tCURLoRA: Tensor CUR Decomposition Based Low-Rank Parameter Adaptation and Its Application in Medical Image Segmentation

Guanghua He[1,2], Wangang Cheng[2], Hancan Zhu[2(✉)], Xiaohao Cai[3], and Gaohang Yu[1(✉)]

[1] Department of Mathematics, Hangzhou Dianzi University, Hangzhou 310018, China
`maghyu@163.com`
[2] School of Mathematics, Physics and Information, Shaoxing University, Shaoxing 312000, China
`hancanzhu@yeah.net`
[3] School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK

Abstract. Transfer learning, by leveraging knowledge from pre-trained models, has significantly improved the performance of downstream tasks. However, as deep neural networks continue to scale, full fine-tuning poses substantial computational and storage challenges in resource-constrained environments, limiting its practical adoption. To address this, parameter-efficient fine-tuning (PEFT) methods have been proposed to reduce computational complexity and memory requirements by updating only a small subset of parameters. Among them, matrix decomposition-based approaches such as LoRA have shown promise, but often struggle to fully capture the high-dimensional structural characteristics of model weights. In contrast, high-order tensors offer a more natural representation of neural network parameters, enabling richer modeling of multi-dimensional interactions and higher-order features. In this paper, we propose tCUR-LoRA, a novel fine-tuning method based on tensor CUR decomposition. By stacking pre-trained weight matrices into a third-order tensor and applying tensor CUR decomposition, our method updates only the compressed tensor components during fine-tuning, thereby substantially reducing both computational and storage costs. Experimental results show that tCURLoRA consistently outperforms existing PEFT approaches on medical image segmentation tasks. The source code is publicly available at: https://github.com/WangangCheng/t-CURLora.

Keywords: Parameter-efficient fine-tuning · tensor CUR decomposition · deep learning · transfer learning · medical image segmentation

---

G. He and W. Cheng—Contributed equally.

1   Introduction

Transfer learning has significantly improved the performance of target tasks, especially in data-scarce scenarios, by leveraging knowledge from pre-trained models [19,23]. However, as deep learning models continue to scale, full fine-tuning incurs substantial computational and storage costs in resource-constrained environments, limiting its practical applicability. To address this challenge, parameter-efficient fine-tuning (PEFT) methods have been introduced. By reducing the number of parameters that need to be updated in deep neural networks (DNNs), PEFT methods effectively lower computational complexity and storage demands, and have emerged as a prominent research focus in recent years [6,9,12].

Low-Rank Adaptation (LoRA) is a well-established PEFT method that reduces the number of trainable parameters by introducing low-rank matrices to incrementally update pre-trained weights in DNNs, while maintaining high model performance [12]. To further improve generalization, Hydra extends this idea through a multi-head low-rank adaptation strategy, combining parallel and sequential branching structures to enhance model expressiveness [14]. PiSSA, built upon singular value decomposition (SVD), improves fine-tuning efficiency by focusing on the dominant singular values and their corresponding vectors [18]. In parallel, both CURLoRA and PMSS leverage CUR matrix decomposition to further optimize adaptation. CURLoRA mitigates catastrophic forgetting during continuous learning [7], whereas PMSS improves adaptability to complex tasks by selecting skeletal substructures from pre-trained weight matrices [22].

Current PEFT methods based on low-rank adaptation predominantly rely on matrix decomposition. However, such approaches often struggle to capture the high-dimensional structural properties of DNN weights. In contrast, high-order tensors offer a more natural and expressive representation, enabling the modeling of complex multi-dimensional interactions and higher-order feature dependencies [20]. Incorporating tensor decomposition into PEFT has the potential to not only improve the efficiency of transfer learning, but also to reveal latent structures embedded within high-dimensional weight representations.

In this paper, we propose tCURLoRA, a key substructure fine-tuning method based on tensor CUR decomposition. Specifically, we stack pre-trained weight matrices from multiple layers along the frontal dimension to form a third-order tensor, which captures shared architectural patterns across transformer layers. This tensorized structure facilitates the application of tensor CUR decomposition, enabling more effective modeling of cross-layer correlations than traditional matrix-based approaches. During fine-tuning, only the compressed tensor components are updated, substantially reducing the number of trainable parameters and minimizing both computational and memory overhead. The main contributions of this work are as follows:

– We propose tCURLoRA, which leverages tensor CUR decomposition on stacked pre-trained weights to exploit high-dimensional structures and enable efficient adaptation by updating only the most informative components.

– Experimental results on three transfer learning tasks demonstrate that tCUR-LoRA consistently outperforms existing PEFT baselines in segmentation accuracy under limited data conditions.

2   Method

2.1   Overall Architecture

Figure 1(a) illustrates a PEFT method based on matrix CUR decomposition [7,22]. In this approach, matrix CUR decomposition is independently applied to each pre-trained weight matrix for fine-tuning. Let $\widetilde{W}_i \in \mathbb{R}^{n_1 \times n_2}$, $i = 1, 2, \ldots, n_3$ denote the n_3 pre-trained weight matrices. During fine-tuning, these matrices remain frozen, while their increments ΔW_i are updated. The update rule is given by:

$$W_i = \widetilde{W}_i + \Delta W_i = \widetilde{W}_i + C_i U_i R_i, \quad i = 1, 2, \ldots, n_3, \tag{1}$$

where $C_i \in \mathbb{R}^{n_1 \times c}$ contains c sampled columns from $\widetilde{W}_i$, and $R_i \in \mathbb{R}^{r \times n_2}$ contains r sampled rows. Both C_i and R_i remain fixed during fine-tuning, while $U_i \in \mathbb{R}^{c \times r}$ is initialized to zero and optimized. To simplify hyperparameter tuning, c and r are typically set equal [7,22].

Figure 1(b) shows the proposed tCURLoRA, which extends matrix CUR to the tensor setting in order to better exploit high-dimensional structural patterns. Specifically, pre-trained weight matrices are stacked along the frontal dimension
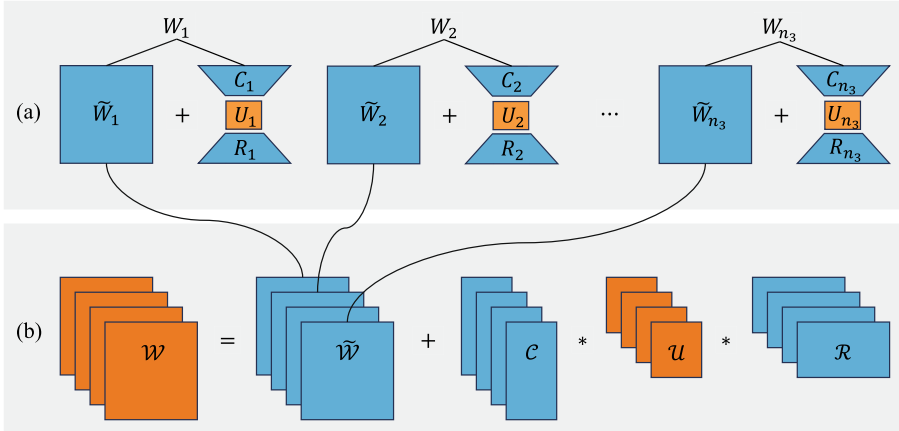

Fig. 1. Comparison of PEFT Methods: Matrix vs. Tensor CUR Decomposition. (a) Matrix CURLoRA fine-tunes weight matrices independently. (b) tCURLoRA stacks weight matrices into a 3D tensor for fine-tuning. Blue indicates frozen parameters, orange indicates updated parameters, and * denotes the tensor product. (Color figure online)

to form a third-order tensor $\widetilde{\mathcal{W}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. Fine-tuning is performed via tensor CUR decomposition, expressed as:

$$\mathcal{W} = \widetilde{\mathcal{W}} + \Delta \mathcal{W} = \widetilde{\mathcal{W}} + \mathcal{C} * \mathcal{U} * \mathcal{R}, \tag{2}$$

where $\mathcal{C} \in \mathbb{R}^{n_1 \times r \times n_3}$ and $\mathcal{R} \in \mathbb{R}^{r \times n_2 \times n_3}$ are fixed low-rank tensors obtained from the tensor CUR decomposition of $\widetilde{\mathcal{W}}$, and $\mathcal{U} \in \mathbb{R}^{r \times r \times n_3}$ is a learnable tensor initialized to zero. The symbol "$*$" denotes the tensor product. Here, r represents the number of sampled rows and columns shared across slices.

2.2   Tensor CUR Decomposition

The tensor CUR decomposition adopted in this work is based on the tensor product (t-product) framework [13,16].

Definition 1. *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times l \times n_3}$ be two third-order tensors. The t-product $\mathcal{A} * \mathcal{B}$ yields a tensor of size $n_1 \times l \times n_3$, defined as:*

$$\mathcal{A} * \mathcal{B} = \text{fold}\left(\text{circ}(\mathcal{A})\text{MatVec}(\mathcal{B})\right),$$

where the operators circ, MatVec, *and* fold *are described below.*

The expressions for $\text{circ}(\mathcal{A})$ and $\text{MatVec}(\mathcal{B})$ are given by:

$$\text{circ}(\mathcal{A}) = \begin{bmatrix} A_1 & A_{n_3} & \cdots & A_2 \\ A_2 & A_1 & \cdots & A_3 \\ \vdots & \vdots & \ddots & \vdots \\ A_{n_3} & A_{n_3-1} & \cdots & A_1 \end{bmatrix}, \quad \text{MatVec}(\mathcal{B}) = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_{n_3} \end{bmatrix},$$

where $A_i = \mathcal{A}(:,:,i)$ and $B_i = \mathcal{B}(:,:,i)$ denote the ith frontal slices of tensors $\mathcal{A}$ and $\mathcal{B}$, respectively. The fold operation transforms the matrix $\text{MatVec}(\mathcal{B})$ back into the original tensor $\mathcal{B}$, i.e., $\text{fold}(\text{MatVec}(\mathcal{B})) = \mathcal{B}$.

It is known that a block circulant matrix can be diagonalized via the Discrete Fourier Transform (DFT) [8], i.e., $(F \otimes I_{n_1})\text{circ}(\mathcal{A})(F^* \otimes I_{n_2})$ results in a block-diagonal matrix, where $F \in \mathbb{R}^{n_3 \times n_3}$ is the DFT matrix, F^* is its conjugate transpose, $\otimes$ denotes the Kronecker product, and I_{n_1} is the $n_1 \times n_1$ identity matrix. In addition, we have:

$$\text{circ}(\mathcal{A})\text{MatVec}(\mathcal{B}) = (F^* \otimes I_{n_1})\left((F \otimes I_{n_1})\text{circ}(\mathcal{A})(F^* \otimes I_{n_2})\right)(F \otimes I_{n_2})\text{MatVec}(\mathcal{B}).$$

Thus, the t-product can be efficiently computed as follows: (i) apply the DFT along the third dimension of tensors $\mathcal{A}$ and $\mathcal{B}$ to transform them into the frequency domain; (ii) perform matrix multiplication on corresponding frontal slices; (iii) apply the inverse DFT along the third dimension of the resulting tensor to obtain the final output.

The tensors $\mathcal{C}$ and $\mathcal{R}$ in Equation (2) are derived via the tensor CUR decomposition of $\widetilde{\mathcal{W}}$, following the procedure in [5]. Specifically, we first apply the Fast Fourier Transform (FFT) along the third dimension of $\widetilde{\mathcal{W}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ to obtain

$$\widehat{\mathcal{W}} = \text{fft}(\widetilde{\mathcal{W}}, [], 3).$$

To evaluate the importance of columns, we define a column score α_j as:

$$\alpha_j = \frac{\sum_{k=1}^{n_3} \|\widehat{\mathcal{W}}(:,j,k)\|_2}{\sum_{j=1}^{n_2} \sum_{k=1}^{n_3} \|\widehat{\mathcal{W}}(:,j,k)\|_2}, \quad j = 1, 2, \ldots, n_2, \tag{3}$$

where $\|\cdot\|_2$ denotes the ℓ_2-norm of a vector. The top r columns with the largest scores are selected to form index set J. Similarly, we define the row score β_i as:

$$\beta_i = \frac{\sum_{k=1}^{n_3} \|\widehat{\mathcal{W}}(i,J,k)\|_2}{\sum_{i=1}^{n_1} \sum_{k=1}^{n_3} \|\widehat{\mathcal{W}}(i,J,k)\|_2}, \quad i = 1, 2, \ldots, n_1. \tag{4}$$

The top r rows form index set I. Using sets I and J, we extract:

$$\widehat{\mathcal{C}} = \widehat{\mathcal{W}}(:,J,:), \quad \widehat{\mathcal{U}} = \widehat{\mathcal{W}}(I,J,:), \quad \widehat{\mathcal{R}} = \widehat{\mathcal{W}}(I,:,:).$$

Finally, we apply the inverse FFT along the third dimension to obtain the final decomposition components:

$$\mathcal{C} = \mathrm{ifft}(\widehat{\mathcal{C}}, [], 3), \quad \mathcal{U} = \mathrm{ifft}(\widehat{\mathcal{U}}, [], 3), \quad \mathcal{R} = \mathrm{ifft}(\widehat{\mathcal{R}}, [], 3),$$

leading to the approximation $\widetilde{\mathcal{W}} \approx \mathcal{C} * \mathcal{U}^\dagger * \mathcal{R}$, where $\mathcal{U}^\dagger$ denotes the Moore–Penrose inverse of $\mathcal{U}$.

2.3   tCURLoRA for Fine-Tuning UNETR Parameters

We apply tCURLoRA to fine-tune the UNETR network [10], which comprises a Transformer-based encoder with 12 layers and a convolutional decoder, connected by deconvolution and convolution layers. Given that most parameters reside in the Transformer modules, tCURLoRA fine-tunes the Transformer modules while fully updating the decoder parameters and keeping the skip connection parameters frozen.

Each Transformer layer consists of two primary components: Multi-Head Self-Attention (MHSA) and a Multi-Layer Perceptron (MLP). In the MHSA module, the queries, keys, values, and outputs are generated using projection matrices $W_a, W_k, W_v, W_o \in \mathbb{R}^{d \times d}$. These are divided into N_h attention heads, where each head uses its own set of matrices $W_q^{(i)}, W_k^{(i)}, W_v^{(i)}, W_o^{(i)} \in \mathbb{R}^{d \times d_h}$, with $d_h = d/N_h$ and $i = 1, 2, \ldots, N_h$. The output of the MHSA module is computed as:

$$MHSA(X) = \sum_{i=1}^{N_h} \mathrm{softmax}\left(\frac{X W_q^{(i)} W_k^{(i)^T} X^T}{\sqrt{d_h}} \right) X W_v^{(i)} W_o^{(i)^T}.$$

The MLP module contains two fully connected layers (biases omitted), expressed as:

$$MLP(X) = \mathrm{GELU}(X W_{up}) W_{down},$$

where GELU is the Gaussian Error Linear Unit activation function [11], and $W_{up} \in \mathbb{R}^{d \times 4d}$, $W_{down} \in \mathbb{R}^{4d \times d}$ are the weights of the two layers.

Each Transformer layer therefore contains four $d \times d$ matrices from the MHSA module and two matrices–$d \times 4d$ and $4d \times d$–from the MLP module. Across 12 layers, this results in 48 $d \times d$ matrices, 12 $d \times 4d$ matrices, and 12 $4d \times d$ matrices. These are concatenated along the frontal dimension to form three tensors: $\mathcal{W}_{sa} \in \mathbb{R}^{d \times d \times 48}$, $\mathcal{W}_{up} \in \mathbb{R}^{d \times 4d \times 12}$, and $\mathcal{W}_{down} \in \mathbb{R}^{4d \times d \times 12}$. In tCURLoRA, these tensors are fine-tuned independently to enable efficient model optimization.

3   Experiments

3.1   Datasets

The UNETR model is first pre-trained on the BraTS2021 dataset [1], which contains multimodal magnetic resonance imaging (MRI) data, including T1, T1ce, T2, and FLAIR modalities. For each patient, the image resolution is $240 \times 240 \times 155$ with a voxel size of $1 \times 1 \times 1$ mm^3. The dataset provides detailed tumor annotations for 1,251 cases, segmented into three regions: enhancing tumor (ET), peritumoral edema/infiltrative tissue (ED), and necrotic tumor core (NCR).

We then apply the tCURLoRA method to transfer the pre-trained UNETR segmentation model to three downstream datasets: EADC-ADNI [4], LPBA40 [21], and UPENN-GBM [2]. The EADC-ADNI dataset, derived from the ADNI database, includes MRI scans of 135 patients with a resolution of $197 \times 233 \times 189$ and a voxel size of $1 \times 1 \times 1$ mm^3. The ADNI project, launched in 2003, was initially designed to assess biomarkers for tracking mild cognitive impairment (MCI) and early Alzheimer's disease (AD), and now focuses on validating biomarkers for clinical trials and enhancing data diversity (http://adni.loni.usc.edu/). Hippocampal annotations are based on the harmonized segmentation protocol proposed by the European Alzheimer's Disease Consortium and ADNI [4] (http://www.hippocampal-protocol.net). During quality control, five cases with annotation inconsistencies were identified and excluded to ensure labeling accuracy.

The LPBA40 dataset, developed by the Laboratory of Neuroimaging (LONI), comprises 3D brain MRI scans from 40 healthy adults, with a resolution of $256 \times 124 \times 256$ and voxel dimensions of $0.8938 \times 1.500 \times 0.8594$ mm^3. It provides detailed manual annotations for 56 brain tissues and structures. In this study, only hippocampal annotations are used to evaluate the proposed method.

The UPENN-GBM dataset contains MRI scans from 630 glioblastoma patients collected at the University of Pennsylvania Health System and made publicly available via the Cancer Imaging Archive (TCIA) [2]. All scans were acquired pre-operatively using a 3T MRI scanner, with imaging modalities including T1, T2, contrast-enhanced T1 (T1GD), and FLAIR, along with corresponding segmentation labels. A subset of 147 scans was manually annotated by clinical experts to delineate three tumor sub-regions: necrotic core (NC), peritumoral edema (ED), and enhancing tumor (ET). In this study, we use these 147 expert-annotated cases, selecting the T1GD modality and defining the Whole Tumor (WT) region–comprising all three sub-regions–as the segmentation target.

All images from the aforementioned datasets were skull-stripped and registered to the MNI152 standard space, resulting in a uniform voxel size of $1 \times 1 \times 1$ mm^3.

3.2   Experimental Details

We conducted experiments using PyTorch on two NVIDIA GeForce RTX 4090D GPUs. During pre-training, we merged the three annotated tumor sub-regions in the BraTS2021 dataset into a single tumor region for binary segmentation, using only the T1ce modality. We split the 1,251 publicly annotated cases into training (1,000), validation (125), and test (126) sets following an 8:1:1 ratio. We configured the hyperparameters according to the UNETR paper [10].

In the fine-tuning phase, we randomly selected five samples from each of the EADC-ADNI and LPBA40 datasets for training, with the remaining samples used for testing. For the UPENN-GBM dataset, 10% of the samples were used for training and the remaining 90% for testing. This setting reflects the relatively consistent anatomical structure of the hippocampus compared to the higher variability of brain tumors, thereby simulating different levels of data scarcity. We trained the models using the Adam optimizer with a batch size of 4 and a polynomial learning rate decay schedule, starting with an initial learning rate of 0.001 and applying a decay factor of 0.9 per iteration. Random cropping of size $128 \times 128 \times 128$ was applied during training. Data augmentation techniques included: (1) random mirroring with a probability of 50%, (2) intensity shifts within the range $[-0.1, 0.1]$, and (3) scaling within the range $[0.9, 1.1]$. The network was trained for 1,000 epochs using Dice loss with L2 regularization (weight decay of 10^{-5}).

During testing, $128 \times 128 \times 128$ patches were extracted using a non-overlapping sliding window strategy. The final segmentation was obtained by averaging the outputs from the last four training epochs. Post-processing was performed to eliminate false positives by identifying connected components and removing those with a volume smaller than $1 \, cm^3$, which were considered background.

3.3   Experimental Results

We compared the proposed tCURLoRA with several PEFT methods, including Full fine-tuning (Full), Linear probing (Linear), LoRA [12], Adapter [17], SSF [15], LoTR [3], PISSA [18], and CURLoRA [7]. For rank-based methods (LoRA, Adapter, LoTR, PISSA, CURLoRA, and tCURLoRA), we tuned the rank r on the EADC-ADNI dataset. The optimal values were $r = 32$ for LoRA and Adapter, $r = 2$ for LoTR, PISSA, and CURLoRA, and $r = 8$ for tCURLoRA. These settings were used in subsequent experiments unless otherwise specified.

Table 1 presents the segmentation performance on the EADC-ADNI, LPBA40, and UPENN-GBM datasets. The proposed tCURLoRA method comprises 2.683 million (M) parameters, comparable to SSF, LoTR, PISSA, and CURLoRA, while requiring only 2.98% of the parameters used in full fine-tuning,

Table 1. Comparison of segmentation results of different fine-tuning methods on three datasets, in terms of Dice (%) and HD95 (mm) metrics. The best results are highlighted in bold.

Method	#Params (M)	EADC-ADNI		LPBA40		UPENN-GBM	
		Dice	HD95	Dice	HD95	Dice	HD95
Full	90.011	83.79	5.839	79.91	7.175	69.95	32.280
Linear	59.348	83.46	5.344	80.62	6.483	72.42	29.401
LoRA [12]	7.397	84.35	5.334	80.17	6.618	72.51	29.799
Adapter [17]	7.987	84.08	5.663	79.72	6.793	73.46	33.357
SSF [15]	2.883	83.73	5.326	79.08	6.904	72.29	**28.762**
LoTR [3]	2.703	84.05	5.394	80.21	7.011	71.14	32.093
PiSSA [18]	2.974	84.45	5.603	80.62	6.584	72.56	31.331
CURLoRA [7]	2.679	84.64	5.549	79.96	6.659	72.73	29.387
tCURLoRA (ours)	2.683	**84.95**	**4.855**	**81.12**	**6.305**	**74.28**	30.550

thereby significantly improving training efficiency. Compared with full fine-tuning, tCURLoRA improves the Dice coefficient by 1.16%, 1.21%, and 4.33%, and reduces the HD95 metric by 0.984 mm (16.85%), 0.870 mm (12.13%), and 1.73 mm (5.36%) on the three datasets, respectively, indicating enhanced segmentation accuracy. Furthermore, tCURLoRA achieves the highest Dice scores across all datasets, further demonstrating its superior performance in medical image segmentation.

We also evaluated training efficiency in terms of per-epoch runtime and memory consumption. Under identical experimental settings, tCURLoRA achieves 495 ms per epoch and utilizes 11.72 GB of memory, ranking second among all PEFT methods, just behind CURLoRA. In comparison, LoRA requires 562 ms and 15.90 GB, while full fine-tuning takes 684 ms and 18.28 GB. These results underscore the practical advantages of tCURLoRA in resource-constrained environments. Figure 2 presents qualitative segmentation results, including 2D slices and 3D surface renderings. The predictions generated by tCURLoRA closely match the ground truth (GT), particularly in regions with complex anatomical structures, effectively preserving fine details while minimizing segmentation errors.
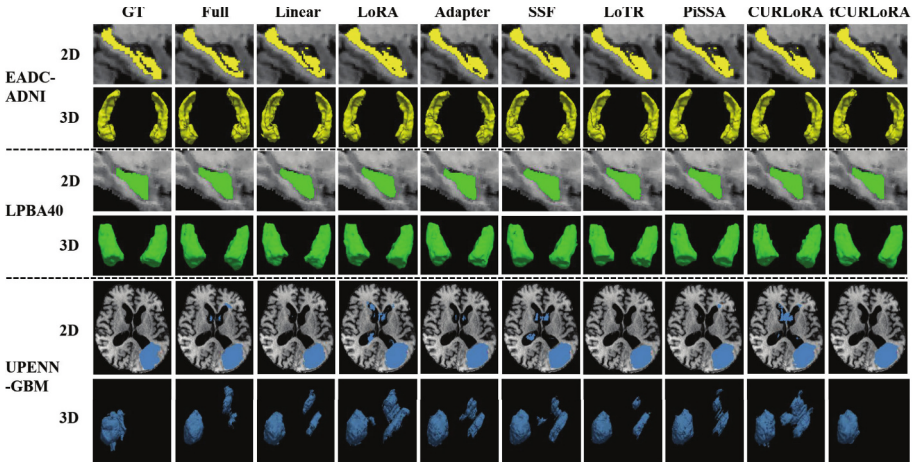
Fig. 2. Qualitative Segmentation Results: 2D Slices and 3D Surfaces Across Datasets.

4   Conclusion

This study proposes tCURLoRA, a tensor CUR decomposition-based fine-tuning method that improves the efficiency of DNNs, with a focus on medical image segmentation. By reducing learnable parameters and controlling computational complexity, tCURLoRA addresses challenges in training cost, memory usage, and scalability, while maintaining high segmentation accuracy. Future work may explore its extension to broader tasks and datasets, particularly in multimodal and large-scale segmentation scenarios. Further optimization could enhance its adaptability to real-world applications, making tCURLoRA a practical tool for efficient deep learning.

References

1. Baid, U., et al.: The rsna-asnr-miccai brats 2021 benchmark on brain tumor segmentation and radiogenomic classification. arXiv preprint arXiv:2107.02314 (2021)
2. Bakas, S., et al.: Multi-parametric magnetic resonance imaging (mpmri) scans for de novo glioblastoma (GBM) patients from the university of pennsylvania health system (upenn-gbm). Cancer Imag. Arch. (2021). https://doi.org/10.7937/TCIA.709X-DN49

3. Bershatsky, D., Cherniuk, D., Daulbaev, T., Mikhalev, A., Oseledets, I.: Lotr: Low tensor rank weight adaptation. arXiv preprint arXiv:2402.01376 (2024)

4. Boccardi, M., et al.: Training labels for hippocampal segmentation based on the eadc-adni harmonized hippocampal protocol. Alzheimer's Dementia **11**(2), 175–183 (2015). https://doi.org/10.1016/j.jalz.2014.12.002

5. Chen, J., Wei, Y., Y.X.: Tensor cur decomposition under t-product and its perturbation. Num. Funct. Anal. Optim. **43**(6), 698–722 (2022). https://doi.org/10.1080/01630563.2022.2056198

6. Ding, N., et al.: Parameter-efficient fine-tuning of large-scale pre-trained language models. Nature Mach. Intell. **5**(3), 220–235 (2023). https://doi.org/10.1038/s42256-023-00626-4

7. Fawi, M.: Curlora: Stable LLM continual fine-tuning and catastrophic forgetting mitigation. arXiv preprint arXiv:2408.14572 (2024)

8. Golub, G.H., Van Loan, C.F.: Matrix computations. JHU press (2013)

9. Han, Z., Gao, C., Liu, J., Zhang, J., Zhang, S.Q.: Parameter-efficient fine-tuning for large models: a comprehensive survey. arXiv preprint arXiv:2403.14608 (2024)

10. Hatamizadeh, A., et al.: UNETR: Transformers for 3D medical image segmentation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 574–584 (January 2022)

11. Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)

12. Hu, E.J., et al.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)

13. Kilmer, M.E., Braman, K., Hao, N., Hoover, R.C.: Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. SIAM J. Matrix Anal. Appl. **34**(1), 148–172 (2013). https://doi.org/10.1137/110837711

14. Kim, S., Yang, H., Kim, Y., Hong, Y., Park, E.: Hydra: Multi-head low-rank adaptation for parameter efficient fine-tuning. Neural Netw. **178**, 106414 (2024). https://doi.org/10.1016/j.neunet.2024.106414

15. Lian, D., Zhou, D., Feng, J., Wang, X.: Scaling & shifting your features: A new baseline for efficient model tuning. In: Advances in Neural Information Processing Systems, vol. 35, pp. 109–123. Curran Associates, Inc. (2022)

16. Liu, Y., Liu, J., Long, Z., Zhu, C.: Tensor computation for data analysis. Springer (2022)

17. Luo, G., et al.: Towards efficient visual adaption via structural re-parameterization. arXiv preprint arXiv:2302.08106 (2023)

18. Meng, F., Wang, Z., Zhang, M.: PiSSA: Principal singular values and singular vectors adaptation of large language models. In: Advances in Neural Information Processing Systems, vol. 37, pp. 121038–121072. Curran Associates, Inc. (2024)

19. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10), 1345–1359 (2010). https://doi.org/10.1109/TKDE.2009.191

20. Panagakis, Y., et al.: Tensor methods in computer vision and deep learning. Proc. IEEE **109**(5), 863–890 (2021). https://doi.org/10.1109/JPROC.2021.3074329

21. Shattuch, D.W., et al.: Construction of a 3D probabilistic atlas of human cortical structures. Neuroimage **39**(3), 1064–1080 (2008). https://doi.org/10.1016/j.neuroimage.2007.09.031

22. Wang, Q., Hu, X., Xu, W., Liu, W., Luan, J., Wang, B.: Pmss: Pretrained matrices skeleton selection for LLM fine-tuning. arXiv preprint arXiv:2409.16722 (2024)

23. Weiss, K., Khoshgoftaar, T.M., Wang, D.D.: A survey of transfer learning. J. Big Data **3**(1), 1–40 (2016). https://doi.org/10.1186/s40537-016-0043-6