On Principal Singular values and Singular vectors Adaptation of Large Language Models

Connie Ens Junjie Ma

December 29, 2024

1 Introduction

Low-rank adaptation(LoRA)[1] proposes the low-rank decomposition of the adjustments ΔW made to pre-trained weight matrices $W \in M_{m,n}(\mathbb{R})$, modifying the forward process by:

$$Y = X\left(W + \Delta W\right) = X\left(W + AB\right), \tag{1}$$

where $A \in M_{m,r}(\mathbb{R}), B \in M_{r,n}(\mathbb{R})$ with $r \ll \min(m,n)$. The adjustments in Equation 1 are restricted to A and B, significantly reducing both the number of trainable parameters and the GPU memory demands for fine-tuning. Many types of research emerged after LoRA, of which some attempted to find optimal initialization of A and B[2] while others attempted to find optimal decomposition and to improve computational efficiency. Along with DoRA[3], QLoRA[4], LQLora[5] and ADALoRA[6], PiSSA falls into the second category. In this report, we aim to first briefly summarize PiSSA and then mathematically formalize the low-rank decomposition method adopted in PiSSA. We then propose an alternative and more efficient decomposition method (CUR) in the Frobinus norm. Lastly, we propose a more robust low-rank decomposition in the ℓ_1 norm.

2 Summary of PiSSA

2.1 PiSSA

The **Pri**nciple **S**ingular Values and **S**ingular vectors **A**daption(PiSSA)[7] method offers an alternative approach to the full fine-tuning and LoRA by decomposing pre-trained weight matrices rather than the adjustments. The Singular Value Decomposition(SVD)[7, 8] of a pre-trained weight matrix $W \in M_{m,n}(\mathbb{R})$ is defined by:

$$W = U\Sigma V^\top, \tag{2}$$

where $U \in M_{m,\min(m,n)}(\mathbb{R})$, $V \in M_{n,\min(m,n)}(\mathbb{R})$ consist of singular vectors and their columns are orthonormal to each other, and $\Sigma = \mathrm{diag}(\boldsymbol{\sigma}) \in M_{\min(m,n),\min(m,n)}(\mathbb{R})$ is a diagonal matrix with the corresponding singular values ranked in a descending order with $\boldsymbol{\sigma} \in \mathbb{R}_{\geq 0}^{\min(m,n)}$. Hence, according to the ordering of Σ and corresponding left & right singular vectors in U and V, W can be separated into the principal singular values and vectors and the residual singular values and vectors, specifically,

$$W^{\mathrm{Pri}} = (U_{[:,:r]}\Sigma^{1/2}_{[:r,:r]})(\Sigma^{1/2}_{[:r,:r]}V^\top_{[:,:r]}) = AB; \quad W^{\mathrm{Res}} = W - W^{\mathrm{Pri}}, \tag{3}$$

where $A = U_{[:,:r]}\Sigma^{1/2}_{[:r,:r]} \in M_{m,r}(\mathbb{R})$, $B = \Sigma^{1/2}_{[:r,:r]}V^\top_{[:,:r]} \in M_{r,n}(\mathbb{R})$.

The authors[7] claim that by choosing the r such that the top-r singular values $\boldsymbol{\sigma}_{[:r]}$ are significantly larger than the rest of singular values $\boldsymbol{\sigma}_{[r:]}$, W^{Pri} effectively retains the essential structure and directions of the weight matrix W. Consequently, during the fine-tuning, the frozen part is no longer the pre-trained weight matrix W, but the residual W^{Res} which is less critical compared to W^{Pri}. This allows us to fine-tune the essential part of the pre-trained weight matrices directly. In contrast to LoRA[1] and Equation 1, the modified forward process is:

$$Y = X\left(W^{\mathrm{Res}} + W^{\mathrm{Pri}}\right) = X\left(W^{\mathrm{Res}} + AB\right),$$

while the gradients remain the same as $\frac{\partial \mathcal{L}}{\partial A} = X^\top \left(\frac{\partial \mathcal{L}}{\partial Y}\right) B^\top$ and $\frac{\partial \mathcal{L}}{\partial B} = A^\top X^\top \left(\frac{\partial \mathcal{L}}{\partial Y}\right)$, for some pre-defined loss function $\mathcal{L}$. In addition, the authors argue that tuning the principal components of W enables PiSSA to better fit the training data and facilitate convergence because principal singular vectors indicate the directions in which W is most stretched or impacted. Furthermore, the loss and gradient norm curves of PiSSA follow similar patterns to those of full parameter fine-tuning, which indicates that PiSSA systematically behaves similarly to full parameter fine-tuning.

Moreover, the authors[7] emphasize that PiSSA preserves the benefits of LoRA, enabling fine-tuning with fewer trainable parameters while avoiding initialization issues associated with Gaussian errors and zero entries. In [1] and [2], to prevent xA from exploding when width $n \to \infty$, the variance of Gaussian initialization is set to be proportional to n^{-1} which makes it tightly centred around 0. This might result in small and random gradients at the initial stage of the optimization, leading to a slower convergence and possibly a local minimum.

2.2 QPiSSA

As a counterpart of LoRA[1], the QLoRA[9] integrates the low-rank adaption and quantization. The authors [7] also propose PiSSA with Quantization(QPiSSA). The quantization error of QLoRA is measured by:

$$\text{Error} = ||W - (nf4(W) + AB)||_* = ||W - nf4(W)||_*,$$

where $||M||_* := \sum_{i=1}^{\min\{m,n\}} \sigma_i(M)$ and $\sigma_i(M)$ is the i^{th} singular value of M. In contrast, QPiSSA quantize only the residual weight matrix and has error

$$\text{Quantization Error of QPiSSA} = ||W - (nf4(W^{\text{res}}) + AB)||_* = ||W^{\text{res}} - nf4(W^{\text{res}})||_*,$$

Because W^{pri} maintains full precision during tuning, W^{res} has a much narrower and more Gaussian-like distribution than that of W. Consequently, QPiSSA significantly reduces quantization errors compared to other methods, such as QLoRA, that quantize the entire weight matrix W.

2.3 Experiments and Results

To validate the PiSSA, the authors[7] conducted various experiments on both Natural-Language Generation(NLG) and Natural-Language Understanding(NLU) Tasks. PiSSA was fine-tuned to various datasets to evaluate mathematical problem-solving capabilities, coding proficiency capabilities, and conversational ability. Fine-tuning PiSSA is shown to be consistently outperforming fine-tuning LoRA. The performance improvement is robust across different training scenarios, such as varying sizes of the training dataset, epochs, altering quantization precisions, various model sizes and model types and changing proportions of trainable parameters. In terms of natural language understanding capability, out of 16 experiments they have conducted, PiSSA outperforms LoRA in 14 of them. Other than one matching performance, the one exceptional case showed a lower average loss than LoRA in the final epoch, which still demonstrated PiSSA's stronger model fitting ability than LoRA. Authors hence conclude that other than the fact that PiSSA outperforms LoRA consistently, PiSSA enables faster and better convergence while best mirroring the full parameter fine-tuning process. PiSSA also shares the same architecture as LoRA. Hence, users can easily modify the existing pipeline.

3 Reformulating the problem and Random PiSSA

The PiSSA procedure can be given a mathematical grounding by reformulating it as an optimization problem as follows: given a size $m \times n$ weight matrix W, we are seeking a rank r matrix that is close to the full rank matrix W with some measure of "closeness" for matrices. To formalize this, let us firstly define the Frobenius norm of $A \in M_{m,n}(\mathbb{R})$ and give a few equivalent formulations:

$$||A||_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{i,j}|^2} = \sqrt{\text{trace}(A^T A)} = \sqrt{\sum_{i=1}^{\min(m,n)} \sigma_i^2(A)} \tag{4}$$

where A^T is the transpose of A. Given any matrix $W \in M_{m,n}(\mathbb{R})$ consider the problem

$$\min_{A \in M_{m,n}(\mathbb{R})} ||W - A||_F \qquad \text{subject to} \qquad \text{rank}(A) \leq r \tag{5}$$

and denote the solution to this problem by $\widehat{W}$. By the Eckart–Young–Mirsky theorem[10], the solution to this problem is precisely the truncated singular value decomposition W^{Pri} from Equation 3 used in the PiSSA method.[1].

This result answers the question raised by the authors[7]" why PiSSA outperforms LoRA?". LoRA freezes the weight matrix and then randomly initializes the rank r matrix of displacements from this weight. On the other hand, PiSSA finds the closest rank r matrix to W, and continues training it while freezing a residual component. Thus, PiSSA continues to train the most significant rank r matrix, as measured by closeness to the weight matrix, rather than just a random rank r displacement.

PiSSA silently made a crucial assumption that the new dataset is structurally similar to the dataset on which the model was previously trained since it assumes that the most significant singular vectors/values for fine-tuning are the same as those that were obtained in the original training. However, such an assumption is not necessarily true, especially if the model is being fine-tuned to perform a substantially different task than it was originally trained on. In such a case, randomly selecting a few terms in W^{res}, which have smaller singular values, could potentially improve the training accuracy beyond PiSSA as now there is a bit more freedom to adapt to the new and different dataset. Our first attempt at an improvement is Random PiSSA. Instead of taking the top-r singular values $\boldsymbol{\sigma}_{[:r]}$ along with the corresponding U, V, here we take the top-$(r-k)$ singular values and the corresponding U, V then randomly select k singular values outside the top r together with the corresponding U, V to obtain a rank r approximation.

4 CURA

Two notable issues related to the `SVD` are the complexity and interpretability. To tackle this, we adapt the framework to `CUR` decomposition. For any pre-trained weight matrix $W \in M_{m,n}(\mathbb{R})$ and consider two index sets $\mathcal{I} \subseteq \{1, 2, \cdots, m\}, \mathcal{J} \subseteq \{1, 2, \cdots, n\}$ with $|\mathcal{J}| = |\mathcal{I}| = r$, we define its rank$-r$ CUR decomposition to be

$$W_{\text{CUR}} = W_{[:,\mathcal{J}]} W^{+}_{[\mathcal{I},\mathcal{J}]} W_{[\mathcal{I}:]} = CUR,$$

where $C = W_{[:,\mathcal{J}]}, U = W^{+}_{[\mathcal{I},\mathcal{J}]}, R = W_{[\mathcal{I}:,]}$ and $W^{+}_{[\mathcal{I},\mathcal{J}]}$ denotes the pseudo-inverse. There are numerous recently developed algorithms for constructing `CUR` decomposition[11], but many of them heavily rely on the `SVD`. For efficient initialization, we select the rows and columns by randomized sampling without replacement under a probability distribution proportional to the ℓ_2 norm of rows and columns[12],

$$\mathbf{P}^{\text{Col}}_j = \frac{||W_{[:,j]}||_2^2}{||W||_F^2}; \quad \mathbf{P}^{\text{Row}}_i = \frac{||W_{[i,:]}||_2^2}{||W||_F^2}, \quad i \in \{1, 2, \cdots, m\}, j \in \{1, 2, \cdots, n\},$$

and construct U using the usual pseudo-inverse. Similar to PiSSA, we modify the Forward Process by

$$Y = X(W^{\text{Res}}_{\text{CUR}} + W^{\text{Pri}}_{\text{CUR}}) = X(W^{\text{Res}}_{\text{CUR}} + AB)$$

where $W^{\text{Res}}_{\text{CUR}} = W - W^{\text{Pri}}_{\text{CUR}}$, $W^{\text{Pri}}_{\text{CUR}} = CUR = AB$ and $A = C, B = UR$. During the fine-tuning, we freeze the residual part $W^{\text{Res}}_{\text{CUR}}$, and the only trainable parameters are low-rank matrices A and B. Similar to PiSSA, we directly fine-tune the essential part of the pre-trained weight matrices but use a more efficient initialization that might result in an even greater Gradient Norm.

To validate and compare CURA with PiSSA and LoRA, we fine-tuned the Vision Transformer(ViT)[13] using three different frameworks and on the MNIST data set[14]. The comparison of the accuracy over the initial stages can be found in Figure 1, which shows that the CURA outperforms the LoRA and PiSSA by yielding the highest accuracy over the first 100 training steps. The trajectory of the training loss can be found in Figure 2, which again illustrates that the CURA results in a similar or even more significant drop in the loss during the initial stages.

[1] we omit the proof in this report due to page limit; one can find the proof in the cited paper

5 Robust PiSSA

In functional analysis, there are many matrix norms other than the Frobenius norm. One of the obvious disadvantages of the Frobenius norm is that it is highly sensitive to the presence of outliers. It is well known that the ℓ_1 norm is less sensitive to outliers and is widely used in machine learning to create more robust algorithms. For a matrix $A \in M_{m,n}(\mathbb{R})$ define

$$||A||_1 = \sum_{i=1}^{m}\sum_{j=1}^{n}|A_{i,j}| \tag{6}$$

Our next improvement to PiSSA replaces the Frobenius norm with the ℓ_1 norm in problem 5 and so initializes the low-rank approximation via the solution $\widehat{A} \in M_{m,n}(\mathbb{R})$ of

$$\min_{\substack{C\in M_{m,n}(\mathbb{R})\\ \text{rank}(A)\leq r}} ||W-C||_1 = {}^{2} \min_{\substack{A\in M_{m,r}(\mathbb{R})\\ B\in M_{r,n}(\mathbb{R}}} ||W-AB||_1 \tag{7}$$

Precisely, given a weight matrix W, let $W^{\text{Pri}} = \widehat{A}\widehat{B}$ and let $W^{\text{Res}} = W - W^{\text{Pri}}$ where $\widehat{A}, \widehat{B}$ is the solution of Equation 7.

To illustrate that the solutions to Problem 5 and 7 can be different let $A, A^F, B \in M_{3,3}(\mathbb{R})$

$$A = \begin{bmatrix}3 & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 1\end{bmatrix}, A^F = \begin{bmatrix}3 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 0\end{bmatrix}, B = \begin{bmatrix}0 & 0 & 0\\ 0 & 1 & 1\\ 0 & 1 & 1\end{bmatrix} \tag{8}$$

Clearly, A^F is the rank 1 truncated SVD decomposition of A. Note however that $||A - A^F||_1 = 4$. while $||A - B||_1 = 3 < \|A - A^F\|$. Thus, the solution A^F of Equation 5 can't possibly be the solution to Equation 7 as B also has rank 1 but has a smaller ℓ_1 norm.

Unlike low-rank approximation in the Frobenius norm, there is no analytical solution for ℓ_1 low-rank approximation. In fact, the problem was recently shown to be NP-hard[15]. Sketching algorithms have, however, been developed which guarantee a solution to within some bound a certain percentage of the time [16]. For comparison, we implemented a simple Pytorch algorithm given in Algorithm 1 to solve this problem using Huber Loss as an alternative to ℓ_1 loss to maintain differentiability at the origin. The benchmarks in Figure 3 indicate that this Pytorch algorithm is significantly faster while still obtaining a result with a smaller mean absolute error.

The method using Algorithm 1 to initialize weight matrices was directly implemented in the PEFT library and benchmarked using the Google Vision Transformer model and the PACS dataset. Results are compared for LoRA, PiSSA, Robust PiSSA and Random PiSSA in Appendix C. Figure 4 shows training losses are lowest for Robust PiSSA for ranks 6 and 8, but PiSSA is slightly lower for rank 10. Similarly, Figure 5 (and Table 1) show that Robust PiSSA is more accurate than PiSSA for ranks 6 and 8, but PiSSA outperforms slightly in rank 10. Random PiSSA slightly outperforms PiSSA in rank eight but otherwise performs similarly or worse to PiSSA but still better than LoRA. LoRA has the lowest accuracy in all cases. These results agree with the idea expressed in [16] that outliers are more of a problem for lower-rank approximations than higher-rank ones, and so it is expected that more benefit is obtained from robust PiSSA for lower ranks.

6 Conclusion

In this report, we summarized the journal article PiSSA. We mathematically reinterpreted the method as a low-rank approximation of a weight matrix in the Frobenius norm and proposed a low-rank approximation method in ℓ_1 norm, which is intended to be more robust than PiSSA, especially for low ranks. The technique was benchmarked using the Google Vision transformer model and finetuned on the PACS dataset with results indicating that, indeed, for ranks 6 and 8, the robust PiSSA method outperforms PiSSA (and thus also LoRA.) Additionally, we also proposed a more efficient decomposition method, CURA, within the Frobenius norm. CURA outperformed LoRA and PISSA as it has higher accuracy over the first 100 training steps.

[2] Any rank r $m \times n$ matrix can be written in the form AB for $A \in M_{m,r}(\mathbb{R})$ and $B \in M_{r,n}(\mathbb{R})$

References

[1] E. J. Hu, yelong shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2022.

[2] S. Hayou, N. Ghosh, and B. Yu, "The impact of initialization on lora finetuning dynamics," 2024.

[3] S.-Y. Liu, C.-Y. Wang, H. Yin, P. Molchanov, Y.-C. F. Wang, K.-T. Cheng, and M.-H. Chen, "Dora: Weight-decomposed low-rank adaptation," 2024.

[4] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.

[5] H. Guo, P. Greengard, E. P. Xing, and Y. Kim, "Lq-lora: Low-rank plus quantized matrix decomposition for efficient language model finetuning," 2024.

[6] Q. Zhang, M. Chen, A. Bukharin, N. Karampatziakis, P. He, Y. Cheng, W. Chen, and T. Zhao, "Adalora: Adaptive budget allocation for parameter-efficient fine-tuning," 2023.

[7] F. Meng, Z. Wang, and M. Zhang, "Pissa: Principal singular values and singular vectors adaptation of large language models," 2024.

[8] D. Serre, *Matrices*. Springer Science & Business Media, 10 2010.

[9] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[10] E. Carl and Y. Gale, "The approximation of one matrix by another of lower rank creator," *Psychometrika Vol.1 (3), p.211-218 DOI: 10.1007/BF02288367*, 1936-09.

[11] D. C. Sorensen and M. Embree, "A deim induced cur factorization," 2015.

[12] K. Hamm and L. Huang, "Stability of sampling for cur decompositions," 2020.

[13] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[14] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[15] N. Gillis and S. A. Vavasis, "On the complexity of robust pca and l1-norm low-rank matrix approximation," *Mathematics of Operations Research*, vol. 43, p. 1072–1084, Nov. 2018.

[16] Z. Song, D. P. Woodruff, and P. Zhong, "Low rank approximation with entrywise ℓ_1-norm error," 2020.

A CURA Results

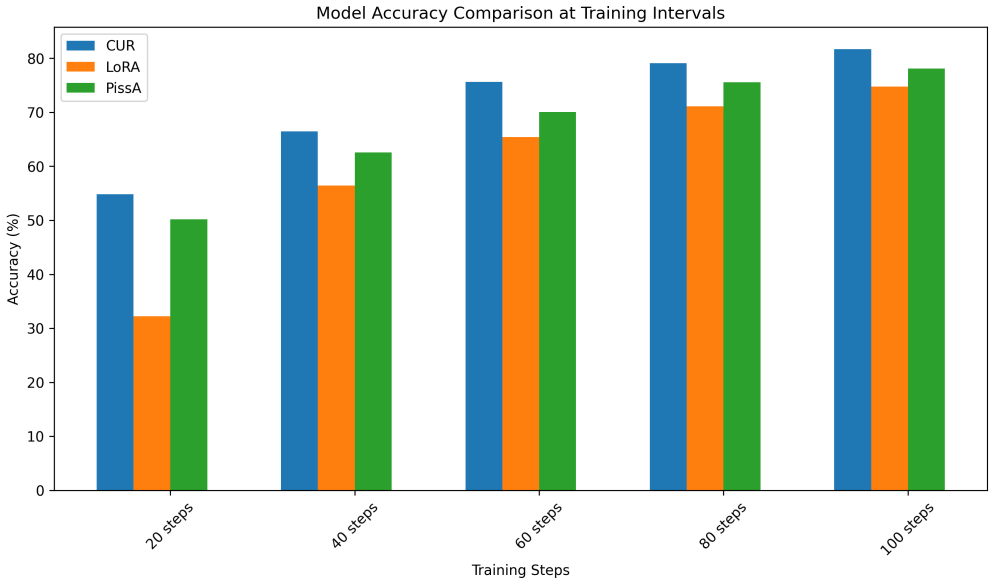

Figure 1: Comparison of the accuracy between CURA, LoRA, and PiSSA during the first 100 fine-tuning steps, which shows CURA outperforms LoRA and PiSSA at the initial stage.

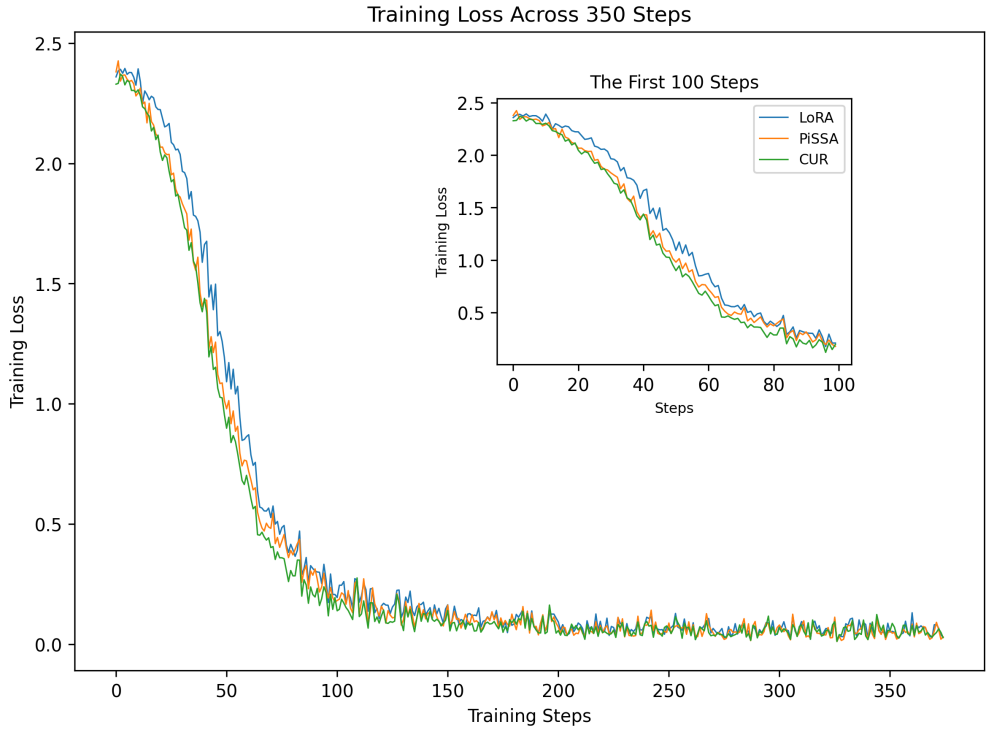

Figure 2: Comparison of the training loss between CURA, LoRA, and PiSSA, which shows CURA and PiSSA had a larger decrease at the initial stage.

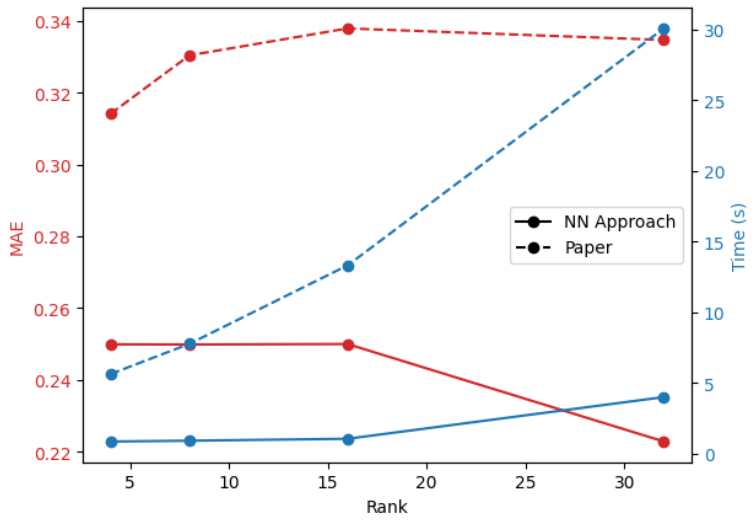

Figure 3: Comparison between the Pytorch minimization algorithm in Algorithm 1 and the sketching algorithm from the paper [16]. The Pytorch algorithm obtains a lower error rank r matrix while taking less time.

B ℓ_1 Low Rank Approximation in Pytorch

Result: $A \in \mathrm{Mat}_{m,r}(\mathbb{R}), B \in \mathrm{Mat}_{r,n}(\mathbb{R})$
Randomly initialize A, B
while *counter* $<$ *Number of Iterations* **do**
  forward(x) = AB;
  loss = Huber(W, AB)
  loss.backward()
end

Algorithm 1: Solving The ℓ_1 minimization problem. The use of the Huber loss as an alternative to mean absolute error avoids issues with gradients at the origin.

C Fine-Tuning Results

This section presents the results of fine-tuning the Google Vision Transformer model on the PACs dataset. Five trials, each with a different random seed, are chosen (for reproducibility, the chosen seeds are 0, 100, 200, 300 and 400.) For each seed, the dataset is broken into a training dataset (80% of the data) and a testing dataset (20% of the data.) The model is then fine-tuned for five epochs on the training dataset and evaluated on the testing dataset. Results are then averaged over the five trials.

	10	6	8
LoRA	0.948074	0.950275	0.950475
PiSSA	**0.952976**	0.951476	0.951776
Random PiSSA	0.951976	0.951476	0.952676
Robust PiSSA	0.951976	**0.953277**	**0.953077**

Table 1: Accuracy of the various methods averaged over the five trials. In each column, the method with the highest accuracy is bolded.

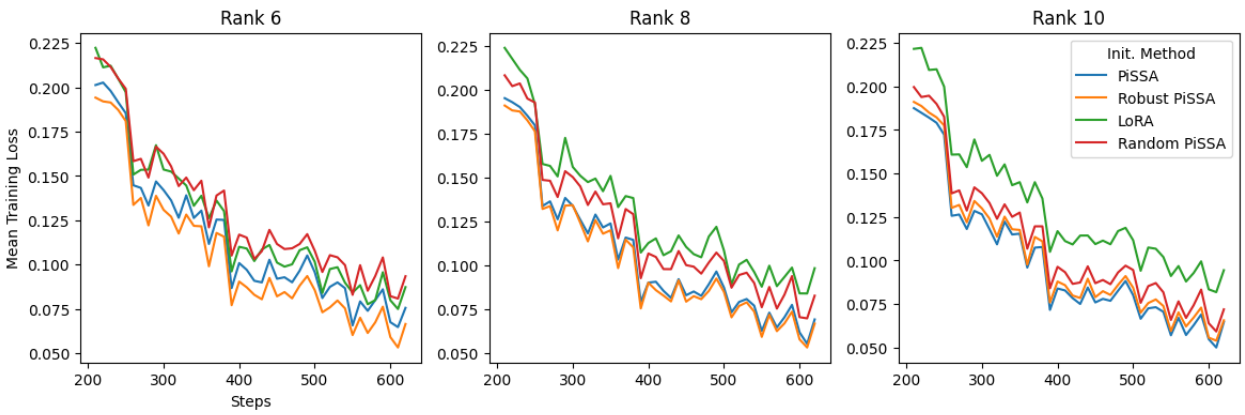

Figure 4: Training loss as a function of step averaged over the five trials.

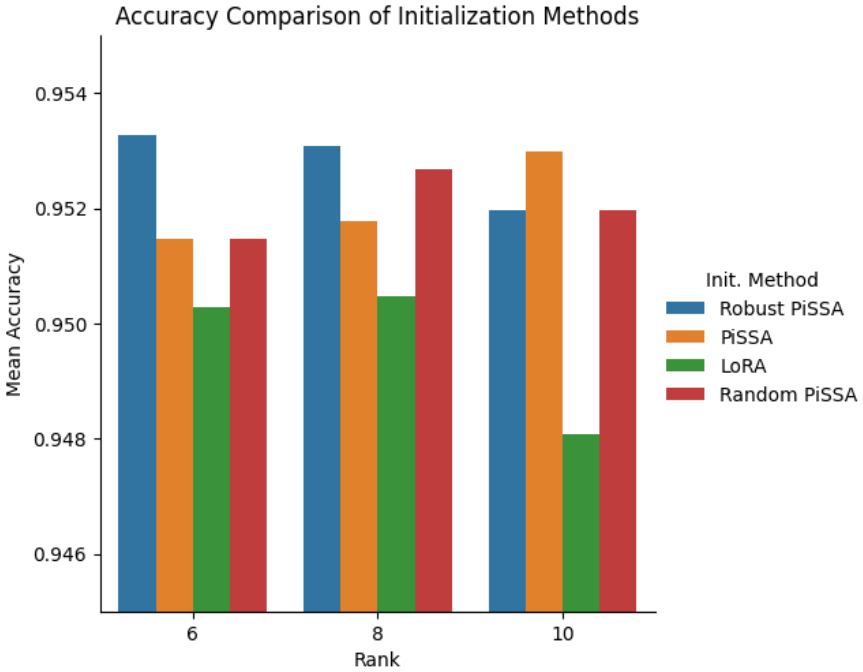

Figure 5: Accuracy of the various methods averaged over the five trials.