SPARSE SPECTRAL TRAINING AND INFERENCE ON EUCLIDEAN AND HYPERBOLIC NEURAL NETWORKS

Jialin Zhao[1,2], **Yingtao Zhang**[1,2], **Xinghang Li**[2], **Huaping Liu**[2], **Carlo Vittorio Cannistraci**[1,2,3] *
[1]Center for Complex Network Intelligence, Tsinghua Laboratory of Brain and Intelligence
[2]Department of Computer Science
[3]Department of Biomedical Engineering, Tsinghua University, Beijing, China

ABSTRACT

The increasing GPU memory demands of large language models call for more memory-efficient training methods. Existing approaches like LoRA struggle with low-rank constraints in pre-training, while ReLoRA suffers from saddle point issues. We propose **Sparse Spectral Training (SST)**, a memory-efficient **pre-training** framework that *updates all singular values*, *selectively updates singular vectors* via multinomial sampling, and *leverages singular value decomposition (SVD) for initialization and periodic reinitialization*, reducing distortion compared to other low-rank methods. Across tasks including language generation, machine translation, and graph learning, SST outperforms existing memory-efficient training methods and is often comparable to full-rank training. On LLaMA-1.3B, SST reduces the perplexity gap to full-rank training by **97.4%**, demonstrating its effectiveness for scalable, memory-efficient model pre-training.

1 INTRODUCTION

The increasing scale of large language models (Kaplan et al., 2020; Brown et al., 2020; Touvron et al., 2023b) poses significant challenges for training from scratch due to extreme GPU memory demands. While parameter-efficient fine-tuning (PEFT) methods like LoRA (Hu et al., 2022) reduce memory consumption by restricting updates to a low-rank subspace, this constraint limits model expressiveness, particularly in pre-training. Recent methods such as ReLoRA (Lialin et al., 2024), COLA (Xia et al., 2024), and PLoRA (Meng et al., 2024b) mitigate this issue by periodically merging low-rank parameters, but they still suffer from *saddle point issues*, leading to slow and unstable convergence.

We propose **Sparse Spectral Training (SST)**, a novel framework that optimizes memory efficiency while closely approximating full-rank training. Unlike prior approaches (Hu et al., 2022; Lialin et al., 2024; Zhang et al., 2023; Ding et al., 2023) that operate in a fixed low-rank subspace, SST updates **all** singular values at each step while **selectively** updating singular vectors sampled from a multinomial distribution based on singular value magnitude. Additionally, SST utilizes **singular value decomposition (SVD) to initialize and periodically reinitialize** low-rank parameters, reducing distortion and improving convergence.

Our extensive experiments demonstrate SST's effectiveness across diverse architectures and tasks. On the OPT and LLaMA model family, SST reduces the perplexity gap between low-rank methods and full-rank training by **50%–97.4%**. In machine translation, SST narrows the BLEU gap by **66.7%**. We also pioneer parameter-efficient pre-training in hyperbolic space, where SST even **outperforms full-rank training** in most cases. A detailed discussion of related work is provided in Appendix M.

2 LOW RANK ADAPTATION

This section introduces LoRA (Hu et al., 2022), ReLoRA (Lialin et al., 2024), and GaLore (Zhao et al., 2024), highlighting their limitations, which are addressed by SST in Section 3.

*Correspondence to: Jialin Zhao <jialin.zhao97@gmail.com>, Carlo Vittorio Cannistraci <kalokagathos.agon@gmail.com>.

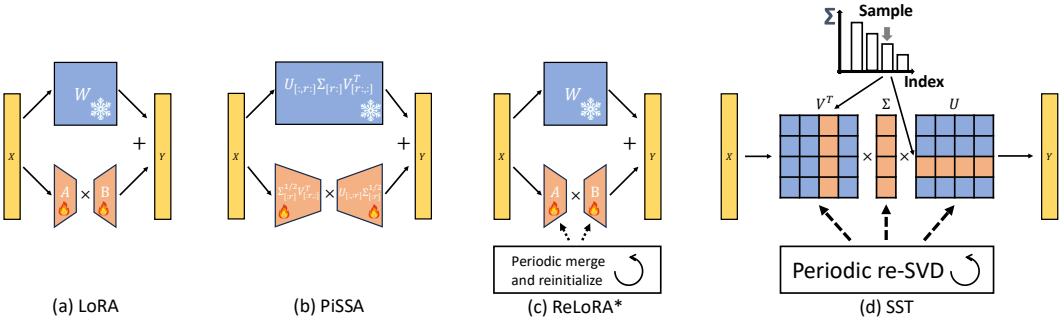

Figure 1: **Comparison of SST with LoRA, PiSSA, and ReLoRA*.** LoRA applies a low-rank update to frozen weights, while ReLoRA* iteratively initializes and merges low-rank matrices. PiSSA initializes low-rank parameters using SVD but always updates the same set of singular vectors. SST adopts a *sampling-update-swapping* paradigm, dynamically selecting singular vectors via multinomial sampling, updating all singular values, and using periodic re-SVD to maintain orthogonality, balancing exploration and exploitation in pre-training.

2.1 LoRA

LoRA fine-tunes a pre-trained model by learning a low-rank update $\Delta\mathbf{W} = \mathbf{BA}$ to a frozen weight matrix $\mathbf{W}_0$, reducing memory usage:

$$\mathbf{h} = (\mathbf{W}_0 + \Delta\mathbf{W})\mathbf{x} = (\mathbf{W}_0 + \mathbf{BA})\mathbf{x} \tag{1}$$

Limitation. LoRA is constrained by its low-rank structure. Consider $\mathbf{W}^*$ as the optimal weight matrix which minimizes loss. The deviation from the current weights is $\Delta\mathbf{W}^* = \mathbf{W}^* - \mathbf{W}_0$. Performing a singular value decomposition on $\Delta\mathbf{W}^*$ yields $\Delta\mathbf{W}^* = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$. $\mathbf{\Sigma}$ is a diagonal matrix with entries $\{\sigma_1, \sigma_2, ..., \sigma_m\}$. Then the Eckart–Young–Mirsky theorem (Eckart & Young, 1936) states:

$$\|\Delta\mathbf{W}^* - \Delta\mathbf{W}\|_{\mathrm{F}} \geq \sqrt{\sigma_{r+1}^2 + \cdots + \sigma_m^2} \tag{2}$$

This suggests that LoRA can only closely approximate the performance of full-rank training in simple tasks like fine-tuning, where $\sigma_i \approx 0, i \in \{r+1, ..., m\}$.

2.2 ReLoRA*

ReLoRA, COLA, and PLoRA mitigate LoRA's limitations by periodically merging low-rank updates into $\mathbf{W}_0$. However, ReLoRA requires a full-rank warm-up, and all methods suffer from zero-initialized low-rank matrices, causing saddle point issues. In this paper, we unify these methods into a generalized, end-to-end parameter-efficient pre-training paradigm, which we refer to as ReLoRA* and formalize in Algorithm 1.

Limitation. ReLoRA* only learns a small subset of singular values per iteration, and zero initialization results in zero gradients at reinitialization, slowing convergence.

2.3 GaLore

GaLore projects gradients into a low-rank space using a projection matrix $\mathbf{P}_t$ derived from the top-r singular vectors of the gradient of $\mathbf{W}$, recalculated every T steps.

Limitation. GaLore selects top-r singular vectors at each step, making it sensitive to data noise. In experiments, it showed instability for low rank, leading to divergence on OPT-350M. Consequently, we chose to include the detailed explanation and comparison with GaLore in Appendix G rather than in the main text.

3 Sparse Spectral Training

To overcome the limitations of prior methods, this section presents Sparse Spectral Training (SST).

(a) Full-rank (b) SST (c) ReLoRA*

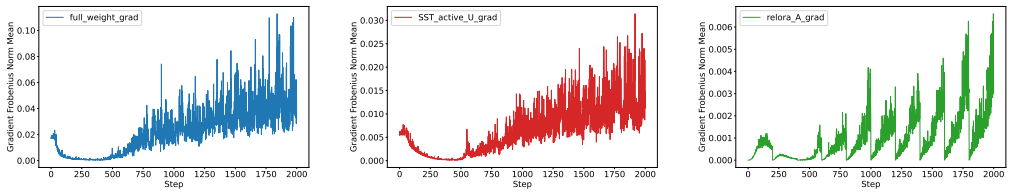

Figure 2: **ReLoRA* encounters periodic saddle points.** This plot shows the Frobenius norm of gradients for (a) full-rank training, (b) sampled $\mathbf{U}$ in SST, and (c) $\mathbf{A}$ in ReLoRA*. Both SST and ReLoRA* use an iteration interval of 200. ReLoRA* exhibits saddle points at each restart, with a lower gradient correlation (**0.58**) to full-rank training compared to SST (**0.85**), highlighting SST's closer alignment with full-rank optimization.

3.1 Sparse Spectral Layer

SST applies sparse updates in the spectral domain by decomposing each linear layer as:

$$\mathbf{h} = \mathbf{W}\mathbf{x} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}\mathbf{x}, \quad [\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^{\mathrm{T}}] = \mathrm{SVD}(\mathbf{W}) \tag{3}$$

where $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^{\mathrm{T}}$ are full-rank SVD components of $\mathbf{W}$, replacing it entirely in the model. Unlike LoRA-based methods, SST maintains full-rank representations while enabling selective updates. SVD is computed only at initialization and periodically reinitialized (Eq. 7) to maintain efficiency (see Table 20), as over time, $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}^{\mathrm{T}}$ may drift from the true singular components.

3.2 Gradient Update of $\mathbf{U}$, $\mathbf{V}^{\mathrm{T}}$ with $\mathbf{\Sigma}$

Updating $\mathbf{\Sigma}$. The diagonal matrix $\mathbf{\Sigma}$, treated as a vector of dimension m, is updated at every step due to its low memory cost:

$$\mathbf{\Sigma}^{t+1} = \max(\mathbf{\Sigma}^t - \eta \nabla \mathcal{L}_{\mathbf{\Sigma}}, 0) \tag{4}$$

where η is the learning rate, and $\nabla \mathcal{L}_{\mathbf{\Sigma}}$ is the backpropagated gradient. The $\max$ function ensures that $\mathbf{\Sigma}$ values remain non-negative.

Selective Updates of $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$. Singular vectors are updated selectively using multinomial sampling:

$$\mathbf{U}_{\cdot i}^{t+1} = \frac{\mathbf{U}_{\cdot i}^{t} - \eta \nabla \mathcal{L}_{\mathbf{U}_{\cdot i}}}{|\mathbf{U}_{\cdot i}^{t} - \eta \nabla \mathcal{L}_{\mathbf{U}_{\cdot i}}|}, \quad \mathbf{V}_{\cdot i}^{t+1} = \frac{\mathbf{V}_{\cdot i}^{t} - \eta \nabla \mathcal{L}_{\mathbf{V}_{\cdot i}}}{|\mathbf{V}_{\cdot i}^{t} - \eta \nabla \mathcal{L}_{\mathbf{V}_{\cdot i}}|}, \quad \text{if } i \in S, \quad S \sim \mathrm{Multinomial}(r, \mathbf{\Sigma}) \tag{5}$$

where S contains the indices of r selected singular vectors. To preserve unit norm, normalization is applied post-update.

Enhanced Gradient Optimization. We enhance default gradients for $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$ by decoupling magnitude ($\mathbf{\Sigma}$) from direction (derivation included in Appendix D):

$$\nabla \mathcal{L}_{\mathbf{U}_{\cdot i}} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \mathbf{V}_{\cdot i} \mathbf{\Sigma}_i \quad \Rightarrow \quad \tilde{\nabla} \mathcal{L}_{\mathbf{U}_{\cdot i}} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}} \mathbf{V}_{\cdot i} \tag{6}$$

This allows singular vectors with lower singular values to retain substantial gradients.

Periodic re-SVD. To maintain orthogonality, periodic singular value decomposition is performed:

$$[\mathbf{U}^{t+1}, \mathbf{\Sigma}^{t+1}, \mathbf{V}^{t+1\mathrm{T}}] = \mathrm{SVD}(\mathbf{U}^t \mathbf{\Sigma}^t \mathbf{V}^{t\mathrm{T}}) \tag{7}$$

Re-SVD resets singular vectors, preventing degradation into a low-rank subspace. Each re-SVD defines a new **round**, while each selective update defines an **iteration**. The full algorithm is detailed in Algorithm 2. Efficient implementation of SST is detailed in Appendix A.

Table 1: **BLEU scores on IWSLT'14 for Euclidean and hyperbolic Transformers.** Some BLEU scores are zero because that training resulted in NaN losses. Notably, SST consistently outperforms other low-rank methods.

		Euclidean				Hyperbolic			
Dimension	**r**	**Full**	**LoRA**	**ReLoRA***	**SST**	**Full**	**LoRA**	**ReLoRA***	**SST**
64	8	24.27	18.08	18.12	**22.28**	25.69	17.50	0.0	**23.40**
	4		14.05	15.49	**20.27**		0.0	0.0	**23.03**
128	16	25.79	23.30	22.92	**25.12**	24.70	23.70	0.0	**25.22***
	8		20.56	20.61	**24.19**		20.81	0.0	**25.12***
	4		16.37	18.00	**22.80**		17.58	24.42	**24.60**
256	32	23.92	23.76	23.02	**23.97***	19.94	24.16*	0.0	**25.04***
	16		22.88	22.01	**23.42**		23.93*	0.0	**25.52***
	8		20.32	20.36	**22.65**		21.58*	24.02*	**24.67***
	4		16.72	17.85	**21.39**		18.72	24.08*	**24.51***

Table 2: **Validation perplexity on OpenWebText** across various model sizes of OPT and LLaMA along with the number of trainable parameters of each method.

Model	r/d_{model}	**Training Tokens**	**Full**	**LoRA**	**ReLoRA***	**SST**
OPT-125M	64/768	19.7B	23.50 (125.2M)	34.23 (50.9M)	35.80 (50.9M)	**26.98** (51.0M)
OPT-350M	64/1024	19.7B	21.78 (331.2M)	34.26 (57.5M)	39.21 (57.5M)	**27.72** (57.7M)
OPT-1.3B	64/2048	19.7B	15.10 (1.316B)	1716 (164.4M)	29.52 (164.4M)	**22.31** (164.7M)
LLaMA-130M	64/768	2.6B	20.04 (134.11M)	29.71 (60.38M)	31.33 (60.38M)	**23.35** (60.44M)
LLaMA-1.3B	128/2048	13.1B	14.54 (1.339B)	16.50 (250.71M)	17.32 (250.71M)	**14.59** (251.05M)

3.3 Why SVD Decomposition is Important

This section discusses the advantages of using SVD initialization and periodic re-SVD over the zero initialization in ReLoRA*. After each merging step, ReLoRA* resets $\mathbf{B}$ to zero, causing $\mathbf{A}$ to have zero gradients, leading to slow learning at the start of each iteration. As shown in Figure 2, ReLoRA* exhibits periodic drops in gradient norm at the start of each iteration, while SST maintains a stable gradient flow.

4 Experiments

4.1 Machine Translation

Experimental details are provided in Appendix E.1. Table 1 presents BLEU scores for IWSLT'14 across various dimensions and ranks (r). The results confirm that SST consistently outperforms other low-rank methods. On average, SST reduces the BLEU gap (defined as the BLEU score difference from full-rank training) by **66.7%** for Euclidean Transformers on IWSLT'14.

Further comparative results on the Multi30K and IWSLT'17 datasets using the standard dimensions for vanilla Euclidean transformers are documented in Table 24. Here, SST not only surpasses other low-rank methods but also demonstrates superior performance compared to full-rank training.

4.2 Natural Language Generation

Experimental details are provided in Appendix E.2.

Language modeling. Table 2 shows validation perplexity on OpenWebText across different LLM sizes. SST consistently achieves lower perplexity than LoRA and ReLoRA*, significantly reducing the PPL gap with full-rank training by **67.6%** (OPT-125M), **52.4%** (OPT-350M), **50.0%** (OPT-1.3B), **65.8%** (LLaMA-130M), and **97.4%** (LLaMA-1.3B).

Zero-shot evaluations. Table 3 presents results on 16 NLP tasks, where SST consistently outperforms other low-rank methods. On OPT-125M, SST achieves an average score of **44.6**, slightly surpassing full-rank training (**44.5**). SST also attains a **56.3%** win rate against full-rank training, demonstrating its strong generalization in zero-shot settings.

REFERENCES

R.M. Anderson and R.M. May. *Infectious Diseases of Humans: Dynamics and Control*. Infectious Diseases of Humans: Dynamics and Control. OUP Oxford, 1991. ISBN 9780198540403. URL https://books.google.com.tw/books?id=HT0--xXBguQC.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Carlo Vittorio Cannistraci and Alessandro Muscoloni. Geometrical congruence, greedy navigability and myopic transfer in complex networks and brain connectomes. *Nature Communications*, 13(1): 7308, 2022.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th IWSLT evaluation campaign. In Marcello Federico, Sebastian Stüker, and François Yvon (eds.), *Proceedings of the 11th International Workshop on Spoken Language Translation: Evaluation Campaign*, pp. 2–17, Lake Tahoe, California, December 4-5 2014. URL https://aclanthology.org/2014.iwslt-evaluation.1.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pp. 2–14, Tokyo, Japan, December 14-15 2017. International Workshop on Spoken Language Translation. URL https://aclanthology.org/2017.iwslt-1.1.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5672–5686, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.389. URL https://aclanthology.org/2022.acl-long.389.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, 2017. doi: 10.1109/IJCNN.2017.7966217.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Ning Ding, Xingtai Lv, Qiaosen Wang, Yulin Chen, Bowen Zhou, Zhiyuan Liu, and Maosong Sun. Sparse low-rank adaptation of pre-trained language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=jxgz7FEqWq.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pp. 70–74. Association for Computational Linguistics, 2016. doi: 10.18653/v1/W16-3210. URL http://www.aclweb.org/anthology/W16-3210.

Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, and Erich Elsen. Rigging the lottery: Making all tickets winners. In *International Conference on Machine Learning*, pp. 2943–2952. PMLR, 2020.

Octavian Ganea, Gary Becigneul, and Thomas Hofmann. Hyperbolic neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/dbab2adc8f9d078009ee3fa810bea142-Paper.pdf.

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. A framework for few-shot language model evaluation, 12 2023. URL https://zenodo.org/records/10256836.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate, 2022.

Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. Hyperbolic attention networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxHsjRqFQ.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017. doi: 10.18653/v1/P17-4012. URL https://doi.org/10.18653/v1/P17-4012.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.243. URL https://aclanthology.org/2021.emnlp-main.243.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012*, Proceedings of the International Conference on Knowledge Representation and Reasoning, pp. 552–561. Institute of Electrical and Electronics Engineers Inc., 2012. ISBN 9781577355601. 13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012 ; Conference date: 10-06-2012 Through 14-06-2012.

Vladislav Lialin, Sherin Muckatira, Namrata Shivagunde, and Anna Rumshisky. ReloRA: High-rank training through low-rank updates. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=DLJznSp6X3.

Qi Liu, Maximilian Nickel, and Douwe Kiela. Hyperbolic graph neural networks. *Advances in neural information processing systems*, 32, 2019.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. Gpt understands, too. *arXiv:2103.10385*, 2021.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a. URL https://openreview.net/forum?id=6ZBHIEtdP4.

Xiangdi Meng, Damai Dai, Weiyao Luo, Zhe Yang, Shaoxiang Wu, Xiaochen Wang, Peiyi Wang, Qingxiu Dong, Liang Chen, and Zhifang Sui. Periodiclora: Breaking the low-rank bottleneck in lora optimization, 2024b. URL https://arxiv.org/abs/2402.16141.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Decebal Constantin Mocanu, Elena Mocanu, Peter Stone, Phuong H Nguyen, Madeleine Gibescu, and Antonio Liotta. Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. *Nature communications*, 9(1):1–12, 2018.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In Kevin Knight, Ani Nenkova, and Owen Rambow (eds.), *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 839–849, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1098. URL https://aclanthology.org/N16-1098.

Alessandro Muscoloni, Josephine Maria Thomas, Sara Ciucci, Ginestra Bianconi, and Carlo Vittorio Cannistraci. Machine learning meets complex networks via coalescent embedding in the hyperbolic space. *Nature communications*, 8(1):1615, 2017.

Galileo Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. *CoRR*, abs/1912.01703, 2019. URL http://arxiv.org/abs/1912.01703.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *arXiv preprint arXiv:1907.10641*, 2019.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Alexandru Tifrea, Gary Becigneul, and Octavian-Eugen Ganea. Poincare glove: Hyperbolic word embeddings. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Ske5r3AqK7.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023b.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 3274–3287, 2023.

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

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/4496bf24afe7fab6f046bf4923da8de6-Paper.pdf.

Wenhan Xia, Chengwei Qin, and Elad Hazan. Chain of lora: Efficient fine-tuning of language models via residual learning, 2024.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://arxiv.org/abs/1708.07747.

Geng Yuan, Xiaolong Ma, Wei Niu, Zhengang Li, Zhenglun Kong, Ning Liu, Yifan Gong, Zheng Zhan, Chaoyang He, Qing Jin, et al. Mest: Accurate and fast memory-economic sparse training framework on the edge. *Advances in Neural Information Processing Systems*, 34:20838–20850, 2021.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Yingtao Zhang, Jialin Zhao, Wenjing Wu, Alessandro Muscoloni, and Carlo Vittorio Cannistraci. Epitopological learning and cannistraci-hebb network shape intelligence brain-inspired theory for ultra-sparse advantage in deep learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=iayEcORsGd.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient llm training by gradient low-rank projection, 2024.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices, 2023.

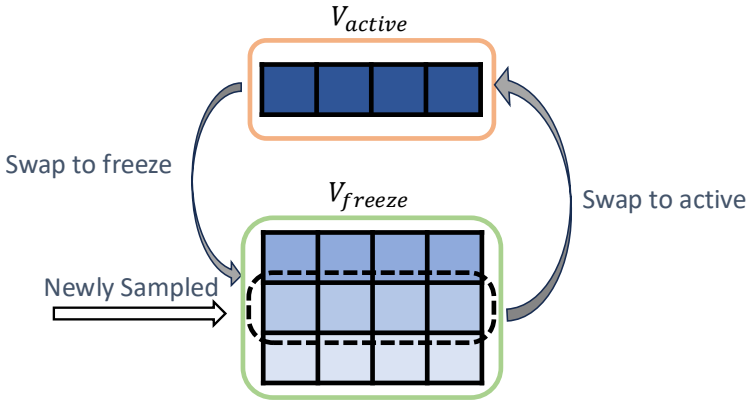

Figure 3: **Illustration of the memory-efficient implementation for SST.** After each sampling step, the sampled vectors are swapped with the active vectors from the previous iteration.

Algorithm 1 ReLoRA*

input Initial weight $\mathbf{W}$ of each layer; total iteration T_1; iteration interval T_2
  for $t_1 = 0, \ldots, T_1 - 1$ **do**
    Initializing: Initialize $\mathbf{B}$ and $\mathbf{A}$ for each layer.
    Subtracting: Subtract $\mathbf{B}$ and $\mathbf{A}$ from $\mathbf{W}$ to maintain the original model output, $\mathbf{W} = \mathbf{W} - \mathbf{BA}$
    Updating: Update $\mathbf{B}$ and $\mathbf{A}$ for T_2 steps while keeping $\mathbf{W}$ frozen.
    Merging: Merge $\mathbf{B}$ and $\mathbf{A}$ back to $\mathbf{W}$, updating $\mathbf{W} = \mathbf{W} + \mathbf{BA}$.
  end for

Table 3: **Zero-shot evaluations** on the same 16 NLP tasks featured in the OPT article (Zhang et al., 2022). Except for the ReCoRD task, which uses F1 score, all other tasks are evaluated using accuracy, with values presented as percentages. Mean scores in bold represent superior performance among the low-rank methods. Additionally, we include the win percentage (including ties) for each low-rank method compared to the full-rank training.

	OPT-125M				**OPT-350M**				**OPT-1.3B**			
	Full	**LoRA**	**ReLoRA***	**SST**	**Full**	**LoRA**	**ReLoRA***	**SST**	**Full**	**LoRA**	**ReLoRA***	**SST**
ARC (Challenge)	21.2	22.9	21.1	21.3	22.0	22.3	21.3	21.1	24.6	24.2	22.9	21.5
ARC (Easy)	35.8	34.2	33.9	34.3	35.9	32.3	33.0	35.7	43.2	26.1	35.9	37.8
BoolQ	59.5	54.2	60.8	62.0	53.6	56.2	62.2	57.7	57.7	37.8	61.4	59.5
CB	51.8	48.2	28.6	48.2	44.6	44.6	33.9	41.1	59.0	41.1	37.5	42.9
COPA	67.0	61.0	57.0	66.0	69.0	61.0	59.0	60.0	70.0	51.0	68.0	65.0
HellaSwag	27.7	26.5	27.1	26.9	28.4	26.6	26.9	27.5	35.0	26.1	27.2	28.1
MultiRC	55.4	57.2	55.9	57.2	52.0	52.6	56.4	57.0	56.8	42.8	57.7	56.9
OpenBookQA	24.6	24.6	23.6	26.2	26.4	24.2	23.0	25.2	29.0	27.0	24.8	25.0
PIQA	58.7	57.2	56.3	58.3	59.2	56.9	56.9	59.0	64.0	50.3	57.1	59.1
ReCoRD	16.7	17.5	22.6	18.5	19.4	17.6	19.0	23.2	13.7	17.6	23.0	18.1
RTE	50.5	56.7	53.1	53.4	52.0	49.1	54.9	50.2	51.6	52.7	52.0	53.8
StoryCloze	55.8	53.8	53.6	54.5	57.2	53.7	53.0	54.6	61.1	49.7	54.0	56.1
WIC	49.8	51.4	50.0	50.0	50.5	50.0	50.0	50.2	50.3	50.0	50.0	50.0
Winograd	52.0	48.7	50.6	50.6	55.0	51.7	50.2	51.3	55.7	50.9	52.4	55.3
Winogrande	49.1	49.2	50.7	50.1	50.7	50.3	50.8	52.0	51.1	47.9	50.0	49.1
WSC	36.5	38.5	36.5	36.5	36.5	37.5	36.5	36.5	39.4	63.5	36.5	36.5
Mean	44.5	43.8	42.6	**44.6**	44.5	42.9	42.9	**43.9**	47.6	41.2	44.4	**44.7**
Win Percentage	-	50.0	43.8	56.3	-	31.3	31.3	31.3	-	18.8	25.0	25.0

A Memory-Efficient Implementation for SST

To achieve similar memory reduction as LoRA, SST stores optimizer states for all $\mathbf{\Sigma}$ and only for the vectors sampled in each iteration from $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$. However, standard implementations of Adam optimizer (Kingma & Ba, 2014) in PyTorch (Paszke et al., 2019) do not support sparse optimizer

states. To address this, we partition $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$ into active and frozen segments. Only active segments store the optimizer states, where $\mathbf{U}_{\text{active}} \in \mathbb{R}^{m \times r}$ and $\mathbf{V}^{\mathrm{T}}_{\text{active}} \in \mathbb{R}^{r \times n}$. The frozen segments, $\mathbf{U}_{\text{freeze}}$ and $\mathbf{V}^{\mathrm{T}}_{\text{freeze}}$, do not store optimizer states. Vectors newly sampled from the frozen segments are swapped with unsampled vectors in the active segments (illustrated in Figure 3). This approach enables SST to function as a time-sharing operating system, effectively balancing resource allocation among the vectors in $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$.

B Algorithm of Sparse Spectral Training

Algorithm 2 Sparse Spectral Training (SST)

input Dataset D; total round T_1; number of iterations T_2; iteration interval T_3

Use Kaiming initialization to initialize origin model's weight $\mathbf{W}_k^{(0)}$, $k = 1, ..., n$, where n is the number of linear layers.

Replace origin model's weight with SVD decomposition

$$[\mathbf{U}_k^{(t_1,0)}, \mathbf{\Sigma}_k^{(t_1,0)}, \mathbf{V}_k^{(t_1,0)^{\mathrm{T}}}] = \text{SVD}(\mathbf{W}_k^{(t_1)})$$

for $t_1 = 0, \ldots, T_1 - 1$ **do**

  for $t_2 = 0, \ldots, T_2 - 1$ **do**

    $I_k = \{1, 2, \ldots, m\}$ be the set of all possible indices of singular vectors

$$S_k^{(t_1,t_2)} \subseteq I_k, \quad S_k^{(t_1,t_2)} \sim \text{Multinomial}(r, \mathbf{\Sigma}_k^{(t_1,t_2 \times T_3)})$$

    for $t_3 = 0, \ldots, T_3 - 1$ **do**

      Represent $t = t_2 \times T_3 + t_3$;

      Sample a mini-batch from D and compute the forward pass by Eq.3 and compute the gradient $\nabla \mathcal{L}$;

      Update $\mathbf{\Sigma}_k^{(t_1,t+1)} = \max(\mathbf{\Sigma}_k^{(t_1,t)} - \eta \nabla \mathcal{L}_{\mathbf{\Sigma}_k}, 0)$

      Update

$$\mathbf{U}_{k,\cdot i}^{(t_1,t+1)} = \frac{\mathbf{U}_{k,\cdot i}^{(t_1,t)} - \eta \tilde{\nabla} \mathcal{L}_{\mathbf{U}_{k,\cdot i}}}{|\mathbf{U}_{k,\cdot i}^{(t_1,t)} - \eta \tilde{\nabla} \mathcal{L}_{\mathbf{U}_{k,\cdot i}}|}, \quad \mathbf{V}_{k,\cdot i}^{(t_1,t+1)} = \frac{\mathbf{V}_{k,\cdot i}^{(t_1,t)} - \eta \tilde{\nabla} \mathcal{L}_{\mathbf{V}_{k,\cdot i}}}{|\mathbf{V}_{k,\cdot i}^{(t_1,t)} - \eta \tilde{\nabla} \mathcal{L}_{\mathbf{V}_{k,\cdot i}}|}, \quad \text{if } i \in S_k^{(t_1,t_2)}$$

      where $\mathbf{U}_{k,\cdot i}$ means column vector i of $\mathbf{U}_k$

    end for

  end for

  Reinitialize with new SVD decomposition

$$[\mathbf{U}_k^{(t_1+1,0)}, \mathbf{\Sigma}_k^{(t_1+1,0)}, \mathbf{V}_k^{(t_1+1,0)^{\mathrm{T}}}] = \text{SVD}(\mathbf{U}_k^{(t_1,T_2 \times T_3 - 1)} \mathbf{\Sigma}_k^{(t_1,T_2 \times T_3 - 1)} \mathbf{V}_k^{(t_1,T_2 \times T_3 - 1)^{\mathrm{T}}})$$

end for

C Experiments on Larger Datasets and Hyperparameter Tuning

To further evaluate the performance of SST, we conducted additional experiments using larger datasets and varied hyperparameter settings. Specifically, we pre-trained LLaMA-130M on the C4 dataset (Raffel et al., 2020), which is about 25 times larger than OpenWebText. We also compared the performance of SST, LoRA, and ReLoRA* under two different learning rates.

Table 4 presents the validation perplexity (PPL) results for LLaMA-130M on both C4 and OpenWebText. The results show that SST consistently outperforms other low-rank methods, achieving lower perplexity across all configurations.

Each method was trained with 2.6 billion tokens. The learning rate of $1\text{e}{-3}$ for full-rank training aligns with the configuration used in the ReLoRA article. For consistency, we applied the same learning rates ($lr = 1\text{e}{-3}$ and $lr = 3\text{e}{-3}$) across LoRA, ReLoRA*, and SST.

Table 4: **Validation perplexity on C4 and OpenWebText** for LLaMA-130M with different learning rates. Bold values indicate the lowest PPL among all low-rank methods.

Dataset	Model	r/d	Full (lr=1e-3)	lr=1e-3			lr=3e-3		
				LoRA	ReLoRA*	SST	LoRA	ReLoRA*	SST
C4	LLaMA-130M	64/768	24.91	35.91	37.34	32.13	30.75	133.06	**29.79**
OpenWebText	LLaMA-130M	64/768	20.04	29.71	31.33	25.89	795.24	230.43	**23.35**

SST consistently achieves lower perplexity than LoRA and ReLoRA* at the same learning rate. Notably, with $lr = 3\text{e}{-3}$, SST surpasses all other low-rank methods, reducing the perplexity gap by **16.4%** on C4 and **65.8%** on OpenWebText. These findings highlight SST's effectiveness and robustness on larger datasets and varied learning rate configurations.

D Proof of Gradient of Sparse Spectral Layer

We can express the differential of $\mathbf{W}$ as the sum of differentials:

$$\mathrm{d}\mathbf{W} = \mathrm{d}\mathbf{U}\,\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} + \mathbf{U}\,\mathrm{d}\mathbf{\Sigma}\,\mathbf{V}^{\mathrm{T}} + \mathbf{U}\mathbf{\Sigma}\,\mathrm{d}\mathbf{V}^{\mathrm{T}} \tag{8}$$

We have chain rule for the gradient of $\mathbf{W}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}}\frac{\partial \mathbf{h}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}}\mathbf{x}^{\mathrm{T}} \tag{9}$$

$$\begin{aligned}
\mathrm{d}\mathcal{L} &= \frac{\partial \mathcal{L}}{\partial \mathbf{W}} : \mathrm{d}\mathbf{W} \\
&= \frac{\partial \mathcal{L}}{\partial \mathbf{W}} : \mathrm{d}\mathbf{U}\,\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} + \frac{\partial \mathcal{L}}{\partial \mathbf{W}} : \mathbf{U}\,\mathrm{d}\mathbf{\Sigma}\,\mathbf{V}^{\mathrm{T}} + \frac{\partial \mathcal{L}}{\partial \mathbf{W}} : \mathbf{U}\mathbf{\Sigma}\,\mathrm{d}\mathbf{V}^{\mathrm{T}} \\
&= \frac{\partial \mathcal{L}}{\partial \mathbf{W}}\mathbf{V}\mathbf{\Sigma} : \mathrm{d}\mathbf{U} + \mathbf{U}^{\mathrm{T}}\frac{\partial \mathcal{L}}{\partial \mathbf{W}}\mathbf{V} : \mathrm{d}\mathbf{\Sigma} + \mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}\frac{\partial \mathcal{L}}{\partial \mathbf{W}} : \mathrm{d}\mathbf{V}^{\mathrm{T}}
\end{aligned}$$

where : is the Frobenius inner product. So we have the gradient of $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}^{\mathrm{T}}$:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}}\mathbf{V}\mathbf{\Sigma}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{V}^{\mathrm{T}}} = \mathbf{\Sigma}\mathbf{U}^{\mathrm{T}}\frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{\Sigma}} = \mathbf{U}^{\mathrm{T}}\frac{\partial \mathcal{L}}{\partial \mathbf{W}}\mathbf{V} \tag{10}$$

In vector perspective, for the i^{th} vector, it is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}_{\cdot i}} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}}\mathbf{V}_{\cdot i}\mathbf{\Sigma}_i, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{V}_{\cdot i}} = \mathbf{\Sigma}_i\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{\mathrm{T}}}\mathbf{U}_{\cdot i}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{\Sigma}_i} = \mathbf{U}_{\cdot i}{}^{\mathrm{T}}\frac{\partial \mathcal{L}}{\partial \mathbf{W}}\mathbf{V}_{\cdot i} \tag{11}$$

where $\mathbf{U}_{\cdot i}$ means the i^{th} column vector of $\mathbf{U}$, and $\mathbf{\Sigma}_i$ is the i^{th} value of the diagonal matrix $\mathbf{\Sigma}$.

E Experiment Details

To validate our Sparse Spectral Training (SST) approach, we conducted experiments on both Euclidean and hyperbolic neural networks, demonstrating the generalization of SST across various neural network architectures and embedding geometries.

We compared SST with full-rank training, LoRA, and ReLoRA*. The key distinctions between ReLoRA* and ReLoRA (Lialin et al., 2024) is that ReLoRA includes a full-rank training as "warm start", which prevents it from being an end-to-end memory-efficient pre-training method.

For all low-rank methods, all linear layers in the baseline models were replaced by low-rank layers. Hyperparameters and implementation details are provided in Appendix E.

As discussed in Section 2.3, the comparison between SST and the contemporaneous work GaLore (Zhao et al., 2024) is provided in Appendix G, as GaLore is unstable during OPT pre-training with $r = 64$. We highlight SST's superior performance across all of our experiment settings. Ablation studies are documented in Appendix H, and a detailed analysis of memory consumption and training time can be found in Appendix I. Additionally, an experiment on image classification tasks is included in Appendix J.

E.1 Machine Translation

We employ the vanilla transformer (Vaswani et al., 2017) as the Euclidean transformer and HyboNet (Chen et al., 2022) as the hyperbolic transformer. Our experiments include three widely-used machine translation datasets: IWSLT'14 English-to-German (Cettolo et al., 2014), IWSLT'17 German-to-English (Cettolo et al., 2017), and Multi30K German-to-English (Elliott et al., 2016). For IWSLT'14, the hyperparameters are aligned with those from HyboNet.

E.2 Natural Language Generation

Language modeling. We utilize the OPT (Zhang et al., 2022) and LLaMA (Touvron et al., 2023a) architecture as the baseline for our language generation experiments. For LLaMA, we follow the experiment setup from (Zhao et al., 2024). All models are pre-trained on OpenWebText (Gokaslan & Cohen, 2019), an open-source reproduction of OpenAI's WebText. We applied a rank of $r = 64$ for all OPT models and LLaMA-130M, and $r = 128$ for LLaMA-1.3B.

Zero-shot evaluations. Each pretrained model performs zero-shot evaluations on all 16 NLP tasks used in the OPT article (Zhang et al., 2022), including ARC Easy and Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), StoryCloze (Mostafazadeh et al., 2016), SuperGLUE (Wang et al., 2019), WinoGrad (Levesque et al., 2012), and WinoGrande (Sakaguchi et al., 2019). Evaluations are conducted using the LM Evaluation Harness framework (Gao et al., 2023). Except for the ReCoRD task, which uses F1 score, all other tasks are evaluated using accuracy.

E.3 Implementation Details for SST

Sampling of $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$. In our experiments, we employ a more exploratory approach when sampling $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$:

$$p(i) = \frac{1}{2}\left(\frac{1}{m} + \frac{\mathbf{\Sigma}_i}{\sum_j \mathbf{\Sigma}_j}\right) \tag{12}$$

where $p(i)$ is the possibility to sample index i vector of $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$. This adjustment ensures that vectors associated with lower singular values still have a substantial likelihood of being sampled, preventing their probabilities from becoming excessively low and promoting a more balanced exploration across the spectral components.

Optimizer state reset and warmup. Before each iteration, Sparse Spectral Training (SST) resets all optimizer states for $\mathbf{U}$, $\mathbf{V}^{\mathrm{T}}$ and $\mathbf{\Sigma}$. For example, for optimizers like Adam, this involves clearing the first and second moments as well as the timestep. Consequently, a brief warmup period is essential at the beginning of each iteration to accommodate the reset states. This warmup period is typically 20 steps, guided by the exponential decay rate β used in the Adam optimizer.

Hyperbolic SST. The formula of hyperbolic linear layer in (Chen et al., 2022) is:

$$\mathbf{h} = f_{\mathbf{x}}(\mathbf{M})\mathbf{x} = \begin{bmatrix} \frac{\sqrt{\|\mathbf{W}\mathbf{x}\|_2 - \frac{1}{K}}}{\mathbf{v}^\top \mathbf{x}} \mathbf{v}^\top \\ \mathbf{W} \end{bmatrix} \mathbf{x} = \begin{bmatrix} \sqrt{\|\mathbf{W}\mathbf{x}\|_2 - \frac{1}{K}}\mathbf{v}^\top \\ \mathbf{W}\mathbf{x} \end{bmatrix} \tag{13}$$

where $\mathbf{v} \in \mathbb{R}^{n+1}$, $\mathbf{W} \in \mathbb{R}^{m \times (n+1)}$ and K is the curvature. The formula of Hyperbolic SST is:

$$h = \begin{bmatrix} \sqrt{\|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}\mathbf{x}\|_2 - \frac{1}{K}}\mathbf{v}^{\top} \\ \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}\mathbf{x} \end{bmatrix} \tag{14}$$

E.4 Hyperparameters of Machine Translation

IWSLT'14. The hyperparameters can be found in Table 5. We employ the same codebase and hyperparameters as those used in HyboNet (Chen et al., 2022), which is derived from OpenNMT-py (Klein et al., 2017). For all methods, last checkpoint is utilized for evaluation. Beam search, with a beam size of 2, is employed to optimize the evaluation process. Experiments were conducted on one A100 GPU.

For SST, iteration interval (T_3) is set to 200. Each iteration begins with a warmup phase lasting 20 steps. The number of iterations per round (T_2) is determined by the formula $T_2 = d/r$, where d represents the embedding dimension and r denotes the rank used in SST.

Table 5: **Hyperparameters on IWSLT'14** for Euclidean and hyperbolic Transformer.

Hyper-parameter	Euclidean	Hyperbolic
Embedding Dimension	64, 128, 256	64, 128, 256
Feed-forward Dimension	256, 512, 1024	256, 512, 1024
Batch Size	10240 tokens	10240 tokens
Gradient Accumulation Steps	4	4
Training Steps	40000	40000
Dropout	0.0	0.1
Attention Dropout	0.1	0.1
Max Gradient Norm	-	0.5
Warmup Steps	6000	6000
Decay Method	noam	noam
Label Smoothing	0.1	0.1
Layer Number	6	6
Head Number	4	4
Learning Rate	5	2
Optimizer	Adam	rAdam

Multi30K and IWSLT'17. The hyperparameters can be found in Table 6. Because of overfitting, model checkpoint with lowest validation loss is utilized for evaluation. A larger learning rate (0.0003) is used for low rank parameters ($\mathbf{U}$, $\mathbf{V}^{\mathrm{T}}$ and $\mathbf{\Sigma}$ for SST, $\mathbf{B}$ and $\mathbf{A}$ for LoRA and ReLoRA*. Experiments were conducted on one A100 GPU.

For SST, interation interval (T_3) is set to 200 for Multi30K and 400 for IWSLT'17. Each iteration begins with a warmup phase lasting 20 steps. The number of iterations per round (T_2) is determined by the formula $T_2 = d/r$, where d represents the embedding dimension and r denotes the rank used in SST.

E.5 Hyperparameters of Natural Language Generation

Hyperparameters for OPT. The hyperparameters for OPT are detailed in Table 7. We employ a linear warmup of 2000 steps followed by a stable learning rate, without decay. A larger learning rate (0.001) is used for only low rank parameters ($\mathbf{U}$, $\mathbf{V}^{\mathrm{T}}$ and $\mathbf{\Sigma}$ for SST, $\mathbf{B}$ and $\mathbf{A}$ for LoRA and ReLoRA*. The total training tokens for each experiment is 19.7B, roughly 2 epochs of OpenWebText. Distributed training is facilitated using the Accelerate (Gugger et al., 2022) library across four A100 GPUs on a Linux server.

For SST, interation interval (T_3) is set to 200. Each iteration begins with a warmup phase lasting 20 steps. The number of iterations per round (T_2) is determined by the formula $T_2 = d/r$, where d represents the embedding dimension and r denotes the rank used in SST.

Table 6: **Hyperparameters on Multi30K and IWSLT'17** for vanilla Transformer.

Hyper-parameter	Multi30K	IWSLT'17
Embedding Dimension	512	512
Feed-forward Dimension	2048	2048
Batch Size	128 sentences	128 sentences
Gradient Accumulation Steps	1	1
Training Steps	100000	150000
Dropout	0.1	0.1
Decay Method	constant	constant
Layer Number	6	6
Head Number	8	8
Learning Rate	0.0001	0.0001
Weight Decay	1	0.1
Optimizer	AdamW	AdamW

Table 7: **Hyperparameters for OPT Models**

Hyper-parameter	OPT-125M	OPT-350M	OPT-1.3B
Embedding Dimension	768	512 (project to 1024)	2048
Feed-forward Dimension	3072	4096	8192
Global Batch Size	240	240	240
Sequence Length	2048	2048	2048
Training Steps	40000	40000	40000
Learning Rate	0.0001	0.0001	0.0001
Warmup Steps	2000	2000	2000
Optimizer	AdamW	AdamW	AdamW
Layer Number	12	24	24
Head Number	12	16	32

Hyperparameters for LLaMA. The hyperparameters for LLaMA are detailed in Table 8. We follow the same experiment setup from (Zhao et al., 2024). We employ a linear warmup of 2000/10000 steps followed by a cosine decay. For LLaMA-130M, the learning rates for LoRA, ReLoRA*, and SST are selected from {1e-3, 3e-3} based on the lowest PPL observed in Table 4. For LLaMA-1.3B, the learning rates for LoRA, ReLoRA*, and SST are fixed at 1e-3. The learning rates for full-rank training are set to 1e-3 for LLaMA-130M and 4e-4 for LLaMA-1.3B, consistent with the configuration in the ReLoRA article.

For SST, interation interval (T_3) is set to 200. Each iteration begins with a warmup phase lasting 20 steps. The number of iterations per round (T_2) is determined by the formula $T_2 = d/r$, where d represents the embedding dimension and r denotes the rank used in SST.

Table 8: **Hyperparameters for LLaMA Models**

Hyper-parameter	LLaMA-130M	LLaMA-1.3B
Embedding Dimension	768	2048
Feed-forward Dimension	2048	5461
Global Batch Size	512	512
Sequence Length	256	256
Training Steps	20000	100000
Learning Rate	0.001	0.0004
Warmup Steps	2000	10000
Optimizer	Adam	Adam
Layer Number	12	24
Head Number	12	32

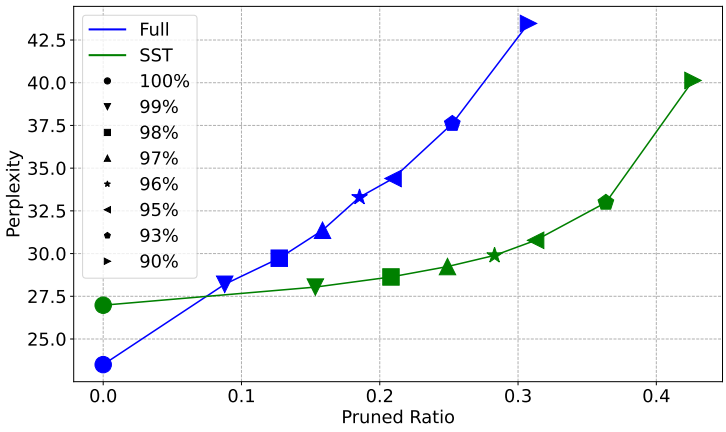

Figure 4: **Singular Value Pruning.** We conduct singular value pruning on full-rank and SST pretrained OPT-125M model. After performing singular value decomposition on weight matrices, we preserve the top k singular values so that the cumulative sum of preserved singular values ranges from $[100\%, 99\%, 98\%, ..., 93\%, 90\%]$ of the original cumulative sum. The pruned ratio of singular values is plotted along the x-axis.

E.6 Hyperparameters of Hyperbolic Graph Neural Networks

We use HyboNet (Chen et al., 2022) as full-rank model, with same hyperparameters as those used in HyboNet. Experiments were conducted on one A100 GPU.

For SST, interation interval (T_3) is set to 100. Each iteration begins with a warmup phase lasting 100 steps. The number of iterations per round (T_2) is determined by the formula $T_2 = d/r$, where d represents the embedding dimension and r denotes the rank used in SST.

We set dropout rate to 0.5 for the LoRA and SST methods during the node classification task on the Cora dataset. This is the only one deviation from the HyboNet configuration.

F Singular Value Pruning

We further conduct an analysis study of the potential for using SST model for further compression. The results, as shown in Figure 4, indicate that the SST model retains lower perplexity across a wider range of pruning ratios compared to the full-rank model. This suggests that the SST method effectively concentrates the informational content of the weights into fewer singular values, making it more suitable for further compression.

This enhanced performance underscores the potential of SST in maintaining essential model characteristics even under significant compression, making it a promising approach for developing lightweight yet powerful language models for inference.

G Evaluating SST and GaLore: Complementary Approaches to Memory Efficiency

Recently, a new approach named Gradient Low-Rank Projection (GaLore) (Zhao et al., 2024) has been proposed to address the memory challenges associated with pre-training large language models. GaLore, by implementing a memory-efficient gradient projection method.

Using the released code of GaLore[1], we conducted comparative experiments on the IWSLT'14 dataset with Transformer models, employing the same configurations as other low-rank methods. We set the scale factor $\alpha = 1$ in these experiments because $\alpha = 0.25$, which is used in the article, performs

[1] https://github.com/jiaweizzhao/GaLore

Table 9: **The BLEU score on IWSLT'14 for Euclidean Transformer, compared with GaLore.** Values highlighted in bold represent the highest performance among the low rank methods, while those marked with an "*" denote performance that exceeds that of the full-rank variants.

Dimension	r	Full	GaLore	SST
64	8	24.27	18.08	**22.28**
	4		14.07	**20.27**
128	16	25.79	23.43	**25.12**
	8		19.71	**24.19**
	4		16.01	**22.80**
256	32	23.92	**24.01***	23.97*
	16		22.82	**23.42**
	8		20.12	**22.65**
	4		15.94	**21.39**

much worse than $\alpha = 1$. As illustrated in Table 9, SST method consistently outperformed GaLore across various model dimensions and ranks, except for $d = 256$, $r = 32$.

In addition, we evaluated validation perplexity on the OpenWebText dataset with OPT-125M and OPT-350M models. We tested GaLore with scale factor $\alpha = 0.25$ (used in GaLore article) and $\alpha = 1$. As shown in Table 10, SST outperformed GaLore at both settings of α on OPT-125M. Since $\alpha = 1$ had better results than $\alpha = 0.25$ on OPT-125M, we used $\alpha = 1$ for training GaLore on OPT-350M. Initially, GaLore trained normally on OPT-350M, but around step 6127, the training loss suddenly increased from approximately 4 to 7 within a few steps, resulting in a very high final perplexity for the GaLore OPT-350M, as shown in Table 10. Training of GaLore on OPT-1.3B is still ongoing, and we will update the results as soon as they are available. Zero-shot evaluations comparing SST with GaLore are presented in Table 11, which also demonstrate SST's superior performance.

Here, we discuss our guess on why SST may have an advantage over GaLore on low-rank settings. GaLore utilizes a projection matrix $P_t \in \mathbb{R}^{m \times r}$ derived from the singular value decomposition (SVD) of a single step's gradient. Only one step's gradient may introduce noise due to data sampling variability. Conversely, SST employs $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$ as projection matrices, which are initialized and reinitialized with the SVD of $\mathbf{W}$. $\mathbf{W}$ could be seemed as the momentum of gradient of $\mathbf{W}$, less noisy than one step's gradient. Furthermore, SST updates all $\mathbf{\Sigma}$ values, regardless of r, making it more robust as r decreases.

Table 10: **Validation perplexity, compared with GaLore** on OpenWebText dataset with OPT-125M and OPT-350M, along with the number of trainable parameters of each method. $r = 64$. Values highlighted in bold represent the highest performance among the low rank methods.

	Full	GaLore $\alpha = 1$	GaLore $\alpha = 0.25$	SST
OPT-125M	23.50 (125.2M)	32.17 (45.6M)	37.08 (45.6M)	**26.98** (51.0M)
OPT-350M	21.78 (331.2M)	1994 (43.4M)	-	**27.72** (57.7M)

H Ablation Study

Impact of $\mathbf{\Sigma}$ updates. We conduct an ablation study to evaluate the impact of various components and configurations within SST on the IWSLT'14 using a Euclidean Transformer with a dimension of 128 and rank r of 4. The results of this study are summarized in Table 12, which highlights the contributions of specific elements to the overall performance measured in BLEU score.

One variation tested involves changing the update mechanism for $\mathbf{\Sigma}$. Instead of updating all $\mathbf{\Sigma}$, only sampled $\mathbf{\Sigma}$ are updated, same as update for $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$. This modification results in a lower BLEU score of 22.40, indicating that full updates of $\mathbf{\Sigma}$ contribute positively to the model's performance.

Table 11: **Zero-shot evaluations, compared with GaLore** with same tasks as Table 3. Mean scores in bold represent superior performance among the low-rank methods. Win percentage (including ties) for each low-rank method is compared to the full-rank training.

	OPT-125M				**OPT-350M**		
	Full	**GaLore** $\alpha = 1$	**GaLore** $\alpha = 0.25$	**SST**	**Full**	**GaLore** $\alpha = 1$	**SST**
ARC (Challenge)	21.2	21.2	20.4	21.3	22.0	25.7	21.1
ARC (Easy)	35.8	33.7	32.8	34.3	35.9	25.7	35.7
BoolQ	59.5	61.8	62.2	62.0	53.6	37.8	57.7
CB	51.8	37.5	35.7	48.2	44.6	41.1	41.1
COPA	67.0	64.0	58.0	66.0	69.0	52.0	60.0
HellaSwag	27.7	27.0	26.6	26.9	28.4	26.2	27.5
MultiRC	55.4	57.2	54.8	57.2	52.0	42.8	57.0
OpenBookQA	24.6	23.6	24.6	26.2	26.4	27.8	25.2
PIQA	58.7	57.1	56.4	58.3	59.2	50.5	59.0
ReCoRD	16.7	15.0	16.4	18.5	19.4	17.5	23.2
RTE	50.5	51.6	56.0	53.4	52.0	52.7	50.2
StoryCloze	55.8	53.5	52.8	54.5	57.2	49.7	54.6
WIC	49.8	50.0	50.0	50.0	50.5	50.0	50.2
Winograd	52.0	50.9	52.4	50.6	55.0	50.2	51.3
Winogrande	49.1	51.7	48.4	50.1	50.7	49.4	52.0
WSC	36.5	36.5	36.5	36.5	36.5	63.5	36.5
Mean	44.5	43.3	42.8	**44.6**	44.5	41.4	**43.9**
Win Percentage	-	43.8	37.5	56.3	-	25.0	31.3

Initialization method. We experiment with a configuration similar to the ReLoRA*, where $\mathbf{h} = (\mathbf{W} + \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}})\mathbf{x}$, with $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$ randomly initialized and $\mathbf{\Sigma}$ initialized to zero. After each round, $\mathbf{U}$, $\mathbf{V}^{\mathrm{T}}$ and $\mathbf{\Sigma}$ are reinitialized. This setup significantly reduces the BLEU score to 16.03, which is similar to the performance of LoRA (16.37) and ReLoRA* (18.00). This demonstrates that the most important feature of SST is that instead of randomly initialized, SST uses SVD of $\mathbf{W}$ as the initialization of $\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$, which is aligned with our analysis in section 3.3.

Table 12: **Ablation Study** on IWSLT'14 dataset with Euclidean Transformer. Dimension is 128 and r is 4.

	BLEU
LoRA	16.37
ReLoRA*	18.00
SST - Instead of update all $\mathbf{\Sigma}$, only update sampled $\mathbf{\Sigma}$	22.40
SST - Use formula similar as ReLoRA*: $\mathbf{h} = (\mathbf{W} + \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}})\mathbf{x}$. ($\mathbf{U}$ and $\mathbf{V}^{\mathrm{T}}$ random initialized, and $\mathbf{\Sigma}$ zero initialized)	16.03
SST	22.80

Impact of iteration interval (T_3). We also conducted additional experiments to study the impact of varying iteration interval T_3 (sampling period). All methods were trained on a vanilla Transformer model with a hidden dimension of 64 and $r = 8$ on the IWSLT'14 dataset. In the original setup (Table 1), T_3 was set to 200 steps per iteration.

Table 13: **Impact of iteration interval** (T_3) on BLEU scores for IWSLT'14.

Steps per Iteration T_3	800	400	200	100	50	25	10
BLEU Score	21.85	23.64	22.47	22.49	22.60	22.46	22.25

As shown in Table 13, both excessively large and small values of T_3 result in decreased performance. A large T_3 may cause SST degrade to LoRA, while a small T_3 leads to frequent resets of the optimizer's momentum, thereby affecting convergence.

Impact of Number of Iterations. We conducted an additional experiment on the IWSLT'14 dataset using a vanilla Transformer to evaluate the impact of the number of iterations per round, with a model dimension of 64 and $r = 8$. The results are summarized in Table 14:

Table 14: **Impact of number of iterations** per round on BLEU scores for IWSLT'14.

Number of Iterations per Round	1	2	4	8	16	32
BLEU Score	22.28	22.21	22.24	22.28	22.30	22.37

The results indicate that different numbers of iterations yield comparable performance. In our experiments, this hyperparameter was not tuned; instead, we fixed it to d/r.

Sampling Mechanisms. To evaluate the impact of different sampling mechanisms on the performance of SST, we conducted additional experiments using a vanilla Transformer with a model dimension of 64 and $r = 8$ on the IWSLT'14 dataset. The evaluation metric is BLEU, where higher scores indicate better performance. Table 15 summarizes the results:

Table 15: BLEU scores for different **sampling mechanisms** on IWSLT'14. Bold indicates the highest performance.

Sampling Mechanism	**MULTINOMIAL**	**UNIFORM**	**SEQUENTIAL**	**TOP_R**
BLEU	**22.28**	22.01	22.13	18.28

Descriptions of Sampling Mechanisms:

- **MULTINOMIAL**: The multinomial random sampling method used in SST.
- **UNIFORM**: Uniform random sampling.
- **SEQUENTIAL**: Iterating through all singular vectors without repetition.
- **TOP_R**: Selecting the top-r singular vectors with the largest singular values.

We also considered a Binomial sampling mechanism; however, it could not guarantee that the number of selected singular vectors would remain consistent with the specified rank, making it unsuitable for direct comparison.

The results indicate that **TOP_R** performs the worst, as its search space collapses into a restricted low-rank subspace. In contrast, as long as all singular vectors are visited, the other methods deliver comparable performance. Among these, **MULTINOMIAL** demonstrates a slight advantage.

Impact of Rank. For all low-rank methods, including LoRA, ReLoRA*, and SST, rank is more of a constraint determined by available resources rather than a hyperparameter to be extensively tuned. Higher ranks generally lead to better performance but at the cost of increased memory consumption. To ensure fairness, the same rank values were used for LoRA, ReLoRA*, and SST in all experiments, as these methods have a similar number of trainable parameters under the same rank.

Additionally, we conducted an experiment on the IWSLT'14 dataset using a vanilla Transformer with a model dimension of 128 to analyze the impact of rank on different methods. The results are presented in Table 16:

The evaluation metric is BLEU, where higher scores indicate better performance. The BLEU score for full-rank training is 25.79. The results demonstrate that as the rank increases, the performance of all methods improves. Notably, SST consistently outperforms other low-rank methods, especially at smaller ranks, highlighting its robustness under resource-constrained settings.

Table 16: **Impact of rank** on BLEU scores for IWSLT'14. Dimension is 128.

Rank (r)	1	2	4	8	16	32	64
LoRA	12.44	14.16	16.37	20.56	23.30	25.12	26.11
ReLoRA	14.53	15.39	18.00	20.61	22.92	24.15	25.25
SST	17.49	20.69	22.80	24.19	25.12	26.08	26.15

Impact of Training Steps. To investigate whether additional training steps benefit SST, we conducted an experiment on the IWSLT'14 dataset using a vanilla Transformer with a model dimension of 64 and $r = 4$. Table 17 presents the BLEU scores for full-rank training and SST under different training steps (evaluated on the model at the last step):

Table 17: BLEU scores under **different training steps.** The default training step in Table 1 is 40,000.

Steps	20,000	40,000	80,000	160,000	320,000	640,000
Full	22.95	24.27	24.85	24.72	24.71	25.05
SST	17.23	20.27	21.91	22.86	23.32	23.92

The results demonstrate that as the number of training steps increases, the gap between full-rank training and SST narrows. Even with $r = 4$, SST approaches the performance of full-rank training at 640,000 steps. These findings confirm that while SST may require more steps to converge at lower ranks, it remains competitive with full-rank training given sufficient steps.

Table 18: **GPU memory consumption** on different sizes of OPT models, including optimizer state and gradient. Model weight uses float32. AdamW optimizer state uses float32 (same data type as used in OPT experiments in Table 2).

	Full	**LoRA/ReLoRA***	**SST**
OPT-125M	1956.05 MB	1118.56 MB	1254.41 MB
OPT-350M	5070.41 MB	2046.41 MB	2573.77 MB
OPT-1.3B	20093.22 MB	7133.24 MB	9345.72 MB

I Memory Consumption and Training Time

Memory consumption. As shown in Table 18, the memory consumption of SST is comparable to LoRA and much smaller than full-rank models. SST has a similar number of trainable parameters (about 0.2% higher) as LoRA (as stated in Table 2), but more frozen parameters (about 45% higher) than LoRA. However, this can be mitigated if we use low precision for the frozen parameters, as in (Dettmers et al., 2024).

Table 19 shows that the memory consumption of SVD decomposition for the largest weight in each model is about 3%, which is small compared with the whole model.

Training time. Table 20 shows that the time spent on SVD in SST is very low, about 0.5%-0.8% compared with the whole training time. SST has comparable training time as LoRA and full-rank model. The increasement of training time of SST is mainly due to SST's linear function, $\mathbf{h} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}\mathbf{x}$, which is slower than original $\mathbf{h} = \mathbf{W}\mathbf{x}$. However, during inference, replacing $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ with a single matrix $\mathbf{W}$ could obtain same computation efficiency as full-rank models. ReLoRA* has comparable computation time as LoRA.

Performance with Fewer Steps. Despite requiring slightly more time per step, SST achieves superior performance with fewer training steps compared to other low-rank methods. The choice of 20% fewer steps for SST corresponds to the maximum additional training time incurred by SST

Table 19: **GPU memory consumption of SVD decomposition in SST.**

Model	Largest Weight Shape	Peak GPU Memory Consumption
OPT-125M	768×3072	41.25 MB (3.3%)
OPT-350M	1024×4096	72.00 MB (2.8%)
OPT-1.3B	2048×8192	288.01 MB (3.1%)

Table 20: **Overall training time** on different sizes of OPT models with 19.7 billion training tokens, using 4 A100 GPU. "Time of SVD in SST" is the overall time of singular value decomposition within SST.

Model	Full	LoRA	SST	Time of SVD in SST
OPT-125M	62.5h	64.4h	65.0h	0.3h (0.5%)
OPT-350M	135.8h	153.3h	170.0h	0.8h (0.5%)
OPT-1.3B	303.4h	324.8h	387.2h	3.0h (0.8%)

compared to other low-rank methods, as shown in Table 20. Table 21 compares the perplexity (PPL) of SST trained with 20% fewer steps to that of other methods trained with full steps.

Table 21: **Validation perplexity with SST trained 20% fewer steps** compared to full steps for other methods.

Model	Full	LoRA	ReLoRA*	SST (20% fewer steps)
OPT-125M	23.50	34.23	35.80	**28.03**
OPT-350M	21.78	34.26	39.21	**29.42**
OPT-1.3B	15.10	1716	29.52	**22.98**
LLaMA-130M	20.04	29.71	31.33	**24.74**
LLaMA-1.3B	14.54	16.50	17.32	**15.65**

These results demonstrate that SST maintains significantly lower perplexity even with fewer training steps, highlighting its efficiency. SST effectively balances its computational overhead while achieving superior performance compared to other low-rank methods. This makes SST a compelling choice for high-quality pretraining.

J Experiment on Image Classification

We conduct additional experiments on image classification tasks using MLP-based models. In this section, we provide a comparison of full-rank training, LoRA, ReLoRA*, and SST on three datasets: MNIST (Lecun et al., 1998), EMNIST (Cohen et al., 2017), and Fashion_MNIST (Xiao et al., 2017).

The architecture of the MLP is $784 - 512 - 512 - 512 -$ #class. Each method is trained for a total of 100 epochs. Learning rate is set to 0.01 for all methods.

We use a rank of 16 for all low-rank methods, which corresponds to 1/32 of the full-rank dimension. For ReLoRA* and SST, one epoch per iteration is used. The results are averaged over three random seeds, and all datasets were evaluated based on test accuracy.

As shown in Table 22, SST outperforms both LoRA and ReLoRA* across all three datasets. SST reduces performance gap between low-rank method and full-rank training by **49%** in average.

K Memory Efficiency Analysis

To better understand the memory efficiency of SST compared to baseline methods, we provide a detailed joint analysis of GPU memory consumption and performance trade-offs.

Table 22: **Image classification tasks** test accuracy.

Dataset	Full	LoRA	ReLoRA*	SST
MNIST	98.63 ± 0.04	97.69 ± 0.10	97.72 ± 0.05	98.33 ± 0.04
EMNIST	85.32 ± 0.24	79.45 ± 0.26	84.12 ± 0.12	84.96 ± 0.11
Fashion_MNIST	90.44 ± 0.06	88.30 ± 0.01	89.08 ± 0.16	89.22 ± 0.06

Memory and Performance Trade-Off. SST's GPU memory consumption is comparable to ReLoRA*, while achieving significant improvements in perplexity (PPL). A comparison of memory reduction and PPL increase is provided in our analysis (Figure 5).

We define the following metrics for clarity:

$$\text{Memory Reduction (\%)} = \frac{\text{Full memory} - \text{Low rank memory}}{\text{Full memory}} \times 100$$

$$\text{PPL Increase (\%)} = \frac{\text{Low rank PPL} - \text{Full PPL}}{\text{Full PPL}} \times 100$$

To provide a more intuitive understanding of SST's memory efficiency, we introduce a new metric called the **efficiency ratio**, defined as:

$$\text{Efficiency Ratio} = \frac{\text{Memory Reduction (\%)}}{\text{PPL Increase (\%)}}$$

This efficiency ratio quantifies how much memory can be reduced at the cost of a 1% increase in PPL. A higher efficiency ratio indicates a more memory-efficient method.

Results. SST achieves a significantly higher efficiency ratio than ReLoRA* across various pretraining tasks. Figure 6 shows the efficiency ratio improvements of SST compared to ReLoRA*:

- **167.4%** (OpenWebText, LLaMA-130M)
- **99.7%** (C4, LLaMA-130M)
- **196.1%** (OpenWebText, OPT-125M)
- **142.3%** (OpenWebText, OPT-350M)
- **65.9%** (OpenWebText, OPT-1.3B)
- **4434.3%** (OpenWebText, LLaMA-1.3B)

Conclusion. These results demonstrate that SST achieves a substantially better trade-off between memory reduction and PPL increase compared to ReLoRA*. This highlights SST's effectiveness in optimizing memory efficiency while maintaining strong model performance, making it a practical choice for resource-constrained pretraining tasks.

L Hyperbolic Graph Neural Networks

Hyperbolic Graph Neural Networks (HGNNs) (Chami et al., 2019; Chen et al., 2022) capitalize on the expansive and hierarchical nature of hyperbolic space to efficiently manage and analyze graph-structured data. This geometric space is particularly suitable for graphs due to its ability to closely mimic the underlying data structures with minimal distortion, offering a substantial improvement over traditional Euclidean methods.

We evaluated the effectiveness of SST on HyboNet (Chen et al., 2022) version of HGNN in node classification and link prediction across four distinct datasets: Airport (Chami et al., 2019), Cora (Sen et al., 2008), Disease (Anderson & May, 1991), and PubMed (Namata et al., 2012). Each experiment was conducted with three random seeds.

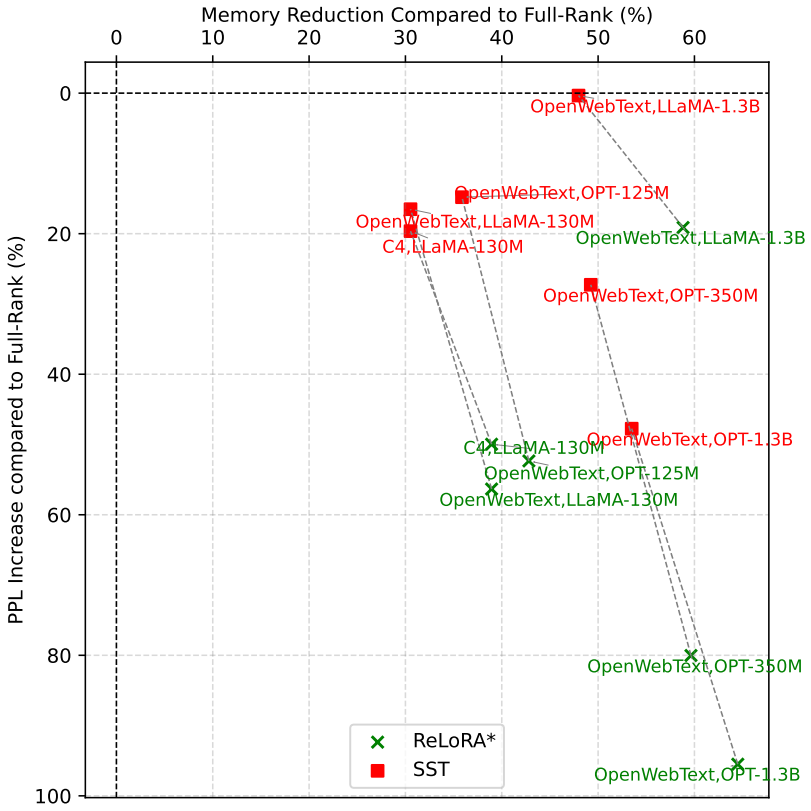

Figure 5: **Memory reduction vs. PPL increase.** Comparison of SST and ReLoRA* on multiple datasets and models.

Table 23: **Node Classification and Link Prediction Results.** Model's dimension $d = 16$. Results are reported as test F1 scores for node classification and test precision for link prediction, expressed in percentages. Values highlighted in bold represent the highest performance among the low-rank methods, while those marked with an "*" denote performance that exceeds that of the full-rank variants.

	Node Classification				Link Prediction			
Method	**Airport**	**Cora**	**Disease**	**PubMed**	**Airport**	**Cora**	**Disease**	**PubMed**
Full $d = 16$	92.88 ± 0.5	81.13 ± 0.2	91.83 ± 0.4	78.1 ± 0.4	95.77 ± 0.08	94.62 ± 0.2	91.49 ± 1.5	96.55 ± 0.03
LoRA $r = 1$	85.75 ± 1.0	45.5 ± 0.3	79.66 ± 1.9	69.17 ± 2.1	94.01 ± 0.2	84.22 ± 0.1	84.29 ± 1.5	89.34 ± 0.4
SST $r = 1$	**88.61 ± 0.5**	**75.07 ± 0.5**	**89.22 ± 1.7**	**77.47 ± 0.3**	**95.37 ± 0.4**	**91.11 ± 0.6**	**93.63 ± 0.7***	**95.57 ± 0.1**
LoRA $r = 2$	**89.06 ± 1.0**	64.73 ± 0.8	83.84 ± 4.3	76.27 ± 0.8	94.75 ± 0.15	88.8 ± 0.5	91.38 ± 0.7	92.14 ± 0.3
SST $r = 2$	87.92 ± 0.09	**77.5 ± 0.7**	**90.64 ± 1.7**	**77.93 ± 0.1**	**95.59 ± 0.2**	**91.89 ± 0.3**	**94.83 ± 0.6***	**95.71 ± 0.1**

The results, detailed in Table 23, demonstrate SST has strong performance in both node classification and link prediction tasks. With $r = 1$, SST reduces the performance gap, by an average of **73.7%** in node classification and **82.5%** in link prediction. In the Disease link prediction task, SST outperforms full-rank training at both $r = 1$ and $r = 2$. Notably, SST's advantage over LoRA is greater at $r = 1$ than at $r = 2$, likely due to SST's sampling strategy being particularly effective in sparser scenarios.

M Related Work

Low-Rank Adaptation. Low-rank adaptation has become a key strategy for reducing the computational and memory requirements of training large-scale neural networks. Hu et al. (2022) introduced

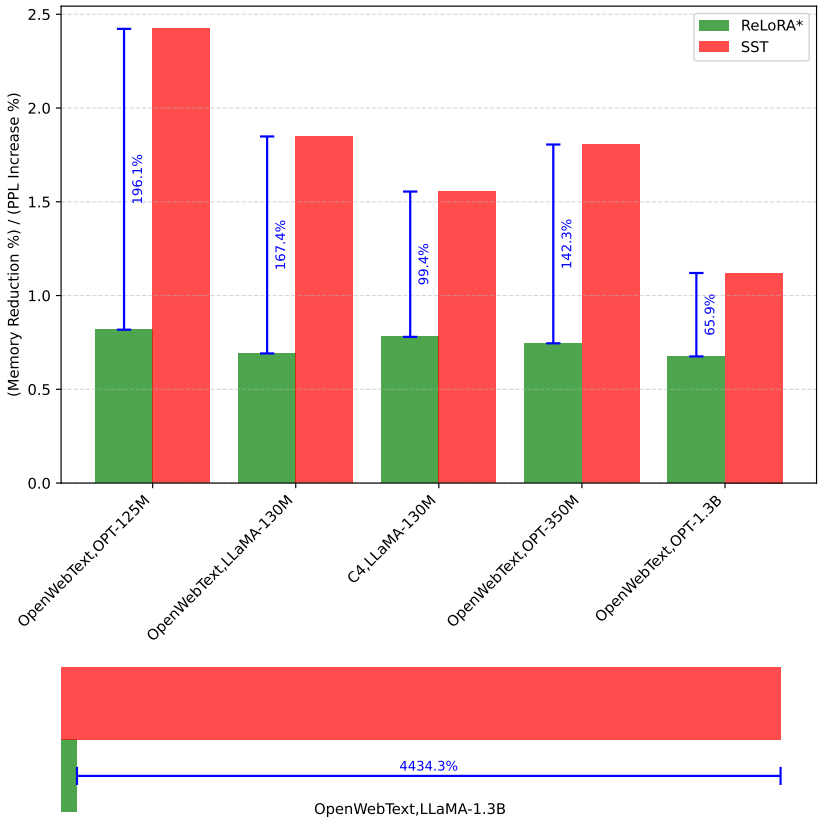

Figure 6: **Efficiency Ratio Improvements.** SST achieves significantly higher efficiency ratios compared to ReLoRA* across various tasks and model sizes. The LLaMA-1.3B result is included at the bottom of the plot due to its large value.

Table 24: **Comparison of BLEU scores on Multi30k and IWSLT'17 datasets** using Euclidean Transformer (dimension $= 512$), $r = 32$. Scores highlighted in bold represent the highest performance achieved by low-rank methods.

	Full	**LoRA**	**ReLoRA***	**SST**
Multi30K	40.7	40.1	41.6	**43.4**
IWSLT'17	31.7	31.9	32.0	**32.3**

Low-Rank Adaptation (LoRA), a technique that fine-tunes pre-trained models by integrating low-rank matrices to significantly reduce the number of parameters updated during training. Various enhancements to LoRA have since been developed to improve its efficiency and broaden its application (Zhang et al., 2023; Dettmers et al., 2024; Zi et al., 2023; Valipour et al., 2023). Lialin et al. (2024) introduced ReLoRA specifically for the pre-training phase, which requires a full-rank warm-up to achieve performance comparable to full-rank training. A similar approach is also found in COLA (Xia et al., 2024) and PeriodicLoRA (Meng et al., 2024b). Additionally, Zhao et al. (2024) introduced GaLore, which projects gradients into a low-rank subspace. Meng et al. (2024a) introduced PiSSA, which applies SVD-based low-rank updates for fine-tuning pre-trained weights by focusing on dominant singular vectors. These advancements highlight the versatility and ongoing evolution of low-rank adaptation techniques in response to the growing complexity of neural network models.

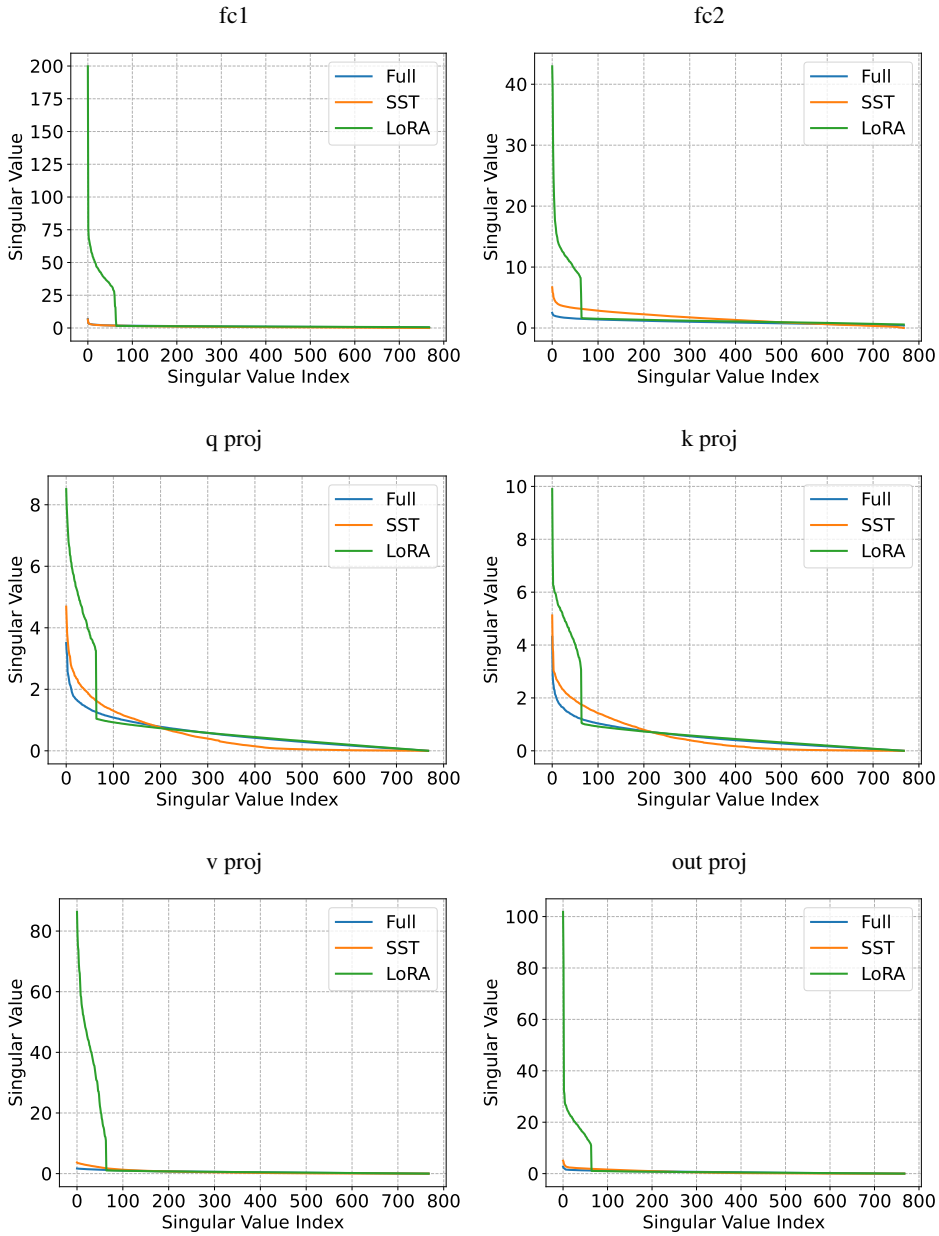

Figure 7: **Singular Value Distribution.** This visualization depicts the distribution of singular values for the OPT-125M model with full-rank, LoRA, and SST, with $r = 64$). The x-axis represents the index of singular values, sorted from largest to smallest, while the y-axis shows the magnitude of each value. It highlights how LoRA predominantly captures and overestimates the top-r singular values, in contrast to SST, which shows a much similar distribution as full-rank training.

N Related Work of Other Parameter-Efficient Training Methods

Apart from low-rank adaptations, researchers have developed a variety of parameter-efficient training techniques to optimize resource consumption while preserving learning effectiveness. Prompt tuning is an effective method that integrates tunable prefixes or soft prompts into the input embeddings of models. It enables lightweight task-specific adaptations with minimal impact on the model's overall architecture (Lester et al., 2021; Liu et al., 2021). Dynamic sparse training (DST), through methods like SET (Mocanu et al., 2018), RIGL (Evci et al., 2020), MEST (Yuan et al., 2021), and CHT (Zhang

et al., 2024), employs a dynamic prune-and-grow strategy that adjusts network topology during training. This approach optimizes training efficiency and can improve generalization by continuously adapting the network's sparse structure. This presents a significant shift from static training methods.

Hyperbolic Neural Networks. Hyperbolic neural networks are an emerging area in deep learning, exploiting the unique properties of hyperbolic space that make it ideal for processing hierarchical and graph-structured data (Muscoloni et al., 2017; Cannistraci & Muscoloni, 2022). Innovations in this area have adapted fundamental neural network mechanisms to function within hyperbolic geometries, as demonstrated by Muscoloni et al. (2017) and Ganea et al. (2018). Further developments by Chen et al. (2022) explore manifold-specific properties to enrich both theoretical understanding and practical deployment. The use of hyperbolic spaces has been shown to significantly improve data representation and generalization across various tasks, marking a notable advancement in managing complex, non-Euclidean data structures (Gulcehre et al., 2019; Liu et al., 2019; Tifrea et al., 2019).