MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

SuperLoRA: Parameter-Efficient Unified Adaptation for Large Vision Models

Chen, Xiangyu; Liu, Jing; Wang, Ye; Wang, Pu; Brand, Matthew; Wang, Guanghui; Koike-Akino, Toshiaki

TR2024-062 June 01, 2024

Abstract

Low-rank adaptation (LoRA) and its variants are widely employed in fine-tuning large models, including large language models for natural language processing and diffusion models for computer vision. This paper proposes a generalized framework called SuperLoRA that unifies and extends different LoRA variants, which can be realized under different hyper-parameter settings. Introducing new options with grouping, folding, shuffling, projection, and tensor decomposition, SuperLoRA offers high flexibility and demonstrates superior performance, with up to 10-fold gain in parameter efficiency for transfer learning tasks.

IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2024

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

SuperLoRA: Parameter-Efficient Unified Adaptation for Large Vision Models

Xiangyu Chen[1,2], Jing Liu[2], Ye Wang[2], Pu (Perry) Wang[2],
Matthew Brand[2], Guanghui Wang[3], Toshiaki Koike-Akino[2]

[1] University of Kansas, Lawrence, KS 66045, USA

[2] Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA

[3] Toronto Metropolitan University, Toronto, ON M5B 2K3, Canada

xychen@ku.edu, {xiachen, jiliu, yewang, pwang, brand, koike}@merl.com, wangcs@torontomu.ca

Abstract

Low-rank adaptation (LoRA) and its variants are widely employed in fine-tuning large models, including large language models for natural language processing and diffusion models for computer vision. This paper proposes a generalized framework called SuperLoRA that unifies and extends different LoRA variants, which can be realized under different hyper-parameter settings. Introducing new options with grouping, folding, shuffling, projection, and tensor decomposition, SuperLoRA offers high flexibility and demonstrates superior performance, with up to 10-fold gain in parameter efficiency for transfer learning tasks.

1. Introduction

Large neural network models are dominating machine learning recently with the emergence of exceptional models, such as large vision models (LVMs) including Vision Transformer (ViT) [10], ConvNeXt [33] and Stable Diffusion [19] for vision tasks, and large language models (LLMs) including GPT [1], PALM2 [4], Gemini [3] and LLaMA2 [39] for natural language processing (NLP). However, the increased resource consumption and data requirement along with model size limits its generalization on downstream tasks. To solve this, Parameter-Efficient Fine-Tuning (PEFT) has been widely explored to fine-tune less parameters while retaining high performance. Among this, adapter-based technique like LoRA (**Low-Rank Adaptation**) [21] demonstrates advantages and flexible convenience.

LoRA [21] approximates the weight updates of the base model by approximating the change ΔW of each weight matrix as the product of two low-rank matrices. This decreases the required parameters from d^2 to $2rd$ when $r \ll d$, where d and r are weight size and the rank, respectively. Most LoRA variants work on solving the inherent *low-rank constraint* of matrix factorization, including LoHA (**Low-rank Hadamard**) [42], LoKr (**Low-rank Kronecker**) [42], and LoTR (**Low Tensor Rank**) [5]. We discuss more related work in Appendix A. However, we find these variants can be nicely unified within our framework—SuperLoRA—with different hyper-parameters as shown in Table 1. Our proposed SuperLoRA framework is depicted in Figure 1, which also yields to some new variants: LoNKr (**Low-rank N-split Kronecker**) and LoRTA (**Low-Rank Tensor Adaptation**). Additionally, we introduce three extended options: 1) reshaping ΔW to any arbitrary multi-dimensional tensor arrays before applying LoRA variants; 2) splitting all ΔW into an arbitrary number of groups, which breaks the boundaries for ΔW across different weights; and 3) projecting fewer number of trainable parameters into larger weights through a projection layer $\mathcal{F}(\cdot)$ with fixed parameters. Accordingly, SuperLoRA provides more flexibility and extended functionality, controlled by a set of hyper-parameters listed in Table 2. Our contributions include:

- We propose a new PEFT framework SuperLoRA which gracefully unifies and extends most LoRA variants.
- With projected tensor rank decomposition, SuperLoRA can adapt all weights across layers jointly with a wide range of adjustable parameter amount.
- We investigate the effect of tensor reshaping, grouping, random projection, and shuffling.
- We demonstrate high parameter efficiency for large ViT and diffusion models in two transfer learning tasks: image classification and image generation.
- Significant parameter reduction by up to 10 folds can be achieved.

2. SuperLoRA

Figure 1 shows the overview of SuperLoRA, which is a generalization of LoRA variants to allow high flexibility in the

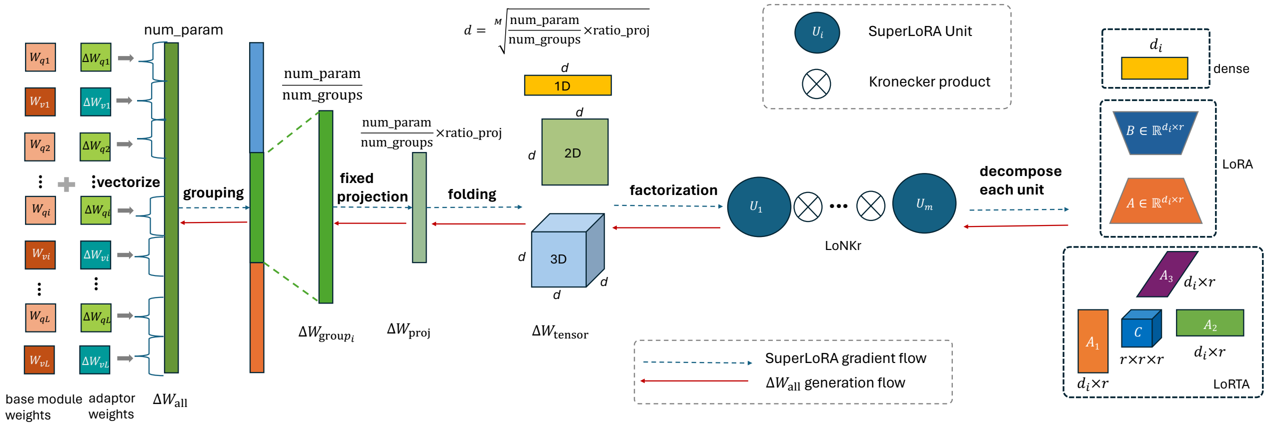

num_param
W_{q1} W_{v1} W_{q2} W_{qi} W_{vi} W_{qL} W_{vL}
ΔW_{q1} ΔW_{v1} ΔW_{q2} ΔW_{qi} ΔW_{vi} ΔW_{qL} ΔW_{vL}
vectorize
grouping
base module weights
adaptor weights
ΔW_{all}
$\frac{\text{num\_param}}{\text{num\_groups}}$
fixed projection
$\frac{\text{num_param}}{\text{num_groups}}\times$ratio_proj
$\Delta W_{\text{group}_i}$
ΔW_{proj}
folding
$d = \sqrt[M]{\frac{\text{num\_param}}{\text{num\_groups}}\times\text{ratio\_proj}}$
d
1D
2D
3D
ΔW_{tensor}
factorization
U_i SuperLoRA Unit
Kronecker product
U_1
U_m
LoNKr
SuperLoRA gradient flow
ΔW_{all} generation flow
decompose each unit
d_i
dense
$B \in \mathbb{R}^{d_i\times r}$
$A \in \mathbb{R}^{d_i\times r}$
LoRA
A_3 $d_i\times r$
A_1 $d_i\times r$
C $r\times r\times r$
A_2 $d_i\times r$
LoRTA

Figure 1. Schematic of SuperLoRA to fine-tune multi-layer attention modules at once with grouping, projection, folding, and factorization.

Table 1. Hyper-parameter settings in SuperLoRA and the resultant LoRA variant

hyper-parameters settings	method
$\mathcal{F}=I$, weight-wise, $K=1$, $C_{g1}=I$, $M=1$	dense FT
$\mathcal{F}=I$, weight-wise, $K=1$, $C_{g1}=I$, $M=2$	LoRA [21]
$\mathcal{F}=I$, weight-wise, $K=2$, $C_{gk}=I$, $M=2$	LoKr [42]
$\mathcal{F}=I$, group-wise, $G=1$, $M>2$	LoTR [5]
$\mathcal{F}=I$, group-wise, $K>2$, $C_{gk}=I$, $M=2$	LoNKr
$\mathcal{F}=I$, group-wise, $K=1$, $M>2$	LoRTA

Table 2. Hyperparameters and notation.

notation	description
r	rank of factorization
$\mathcal{F}$	mapping function
ρ	compression ratio
G	number of groups
M	order of tensor modes
K	number of splits

weight update ΔW. SuperLoRA can be formulated as:

$$\Delta W_{\text{group}_g} = \mathcal{F}\left(\bigotimes_{k=1}^{K}\left(C_{gk}\times_1 A_{gk1}\times_2\cdots\times_M A_{gkM}\right)\right),$$

where $\mathcal{F}(\cdot)$ is a simple projection function applied on the results of SuperLoRA modules. We denote $\times_m$ as mode-m tensor product, and $\otimes$ as Kronecker product. Here, M represents the order of the reshaped weight tensor modes, and high-order Tucker decomposition [41] is employed to formulate this high-order tensor, where $C_{gk}\in\mathbb{R}^{r_1\times r_2\times\cdots\times r_M}$ is M-D core tensor and $A_{gkm}\in\mathbb{R}^{d_m\times r_m}$ are 2D plane factors. SuperLoRA units in Figure 1 are combined with Kronecker product across K splits in a proper shape. Depending on reshaping, each split has multiple choices including a combination of dense fine-tuning (FT: 1D), LoRA (2D), and high-order Tucker decomposition (3D, 4D, etc.).

For SuperLoRA as depicted in Figure 1, we first concatenate all $\Delta W\in\mathbb{R}^{d_i\times d_i}$ across multiple layers to get the total correction of $\Delta W_{\text{all}}\in\mathbb{R}^{\sum_i d_i^2}$. Then, ΔW_{all} is divided into g groups: $\{\Delta W_{\text{group}_g}\}$ for $g\in\{1,2,\ldots,G\}$. Each LoRA module will then produce $\Delta W_{\text{group}_g}$. Finally, stretch $\Delta W_{\text{group}_g}$ to one dimension, fetch corresponding size of ΔW from those $\Delta W_{\text{group}_g}$ and add it to candidate weight matrix, *e.g.*, query and value projection weights for attention modules across layers. Figure 2 shows the grouping mechanism which provides various options, including weight-wise, layer-wise, and general grouping. Reshaping in Figure 2(c) can solve unbalanced fan-in/fan-out issue in Figure 2(b) when stacking multiple weights.

SuperLoRA can further modify the tensor arrays through a simple mapping $\mathcal{F}(\cdot)$: *e.g.*, we can project much smaller ΔW_{lora_g} into larger final $\Delta W_{\text{group}_g}$ to improve the parameter efficiency. We use the fastfood projection [2, 28] as shown in Figure 3, which is written as follows:

$$\begin{aligned}\Delta W_{\text{group}_g} &= \mathcal{F}(\Delta W_{\text{lora}_g})\\ &= \mathsf{vec}[\Delta W_{\text{lora}_g}]\,\mathcal{H}'\,\mathsf{diag}[\mathcal{G}]\,\varPi\,\mathcal{H}\,\mathsf{diag}[\mathcal{B}],\end{aligned}$$

where $\mathsf{vec}[\cdot]$ is a vectorization operator, $\mathsf{diag}[\cdot]$ denotes a diagonalization operator, $\mathcal{H}$ is left-truncated Walsh–Hadamard matrix, $\mathcal{H}'$ is its right-truncated version, $\mathcal{G}$ is a random vector drawn from normal distribution, $\varPi$ is a random permutation matrix, and $\mathcal{B}$ is a random vector drawn from Rademacher distribution. The compression ratio for the projection $\mathcal{F}(\cdot)$ is $\rho=|\Delta W_{\text{lora}_g}|/|\Delta W_{\text{group}_g}|$, where $|\cdot|$ denotes the total number of elements of the tensor. It is a fast Johnson–Lindenstrauss transform with log-linear

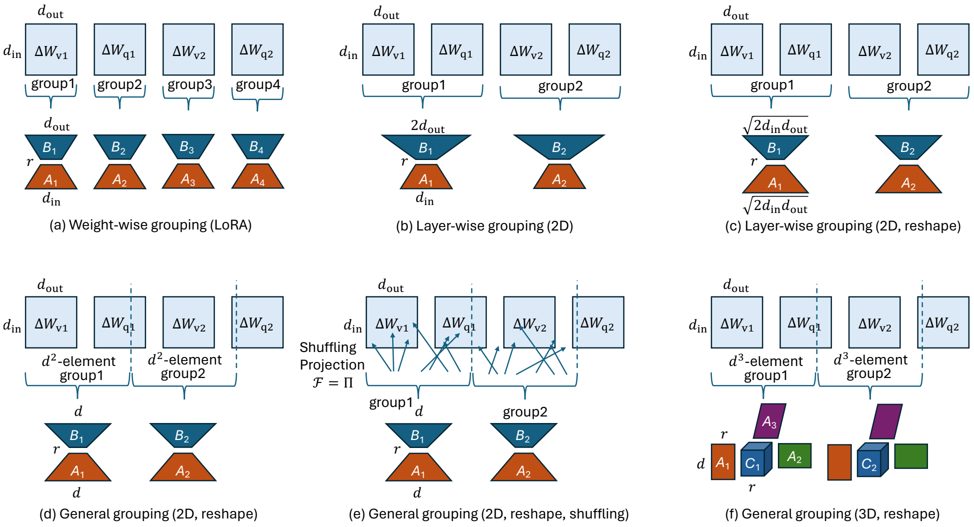


Figure 2. Examples of grouping mechanism.

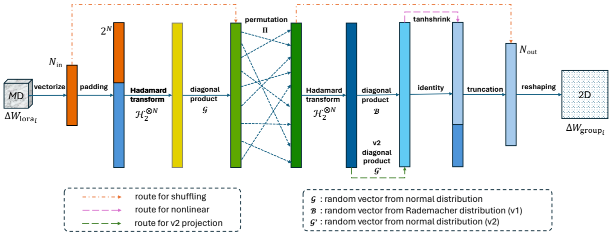


Figure 3. Illustration of fastfood projection and its variants.

complexity due to the fast Walsh–Hadamard transform, and no additional parameters are required when the random seed is predetermined. The projection also includes a shuffling variant as in Figure 2(e). More details of SuperLoRA framework are found in Appendix A.2, and its different variants are discussed in Appendix A.10.

3. Transfer Learning Experiments

Transfer learning for image classification is conducted between ImageNet21k [9] and CIFAR100 [26] based on a ViT-base [10] model. More details of the ViT model are described in Appendix A.3. The query and value projection layers in the attention modules are fine-tuned with SuperLoRA. The model is trained for 5,000 steps with the stochastic gradient descent (SGD) optimizer, with a batch size of 128 and a learning rate of 0.05. The OneCycleLR [38] scheduler is used.

We evaluated SuperLoRA with grouping with/without reshaping to square-like for 2D $\Delta W_{\text{group}_g}$, reshaping version for higher-order $\Delta W_{\text{group}_g}$ including 3D, 4D and 5D. The fixed projection layers are inserted to SuperLoRA with reshaping (2D version) and also dense. Original weight-wise LoRA is also examined for comparison by setting the number of groups to the number of query and value weights (24 for 12-layer ViT-base) as all projection weights for ViT-base are equal size. Each correction weight is of size 768 × 768 as the projection weight for query/value, resulting in 14M parameters. Except for most cases, more ranks are needed to span the parameter axis well, including larger ranks from 34 to 128 and smaller ranks below 8 for LoRTA. Projection compression ratio is from $\rho \in \{0.5, 0.25, 0.1, 0.01\}$, and the fixed projection parameters are shared across all groups in our experiments.

Classification results versus the number of parameters are shown in Figure 4 with Pareto frontier lines. Comparing group-wise SuperLoRA (2D with/without reshape) with weight-wise LoRA, we can find that SuperLoRA versions show better performance in terms of the trade-off between classification accuracy and the number of parameters. Noticeably, we observe three to four times advantage in terms of parameter efficiency for the same accuracy. As the largest number of groups is set to 24 (*i.e.* LoRA), it indicates smaller number of groups are superior. This may be because ViT model is excessively large for the CIFAR100 dataset, with much more redundant weights. Grouping weights and layers together can reduce noise brought by the redundancy. With reshaping $\Delta W_{\text{group}_g}$ to a square matrix, classification accuracy further increases in the lower parameter regime and the range of parameters the model can cover becomes wider as higher rank can be used while maintaining a smaller number of parameters.

To examine the effect of higher-order tensor folding, the order M is set to be 3, 4 and 5 for SuperLoRA (*i.e.* LoRTA) as well as 2. For $M = 2$ cases with 2D tensor, we use

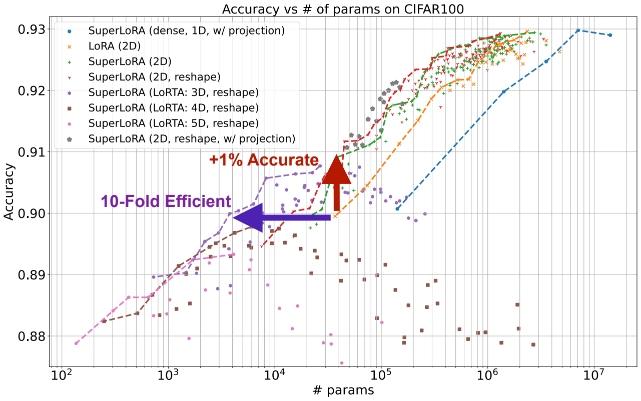


Figure 4. Transfer learning from ImageNet21K to CIFAR100, parameters in classifier head excluded.

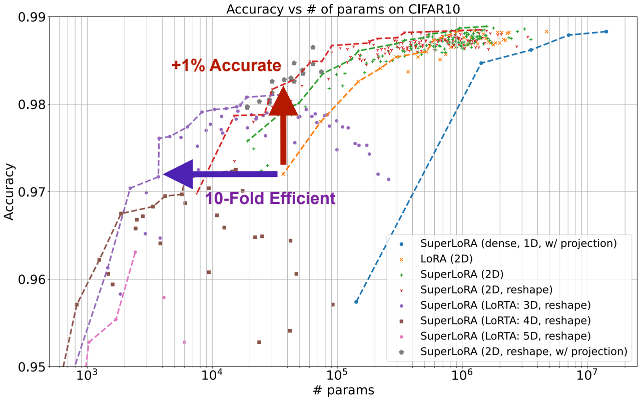


Figure 5. Transfer learning from ImageNet1K to CIFAR10, with frozen classifier head after manual label matching.

identity core tensor like typical LoRA. With the increase of order from 2 to 5, higher order takes place lower-order at fewer-parameter regimes. Moreover, data points for high-order LoRTA show a hill-like trend with the increase of parameters. This may be caused by the inefficient core tensor, which increases parameters rapidly without benefiting the accuracy. When comparing the lowest rank LoRA (which achieves around 0.9 accuracy with about 4×10^4 parameters), our LoRTA (3D) significantly improves the accuracy by about 1% at the comparable number of parameters, and more significantly reduces the number of parameters by 10 folds to keep the comparable accuracy of 0.9.

Finally, we address the impact of the projection layer $\mathcal{F}(\cdot)$. Fixed fastfood projection as in Figure 3 is applied on SuperLoRA. For 1D dense, the plot for a projection ratio of $\{1, 0.5, 0.25, 0.1, 0.01\}$ is placed from right to left in Figure 4. The classification accuracy dropped less than 1% from projection ratio 1 to 0.1 (*i.e.* 90% less parameters), but it is worse than LoRA. To get some results of projection for the number of parameters around 10^4 and 10^5, we select a few settings for SuperLoRA (2D, reshape) with $G = 1$ as shown in the figure with a marker of dark stars. Most projection results demonstrate better accuracy compared with other SuperLoRA settings without projection in the same number of parameters level. This result shows a smaller adapter with fixed projection layer is a strong functionality to improve the parameter efficiency of SuperLoRA.

We also examined another transfer learning task from ImageNet1k to CIFAR10. Most settings are same as Figure 4 for transfer learning from ImageNet21k to CIFAR100. The classifier head is frozen after selecting most relevant labels in ImageNet1k. Details are found in Appendix A.3.2. Classification results can be found in Figure 5. Even though only attention modules are adapted, overall performance is excellent, reaching an accuracy close to 0.99. Besides, SuperLoRA significantly outperforms original LoRA in terms of both classification accuracy and the parameter range it covers as the transfer learning. SuperLoRA (2D, reshape) shows at least 3-fold reduction in the required number of parameters compared to LoRA. Noticeably, when comparing the lowest-rank LoRA with around 0.97 accuracy, SuperLoRA (2D, reshape, w/ projection) improves the accuracy by about 1%, and moreover the required number of parameters can be greatly reduced by 10 folds with SuperLoRA (LoRTA: 3D, reshape) to maintain the comparable accuracy.

We confirmed the remarkable gain of our SuperLoRA on a transfer learning task for image classification with ViT models. In Appendix A.6, we further discussed the geometric analysis of SuperLoRA, and grouping impacts in Appendix A.7. In addition, We evaluated the advantage in another transfer learning task for image generation with diffusion models in Appendix A.8, Appendix A.9, Appendix A.11, and Appendix A.12.

4. Conclusion

We proposed a new unified framework called SuperLoRA, which generalizes and extends LoRA variants including LoKr and LoTR. SuperLoRA provides some extended variants, which we refer to as LoNKr and LoRTA. It offers a rich and flexible set of hyper-parameters, including the rank of factorization, the choice of projection function, projection ratio, the number of groups, the order of tensor, and the number of Kronecker splits. Through transfer learning experiments, we demonstrated that SuperLoRA achieves promising results in parameter efficiency for finetuning at low-parameter regimes. We could reduce the required number of parameters by 3 to 10 folds compared to LoRA. Future work includes studying the projection functions to further improve the efficiency in extremely-low-parameter regimes, and applications to various transfer learning tasks along with different large models such as LLMs.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report, 2023. 1

[2] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*, 2020. 2, 1

[3] Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

[4] Rohan Anil, Andrew M. Dai, and Orhan Firat et al. PaLM 2 technical report, 2023. 1

[5] Daniel Bershatsky, Daria Cherniuk, Talgat Daulbaev, and Ivan Oseledets. LoTR: Low tensor rank weight adaptation. *arXiv preprint arXiv:2402.01376*, 2024. 1, 2

[6] Mikołaj Bińkowski, Danica J Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD GANs. *arXiv preprint arXiv:1801.01401*, 2018. 6

[7] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. AdaptFormer: Adapting vision transformers for scalable visual recognition. *Advances in Neural Information Processing Systems*, 35:16664–16678, 2022. 1

[8] Wei Chen, Zichen Miao, and Qiang Qiu. Parameter-efficient tuning of large convolutional models. *arXiv preprint arXiv:2403.00269*, 2024. 1

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 1, 3

[11] Ali Edalati, Marzieh Tahaei, Ivan Kobyzev, Vahid Partovi Nia, James J Clark, and Mehdi Rezagholizadeh. Krona: Parameter efficient tuning with Kronecker adapter. *arXiv preprint arXiv:2212.10650*, 2022. 1

[12] Demi Guo, Alexander M Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4884–4896, 2021. 1

[13] Tianxiang Hao, Hui Chen, Yuchen Guo, and Guiguang Ding. Consolidator: Mergable adapter with group connections for visual adaptation. In *The Eleventh International Conference on Learning Representations*, 2022. 1

[14] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient low rank adaptation of large models. *arXiv preprint arXiv:2402.12354*, 2024. 1

[15] Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. Towards a unified view of parameter-efficient transfer learning. In *International Conference on Learning Representations*, 2021. 1

[16] Xuehai He, Chunyuan Li, Pengchuan Zhang, Jianwei Yang, and Xin Eric Wang. Parameter-efficient model adaptation for vision transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 817–825, 2023. 1

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 6

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 3, 6

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[20] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019. 1

[21] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2021. 1, 2

[22] Shibo Jie and Zhi-Hong Deng. Fact: Factor-tuning for lightweight adaptation on vision transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1060–1068, 2023. 1

[23] Shibo Jie, Haoqing Wang, and Zhi-Hong Deng. Revisiting the parameter efficiency of adapters from the perspective of precision redundancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17217–17226, 2023. 1

[24] Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. Compacter: Efficient low-rank hypercomplex adapter layers. *Advances in Neural Information Processing Systems*, 34:1022–1035, 2021. 1

[25] Oscar Key, Jean Kaddour, and Pasquale Minervini. Local LoRA: Memory-efficient fine-tuning of large language models. In *Workshop on Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Optimization (WANT@ NeurIPS 2023)*, 2023. 1

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 3

[27] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems*, 32, 2019. 6

[28] Quoc Le, Tamás Sarlós, Alex Smola, et al. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, page 8, 2013. 2, 1

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86 (11):2278–2324, 1998. 6

[30] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, 2021. 1

[31] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021. 1

[32] Jing Liu, Toshiaki Koike-Akino, Pu Wang, Matthew Brand, Ye Wang, and Kieran Parsons. LoDA: Low-dimensional adaptation of large language models. *NeurIPS'23 Workshop on on Efficient Natural Language and Speech Processing*, 2023. 1

[33] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022. 1

[34] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, number 5, page 7. Granada, Spain, 2011. 6

[35] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In *16th Conference of the European Chapter of the Associationfor Computational Linguistics, EACL 2021*, pages 487–503. Association for Computational Linguistics (ACL), 2021. 1

[36] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 6

[37] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training GANs. *Advances in neural information processing systems*, 29, 2016. 6

[38] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, pages 369–386. SPIE, 2019. 3

[39] Hugo Touvron, Louis Martin, and Kevin Stone et al. Llama 2: Open foundation and fine-tuned chat models, 2023. 1

[40] Anton Tsitsulin, Marina Munkhoeva, Davide Mottin, Panagiotis Karras, Alex Bronstein, Ivan Oseledets, and Emmanuel Mueller. The shape of data: Intrinsic distance for data distributions. In *International Conference on Learning Representations*, 2019. 6

[41] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966. 2

[42] Shin-Ying Yeh, Yu-Guan Hsieh, Zhidong Gao, Bernard B W Yang, Giyeong Oh, and Yanmin Gong. Navigating text-to-image customization: From lyCORIS fine-tuning to model evaluation. In *The Twelfth International Conference on Learning Representations*, 2024. 1, 2

[43] Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in low-rank adapters of foundation models. *arXiv preprint arXiv:2402.16842*, 2024. 1

SuperLoRA: Parameter-Efficient Unified Adaptation for Large Vision Models

Supplementary Material

A. Related Work

PEFT algorithms are widely explored in transfer learning tasks in both computer vision [16, 22, 23] and NLP fields [12, 15, 24, 30, 31] as it not only saves memory and time at fine-tuning, but also requires much less data to fine-tune, making it feasible to borrow the capacity from large models in few-data tasks. Adapter-based methods [7, 13, 20, 35], that freeze the base model weights and fine-tune only the additional adapter parameters, stand out since their plug-and-play nature enables many downstream tasks to share the same large model, leaving the adapter to hold only the task-specific information. The widely used method LoRA [21] and its extension [14, 43] assume that the weight correction term can be estimated by low-rank decomposition under the low-dimensional manifold hypothesis.

Addressing the inherent *low-rank constraint* of matrix factorization in LoRA, LoHA [42] divides ΔW into two splits and combines them with Hadamard product, and KronA [11] combines the two splits with a Kronecker product to enlarge the overall rank. LoKr [42] further extended KronA to convolutional layers. LoDA (Low-Dimensional Adaptation) [32] extended LoRA by introducing nonlinearity. Our SuperLoRA can nicely generalize and extend such variants.

Instead of approximating weight-wise updates, LoTR [5] jointly approximates all ΔW across the model with a careful handling to preserve the geometric meaning of each weight. Differently, SuperLoRA relaxes the geometrically meaningful boundaries by caring the total number of parameters and splitting it to any number of groups. For high-order tensor decomposition, LoTR employs more stringent Tensor Train Decomposition to deal with the core tensor explosion, while SuperLoRA coupled Tucker Decomposition with a fixed projection layer. Besides, their proposed methods are restricted to context when ΔW is the same high-order tensor, while with reshaping, SuperLoRA (LoRTA) can be applied to any weight shape.

Most recent work [8] decomposes each convolution kernel into a learnable filter atom and its non-learnable counterparts. The concept of filter atom is similar to the projection layer of SuperLoRA. However, it works on each convolutional kernels separately, resulting in a waste of parameters, while SuperLoRA considers the entire model jointly. Besides, the atom coefficients are obtained from matrix factorization, while SuperLoRA uses a fastfood projection [28], which is faster, simpler and more theoretically justifiable to exploit intrinsic dimensionality [2]. In addition, SuperLoRA can control the size of atoms directly while atoms in their method are restricted in factorization.

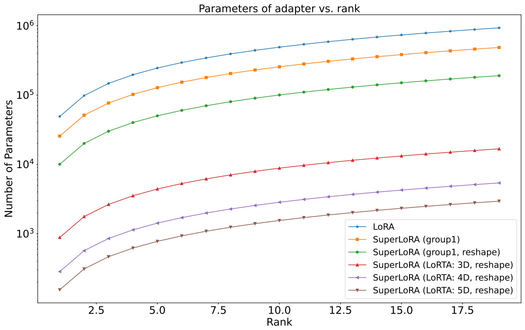


Figure 6. Required number of parameters.

Local LoRA [25] aims to reduce memory consumption at fine-tuning by splitting large model into groups and then fine-tune group-by-group sequentially, but no adjustment on the LoRA structure was proposed. Instead, SuperLoRA focuses on how to split and assign LoRA for each group, which is a viable extension of Local LoRA.

A.1. Low-Rank Adaptation (LoRA)

LoRA [21] assumes the update ΔW of each weight matrix W for fine-tuning can be approximated by low-rank mapping as $\Delta W = AB^\top$ ($[\cdot]^\top$ denotes matrix transpose), which is added to the frozen weight matrix as shown in Figure 7a:

$$W' = W + \Delta W = W + AB^\top, \tag{1}$$

where $A \in \mathbb{R}^{d_1 \times r}$, $B \in \mathbb{R}^{d_2 \times r}$, and the rank r. With a smaller r compared with the matrix dimensions, it only requires $(d_1 + d_2)r$ parameters for each weight matrix, while full fine-tuning (FT) for dense $\Delta W \in \mathbb{R}^{d_1 \times d_2}$ results in $d_1 d_2$ parameters. LoRA has been widely used in fine-tuning large models as much less trainable parameters save memory usage at training while retaining performance, making it easily adapted to downstream tasks with limited resources.

A.2. SuperLoRA

SuperLoRA and LoTR: While LoRA estimates ΔW in a weight-wise independent way, SuperLoRA considers the whole weights ΔW_{all} jointly. It can relax the restriction of the weight shape and geometric meaning of weight axis unlike LoTR. Here, the number of groups can be adjusted to balance between parameter amount and fine-tuning performance. When the number of groups is the number of weights and the group boundary matches the weight boundary, it corresponds to weight-wise LoRA. When the number of groups is $G = 1$, SuperLoRA corresponds to LoTR [5], but with an additional projection mapping $\mathcal{F}$.

Reshaping to regular tensor: Grouping multiple layers together by concatenating ΔW along one axis results in skew $\Delta W_{\text{group}_g}$, limiting the choice of ranks in LoRA modules and leading to worse approximation. For example, stacking query and value weight updates as $[\Delta W_{\text{q}}, \Delta W_{\text{v}}]$ will be of size $d_1 \times 2d_2$, which is less efficient for LoRA as A and B matrices have unbalanced sizes. To solve this, we propose to reshape $\Delta W_{\text{group}_g}$ to a regular tensor: *i.e.*, square-like 2D matrix, cubic-like 3D tensor, or high-order hyper-cubic tensors having same dimension size across all axes. This reshaping can reduce the dimension per axis in the order of $\mathcal{O}[N^{1/M}]$ for N being the number of stacking weights, that in return can allow higher rank size per plane factors. Several examples of grouping and reshaping are discussed in Appendix A.5, and its geometric analysis in Appendix A.6.

SuperLoRA and LoKr/LoNKr: LoKr is depicted in Figure 7b, which can be extended as shown in Figure 7c. We call it LoNKr, which combines K splits composed of sub LoRA units through Kronecker products: *i.e.*, $K > 2$. When $K = 2$, it reduces to LoKr but with an additional flexibility. For example, LoNKr can still adapt multiple attention modules at once with an adjustable group size G, unlike weight-wise adaptation of LoKr.

LoRTA: Folding a matrix $\Delta W_{\text{group}_g}$ into high-order tensor (*e.g.*, 3D, 4D, 5D) can decrease parameters with tensor rank decomposition, like Tucker decomposition, where $\Delta W_{\text{group}_g}$ is represented by M 2D plane factors and one MD core tensor. We refer to this variant of SuperLoRA using Tucker decomposition as LoRTA. For example, when $M = 3$ and $K = 1$, we have 3D tensor rank decomposition for $\Delta W_{\text{group}_g} \in \mathbb{R}^{d1 \times d2 \times d3}$ as follows:

$$\Delta W_{\text{group}_g} = C_{gK} \times_1 A_{gK1} \times_2 A_{gK2} \times_3 A_{gK3}, \quad (2)$$

where $C_{gK} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is a reshaped 3D core tensor, $A_{gKm} \in \mathbb{R}^{d_m \times r}$ is a mode-m 2D plane factor, and $\times_m$ denotes mode-m tensor product. For simplicity, we set a rank $r = r_m$ for any mode $m \in \{1, 2, \ldots, M\}$.

The core tensor may cause the explosion of parameters with larger rank as the number of parameters is exponential as r^M. It may be resolved by restricting the core tensor to be strongly diagonal or identity. For instance, $M = 2$ with identity core tensor $C_{gK} = I$ corresponds to the original LoRA, and $M = r = 1$ identity core tensor corresponds to the dense FT. When using diagonal core tensor, it reduces to Candecomp-Parafac (CP) decomposition. Figure 6 shows the number of required parameters with CP decomposition. One can see that higher-order tensor decomposition can significantly reduce the total number of trainable parameters at a certain rank. We provide another solution without limiting the core tensor by coupling with the projection layer $\mathcal{F}$ below.

Projection: Most LoRA variants assume the resultant ΔW_{lora_g} from LoRA modules is the final ΔW added to W directly. However, we can further modify the ΔW_{lora_g} through a simple mapping: *e.g.*, we can project much smaller ΔW_{lora_g} into larger final $\Delta W_{\text{group}_g}$ to improve the parameter efficiency. We consider a random projection layer based on the fastfood projection [28] to map ΔW_{lora_g} to $\Delta W_{\text{group}_g}$.

Specifically, the fastfood projection is performed as follows:

$$\begin{aligned}\Delta W_{\text{group}_g} &= \mathcal{F}(\Delta W_{\text{lora}_g}) \\ &= \mathsf{vec}[\Delta W_{\text{lora}_g}]\, \mathcal{H}'\, \mathsf{diag}[\mathcal{G}]\, \varPi\, \mathcal{H}\, \mathsf{diag}[\mathcal{B}], \quad (3)\end{aligned}$$

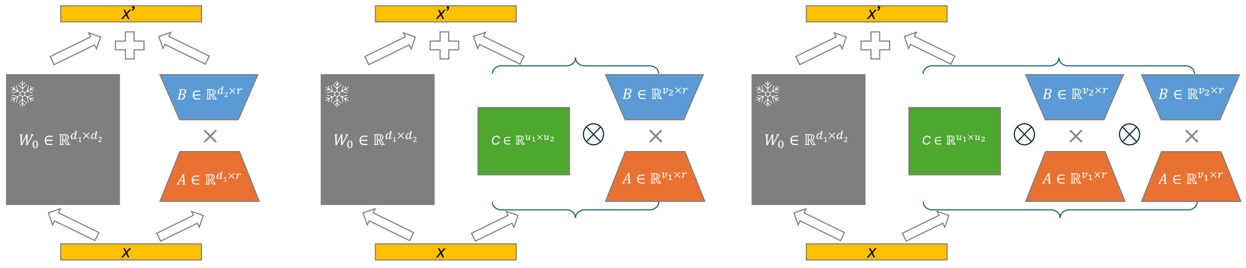
where $\mathsf{vec}[\cdot]$ is a vectorization operator, $\mathsf{diag}[\cdot]$ denotes a diagonalization operator, $\mathcal{H}$ is Walsh–Hadamard matrix, $\mathcal{H}'$ is its truncated version, $\mathcal{G}$ is a random vector drawn from normal distribution, $\varPi$ is a random permutation matrix for shuffling, and $\mathcal{B}$ is a random vector drawn from Rademacher distribution. It is a fast Johnson–Lindenstrauss transform with log-linear complexity due to the fast Walsh–Hadamard transform, and no additional parameters are required when the random seed is predetermined. Further, a nonlinear function such as tanhshrink can be added to make this layer nonlinear. To avoid introducing extra parameters for the projection layer, weights of this projection layer is reproduced on the fly with a known random seed and fixed during training and inference.

Shuffling: Another simple projection is to use a shuffling function without compression. It can be achieved by simplifying the fastfood projection without $\mathcal{H}$, $\mathcal{H}'$, $\mathcal{G}$, and $\mathcal{B}$ but with the random permutation $\varPi$ and projection ratio $\rho = 1$. As SuperLoRA updates all weights at once, we have a flexibility in a way to distribute $\Delta W_{\text{group}_g}$ towards which element of W. To understand how the weight assignment method impacts, we consider a random shuffling case for the projection function $\mathcal{F}$. Several projection variants including shuffling are discussed in Appendix A.10.

A.3. Illustration of ViT model in detail

The ViT model that we used for the classification task is adapted from a public codebase[1]. The detailed structure of the ViT is depicted in Figure 9, where we only fine-tune the projection layers for query and value in the Self-Attention modules. In ViT-base, depth (L in Figure 9) is set to be 12 and dimension is 768. The total number of parameters of the ViT base model is 86.6M.

[1] https://github.com/bwconrad/vit-finetune


x'
$W_0 \in \mathbb{R}^{d_1 \times d_2}$
$B \in \mathbb{R}^{d_2 \times r}$
$A \in \mathbb{R}^{d_1 \times r}$
x
x'
$W_0 \in \mathbb{R}^{d_1 \times d_2}$
$C \in \mathbb{R}^{u_1 \times u_2}$
$B \in \mathbb{R}^{v_2 \times r}$
$A \in \mathbb{R}^{v_1 \times r}$
x
x'
$W_0 \in \mathbb{R}^{d_1 \times d_2}$
$C \in \mathbb{R}^{u_1 \times u_2}$
$B \in \mathbb{R}^{v_2 \times r}$
$A \in \mathbb{R}^{v_1 \times r}$
$B \in \mathbb{R}^{v_2 \times r}$
$A \in \mathbb{R}^{v_1 \times r}$
x

(a) LoRA

(b) LoKr

(c) LoNKr (weight-wise)

Figure 7. Overview of (a) LoRA; (b) LoKr; (c) LoNKr (weight-wise).

A.3.1 ImageNet21k to CIFAR-100 transfer

Firstly, we used a ViT model[2] pretrained on ImageNet21k for CIFAR-100 transfer. A new classifier head to match the number of classes from 21k to 100 is added for classifing CIFAR100 dataset. All layers of the pretrained ViT model are frozen except the SuperLoRA parameters and the new classifier head. The result of SuperLoRA for CIFAR-10 is shown in Figure 4, achieving a significant reduction by 3 to 10 folds in the required number of parameters over LoRA.

To exclude extra fine-tuning budget introduced in the classifier head, automatic label matching is used for ViT model pretrained on ImageNet1k. Specifically, ViT-base model pretrained on ImageNet1k is loaded along with the pretrained classifier head. Then, feed all training data from CIFAR-100 into this pretrained model, and corresponding labels of CIFAR-100 in ImageNet1k are obtained by voting. When there is a tie, the label that has larger gap with the second voting label wins the label. If one label is token, the label with second voting is assigned. In this way, all 100 classes in CIFAR-100 get their corresponding labels in ImageNet1k, and the classifier head can be frozen. Other settings are same as experiments above. The results can be found in Figure 8. Compared with Figure 4, 768 × 100 less parameters are fine-tuned, while most conclusions still hold true.

A.3.2 ImageNet1k to CIFAR-10 transfer

For transfer learning from ImageNet1k to CIFAR-10, we used a ViT base model[3] which is pretrained for ImageNet1k. The classifier head is frozen after selecting most relevant labels in ImageNet1k, *i.e.* [404, 436, 94, 284, 345, 32, 340, 510, 867], corresponding to [airliner, humming bird, siamese cat, ox, golden retriever, tailed frog, zebra, container ship, trailer truck]. It fine-tunes with 3000 steps at most and the best accuracy is reported.

Detailed ranks we tested are as follows:

- LoRA (2D): ranks: 1, 2, 4, 6, 8, . . . , 64, 128
- SuperLoRA (2D): groups: 1, 4, 8, 12; ranks: 1, 2, 4, 6, 8, . . . , 64, 128
- SuperLoRA (2D, reshape):
  - groups: 1, 4, 12, ranks: 1, 2, 4, 6, 8, . . . , 64, 128;
  - group 8, ranks: 1, 2, 4, 6, 8, . . . , 24, 28, 32, 36, . . . , 64
- SuperLoRA (LoRTA: 3D, reshape): groups: 1, 4, 8, 12; ranks: 1–6, 8, 10, 12, . . . , 24
- SuperLoRA (LoRTA: 4D, reshape):
  - group 1; ranks: 1–6, 8, 10, 12, . . . , 22
  - group 4; ranks: 1–6, 8, 10, 12, . . . , 16
  - group 8; ranks: 1–6, 8, 10, 12, . . . , 18
  - group 12; ranks: 1–6, 8, 10, 12
- SuperLoRA (LoRTA: 5D, reshape):
  - groups 1, 4, 8; ranks: 1–6, 8
  - group 12; ranks: 1–6

The result of SuperLoRA for CIFAR-10 is shown in Figure 5, achieving a significant reduction by 3 to 10 folds in the required number of parameters over LoRA.

A.4. Illustration of diffusion model in detail

The classifier-free diffusion model [18] that we used for image generation is adapted from a public codebase[4]. Its U-Net structure is illustrated in Figure 10, which contains 21 attention modules, where the number of input/output channels of the attention modules is either 64 or 128. We only fine-tune the query and value projection layers of those attention modules. The total number of parameters of the U-Net base model is 10.42M, including 300 parameters for the class embedding.

A.5. Illustration of grouping mechanism

Figure 2 illustrates several different cases of the grouping mechanism. Figure 2(a) is the conventional weight-wise grouping, used for typical LoRA. Each weight correction,

[2] https://huggingface.co/google/vit-base-patch16-224-in21k

[3] https://huggingface.co/google/vit-base-patch16-224

[4] https://github.com/coderpiaobozhe/classifier-free-diffusion-guidance-Pytorch

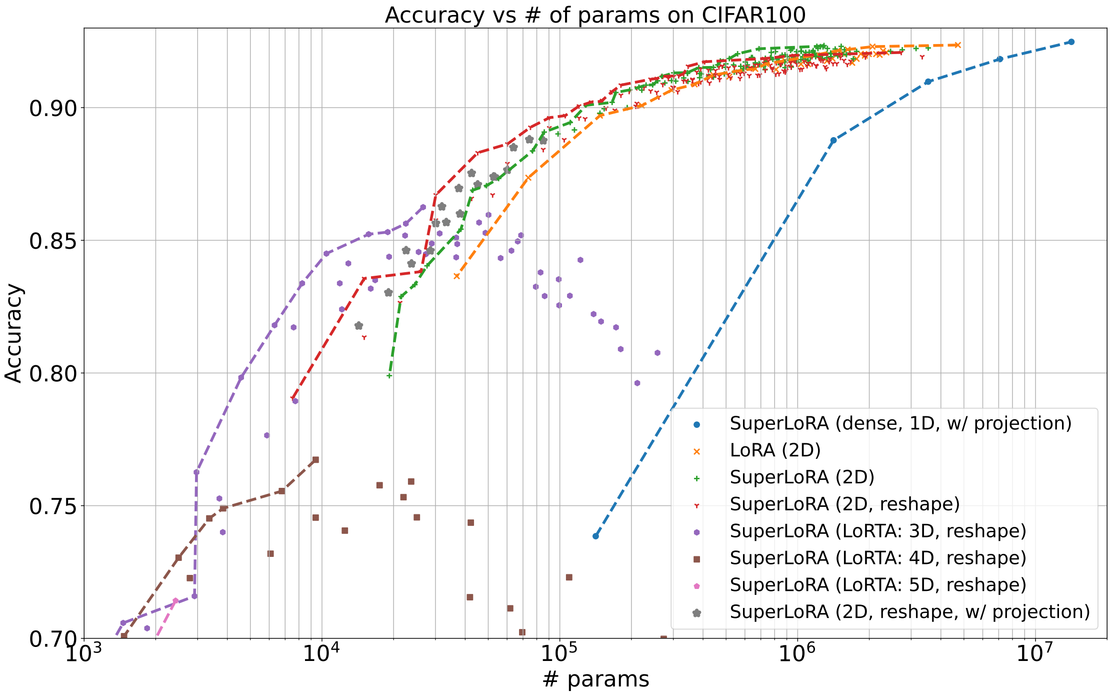


Figure 8. ImageNet1k to CIFAR100 transfer learning with frozen classifier head by automatic label matching.

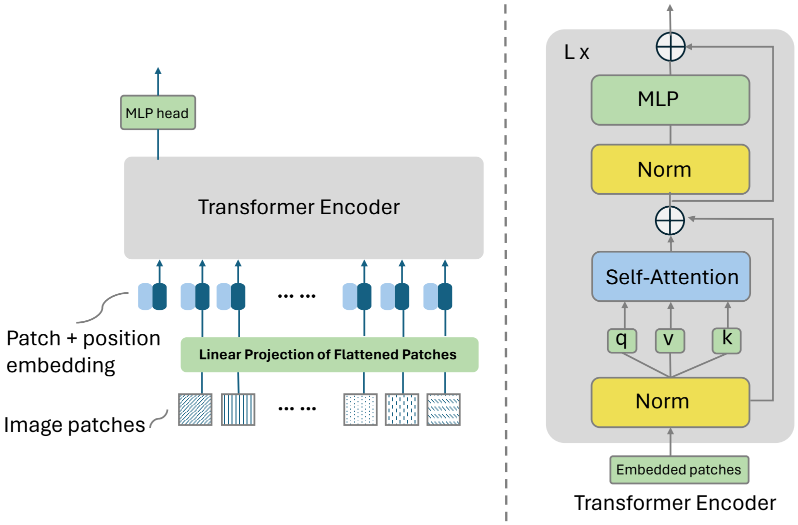


Figure 9. ViT model structure.

i.e. $\Delta W_{\mathrm{v}\ell}$ and $\Delta W_{\mathrm{q}\ell}$ for value and query projections at layer ℓ, is individually represented by a rank-r decomposition: $A_g B_g^\top$ for group g. Figure 2(b) shows layer-wise grouping, where the LoRA unit in each group jointly adapts both value and query projections in each layer. When we stack multiple weight matrices in a naïve way, the 2D array will have unbalanced fan-in/fan-out shape, leading to inefficient low-rank decomposition. Figure 2(c) can solve this issue by reshaping the 2D array into a regular square shape before low-rank decomposition. As the reshaping is

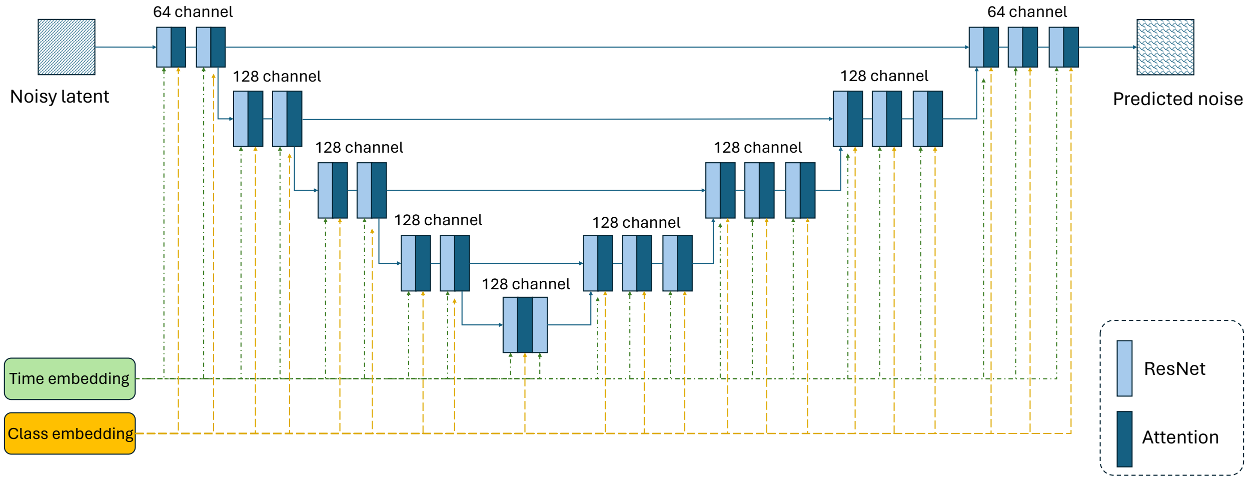


Figure 10. Classifier-free diffusion model structure.

already breaking the geometric meaning of the original 2D weights, the grouping need not necessarily aligned with the weight boundary as shown in a general grouping case of Figure 2(d). Further, applying a projection function $\mathcal{F}(\cdot)$ as shown in Figure 2(e), the element distribution can be shuffled and mixed-up to relax the geometric restriction of original LoRA. LoRTA can further generalize the reshaping by folding the 2D array to any arbitrary M-dimensional tensor array by using the Tucker decomposition as shown in Figure 2(f). Relaxing the geometric constraint can improve the parameter efficiency as shown in this paper. We further make a geometric analysis of our grouping methods.

A.6. Geometric analysis

SuperLoRA adapts multiple attention modules at once, and relaxes the underlying geometric restrictions inherent to the 2D weights for each attention module, by employing grouping, reshaping, and projection (including shuffling). To better understand how SuperLoRA works differently from LoRA, geometric analysis is conducted for the classification task. Specifically, we pick 4 different methods with a comparable number of parameters around 100,000:

- LoRA (2D): #param 147,456, accuracy 0.9113;
- SuperLoRA (2D): #param 115,200, accuracy 0.9170;
- SuperLoRA (2D, reshape): #param 165,572, accuracy 0.9218;
- SuperLoRA (2D, reshape, w/ projection): #param 138,372, accuracy 0.9213.

The weight correction term ΔW is compared to the full dense FT case, which involves 14M parameters achieving an accuracy of 0.9290. We analyze three different geometric measures with respective to the FT weight ΔW_{dense}: i) left-singular similarity; ii) right-singular similarity; and iii) Euclidean distance. Letting U and V denote the left- and right-singular vectors of ΔW for each variant listed above, these metrics are defined as follows:

$$d_{\mathrm{L}} = \frac{1}{\sqrt{k}} \| U_{\text{dense}}[:, :k]^{\top} U_{\text{variant}}[:, :k] \|_2, \tag{4}$$

$$d_{\mathrm{R}} = \frac{1}{\sqrt{k}} \| V_{\text{dense}}[:k, :] V_{\text{variant}}[:k, :]^{\top} \|_2, \tag{5}$$

$$d_{\mathrm{E}} = \frac{\| \Delta W_{\text{dense}} - \Delta W_{\text{variant}} \|_2}{\| \Delta W_{\text{dense}} \|_2}. \tag{6}$$

Note that d_{E} approaches to 0 when ΔW converges to the dense FT case, while d_{L} and d_{R} converge to 1.

The top $k = 5$ principal singular vectors are analyzed as shown in Figure 11. The ViT model has 12 attention modules, and we plot the total of 24 points for the query and value projection weights. The first row shows the query weights for d_{L} *vs.* d_{R}, d_{E} *vs.* d_{R}, and d_{E} *vs.* d_{L} from left to right across the columns. The second and third rows are for the value weights, and both query and value weights, respectively.

We see that the Euclidean distance d_{E} is significantly decreased for SuperLoRA, especially with reshaping applied. It explains the improved accuracy with reshaping. Although grouping, reshaping, and projection can break the geometric meaning of the original 2D weights, the subspace similarity is not completely lost. Especially for query weights, SuperLoRA shows higher right-singular similarity than LoRA. As the embedding vector passes through right-hand side of the weight, principal right-singular vectors perform as a low-rank subspace mapping of the input vector while the left-singular vectors work as mapping the subspace towards the output vector. While SuperLoRA with reshaping tends to preserve higher right-singular similarity, LoRA tends to preserve higher left-singular similarity. Further, it is found that the corrections for query and value weights behave differently with reshaping, *i.e.*, right-singular similarities for the value weights are much larger than for query weights.

A.7. Grouping effect on SuperLoRA (1D, dense, with projection)

As the fixed projection matrix is shared across all groups, the number of groups will affect the size of the projection matrix directly. To explore this influence, dense FT with projection is tested for different splitting, from 1 to 12 groups. According to Figure 12, using 1 group achieves the best overall accuracy and using 4 or 8 groups are comparable to a smaller projection ratio. When the projection matrix is too small, *e.g.*, with 12 groups, accuracy drops greatly. This confirms that jointly updating multiple attention modules is beneficial.

A.8. Image generation transfer task

A.8.1 Settings:

For the image generation task, SuperLoRA is evaluated by transfer learning between SVHN [34] and MNIST datasets [29]. Both datasets have 10 classes corresponding to images of the digits 0 to 9, where the SVHN images have a more complicated color background, while the MNIST images are nearly black-and-white with a plane black background. We mainly work on the transfer learning from SVHN to MNIST. The reverse transfer learning from MNIST to SVHN is discussed in Appendix A.12.

The model we worked on is a classifier-free diffusion model [18] and the correction weights from LoRA variants are added to query and value projection matrices in the attention modules of U-Net backbone [36]. Note that the size of projection weights differs across layers for this U-Net structure, which allows us to examine the performance of SuperLoRA after breaking the boundaries of different weight matrices. More details of the diffusion model are described in Appendix A.4. For comparison, the original weight-wise LoRA and dense FT are also evaluated. For SuperLoRA variant, LoRA, LoNKr and LoRTA consider three versions: weight-wise, group-wise and group-reshaped. The scaling factor α of LoRA is fixed to 2.0 for all variants unless specified. 40 epochs with a batch size of 32 are carried out and results plotted are mainly from epoch 20 noticing convergence becomes stable around epoch 20. The maximum rank is set to 32 by default and a constraint $r < \min(d_1, d_2)$ is imposed. To evaluate the quality of images generated by the fine-tuned diffusion model, we consider several metrics including Inception Score (IS) [37], Fréchet Inception Distance (FID) [17], Multi-Scale Intrinsic Distance (MSID) [40], Kernel Inception Distance (KID) [6], Recall and Precision [27]. Except for the recall and precision metrics, all metrics should be lower for higher-quality image generations. As we found ℓ_1-distance based IS is more consistent to the perceptual visual quality, we mainly focus on IS metric results in the main content, while the results for other metrics can be found in Appendix A.11. For following figures, Pareto frontier lines/dots are mainly shown to provide the limit of each method, while Appendix provides more complete figures with all data points.

A.8.2 Grouping effect:

First, we evaluated how splitting all ΔW_{all} into multiple groups affects the performance. Figure 13 shows the results of dense, original weight-wise LoRA and group-wise SuperLoRA with different number of groups. Sweeping the rank and the number of groups, we plot the image quality metrics in y-axis and the required number of trainable parameters in x-axis. Pareto frontier lines/data points are also shown in the figure.

Figure 13 shows that the dense FT for ΔW presents the best IS, while requiring most parameters. Original weight-wise LoRA is closest to dense, in terms of both IS and parameter amount. However, in low-parameter regimes, SuperLoRA (2D, group1) shows the best results compared with other grouping. While in the middle of parameter amount axis, other splittings including groups $G = 8$ and 12 show slightly better IS compared with LoRA. Besides, splitting ΔW_{all} shows much more data points compared with both LoRA and dense, providing us higher flexibility to adjust the trade-off between quality and parameter efficiency especially when the memory resource is limited.

A.8.3 Reshaping effect:

To evaluate the importance of reshaping, we compare group-wise SuperLoRA with and without reshaping in Figure 14. For weight-wise LoRA, most weight matrices corrected are square already. For all splitting with groups $G = 1, 4$ and 8, we confirmed that reshaping shows smaller number of parameters and better IS compared with their corresponding non-reshaping counterparts. This indicates that reshaping ΔW to regular tensor array (square, cube, and hyper-cube) is vital for SuperLoRA fine-tuning to prevent unbalanced skew tensors when adapting multiple weights at once.

A.8.4 LoKr vs. LoNKr:

In 2D ΔW, we also compared LoKr with our proposed extension LoNKr, a variant of SuperLoRA. We evaluated LoNKr when the number of splits is $K \in \{2, 3, 4\}$, where

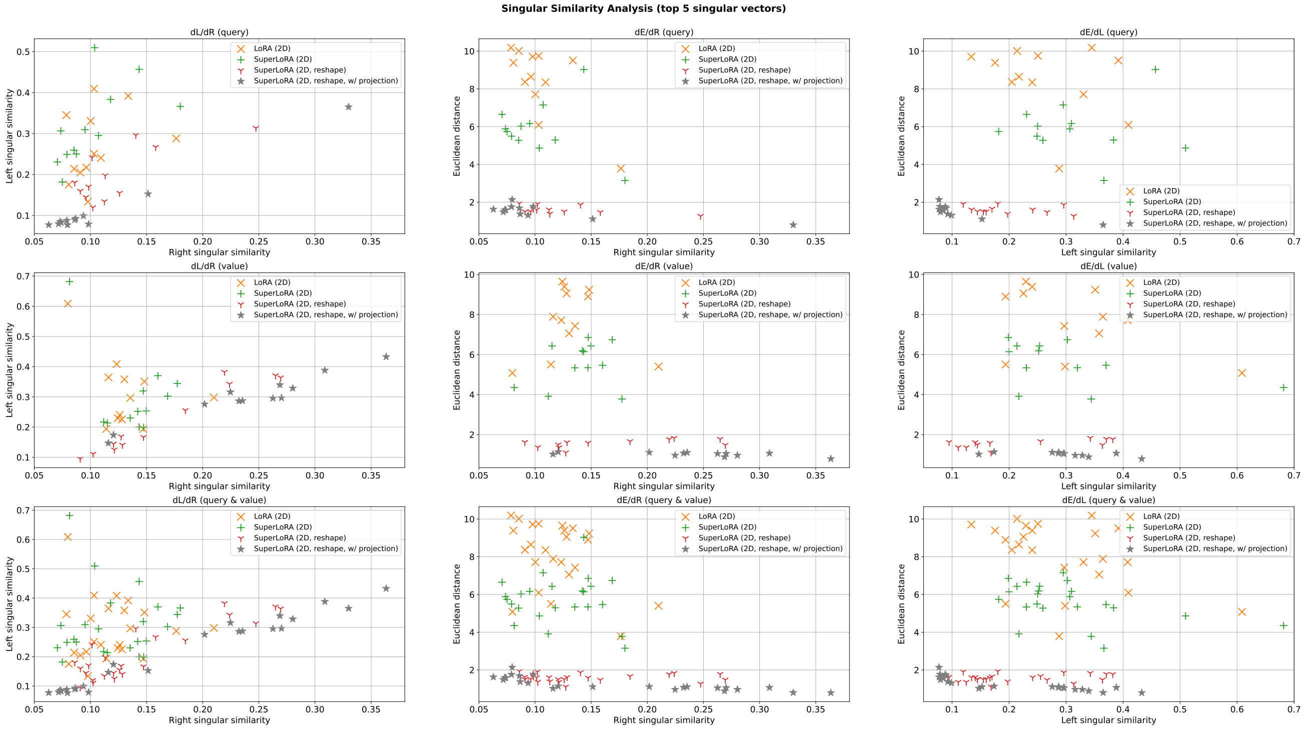


Figure 11. Geometric similarity analysis (top 5 principal singular vectors).

$K = 2$ corresponds to the original LoKr. For the dense factor on the left in LoNKr/LoKr as shown in Figure 7c, dimension is fixed to 6, 8 or 10. Figure 15 shows that more splits provide us more choices in low-parameter regimes, especially for group-wise LoNKr. LoNKr shows much more data points and better IS when the number of parameters is less than $5{,}000$. And the least parameter for LoKr and LoNKr dropped greatly from 500 to 150.

A.8.5 LoRTA:

LoRTA reshapes ΔW_{all} to high-order tensor. We evaluated 3D, 4D and 5D, as data points become much less when the dimension is too small for all planes when order is larger than 5D. From Figure 16, the higher the order of tensor folding, the less data points we have. In both weight-wise and group-wise version, 5D LoRTA reduces the least parameter it requires. Especially for group-wise LoRTA, 5D LoRTA requires less than 80 parameters to produce a result compared with beyond 1000 for 2D LoRTA and beyond 200 for 3D LoRTA, while original LoRA needs about 10^4 parameters, about 120-fold more parameters. To achieve a comparable IS of LoRA having 10^4 parameters, LoRTA (3D) just needs 2×10^3 parameters, *i.e.* 5-fold reduction.

A.8.6 Projection effect:

SuperLoRA can use a projection layer $\mathcal{F}$ which is randomly initialized but fixed at both finetuning and inference. Linear fastfood projection and nonlinear projection with tanshrink applied after the linear projection matrix are evaluated. Besides, a modified version of fastfood projection with random Gaussian instead of random binary $\mathcal{B}$ is also tested for both linear and nonlinear versions, denoted as linear$_{\text{v2}}$ and nonlinear$_{\text{v2}}$ respectively. The projection matrix is shared across all groups. We evaluated number of groups $G \in \{1, 4\}$, rank $r \in \{1, 4, 8\}$ and projection ratio $\rho \in \{0.01, 0.1, 0.5\}$ on SuperLoRA (2D, reshape) and SuperLoRA (LoRTA, reshape) for 3D, 4D and 5D tensor.

Figure 17 demonstrates with smaller projection ratio, required parameters for both SuperLoRA (2D, reshape) and SuperLoRA (LoRTA, group-wise) are pushed to extremely low-parameter regimes. The least parameter required becomes only about 30, compared with 10,000 for original LoRA. Surprisingly, linear version for both methods shows better performance than nonlinear version which are attached in Appendix A.10. Besides, in extremely low-parameter regimes, higher rank with projection layer for SuperLoRA (LoRTA, group-wise) works better than small ranks itself, showing promising direction to explore pro-

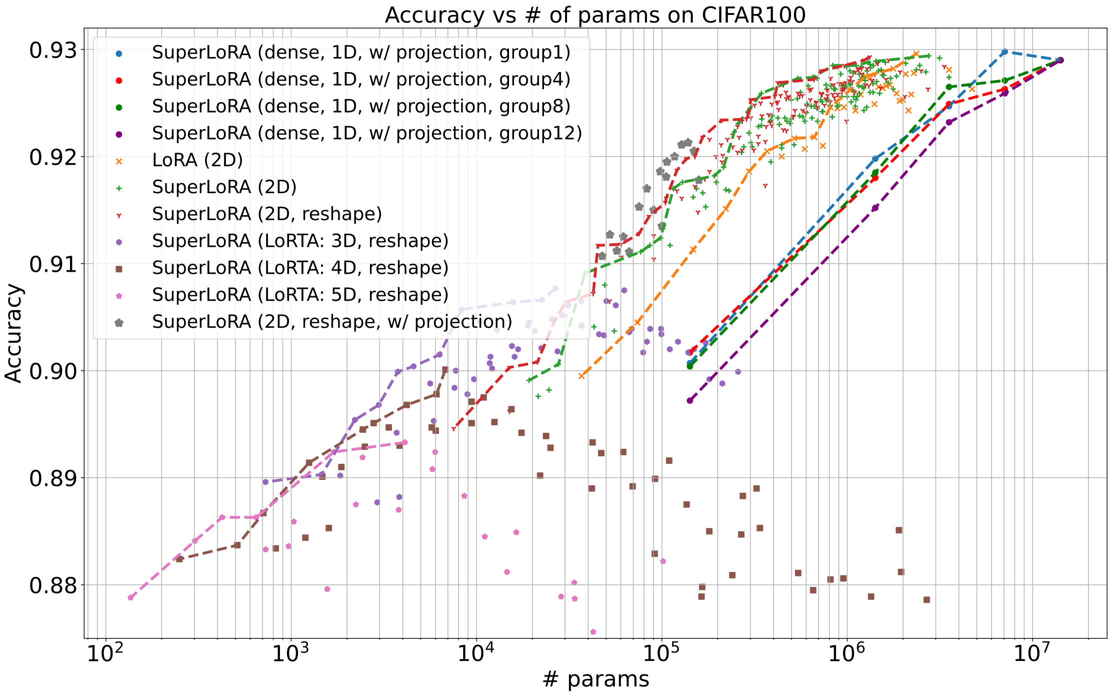


Figure 12. More groups (*i.e*. less fixed projection parameters) on SuperLoRA (1D, dense, w/ projection).

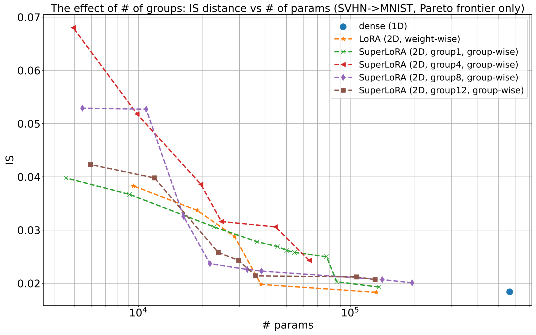


Figure 13. weight-wise *vs*. group-wise

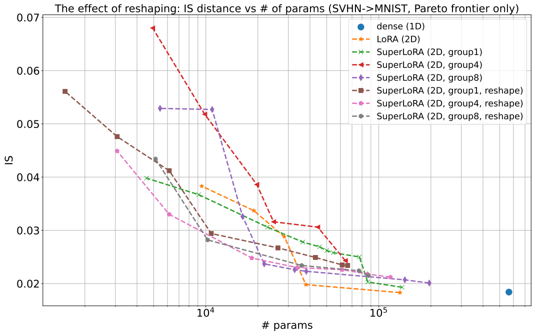


Figure 14. reshaping *vs*. non-reshaping

jection layer in extremely low-parameter regime. In terms of linear *vs*. $\text{linear}_{\text{v2}}$, $\text{linear}_{\text{v2}}$ shows better performance in higher-parameter area while linear works better in lower-parameter area, even better than SuperLoRA (LoRTA) without projection.

A.8.7 Shuffling effect:

As another simple projection, we studied a random shuffling to distribute ΔW_{group} before adding it to corresponding W. We evaluated SuperLoRA (2D) and SuperLoRA (2D, reshape) with/without shuffling for groups $G \in \{1, 4, 8, 16]\}$ and ranks $r \in \{1, 4, 8\}$, where the shuffled indexes are shared across all groups. The shuffling corresponds to one of fastfood projection modes by setting projection ratio to $\rho = 1$ with only permutation matrix $\mathit{\Pi}$. As shown in Figure 18, shuffling inside groups had no harm on IS. It even improved IS for SuperLoRA (2D) in most cases.

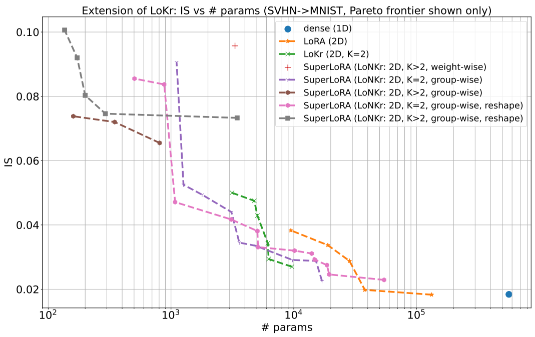


Figure 15. SuperLoRA (LoNKr)

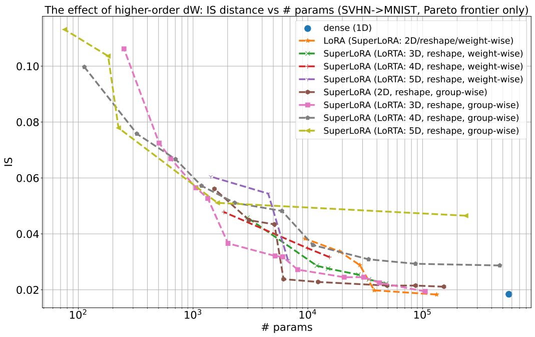


Figure 16. SuperLoRA (LoRTA)

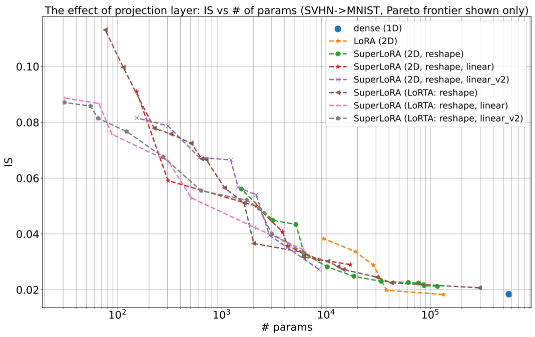


Figure 17. fixed random projection within group

A.9. Visualization

To better understand the superiority of SuperLoRA, especially in low-parameter regimes, we visualize a set of generated images from SuperLoRA, as well as dense FT and LoRA, from a range of parameter setting: high-parameter ($>$ 70,000), middle-parameter (from 5,000 to 10,000), low-parameter (around 1,000) and extremely-low parameter ($<$ 100) regimes. We selected one image with the best IS for each hyper-parameter setting we have tested under same level of parameter amount. Figure 19 shows that all generated images by the transfer learning model from SVHN to MNIST are close to images from MNIST dataset itself with black-white background, removing most domain information of color SVHN. SuperLoRA (2D, group8, rank13) in Figure 19c shows competitive results with LoRA (rank8) using $5{,}000$ less parameters.

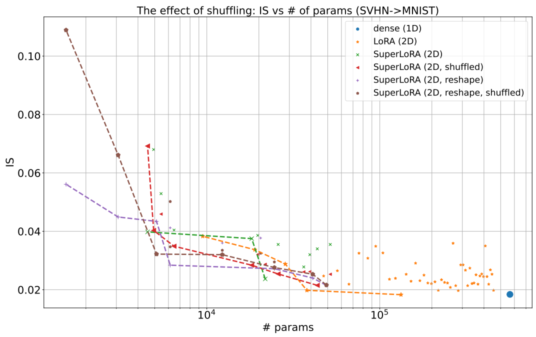


Figure 18. fixed random shuffling within group

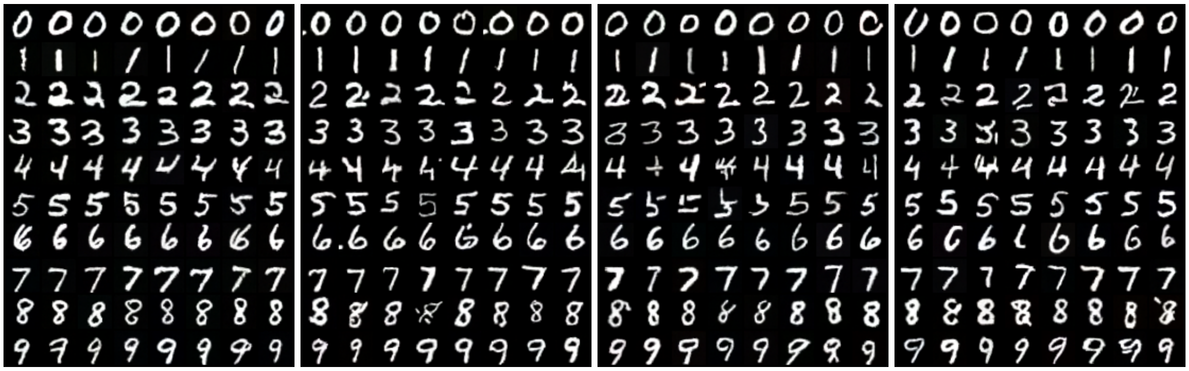
For the middle-parameter regimes, Figure 20 shows visualization of LoNKr, SuperLoRA (2D, reshape), LoRTA (3D, reshape), LoRTA (4D, reshape) and SuperLoRA (2D, reshape, projection). More domain information with colorful digits and background occur occasionally. There are also some missing digits presented in middle-parameter area.

When the number of parameters is as low as $1{,}000$, even though only few choices left like LoNKr and LoRTA, one can always stretch hyper-parameter settings from middle-parameter level coupled with fixed linear projection layer to compress the tensor size. In this way, the strength of middle-parameter level gets extended to low-parameter area. As shown in Figure 21, compared with the visualization from middle-parameter results, more missing digits and more colorful backgrounds are presented.

Finally, we also visualized a few images from extremely-low parameter level less than 100 in Figure 21. Surprisingly, domain transfer in those images is somewhat realized from SVHN to MNIST even with such an extremely few parameter case such as 31, which is more than four orders of magnitude smaller than dense FT.

A.10. Linear *vs*. nonlinear projection

Besides linear projection, we also examined nonlinear projections. We use the "tanhshrink" operation, denoted by $\mathtt{tanhs}(x) := x - \tanh(x)$, after the fixed linear projection, resulting in the 'nonlinear' and 'nonlinear$_{\mathrm{v2}}$' variants. Note that the 'v2' projection uses a Gaussian random vector rather than a binary random vector $\mathcal{B}$ for the fastfood projection as shown in Figure 3. More specifically, we consider



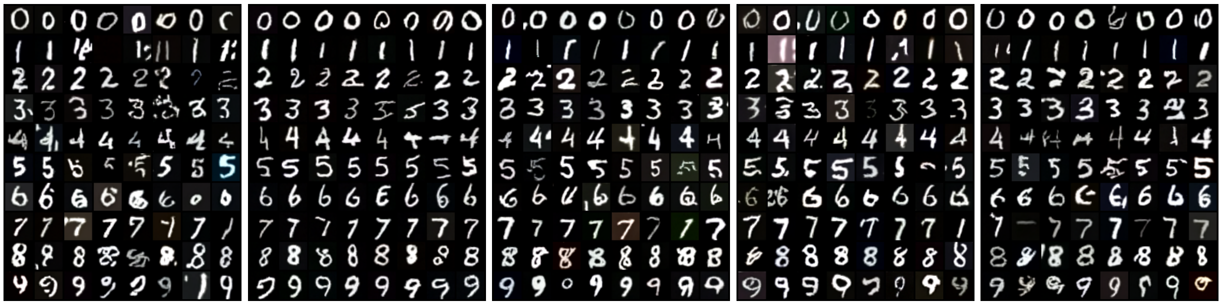
(a) dense
#param 565,248
IS 0.0184

(b) LoRA $r = 8$
#param 75,776
IS 0.03025

(c) SuperLoRA
#param 70,720
IS 0.0305

(d) SuperLoRA
#param 73,728
IS 0.0263

Figure 19. Visualization of generated images under high-parameter level ($>$ 70,000).



(a) LoNKr
#param 5,112
IS 0.036

(b) SuperLoRA
#param 10,752
IS 0.0294

(c) LoRTA
#param 8,160
IS 0.0272

(d) LoRTA
#param 11,100
IS 0.036

(e) SuperLoRA w/ p
#param 8,512
IS 0.0273

Figure 20. Visualization of generated images under middle-parameter level ([5,000, 20,000]).

six variants for the projection function $\mathcal{F}(\cdot)$ in this paper:

- identity (no projection): $\mathcal{F}(x) = x$;
- shuffling: $\mathcal{F}(x) = x\,\mathit{\Pi}$;
- linear: $\mathcal{F}(x) = x\,\mathcal{H}'\,\mathsf{diag}[\mathcal{G}]\,\mathit{\Pi}\,\mathcal{H}\,\mathsf{diag}[\mathcal{B}]$;
- linear$_{\text{v2}}$: $\mathcal{F}(x) = x\,\mathcal{H}'\,\mathsf{diag}[\mathcal{G}]\,\mathit{\Pi}\,\mathcal{H}\,\mathsf{diag}[\mathcal{G}']$;
- nonlinear: $\mathcal{F}(x) = \mathsf{tanhs}\big[x\,\mathcal{H}'\,\mathsf{diag}[\mathcal{G}]\,\mathit{\Pi}\,\mathcal{H}\,\mathsf{diag}[\mathcal{B}]\big]$;
- nonlinear$_{\text{v2}}$: $\mathcal{F}(x) = \mathsf{tanhs}\big[x\,\mathcal{H}'\,\mathsf{diag}[\mathcal{G}]\,\mathit{\Pi}\,\mathcal{H}\,\mathsf{diag}[\mathcal{G}']\big]$.

Here, $\mathit{\Pi}$ performs a random permutation of a vector. The Walsh–Hadamard matrices $\mathcal{H}' \in \mathbb{R}^{N_{\text{in}} \times 2^N}$ and $\mathcal{H} \in \mathbb{R}^{2^N \times N_{\text{out}}}$ are left- and right-truncated versions of a regular Walsh–Hadamard matrix $\mathcal{H}_2^{\otimes N} \in \mathbb{R}^{2^N \times 2^N}$, where $[\cdot]^{\otimes N}$ denotes N-fold Kronecker power and $\mathcal{H}_2 = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right]$. Letting N_{in} and N_{out} be the number of elements for the input and output of the projection function $\mathcal{F}(\cdot)$ with a compression ratio of $\rho = N_{\text{in}}/N_{\text{out}}$, the exponent N is chosen as $N = \text{ceil}[\log_2(\max(N_{\text{in}}, N_{\text{out}}))]$. In practice, the left-truncated Walsh–Hadamard matrix is realized by input zero-padding before fast Walsh–Hadamard transform. The random vector $\mathcal{G}$ is hence of size 2^N, and drawn from the normal distribution. Here, $\mathcal{G}' \in \mathbb{R}^{N_{\text{out}}}$ is another random vector drawn from the normal distribution while $\mathcal{B} \in \{\pm 1\}^{N_{\text{out}}}$ is a random vector drawn from the Rademacher distribution.

Figure 22 shows the comparison of several projection variants. Surprisingly, with the same number of parameters, the linear version outperforms the nonlinear versions in most cases.

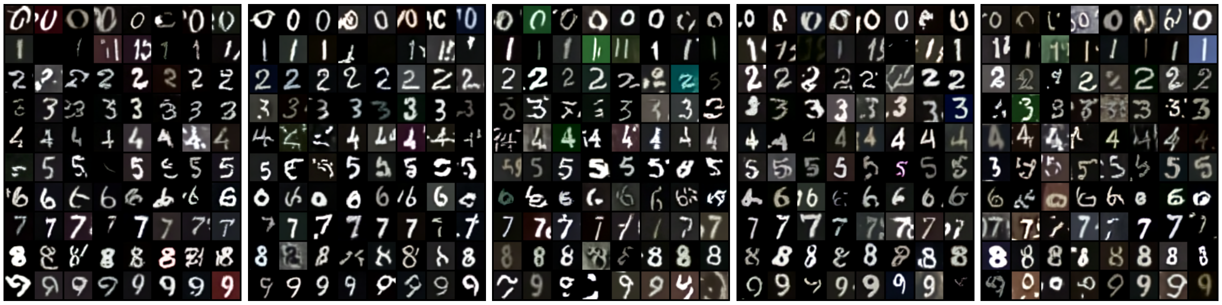
A.11. Transfer learning from SVHN to MNIST

A.11.1 Grouping effect (complete results)

Scatter plots of all metrics (FID, IS, KID, MSID, Improved Precision and Improved Recall) are given in Figure 23. Except IS, all metrics show many examples performing better than dense FT, while worse according to the visualization results, indicating IS is a more reasonable quantitative metric in this case.

A.11.2 Reshaping effect (complete results)

Complete results for reshaping, with scatter plots for all metrics, are shown in Figure 24.



(a) LoNKr
#param 1,080
IS 0.0471

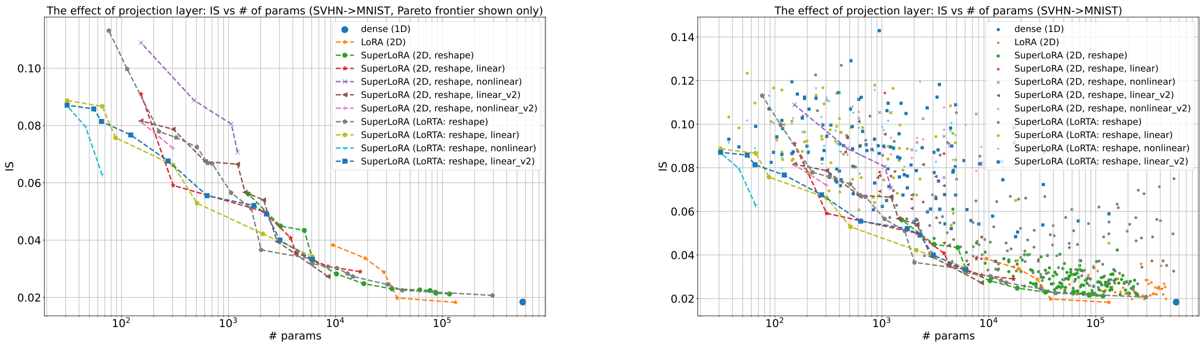
(b) LoRTA
#param 1,060
IS 0.0565

(c) SuperLoRA w/ p
#param 832
IS 0.0607

(d) LoRTA
#param 76
IS 0.1131

(e) LoRTA w/ p
#param 31
IS 0.0871

Figure 21. Visualization of generated images under low-parameter level (1,000) and extremely-low level (< 100).



(a) linear *vs*. nonlinear (IS, Pareto only)

(b) linear *vs*. nonlinear (IS, all points)

Figure 22. Comparison between Linear/Linear$_{\text{v2}}$/Nonlinear/Nonlinear$_{\text{v2}}$ projections.

A.11.3 SuperLoRA (LoNKr, complete results)

Complete results for SuperLoRA (LoNKr), with scatter plots for all metrics, are shown in Figure 25.

A.11.4 SuperLoRA (LoRTA, complete results)

Complete results for SuperLoRA (LoRTA), with scatter plots for all metrics, are shown in Figure 26.

A.12. Transfer learning from MNIST to SVHN

A.12.1 Grouping effect

Transfer learning from MNIST to SVHN is also tested. Figure 27 shows that some metrics cannot function when transferred from a simpler dataset to a more complicated one, *e.g*. FID, IS, KID and Improved Precision, where some ill-posed cases appear. Besides this, we can still find from the Pareto frontiers that SuperLoRA extends LoRA to low-parameter regime and works better occasionally in terms of IS, MSID, Improved Precision and Improved Recall.

A.12.2 Reshaping effect

Figure 28 shows that SuperLoRA with reshaping works better than non-reshaping in most cases in transfer learning from MNIST to SVHN, consistent with the results in transfer learning from SVHN to MNIST.

A.12.3 SuperLoRA (LoNKr)

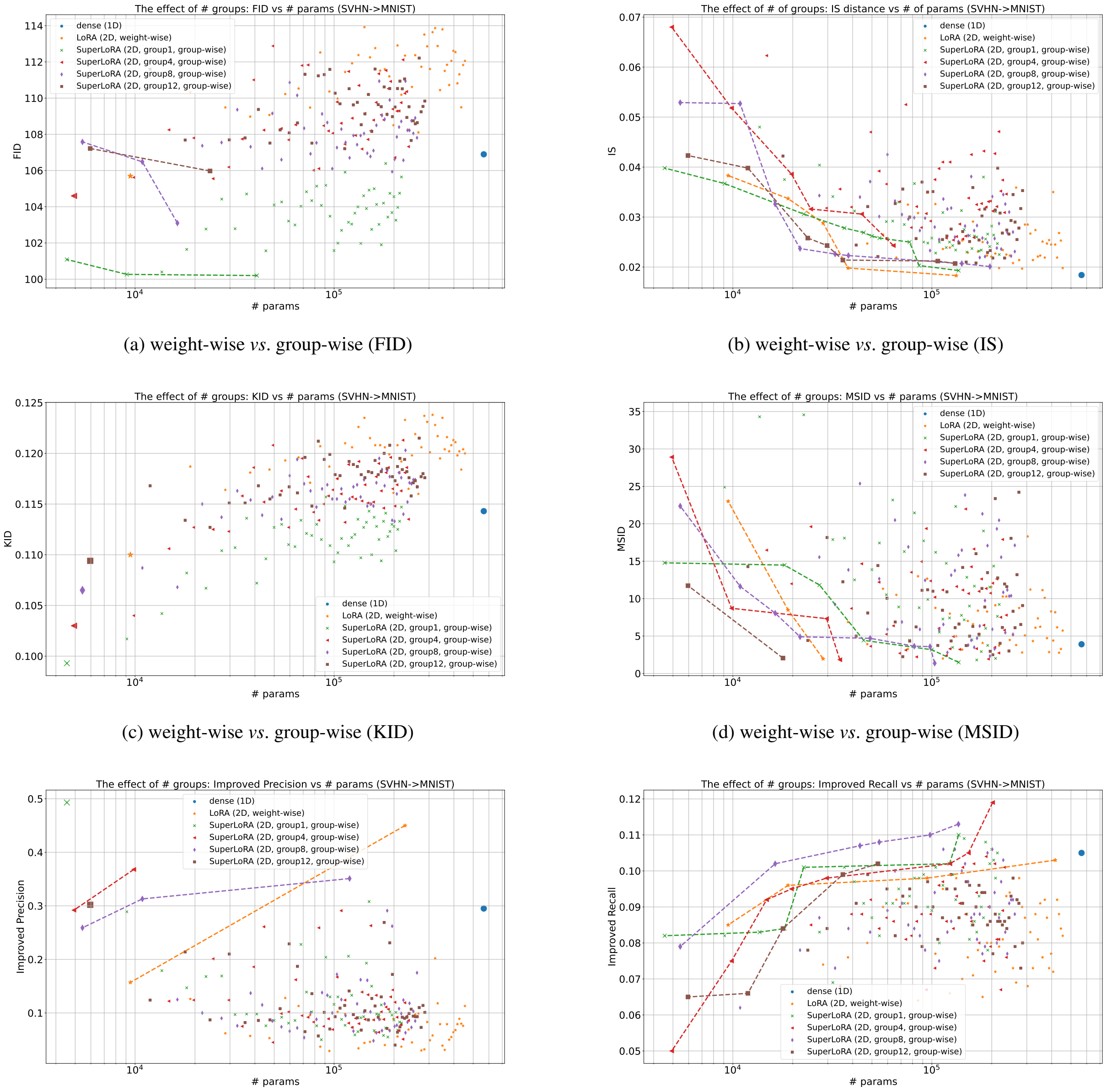
Figure 29 demonstrates the results of SuperLoRA (LoNKr). From MSID figure, we can see that, LoNKr extends LoKr to low-parameter regime, and achieves a better MSID.

A.12.4 SuperLoRA (LoRTA)

From FID and KID in Figure 30, LoRTA pushes required parameters from 10^4 to 10^2 compared with LoRA, providing more flexibility when the memory is limited.

A.13. Effect of groups in LoNKr and LoRTA

From Figure 31 and Figure 32, LoNKr and LoRTA behave differently in terms of the number of groups: for LoNKr, fewer groups are better (than more groups) in



(a) weight-wise *vs*. group-wise (FID)

(b) weight-wise *vs*. group-wise (IS)

(c) weight-wise *vs*. group-wise (KID)

(d) weight-wise *vs*. group-wise (MSID)

(e) weight-wise *vs*. group-wise (Improved Precision)

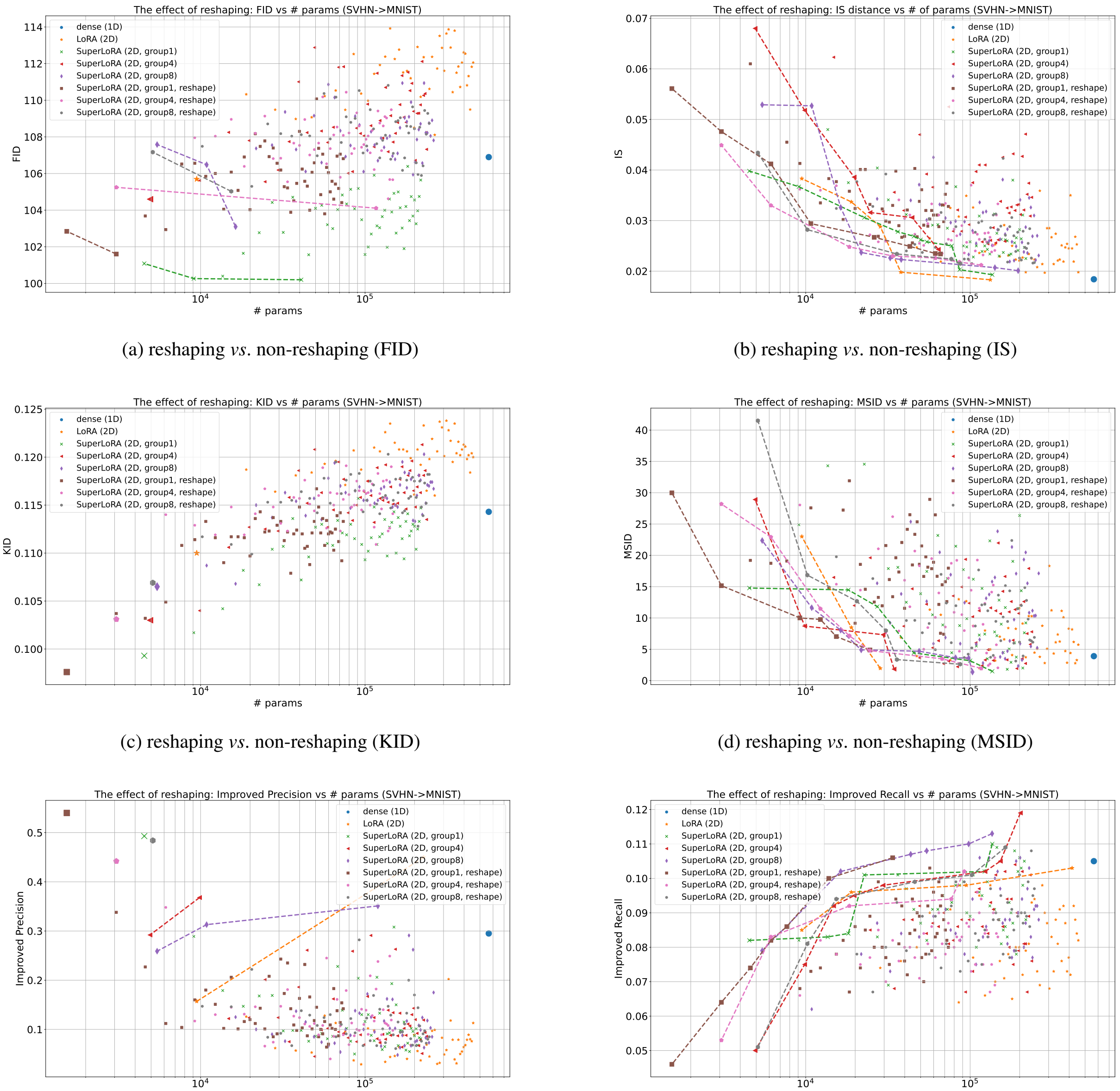
(f) weight-wise *vs*. group-wise (Improved Recall)

Figure 23. Complete comparison between weight-wise LoRA and group-wise SuperLoRA.

low-parameter regime, while they are comparable in high-parameter regime. However, LoRTA prefers less groups.

A.14. Effect of split K in LoNKr

As shown in Figure 33, larger K works better than smaller ones in the low-parameter regime.



(a) reshaping *vs*. non-reshaping (FID)

(b) reshaping *vs*. non-reshaping (IS)

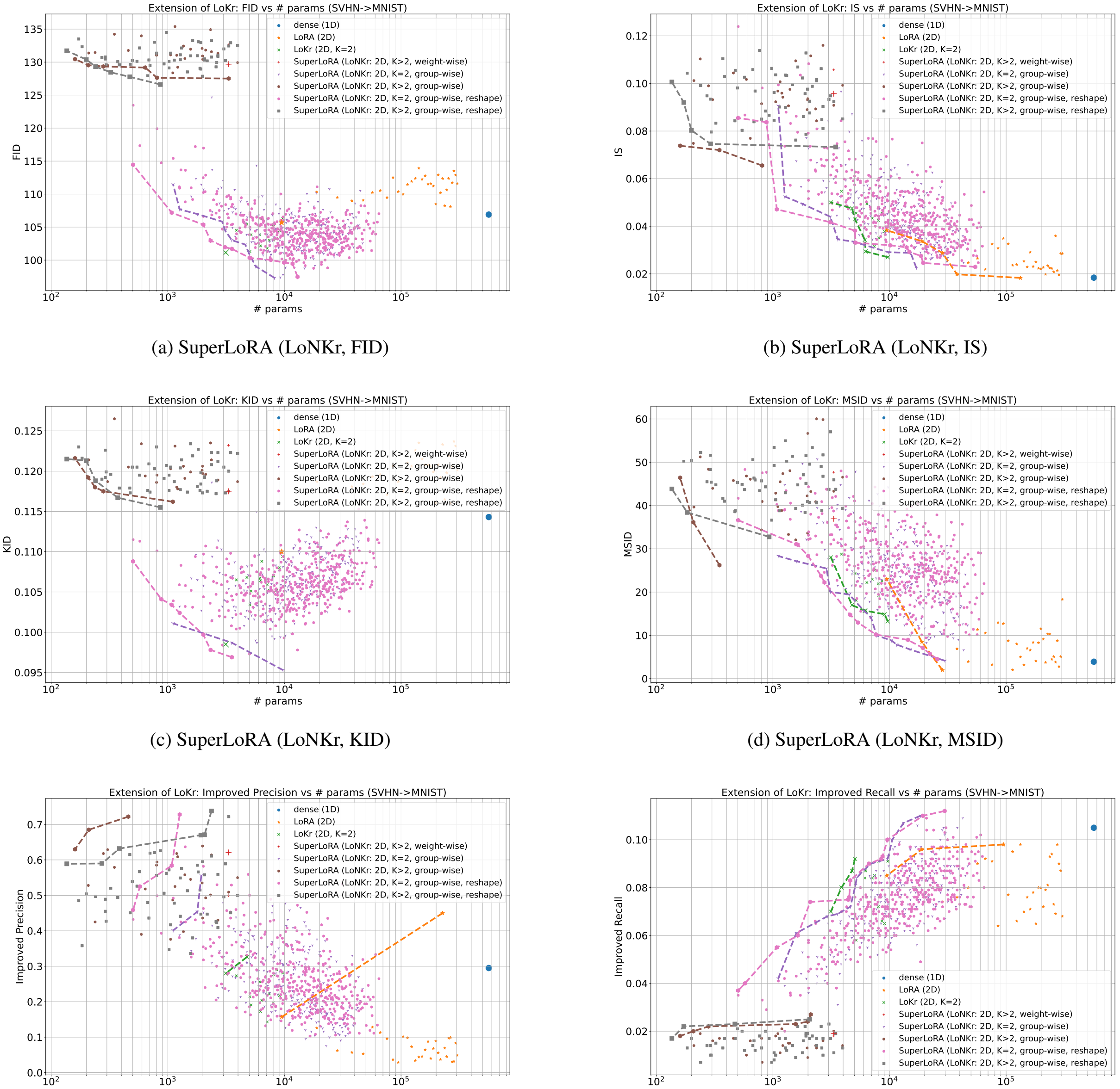
(c) reshaping *vs*. non-reshaping (KID)

(d) reshaping *vs*. non-reshaping (MSID)

(e) reshaping *vs*. non-reshaping (Improved Precision)

(f) reshaping *vs*. non-reshaping (Improved Recall)

Figure 24. Complete comparison between reshaping and non-reshaping SuperLoRA.



(a) SuperLoRA (LoNKr, FID)

(b) SuperLoRA (LoNKr, IS)

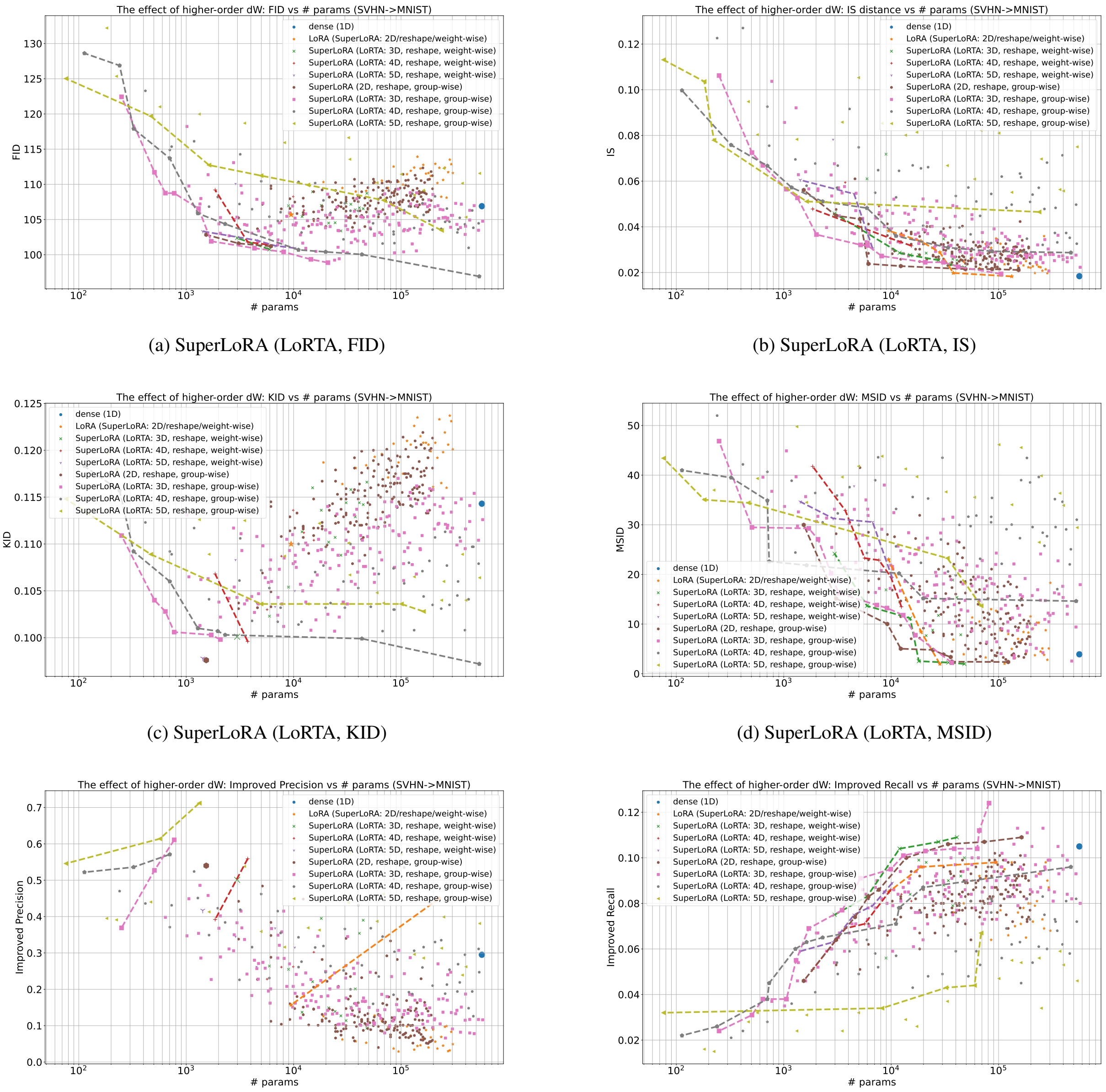
(c) SuperLoRA (LoNKr, KID)

(d) SuperLoRA (LoNKr, MSID)

(e) SuperLoRA (LoNKr, Improved Precision)

(f) SuperLoRA (LoNKr, Improved Recall)

Figure 25. Complete results for LoNKr.



(a) SuperLoRA (LoRTA, FID)

(b) SuperLoRA (LoRTA, IS)

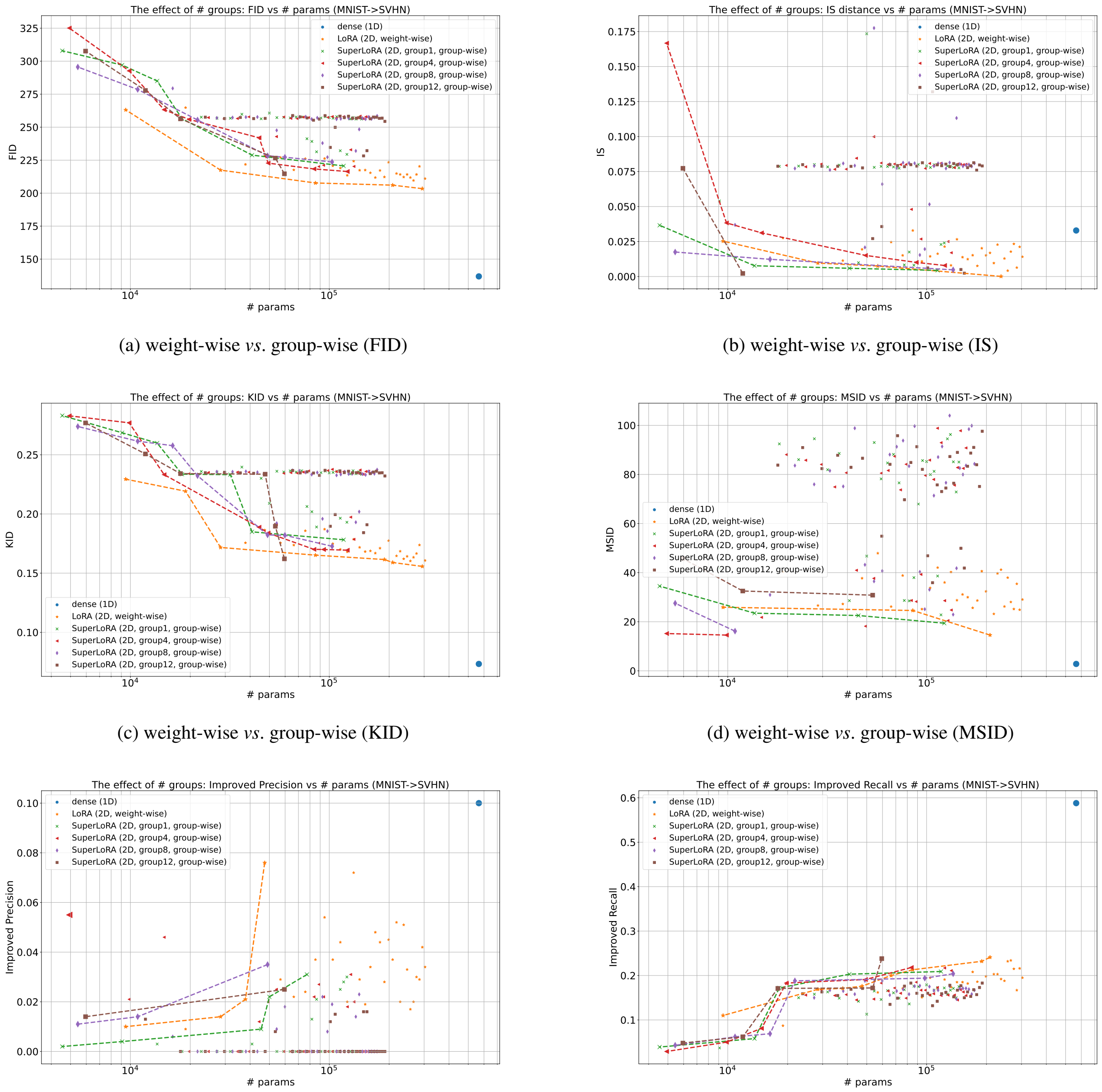
(c) SuperLoRA (LoRTA, KID)

(d) SuperLoRA (LoRTA, MSID)

(e) SuperLoRA (LoRTA, Improved Precision)

(f) SuperLoRA (LoRTA, Improved Recall)

Figure 26. Complete results for LoRTA.



(a) weight-wise *vs*. group-wise (FID)

(b) weight-wise *vs*. group-wise (IS)

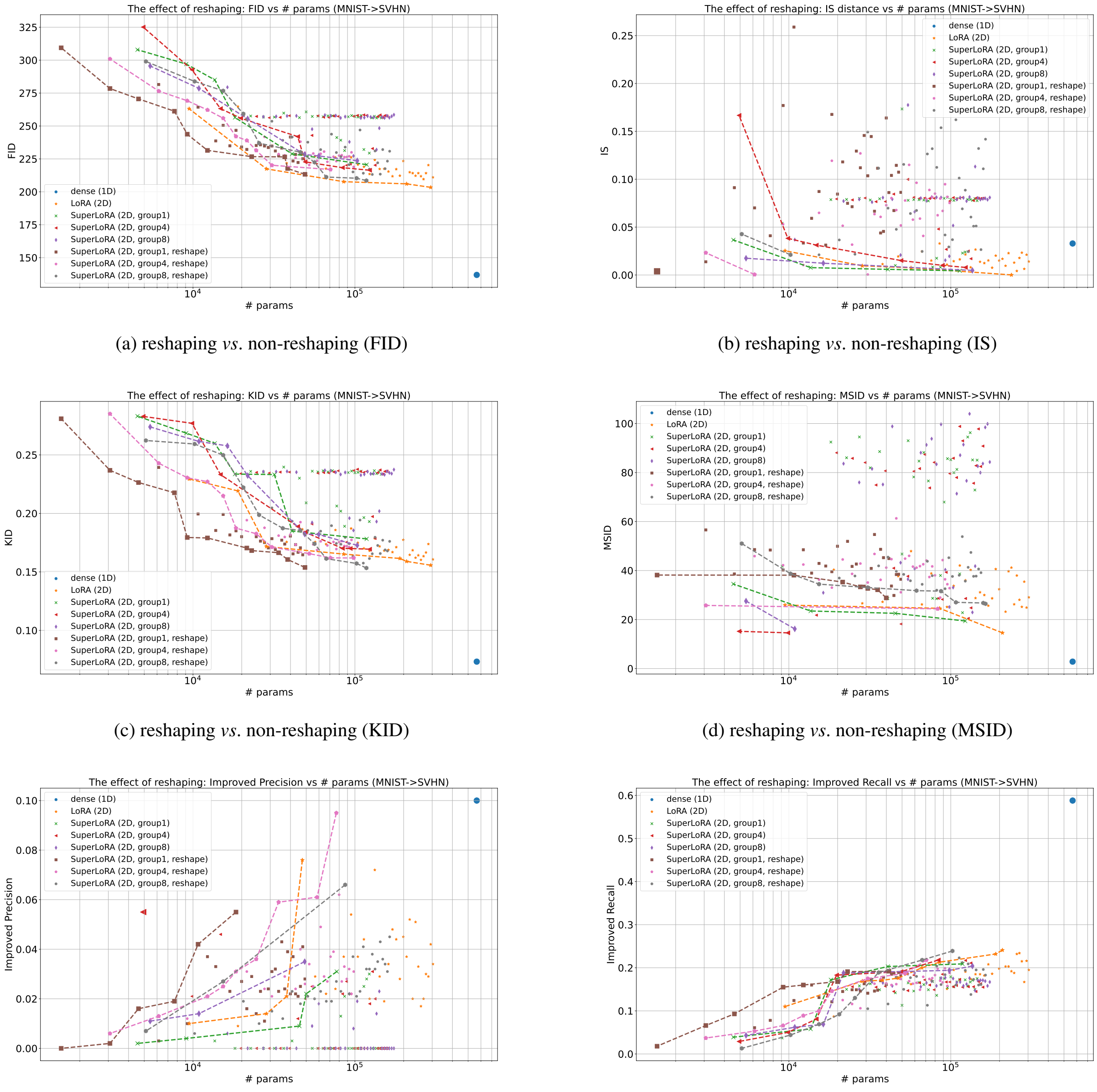
(c) weight-wise *vs*. group-wise (KID)

(d) weight-wise *vs*. group-wise (MSID)

(e) weight-wise *vs*. group-wise (Improved Precision)

(f) weight-wise *vs*. group-wise (Improved Recall)

Figure 27. Complete comparison between weight-wise LoRA and group-wise SuperLoRA for transfer learning from MNIST to SVHN.



(a) reshaping *vs*. non-reshaping (FID)

(b) reshaping *vs*. non-reshaping (IS)

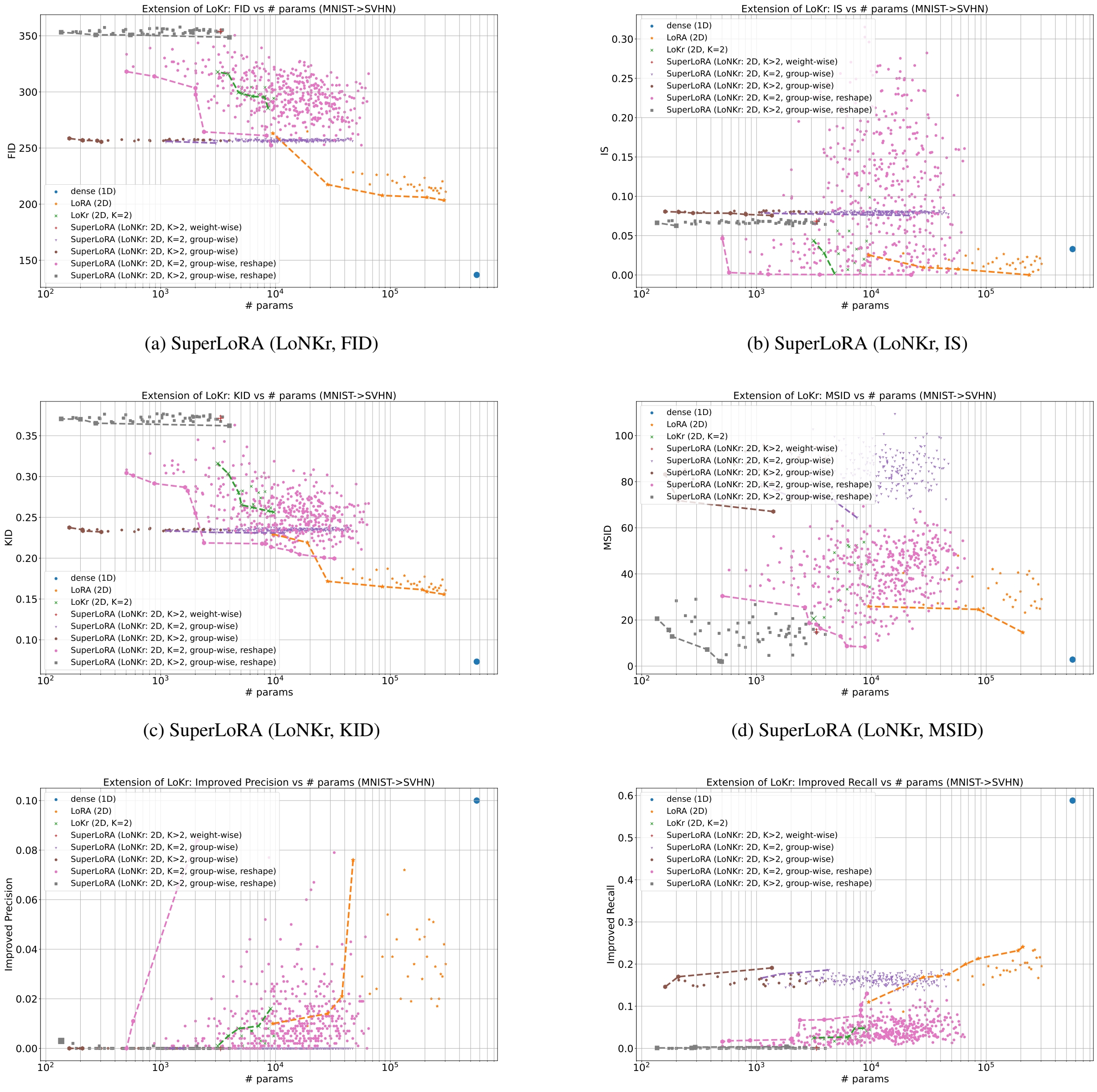
(c) reshaping *vs*. non-reshaping (KID)

(d) reshaping *vs*. non-reshaping (MSID)

(e) reshaping *vs*. non-reshaping (Improved Precision)

(f) reshaping *vs*. non-reshaping (Improved Recall)

Figure 28. Complete comparison between reshaping and non-reshaping SuperLoRA for transfer learning from MNIST to SVHN.



(a) SuperLoRA (LoNKr, FID)

(b) SuperLoRA (LoNKr, IS)

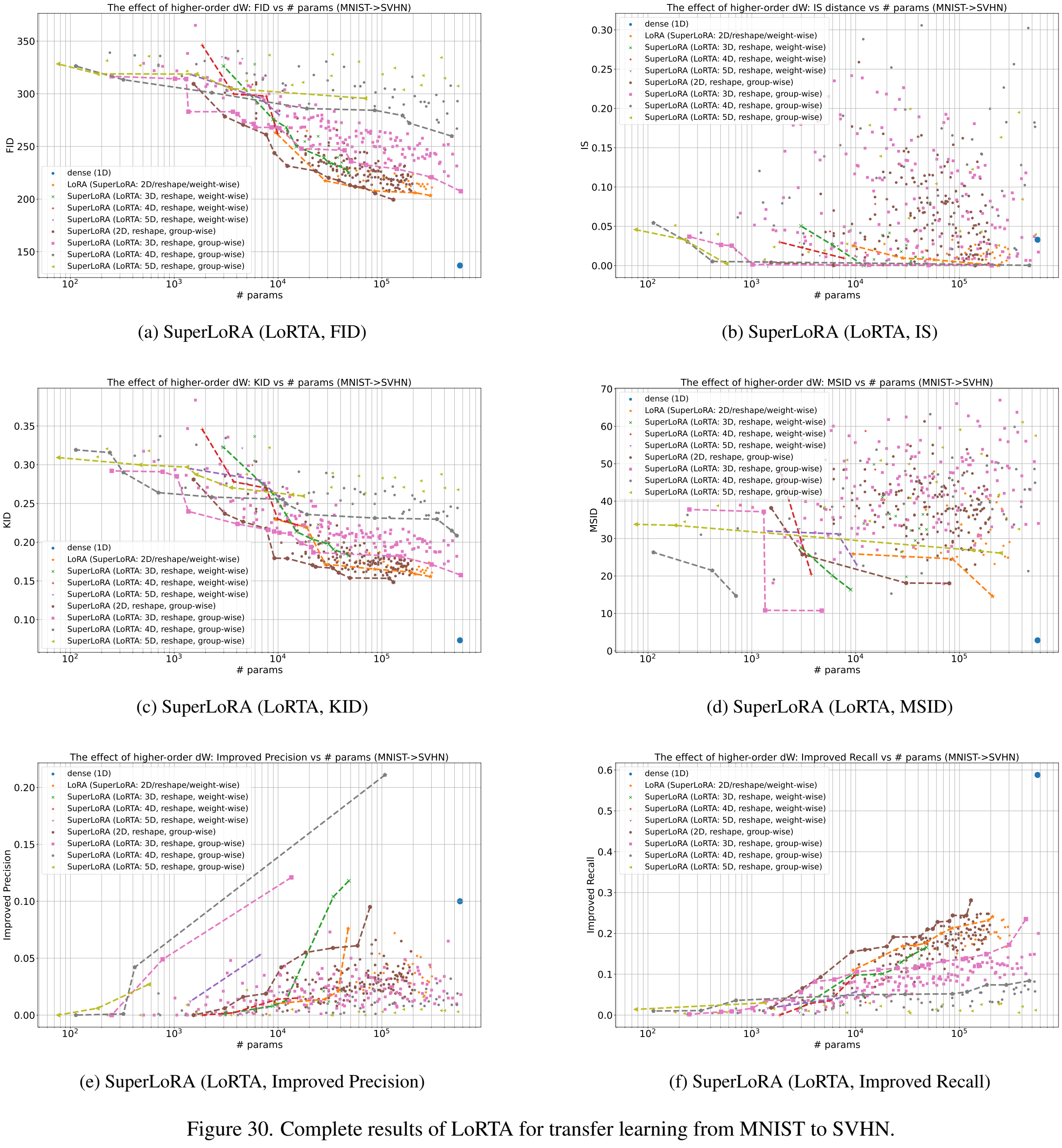
(c) SuperLoRA (LoNKr, KID)

(d) SuperLoRA (LoNKr, MSID)

(e) SuperLoRA (LoNKr, Improved Precision)

(f) SuperLoRA (LoNKr, Improved Recall)

Figure 29. Complete results of SuperLoRA (LoNKr) for transfer learning from MNIST to SVHN.



(a) SuperLoRA (LoRTA, FID)

(b) SuperLoRA (LoRTA, IS)

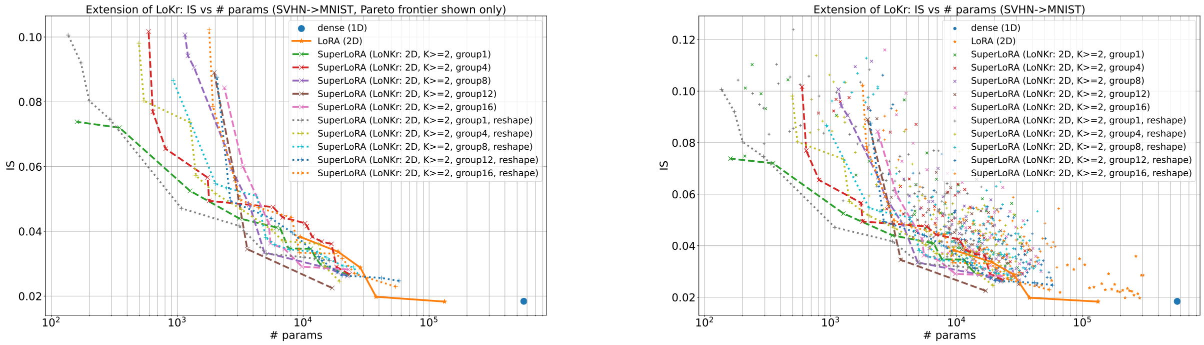
(c) SuperLoRA (LoRTA, KID)

(d) SuperLoRA (LoRTA, MSID)

(e) SuperLoRA (LoRTA, Improved Precision)

(f) SuperLoRA (LoRTA, Improved Recall)

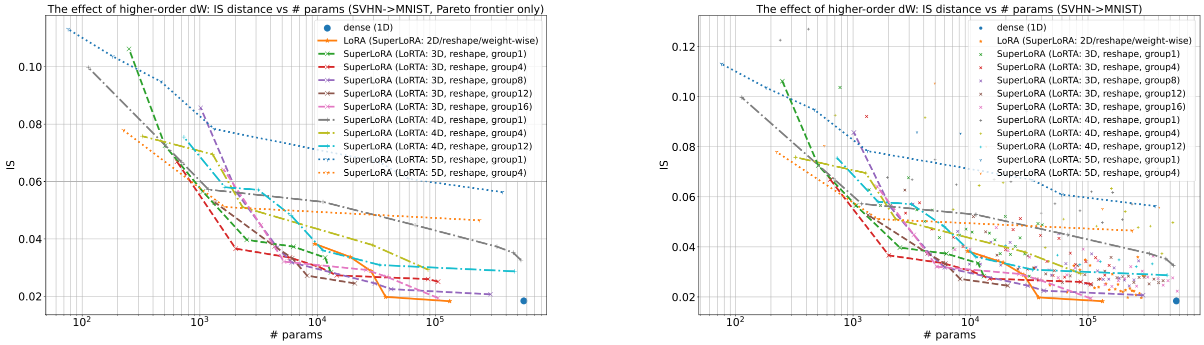
Figure 30. Complete results of LoRTA for transfer learning from MNIST to SVHN.



(a) SuperLoRA (LoNKr, Pareto frontier only)

(b) SuperLoRA (LoNKr)

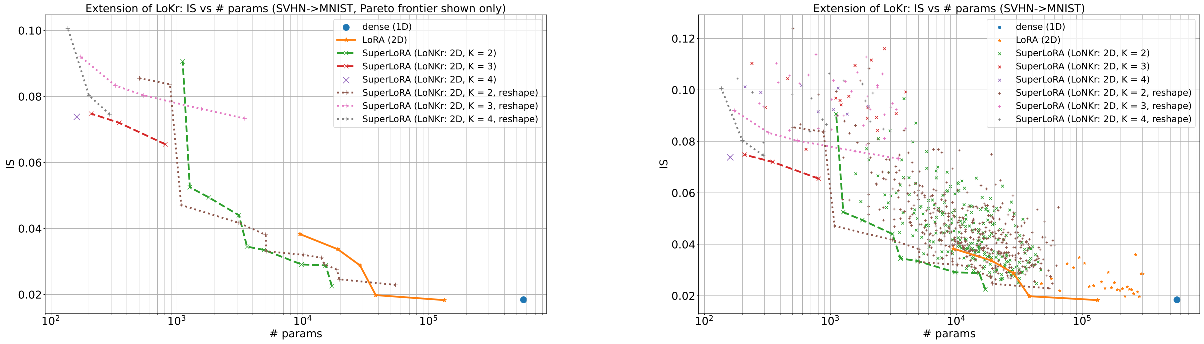
Figure 31. Effect of groups in LoNKr.



(a) SuperLoRA (LoRTA, Pareto frontier only)

(b) SuperLoRA (LoRTA)

Figure 32. Effect of groups in LoRTA.



(a) SuperLoRA (LoNKr, Pareto frontier only)

(b) SuperLoRA (LoNKr)

Figure 33. Effect of K in LoNKr.