

Singular Value Adaptation: Fixing LoRA With SVD Initialization

Baykam Say[✉], Mato Gudelj[✉], and Coşku Barış Coşlu[✉]

School for Computation, Information and Technology, Technical University of Munich

✉ baykam.say@tum.de

✉ mato.gudelj@tum.de

✉ baris.coslu@tum.de

August 6, 2023

Abstract — This work introduces Singular Value Adaptation (SiVA), a novel Parameter-Efficient Fine Tuning (PEFT) method for large-scale pre-trained transformer models. SiVA improves upon LoRA by leveraging singular value decomposition for initialization, thereby sidestepping the zero-initialization pitfalls inherent in default LoRA initialisation. This results in faster convergence while preserving full fine-tuning performance. Additionally, we demonstrate that the use of input prompts can boost the performance of SiVA without any computational drawbacks.

1 Introduction

The advent of large-scale transformer models has revolutionized performance benchmarks in natural language processing (NLP), computer vision (CV), and integrated vision-and-language (VL) tasks. However, as these models grow, so does the computational burden associated with their fine-tuning. Parameter-Efficient Fine Tuning (PEFT) has thus become an essential area of research, aiming to efficiently adapt pre-trained models to novel tasks with minimal computational overhead.

This paper specifically focuses on reparametrization-based methods, which are gaining traction in the industry for their balance between efficiency and performance. We introduce a novel method, SiVA (Singular Value Adaptation), which leverages singular value decomposition to fine-tune models more effectively. SiVA aims to optimize the trade-offs between model efficiency and performance, building on the foundation set by existing methods like LoRA.

2 Related Work

2.1 Reparametrization-based Methods

Research has shown that the intrinsic dimensions of a model reduce when it is fine-tuned for a specific task [1]. Therefore, in theory, a low-dimension

reparametrization that is as effective as the full parameter space can be used for fine-tuning [1].

2.1.1 LoRA

LoRA [4] uses this information and the hypothesis that the updates made during fine-tuning also have a low intrinsic dimension to create a low-rank weight updating PEFT method. It works on the weight matrices of linear layers by inserting two trainable lower-rank matrices (LoRA weights) that represent the change of the original matrix as seen on Figure 1.

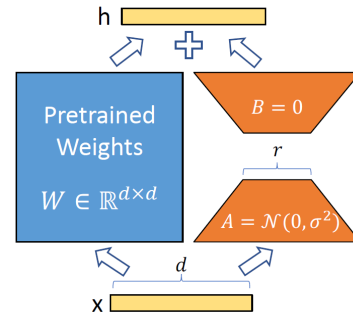


Figure 1 The reparametrization of LoRA [4]. Only A and B are trained.

During training, the pretrained weight matrix (W_0) of the linear layer is frozen, and the LoRA weights, represented by a down projection (A) and an up projection (B) layer, are trained. During the forward pass, the two lower-rank matrices are multiplied together and then added to the original matrix to generate the updated weight matrix for that specific linear layer. For a linear transformation $h = W_0x$, the modified forward pass of LoRA yields:

$$h = W_0x + BAx \quad (1)$$

The down projection layer is randomly initialized, while the up projection layer is initialized to zero. This ensures that their initial multiplication, which represents the change of the weight matrix, is zero. For inference, LoRA weights can be merged with the

original weight matrix to get an unmodified model that has the updated weight matrices.

2.1.2 LoRA-like Models

Recent advancements in reparametrization-based methods have all used the same approach of freezing the original weight matrix and representing the updates on lower-rank update matrices. These methods include KronA [3], which uses the Kronecker products of two low-rank matrices to represent the updates; IA³ [5], which reimplements the forward pass of LoRA as $h = (W_0(x \circ A)) \circ B$ and applies LoRA to only the key and value layers in self-attention and the feed-forward layers in the transformer; and QLoRA [2], which keeps the original weights in a lower-precision data type while applying LoRA. We compare our method directly to LoRA as the other reparametrization-based methods use the same approach and only introduce incremental improvements over LoRA.

3 Methods

3.1 Singular Value Adaptation (SiVA)

During our research, we focused on the main weakness of LoRA-like models: using random and zero initializations for their layers. While this allows them to initially represent the weight updates as zero, it also harms the training as the optimizer struggles with zero initialization. As a solution, we propose a novel low-rank fine-tuning method: Singular Value Adaptation (SiVA). SiVA represents the whole weight matrix in a lower rank using singular value decomposition instead of representing just the change in the weights like LoRA.

3.1.1 Model Initialization

The initialization of SiVA is shown in Figure 2. To accomplish this, it first decomposes the original weight matrix W_0 into U, S, V matrices using SVD, where S is a one-dimensional matrix consisting of the diagonal elements of the rectangular diagonal matrix Σ and $V = V_0^H$:

$$W_0 = U \Sigma V_0^H \quad (2)$$

It then splits U, S, V matrices into training and reconstruction matrices. Training (U_t, S_t, V_t) matrices are rank r matrices where r represents the rank of the training and they contain the highest singular value row and columns:

$$U_t = U[:, [1, 2, \dots, r]] \quad (3)$$

$$S_t = S[1, 2, \dots, r] \quad (4)$$

$$V_t = V[[1, 2, \dots, r], :] \quad (5)$$

Reconstruction (U_r, S_r, V_r) matrices are rank $d - r$ matrices where d is the decomposition rank and they contain the remaining rows and columns until d . d is a hyperparameter to allow lossy reconstruction of the weight matrix to reduce overfitting of the initial model similar to pruning with SVD.

$$U_r = U[:, [r, r + 1, \dots, d]] \quad (6)$$

$$S_r = S[r, r + 1, \dots, d] \quad (7)$$

$$V_r = V[[r, r + 1, \dots, d], :] \quad (8)$$

The original matrix is then reconstructed using the reconstruction matrices and frozen. The resulting matrix W_r denotes the lost granularity of the low-rank representation:

$$W_r = U_r \circ S_r V_r \quad (9)$$

Only U_t and V_t matrices are trained. Since they are the most impactful parts of the original weight matrix according to singular value decomposition, their updates represent the updates over the original weight matrix as accurately as possible.

3.1.2 Forward Pass

During the forward pass, training matrices are reconstructed and added to W_r to get the updated weight matrix without any loss of granularity from the lower-rank representation. For $h = W_0 x$, the modified forward pass yields:

$$h = W_r x + U_t \circ S_t V_t x \quad (10)$$

Figure 3 shows the comparison between LoRA’s and SiVA’s forward passes. Recall that the modified forward pass of LoRA is formulized as $h = W_0 x + B A x$. While A, B are same size matrices as U_t, V_t ; A, B represent only the change of the original matrix whereas U_t, V_t represent the whole matrix.

3.1.3 Optimal Hyperparameters

SiVA introduces two new hyperparameters. For the training rank, we use $r = 4$ to reach the full fine-tuning performance on RoBERTa-large. We hypothesize that

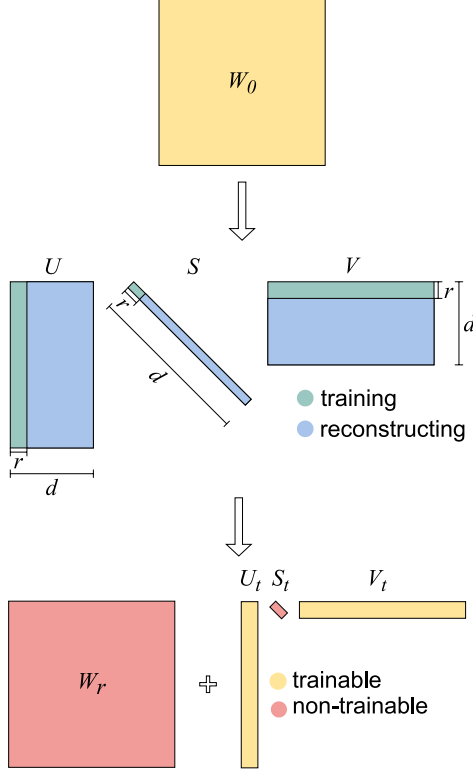


Figure 2 SiVA initialization diagram. The original weight matrix W_0 is decomposed using SVD. Rank r submatrices of the decomposition are selected according to the highest singular values (U_t, S_t, V_t). Original matrix is reconstructed without including (U_t, S_t, V_t). Only U_t, V_t are trained.

as the model size grows, r can get as low as 1 without losing any performance similar to LoRA. We use the number of output channels of W_0 for the decomposition rank d . Selecting a lower d reduces accuracy. Moreover, we experiment with training a subset of linear layers with SiVA and freezing the rest. We find the best balance between accuracy and efficiency when training only the key and query projection matrices in self attention.

3.1.4 Comparing to LoRA and Full Fine-Tuning

Similar to LoRA, training matrices of SiVA can be merged after training to get an unmodified model to remove any inference latency. They can also be swapped with other training matrices to modify the same base model for different fine-tuned tasks, eliminating the need for keeping multiple fine-tuned large models for different tasks. However, unlike LoRA, since SiVA does not have zero or random initialization, the optimization is easier and the training converges much faster.

As it can be seen from Figure 4, the training graphs of full fine-tuning and SiVA are almost identical, to

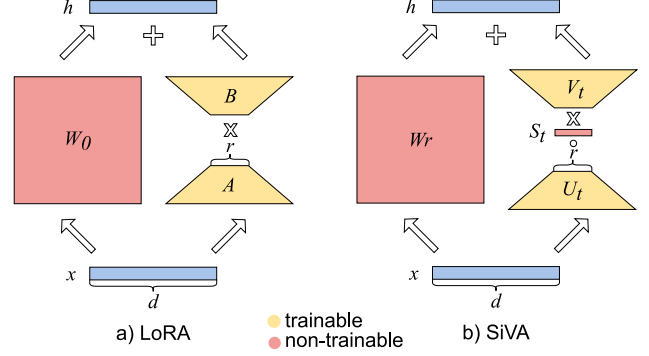


Figure 3 Visual representation of the forward passes of LoRA and SiVA. In both figures, the trainable parts (A, B for LoRA and U_t, V_t for SiVA) are multiplied and added to the untrainable part to get the updated weight matrix. A linear transformation is applied with the resulting updated matrix to get the results. LoRA’s trainable parts represent the change in the original matrix while SiVA’s trainable parts represent the whole matrix.

the fluctuations of the unsmoothed graphs, whereas LoRA struggles to have any meaningful optimizations in the first quarter of the epoch due to the random and zero initializations of its layers. Even if we shift the LoRA graph left 1000 iterations to mitigate 50% accuracy of the first quarter, we can observe that the accuracy and loss are still worse compared to full fine-tuning and SiVA. Due to the similarity of the training graphs of SiVA and full fine-tuning with the same hyperparameters, we claim that SiVA is analogous to full fine-tuning in a lower dimension.

4 Results

Table 1 shows a comparison of our method and the respective baselines. We evaluate the performance in terms of accuracy and F1-score on the SST-2 dataset, as well as peak memory usage and training samples per second. For all experiments, we fine-tune a RoBERTa-large model using a batch size of 16.

5 Conclusion and Future Work

5.1 Conclusion

In this paper, we introduced SiVA, a low-rank update method for Parameter-Efficient Fine Tuning (PEFT). Utilizing Singular Value Decomposition (SVD), SiVA demonstrates faster convergence compared to existing methods like LoRA, while maintaining full-fine-tuning accuracy and low memory usage. Additionally, we observed that employing modified prompts

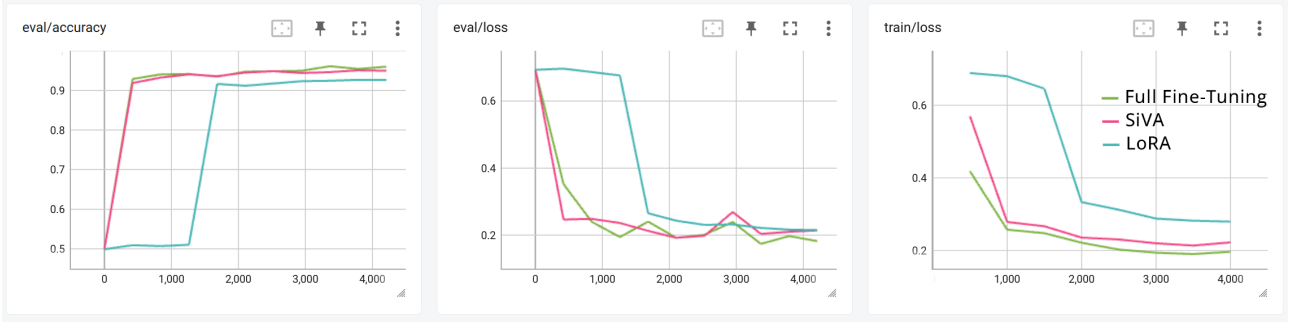


Figure 4 The training graphs for full fine-tuning, SiVA, and LoRA for RoBERTa-large [6] on the SST-2 [7] dataset for one epoch. Full fine-tuning and SiVA follow each other closely in all graphs while LoRA has worse performance. All the hyperparameters are the same, the original model initialization (including the head) is also the same.

Table 1 Comparison of different methods when fine-tuning a RoBERTa-large model on the SST-2 dataset using a batch size of 16. Samples per Second is measured on an RTX 3070 Mobile and computed as an average over the training process.

Method	Accuracy	F1	Memory Usage (MB)	Samples per Second
Full Fine-Tuning	96.44	96.52	8188	51.8
LoRA	96.22	96.30	4526	64.6
Last 3 Layers	94.04	94.18	2117	179.6
SiVA	96.56	96.63	4529	73.3
SiVA - key value	95.76	95.86	2920	106.4

during the fine-tuning phase positively impacted performance metrics without any drawbacks. For tasks requiring high-accuracy fine-tuning, SiVA emerges as a compelling choice.

5.2 Future Work

For future investigations, we recommend subjecting SiVA to more benchmarks by deploying it on various architectures and larger models. The method’s model-agnostic nature also opens doors for its application in non-NLP domains and on non-transformer models. Furthermore, adapting 4-bit quantization techniques such as QLoRA for SiVA could make it an even more efficient solution, widening its applicability for low-resource settings.

References

- [1] Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. *Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning*. 2020. arXiv: 2012.13255 [cs.LG].
- [2] Tim Dettmers et al. *QLoRA: Efficient Finetuning of Quantized LLMs*. 2023. arXiv: 2305.14314 [cs.LG].
- [3] Ali Edalati et al. *KronA: Parameter Efficient Tuning with Kronecker Adapter*. 2022. arXiv: 2212.10650 [cs.CL].
- [4] Edward J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].
- [5] Haokun Liu et al. *Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning*. 2022. arXiv: 2205.05638 [cs.LG].
- [6] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692. URL: <http://arxiv.org/abs/1907.11692>.
- [7] Alex Wang et al. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: 1804.07461 [cs.CL].