPT, and Universal Transformer. Nonetheless, the Transformer's abundance of parameters presents challenges for training and deploying in resource-constrained environments, necessitating the compression of these large pre-trained models. Within the Transformer architecture, the multi-head attention mechanism stands as a pivotal component, characterized by a substantial parameter count. To address this, Tucker decomposition is proposed, initializing a low-rank core tensor, to reconstruct a more compact attention representation. Here, the matrices Q, K, and V can be viewed as factor matrices. The method for constructing the multi-head mechanism and compressing the model involves Block-Term Tensor Decomposition (BTD), a fusion of CP decomposition and Tucker decomposition. Notably, in BTD, the three-factor matrices Q, K, and V are shared when constructing each 3-order block tensor, resulting in significant parameter reduction while maintaining model performance.

This paper introduces several significant contributions: 1) The investigation establishes that the output of scaled dot-product attention, when viewed as a function, can be accurately represented through a set of orthonormal base vectors, thus elucidating a fundamental aspect of attention mechanisms. 2) A novel self-attention technique, termed multi-linear attention, is introduced. This method amalgamates two innovative compression strategies, namely parameter sharing and low-rank approximation. By leveraging these techniques, multi-linear attention offers a more efficient representation while maintaining performance. 3) Multi-linear attention establishes a robust relationship between three-factor matrices, encapsulating queries, keys, and values, respectively. This interconnection enhances the model's capacity to capture comprehensive attention information, thereby

improving overall performance. Additionally, it is demonstrated that our model successfully reconstructs scaled dot-product attention within the original Transformer architecture.

To assess the efficacy of our model, we conduct experiments on two prominent NLP tasks: language modeling and neural machine translation. In these experiments, we substitute the traditional multi-head attention mechanism with our proposed model, termed multi-linear attention. Our findings indicate that the standard multi-head attention can be significantly compressed when utilizing the multi-linear attention approach, particularly evident when testing on the One-Billion dataset. Consequently, our results demonstrate that multi-linear attention not only achieves a substantial reduction in parameter count but also delivers promising performance across various experiments, particularly in language modeling tasks.

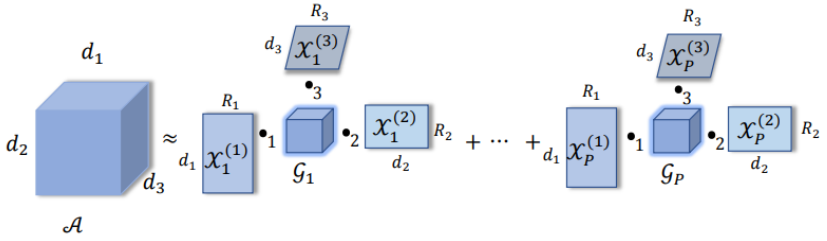


Fig. 1 The representation of Block-Term tensor decomposition for a 3-order tensor.

The Block-Term tensor decomposition offers a representation for a three-order tensor, denoted as A ∈ R^d1×d2×d3. This decomposition approximates the tensor A using a Tucker decomposition framework. Here, P represents the CP rank, which signifies the number of components in the CP decomposition, while R1, R2, and R3 denote the Tucker ranks among the three dimensions of the tensor. In this context, we assume that R = R1 = R2 = R3, implying that the Tucker ranks are equal across all dimensions. This assumption simplifies the decomposition process and facilitates a more uniform representation of the tensor. By employing this assumption, we aim to achieve a more cohesive and manageable decomposition, enabling a clearer understanding of the underlying structure and characteristics of the tensor data.

II. RELATED WORK

In recent years, the field of language modeling has seen significant advancements, particularly with the emergence of Transformer-based architectures and their various adaptations. Unlike traditional approaches such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), Transformers, and their derivatives have demonstrated remarkable performance in language processing tasks. One notable strength of Transformer networks is their ability to capture long-range dependencies, although they face challenges with fixed-length context in language modeling scenarios [1]. Vaswani et al. introduced innovations to address this limitation by incorporating segment-level recurrence mechanisms and novel positional encoding schemes [2]. BERT, another significant development, focuses on bidirectional encoder representations from transformers, achieving state-of-the-art results across multiple NLP tasks [3]. However, the proliferation of parameters in these models presents challenges for training in resource-constrained environments. Despite their successes, Transformers have struggled to generalize effectively in certain tasks like string copying and logical inference [4]. To mitigate this issue, Universal Transformers proposes a self-attentive recurrent sequence model that enhances training speed and balances model expressivity through weight sharing inspired by CNNs and RNNs [5]. Nonetheless, this approach also entails a substantial parameter overhead.

Hence, it's crucial to address the challenge of reducing memory usage and computational demands in neural network models. Model compression methods typically fall into categories such as parameter pruning and sharing [10], low-rank approximation [11], knowledge transfer [12], and transferred convolutional filters [13]. Tensor decomposition methods play a vital role in decomposing high-order tensors, enabling the derivation of diverse neural network language model structures [14]. Particularly, leveraging low-rank approximation within tensor decomposition has proven effective for compressing neural networks. For instance, researchers [15] have successfully applied tensor decomposition techniques to minimize the reconstruction error of original parameters, notably in convolutional neural networks (CNNs). However, sequential compression of multiple layers in these approaches often leads to error accumulation and significant deviations in output feature maps from their original values. To mitigate this, our compression method integrates parameter sharing during the

construction of attention layers, ensuring that the output size remains consistent with that of the self-attention mechanism in the Transformer model, thereby addressing these issues. Furthermore, Tensorizing Neural Networks [16] involves reshaping weights of fully connected layers into high-dimensional tensors, represented in Tensor Train format [17]. This methodology has been extended to convolutional [18] and recurrent neural networks [19], providing a comprehensive compression solution. Recent advancements in compression techniques include efficient methods for compressing embedding and softmax layers based on structured low-rank matrix approximation [20]. While TT-embedding [21] focuses on compressing larger embedding layers in Transformer-XL [9], Sparse Transformer adopts sparse techniques to reduce attention matrix parameters. Our method differs from these approaches by combining low-rank approximation and parameter sharing to construct a tensorized Transformer, offering a novel approach to model compression in NLP tasks [22].

Pre-trained language models (PLMs) have emerged as leading solutions in natural language processing (NLP), showcasing state-of-the-art performance across various tasks [27]. Notably, BERT, built upon the Transformer architecture, introduced the "pre-training + fine-tuning" paradigm, significantly enhancing NLP benchmarks such as GLUE [28] [29] [30]. Subsequent models like T5 and RoBERTa further augmented performance by leveraging increased data, parameters, and pre-training steps [31] [32]. Furthermore, advancements in model scaling, as demonstrated by GPT-3, have shown substantial improvements in few-shot performance [33]. In our approach, we enhance PLM performance solely through model scaling during fine-tuning, without incurring additional inference latency. This strategy offers a streamlined approach to improving PLM performance while maintaining efficiency. Recent studies have demonstrated the benefits of over-parameterization in various aspects of model training. Over-parameterization has been found to contribute to better model initialization, leading to improved model convergence and generalization [34] [35] [36]. Following the introduction of the lottery theory hypothesis, which suggests that over-parameterization can enhance training efficiency [37], numerous works have highlighted its potential to improve model performance [38] [39].

III. RESEARCH METHODOLOGY

Block-Term Tensor Decomposition (BTD) is a amalgamation of *CP decomposition* [7] & *Tucker decomposition* [8]. Given a n-order tensor $\mathcal{A} \in \mathbb{R}^{d1\times\ldots..dn}$. A high-order tensor can be decomposed into P block standings by the technique named 'BTD'.

- z is denoted as the tenor-tensor product on the z-th order [19] & $z \in \{1,\ldots,d\}$.
- z between a core tensor $G_i \in \mathbb{R}^{R1\ldots Rd}$ & d; factor matrices $X_i^{(k)} \in \mathbb{R}^{dk\times Rk}$, where $i \in [1,P]$ & $k \in [1,d]$. The devising of *BTD decomposition* is as follows:

$$\mathcal{A} = \sum_{i=1}^{P} G_i \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where P is the CP rank, & d is the Core-order. In this research work, the tensor is 3-order. Figure 1 validates the instance of how a 3-order tensor $\mathcal{A}$ can be disintegrated into P block terms.

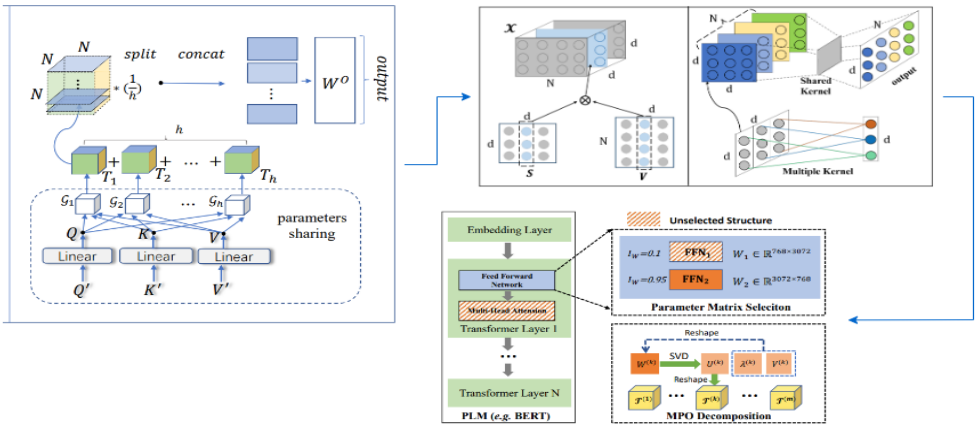


Fig. 2 The proposed architecture

The proposed architecture includes multi-linear attention based on Block-Term tensor decomposition then feature extraction using convolution, then an over-parameter framework (OPF) in fine-tuning PLMs is incorporated. IW denotes the estimated important score of parameter matrices. This presents an illustrative example of how a parameter matrix W is selected for over-parameterization and is decomposed into a set of high-order tensors {T (k)} m k=1.

To establish the multi-head mechanism and compress parameters across multiple mapping groups, a set of linear projections is employed, sharing their output. Illustrated in Figure 2, the learned linear projection effectively maps queries, keys, and values to matrices comprising basis vectors. Subsequently, Block-Term tensor decomposition is utilized to construct the multi-head mechanism. The proposed model, termed multi-linear attention, is defined as follows:

$$MultiLinear(G;\ Q', K', V') = SplitConcat\left(\frac{1}{h} * (T1 + \ldots + Th)\right) W^0 \ldots\ldots\ldots\ldots\ldots (2)$$

The function SplitConcat(·) facilitates concatenation following tensor splitting for a 3-order tensor. Figure 2 illustrates the fundamental concept behind multi-linear attention. The parameter matrix WO corresponds to the output of multi-linear attention and functions as a fully connected layer. AttenTD(·) represents the Single-block attention function, a component of multi-linear attention. Parameters matrices Wq, Wk, and Wv are shared in constructing multi-linear attention.

The time complexity of the attention function outlined in Eq. 2 is O(N^2d), where N represents the sequence length and d signifies the representation dimension. In the context of multi-linear attention, we can reorganize computations to achieve a model complexity of O(N^3), with N still representing the sequence length. Notably, the minimum number of sequential operations in multi-linear attention for various layers approximates that of self-attention in the Transformer model.

Scaled Dot-Product Attention (Token-Wise Attention) uses a particular self-attention, called Scaled Dot-Product Attention. The input of attention consists of queries matrix Q, keys matrix K, and values matrix V. The attention can be written as follows:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{dk}})V \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

where matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{N \times d}$, and d is the dimensionality of head. The token-wise attention computes the dot products of the query with all keys, where the time complexity is quadratic in the sequence length. It focuses on the weights between tokens and then applies a softmax function to obtain the weights on the values.

Multi-Head Attention Transformer uses multi-head attention to allow the model to jointly attend to information from a different representation:

$$MultiHead(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(head_1, \ldots., head_k) W^O \ldots\ldots\ldots\ldots (4)$$

where matrices $W^O \in \mathbb{R}^{hd \times dmodel}$, h is the number of heads, and $d_{model} = h \times d$ is the dimensionality of model.

Algorithm: Fine-tuning a Language Model

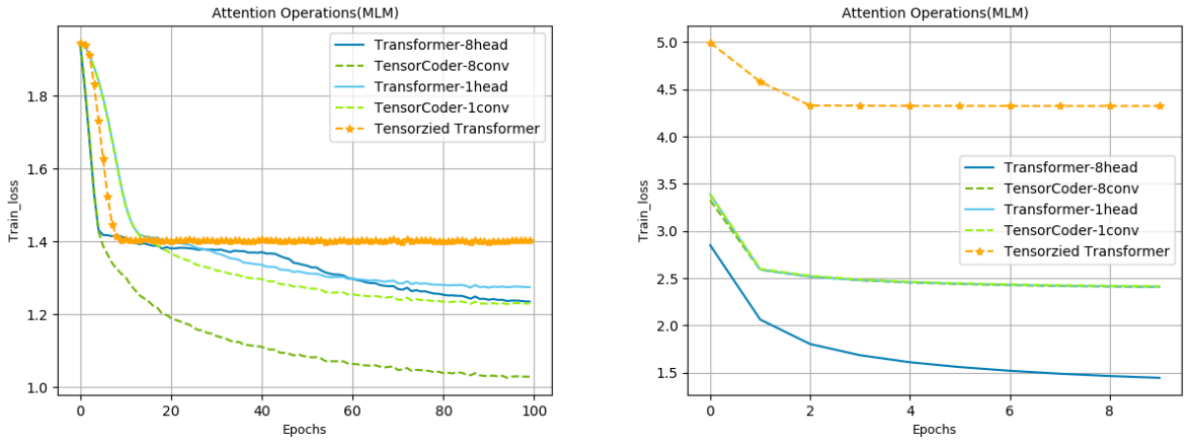
Input: Parameters matrices set of a **PLM {W}**.
1: **Divide {W}** into several groups by module.
2: *if* is Static Strategy then
3: Fine-tuning the PLM until converged.
4: *Compute* **IW for {W}** using Eq. (2).
5: **Sort {W}** in each group according to IW.
6: Perform **MPO** on the top-N matrices.
7: Train the other PLM until converged.
8: *else*
9: Define **S = {}**
10: *while* **Len(S) < N** do
11: Train the PLM for t steps.

12: Compute **IW** for **{W}** using Eq. (4).
13: Sort **{W}** in each group according to **IW**.
14: Add top-n matrices into **S**, and perform **MPO**.
15: end *while*
16: Continually train the **PLM** until converged.
17: end *if*

While the MPO method offers efficiency and flexibility, its extensive utilization for over-parameterizing all parameter matrices in small PLMs remains costly. To harness the advantages of over-parameterization effectively, we focus on selecting the most crucial parameter matrices from PLMs for decomposition. This selection process involves two strategies: a static selection strategy and a dynamic selection strategy. The static strategy pre-selects important parameter matrices, while the dynamic strategy dynamically chooses them during fine-tuning. These approaches enable the concentration of over-parameterization benefits on key parameters, optimizing resource utilization in small PLMs.

IV. RESULTS & DISCUSSION

Masked language modeling (MLM) tasks involve randomly masking a certain percentage of input tokens and predicting only those masked tokens [23]. Often referred to as a Cloze task in literature [24], MLM differs from standard conditional language models as it can be trained bidirectionally. Like BERT [25], our experiments involve masking 15% of all tokens randomly in each sequence. Furthermore, we adopt varying processing methods for masked words: 80% are replaced with the [MASK] token, 10% with a random word, and 10% are kept unchanged. These methods bias the model towards accurately predicting the observed word. For our experiments, we utilize two datasets: PTB and WikiText-103. The PTB dataset comprises 929k training tokens, 73k validation words, and 82k test words, making it a widely used dataset for language model learning. On the other hand, the WikiText-103 dataset contains 267,735 unique tokens and features 103M training tokens extracted from 28.5k articles. This dataset is suitable for models capable of capturing long-term dependencies and retains the original case, punctuation, and numbers. No additional processing is required other than replacing newlines with tokens. Models undergo evaluation based on Negative Log-Likelihood Loss (NLL), also known as multi-class cross entropy. Lower loss values indicate better model performance. In addition to loss metrics, model complexity is assessed in terms of the number of floating-point operations (FLOPs) [26], which primarily involves counting the sum of multiplication and addition operations. In the context of the Masked Language Modeling (MLM) task, only the encoder structure of the Transformer is utilized. To streamline the model, we replace multi-head attention in each encoder layer with dimension-wise attention, while keeping other components unchanged. This adjustment helps maintain efficiency while focusing on the essential aspects of the MLM task.



(a) PTB Loss (b) Wiki-103 Loss

Fig. 3: Loss of TensorCoder vs Transformer on PTB and Wiki-103 dataset.

ANALYSIS OF COMPLEXITY:

N is the length of a sequence, and d is the dimension of token embedding. The complexity analysis of our model is introduced from two aspects: first, in the encoder part of TensorCoder, the time complexity of dimension-wise attention matrix (in Eq. 4), tensor representation by KR product (in Eq. 1), and feature extraction (in Eq. 3) all are $O(Nd^2)$. Therefore, the time complexity of TensorCoder's encoder part is $O\ (Nd^2)$. In some tasks (i.e., Masked language modeling), they only use the encoder part, TensorCoder has a great advantage compared with Transformer; Second, in the decoder of TensorCoder, the complexity of dimension-wise attention tensor S (in Eq. 2) is $O\ (N^2d^2)$, the complexity of third-order tensor construction process (in Eq. 1) and final feature extraction both are $O\ (Nd^2)$. Therefore, the complexity of TensorCoder is $O\ (Nd^2 + N^2d^2)$ in the decoder part.

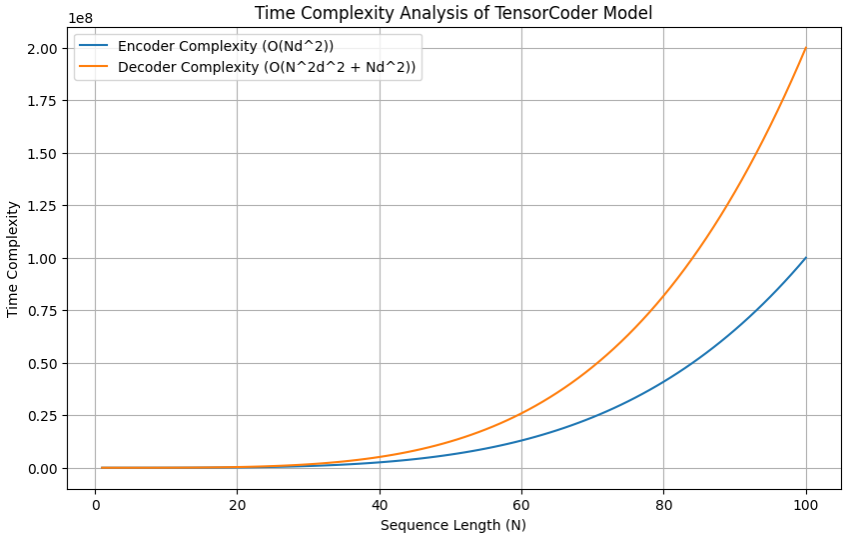
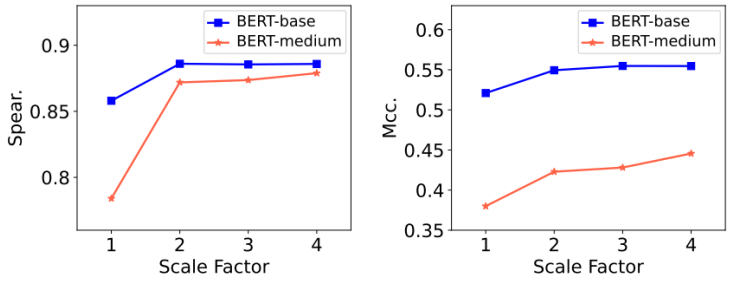


Fig. 4: Time Complexity Analysis of TensorCoder Model

Performance Comparison w.r.t. Parameter Increasing Rate. During fine-tuning, our approach can increase the number of model parameters for improving the over-parameterization of PLMs. As our approach is a general and flexible way to increase the model parameters into arbitrary scales, here we investigate how the performance changes w.r.t. a different number of increased model parameters.



(a) Performance on STS-B. (b) Performance on CoLA.

Fig. 5: Comparison of different scale factors of parameter number after over-parameterizing BERT-medium and BERT-base in STS-B and CoLA tasks.

Table 1: Comparison of different learning rates on RTE, CoLA and STS-B tasks using proposed approach.

Learning Rate	**5e-6**	**1e-5**	**3e-5**	**5e-5**	**1e-4**
RTE	72.09	72.42	72.22	72.41	70.34
CoLA	59.96	60.74	60.84	60.81	59.41
STS-B	88.52	88.69	89.51	88.35	88.54

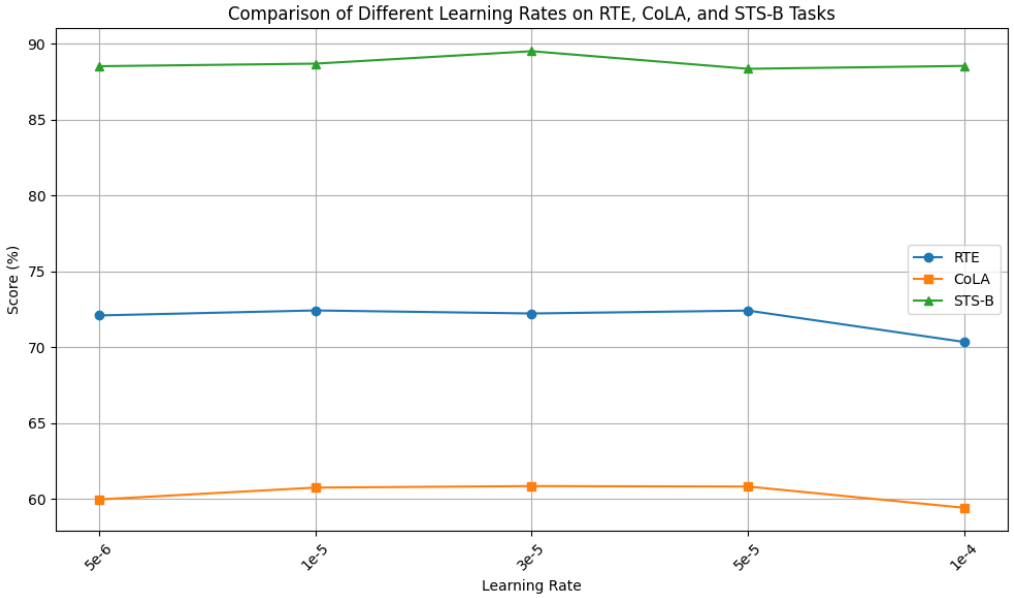


Fig. 6: Comparison of different learning rates on RTE, CoLA, STS-B Tasks

Hyperparameter Tuning: In the Orthogonal Parameterization Framework (OPF) employing the dynamic strategy, the values of two crucial hyperparameters - the total number of selected parameter matrices (N) and the selection number at one time (n) within each parameter matrix group - significantly impact model performance. A larger N implies more parameter matrices are selected and over-parameterized, while a larger n indicates that more matrices are over-parameterized simultaneously. To assess the impact of these values on model performance, experiments were conducted on the CoLA task using BERT-base as the backbone. As depicted in Figure 6, a steady improvement in performance is observed with increasing values of N, eventually reaching a plateau. This trend suggests that over-parameterizing too few matrices fails to sufficiently enhance the performance of the pre-trained language model (PLM). Conversely, excessively large values of n tend to degrade performance. This deterioration may arise from over-parameterizing too many matrices simultaneously, leading the dynamic strategy to resemble the static one. Nevertheless, the approach consistently outperforms the baseline method across different values of N and n. This indicates that the approach exhibits robustness to variations in these hyperparameters, underscoring its effectiveness and stability in practical applications.

V. CONCLUSION

In conclusion, the novel self-attention encoder layer, multi-linear attention, introduced in this study, offers a compressed alternative to the original multi-head attention, thereby presenting a new encoding scheme. Central to the contribution is the Tensorized Transformer structure based on Block-Term tensor decomposition, which integrates a series of 3-order tensors while leveraging low-rank approximation and parameter-sharing concepts. Notably, the model achieves superior compression ratios and demonstrates enhanced performance, particularly in language modeling tasks, compared to existing Transformer-based methods. These findings suggest promising prospects for the broader applicability of the approach across various natural language processing tasks, especially in resource-constrained environments. Furthermore, the proposal of TensorCoder, an encoder-decoder language model, substitutes token-wise attention with dimension-wise attention, thereby reducing complexity to $O(Nd\text{^}2)$. This linear increase in complexity concerning N contrasts with the quadratic complexity of token-wise attention, making TensorCoder well-suited for long-sequence tasks while enhancing the efficiency of pre-trained language models. Moreover, the Over-Parameterization Framework (OPF) offers a novel strategy to enhance the performance of small pre-trained language models (PLMs) by scaling up the number of parameters exclusively during fine-tuning. By incorporating the matrix product operator method and devising static and dynamic selection strategies for over-parameterization, OPF significantly boosts the performance of small PLMs, even surpassing larger counterparts. Future research directions will focus on exploring more efficient tensor decomposition methods for PLM over-parameterization and extending the application of OPF to other critical backbone models across computer vision and multimodal domains.

REFERENCES

[1] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, 2018.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008, 2017.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[4] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf, 2018.

[5] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. Published at ICLR2019, 2018.

[6] Guangxi Li, Jinmian Ye, Haiqin Yang, Di Chen, Shuicheng Yan, and Zenglin Xu. Bt-nets: simplifying deep neural networks via block term decomposition. arXiv preprint arXiv:1712.05689, 2017.

[7] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. Psychometrika, 35(3):283–319, 1970.

[8] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. Psychometrika, 31(3):279–311, 1966.

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

[10] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In Advances in neural information processing systems, pages 1135–1143, 2015.

[11] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 6655–6659. IEEE, 2013.

[12] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 535–541. ACM, 2006.

[13] Taco Cohen and Max Welling. Group equivariant convolutional networks. In International conference on machine learning, pages 2990–2999, 2016.

[14] Peng Zhang, Zhan Su, Lipeng Zhang, Benyou Wang, and Dawei Song. A quantum many-body wave function inspired language modeling approach. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pages 1303–1312. ACM, 2018.

[15] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In Proceedings of the British Machine Vision Conference. BMVA Press, 2014.

[16] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In Advances in neural information processing systems, pages 442–450, 2015.

[17] Ivan V Oseledets. Tensor-train decomposition. SIAM Journal on Scientific Computing, 33(5):2295–2317, 2011.

[18] Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov. Ultimate tensorization: compressing convolutional and fc layers alike. arXiv preprint arXiv:1611.03214, 2016.

[19] Yinchong Yang, Denis Krompass, and Volker Tresp. Tensor-train recurrent neural networks for video classification. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 3891–3900. JMLR. org, 2017.

[20] Ehsan Variani, Ananda Theertha Suresh, and Mitchel Weintraub. West: Word encoded sequence transducers. arXiv preprint arXiv:1811.08417, 2018.

[21] Valentin Khrulkov, Oleksii Hrinchuk, Leyla Mirvakhabova, and Ivan Oseledets. Tensorized embedding layers for efficient model compression. arXiv preprint arXiv:1901.10787, 2019.

[22] Alec Radford Rewon Child, Scott Gray and Ilya Sutskever. Generating long sequences with sparse transformer. arXiv preprint arXiv:1904.10509, 2019.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics, 2019.

[24] Wilson L. Taylor. Cloze procedure: A new tool for measuring readability. Journalism & Mass Communication Quarterly, 30(30):415–433, 1953.

[25] Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. Non-autoregressive neural machine translation. In ICLR 2018: International Conference on Learning Representations 2018, 2018.

[26] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In ICLR 2017: International Conference on Learning Representations 2017, 2017.

[27] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A survey of large language models. arXiv preprint arXiv:2303.18223.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pages 5998–6008.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics.

[30] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res., 21(140):1–67.

[32] 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.

[33] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. Advances in neural information processing systems, 33:1877–1901.

[34] Devansh Arpit and Yoshua Bengio. 2019. The benefits of over-parameterization at initialization in deep relu networks. arXiv preprint arXiv:1901.03611.

[35] Simon S. Du, Xiyu Zhai, Barnabás Póczos, and Aarti Singh. 2019. Gradient descent provably optimizes over-parameterized neural networks. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

[36] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019b. A convergence theory for deep learning via over-parameterization. In International Conference on Machine Learning, pages 242–252. PMLR.

[37] Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net.

[38] Eran Malach, Gilad Yehudai, Shai Shalev-Schwartz, and Ohad Shamir. 2020. Proving the lottery ticket hypothesis: Pruning is all you need. In International Conference on Machine Learning, pages 6682–6691. PMLR.

[39] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. 2022. Pali: A jointly-scaled multilingual language-image model. arXiv preprint arXiv:2209.06794.