OSoRA: Output-Dimension and Singular-Value Initialized Low-Rank Adaptation

Anonymous ACL submission

Abstract

Fine-tuning Large Language Models (LLMs) has become increasingly challenging due to their massive scale and associated computational costs. Parameter-Efficient Fine-Tuning (PEFT) methodologies have been proposed as computational alternatives; however, their implementations still require significant resources. In this paper, we present OSoRA (Output-Dimension and Singular-Value Initialized Low-Rank Adaptation), a novel PEFT method for LLMs. OSoRA extends Low-Rank Adaptation (LoRA) by integrating Singular Value Decomposition (SVD) with learnable scaling vectors in a unified framework. It first performs an SVD of pre-trained weight matrices, then optimizes an output-dimension vector during training, while keeping the corresponding singular vector matrices frozen. OSoRA substantially reduces computational resource requirements by minimizing the number of trainable parameters during fine-tuning. Comprehensive evaluations across mathematical reasoning, common sense reasoning, and other benchmarks demonstrate that OSoRA achieves comparable or superior performance to state-of-the-art methods like LoRA and VeRA, while maintaining a linear parameter scaling even as the rank increases to higher dimensions. Our ablation studies further confirm that jointly training both the singular values and the output-dimension vector is critical for optimal performance.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across various Natural Language Processing (NLP) tasks. However, as these models escalate in size to hundreds of billions of parameters, fine-tuning them requires prohibitive computational resources (Abacha et al., 2025; Brown et al., 2020). This computational challenge has catalyzed the development of Parameter-Efficient Fine-Tuning (PEFT) methodologies, which enable fine-tuning LLMs by selectively updating only a minimal subset of parameters.

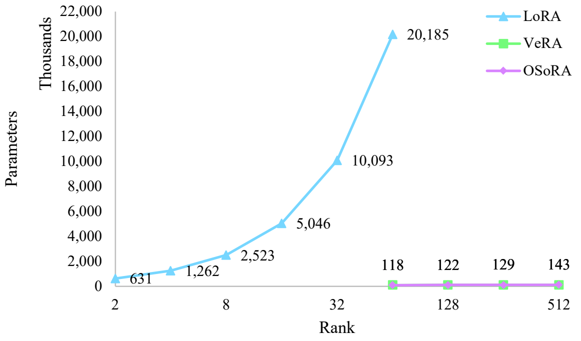

Figure 1: Parameter count comparison among adaptation methods at varying ranks on Qwen2-7B model. The results demonstrate that LoRA exhibits exponential growth in trainable parameters with increasing rank, whereas both VeRA and OSoRA maintain efficient linear scaling in their parameter count.

Recent PEFT approaches include Low-Rank Adaptation (LoRA) (Hu et al., 2022), which constrains weight updates to low-rank decompositions; Vector-based Random Matrix Adaptation (VeRA) (Kopiczko et al., 2024), which further improves efficiency by training only scaling vectors; and PiSSA (Meng et al., 2024), which uses Singular Value Decomposition (SVD) to update important weight matrix components. Despite these advances, existing methods still face limitations in balancing parameter efficiency with adaptation quality.

We introduce OSoRA (Output-Dimension and Singular-Value Initialized Low-Rank Adaptation), a novel PEFT method that combines SVD-based decomposition with a learnable scaling vector. OSoRA decomposes pretrained weight matrices using SVD, then selectively updates only the singular values and a single output-dimension vector during training. This approach significantly reduces trainable parameters while maintaining competitive performance.

Our contributions include:

- A novel PEFT method combining SVD-based decomposition with a learnable scaling vector
- Demonstration that updating only singular values and a single vector is sufficient for effective adaptation
- Comprehensive experiments showing OSoRA achieves comparable or superior performance to state-of-the-art methods with fewer parameters

Our work makes LLMs adaptation more accessible and efficient, enabling fine-tuning of large models on limited computational resources without sacrificing adaptation quality.

2 Related Work

PEFT began with inserting small adapter modules into each transformer block (Houlsby et al., 2019). Concurrently, prompt-based methods such as Prompt-Tuning (Lester et al., 2021), P-Tuning (Liu et al., 2022), and P-Tuning v2 (Liu et al., 2021) showed that a handful of continuous tokens prepended to the input can steer frozen language models toward new tasks while keeping all backbone weights intact. These two lines established the principle that high-capacity language models can often be adapted with orders-of-magnitude fewer trainable parameters than full fine-tuning.

LoRA (Hu et al., 2022) popularized the idea of constraining weight updates to a rank-r product of two small matrices, reducing trainable parameters from $O(dk)$ to $O(r(d+k))$ and sparing most optimizer state. On top of this foundation, AdaLoRA (Zhang et al., 2023) allocates rank budget across layers on the fly, and QLoRA (Dettmers et al., 2023) combines LoRA with 4-bit quantization so that both training and inference fit on consumer GPUs. VeRA (Kopiczko et al., 2024) keeps the low-rank bases frozen and learns only two scaling vectors, achieving the same $r+d$ trainable parameters as our method while introducing variance-preserving random projections that improve generalization.

Several works seek more informative update directions than random bases. DoRA (Liu et al., 2024) fine-tunes the norm of each weight and updates its direction, improving stability. PiSSA (Meng et al., 2024) leverages SVD to decompose weight matrices and selectively updates only the principal singular values and their corresponding vectors, preserving the model's inherent knowledge while enabling effective adaptation.

OSoRA unifies the advantages of the two branches above. Like LoRA and VeRA, it constrains updates to a low-rank form and requires only $r+d$ trainable scalars, preserving memory and computational efficiency. Unlike methods that rely on random or learned bases, OSoRA initializes its subspace with the top-r singular vectors of the pretrained weights, capturing the model's dominant variation directions from the outset. It further introduces two learnable vectors that can be transformed into diagonal matrices - one over output dimensions and one over rank components. This synthesis yields a PEFT method that maintains LoRA's simplicity, matches VeRA's parameter efficiency, and inherits the informed initialization benefits demonstrated by PiSSA.

3 Method

In this section, we introduce Output-Dimension and Singular-Value Initialized Low-Rank Adaptation (OSoRA), a novel approach for efficient fine-tuning of pre-trained models. OSoRA builds upon and extends state-of-the-art methods such as VeRA (Kopiczko et al., 2024) and LoRA (Hu et al., 2022). The key innovation of OSoRA is the strategic reparameterization of low-rank matrices using SVD. Specifically, we maintain frozen pairs of matrices derived from singular vectors, while only updating the singular value vectors and a single output-dimension vector initialized as all-ones during training, as illustrated in Figure 2. Like VeRA and LoRA, OSoRA allows the trained vectors and low-rank matrices to be seamlessly merged into the original weights, eliminating any additional computational overhead during inference.

3.1 Preliminaries

LoRA fine-tunes LLMs using a product of two low-rank matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$. For a pretrained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, LoRA constrains the weight update ΔW to a low-rank decomposition, as shown in Eq. (1):

$$y = W_0x + \Delta W x = W_0 x + \underline{BA}x \qquad (1)$$

where underlined parameters indicate trainable components. This approach allows the original weight matrix W_0 to remain frozen while only optimizing the low-rank matrices A and B. Since

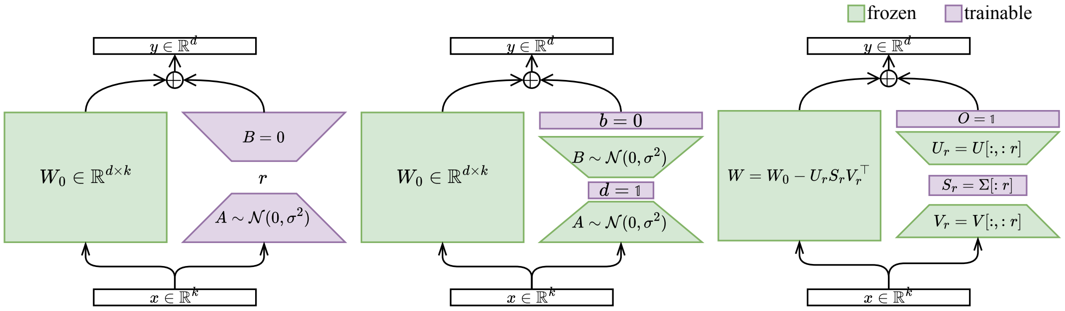

Figure 2: Schematic comparison of LoRA (left), VeRA (middle) and OSoRA (right). LoRA adapts pretrained weights $W_0 \in \mathbb{R}^{d \times k}$ by training low-rank matrices $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$. VeRA keeps these matrices frozen but introduces learnable scaling vectors $d \in \mathbb{R}^r$ and $b \in \mathbb{R}^d$. OSoRA applies SVD to decompose W_0 into singular vectors $U_r \in \mathbb{R}^{d \times r}$ and $V_r \in \mathbb{R}^{k \times r}$ with corresponding singular values $S_r \in \mathbb{R}^r$. During fine-tuning, only S_r and a learnable all-ones vector $O \in \mathbb{R}^d$ are updated, while the singular vector matrices remain fixed.

$r \ll \min(d, k)$, these matrices contain significantly fewer parameters than the original weight matrix, making the fine-tuning process computationally efficient.

Building upon LoRA, VeRA further reduces parameter count and can be formulated as:

$$y = W_0 x + \Delta W x = W_0 x + \underline{\Lambda_b} B \underline{\Lambda_d} A x \qquad (2)$$

where $\underline{\Lambda_b}$ and $\underline{\Lambda_d}$ are diagonal matrices constructed from learnable vectors $b \in \mathbb{R}^d$ and $d \in \mathbb{R}^r$, respectively. Unlike LoRA, VeRA uses frozen, randomly initialized matrices $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$, with adaptation occurring solely through the scaling vectors.

3.2 Method Formulation

OSoRA performs SVD to decompose the pretrained weight matrix W_0, as shown in Eq. (3):

$$W_0 = U \Sigma V^\top \qquad (3)$$

where $U \in \mathbb{R}^{d \times d}$ and $V \in \mathbb{R}^{k \times k}$ are orthogonal matrices containing the left and right singular vectors of W_0, respectively, and $\Sigma \in \mathbb{R}^{d \times k}$ is a diagonal matrix containing the singular values of W_0 in descending order.

OSoRA selectively adapts only the top r singular values and introduces a learnable scaling vector, while keeping the corresponding singular vectors fixed. The adaptation can be formulated as:

$$y = W_0' x + \Delta W x = W_0' x + \underline{\Lambda_O} U_r \underline{\Lambda_{S_r}} V_r^\top x \quad (4)$$

where $\underline{\Lambda_O} \in \mathbb{R}^{d \times d}$ is a diagonal matrix constructed from a learnable scaling vector $O \in \mathbb{R}^d$ (initialized as all-ones), $U_r \in \mathbb{R}^{d \times r}$ and $V_r \in \mathbb{R}^{k \times r}$ are the fixed left and right singular vectors corresponding to the top r singular values, and $\underline{\Lambda_{S_r}} \in \mathbb{R}^{r \times r}$ is a diagonal matrix constructed from the learnable singular values $S_r \in \mathbb{R}^r$. W_0' represents the frozen component of the weight matrix after excluding the contribution of the top r singular values and the corresponding singular vectors, which can be written as:

$$W_0' = W_0 - \underline{\Lambda_O} U_r \underline{\Lambda_{S_r}} V_r^\top \qquad (5)$$

3.3 Memory and Computational Considerations

While OSoRA significantly reduces the number of trainable parameters to just $r + d$ during fine-tuning, it's important to clarify the overall memory footprint during training. Although only the singular values $S_r \in \mathbb{R}^r$ and the scaling vector $O \in \mathbb{R}^d$ are learnable, the method still requires storing the frozen singular vectors $U_r \in \mathbb{R}^{d \times r}$ and $V_r \in \mathbb{R}^{k \times r}$ in memory during training. These matrices contain $dr + kr$ elements, which is comparable to the memory requirements of LoRA and VeRA.

The total memory footprint during training can be expressed as:

$$\mathcal{M}_{\text{OSoRA}} = (r + d) + (dr + kr) \qquad (6)$$

where the first term $(r + d)$ represents the trainable parameters, and the second term $(dr + kr)$ represents the frozen singular vectors that must be stored in memory.

This clarification is important because while the trainable parameter count is significantly reduced, the overall memory and computational requirements during training remain similar to other low-rank adaptation methods. However, the key advantage of OSoRA is that after training, the adapted weights can be computed and merged into a single matrix:

$$W = W_0' + \underline{\Lambda_O} U_r \underline{\Lambda_{S_r}} V_r^\top \tag{7}$$

This means that while the singular vectors U_r and V_r need to be kept in memory during training, they do not need to be saved when storing checkpoints or the final adapted weights, significantly reducing storage requirements. During inference, only the merged weight matrix W is needed, eliminating any additional memory or computational overhead compared to using the original pretrained weights.

3.4 Necessity of Dual Vectors O and S_r

These vectors serve distinct purposes and operate in different dimensions:

$O \in \mathbb{R}^d$ controls scaling along the output dimension, allowing the model to selectively emphasize or de-emphasize specific output features.

$S_r \in \mathbb{R}^r$ controls the importance of each rank component, effectively weighting the contribution of each singular vector pair.

Since $r \ll d$ in typical applications (e.g., $r = 256$ while $d = 4096$), these vectors operate in spaces of different dimensionality and cannot be collapsed into a single vector. This dual-vector approach provides OSoRA with greater expressivity.

Furthermore, initializing S_r with the top singular values from the pretrained weights provides OSoRA with a principled starting point that captures the most important directions of variation in the original weight matrix, while O allows for fine-grained control over how these directions affect each output dimension. This effectively enables fine-tuning within the most important low-rank subspace, while O is responsible for regulating energy distribution across the complete output space.

3.5 Parameter Efficiency Analysis

OSoRA achieves significant parameter efficiency compared to other methods. The total number of trainable parameters in OSoRA is $r + d$, where r is the rank and d is the output dimension of the weight matrix.

Comparison with LoRA LoRA requires $r(d+k)$ trainable parameters, where k is the input dimension. The ratio of parameters between OSoRA and LoRA is:

$$\frac{\mathcal{P}_{\text{OSoRA}}}{\mathcal{P}_{\text{LoRA}}} = \frac{r+d}{r(d+k)} = \frac{1}{d+k} + \frac{d}{r(d+k)} \tag{8}$$

For large values of r, d, and k (typical in LLMs), this ratio becomes very small, demonstrating OSoRA's superior parameter efficiency.

Comparison with VeRA VeRA requires $r + d$ trainable parameters, the same as OSoRA. However, OSoRA's initialization from the pretrained weights' SVD provides a more informed starting point for fine-tuning, potentially leading to better performance with the same parameter count.

3.6 Optimization Dynamics

The optimization dynamics of OSoRA differ from those of other methods due to its unique parameterization. When updating the singular values S_r and the scaling vector O, the gradients flow through the fixed singular vectors U_r and V_r, which capture the principal directions of variation in the original weight matrix.

Let $\mathcal{L}$ be the loss function. The gradients with respect to the trainable parameters are:

$$\frac{\partial \mathcal{L}}{\partial S_r} = \text{diag}(U_r^\top \Lambda_O \frac{\partial \mathcal{L}}{\partial \Delta W} V_r) \tag{9}$$

$$\frac{\partial \mathcal{L}}{\partial O} = \text{diag}(\frac{\partial \mathcal{L}}{\partial \Delta W} U_r \Lambda_{S_r} V_r^\top) \tag{10}$$

These gradients show that the updates to S_r are influenced by how well the corresponding singular vectors align with the desired weight update direction, while updates to O are influenced by the overall contribution of each output dimension to the loss.

4 Experiments

In this section, we present a comprehensive evaluation of OSoRA through a series of experiments. We first compare OSoRA against state-of-the-art PEFT methods including LoRA, VeRA, DoRA, and other baselines on Common Sense Reasoning and Mathematics benchmarks. We then examine OSoRA's robustness across different rank configurations to assess its stability and performance characteristics.

Additionally, we perform detailed ablation studies to analyze the contribution of each component in our method, with particular focus on how different initialization strategies affect the overall performance.

Model	Method	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC_e	ARC_c	OBQA	Avg.
LLaMA2-13B	LoRA	65.84	73.78	53.68	48.63	51.78	79.01	59.32	60.00	61.51
	DoRA	61.07	**73.94**	54.25	49.98	51.46	79.19	**61.02**	60.40	61.41
	PiSSA	66.09	70.18	45.39	51.94	**52.33**	**82.19**	59.32	**62.40**	61.23
	VeRA	67.16	67.63	48.41	48.78	51.85	79.19	55.25	57.80	59.38
	OSoRA	**74.10**	65.07	**54.76**	**55.13**	50.83	73.72	50.51	56.00	60.02
Qwen1.5-7B	LoRA	83.43	72.47	44.68	71.78	61.96	87.83	77.29	75.20	71.83
	DoRA	83.24	70.95	44.68	**71.82**	61.88	88.01	77.29	76.00	71.73
	PiSSA	84.04	74.32	**44.73**	71.53	61.64	87.65	78.31	74.20	72.05
	VeRA	83.79	78.24	38.74	69.00	**62.27**	88.01	74.92	76.00	71.50
	OSoRA	**84.31**	**78.84**	38.84	69.73	61.56	**88.54**	**78.64**	**76.20**	72.08
Qwen2.5-32B	LoRA	89.85	**90.75**	46.16	92.11	79.16	**97.53**	93.90	89.40	84.86
	DoRA	**90.03**	90.59	46.26	92.06	79.08	97.35	**93.56**	89.80	84.84
	PiSSA	89.76	89.61	**46.62**	91.91	77.66	97.53	91.86	88.60	84.19
	VeRA	87.37	84.87	43.04	92.67	**80.66**	95.41	90.85	89.20	83.00
	OSoRA	88.10	85.85	43.04	**92.76**	78.69	96.83	89.15	**91.40**	83.23

Table 1: Accuracy comparison of LLaMA2-13B, Qwen1.5-7B, and Qwen2.5-32B with different PEFT methods on eight commonsense reasoning tasks. The best results are highlighted in bold.

4.1 Common Sense Reasoning

We evaluate OSoRA on a comprehensive suite of benchmarks: BoolQ (Clark et al., 2019), PIQA (Bisk et al., 2019), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2021), ARC_e and ARC_c (Clark et al., 2018), and OpenBookQA (Mihaylov et al., 2018). We utilize three language models: LLaMA2-13B Chat (Touvron et al., 2023), Qwen1.5-7B Chat (Team, 2024), and Qwen2.5-32B Instruct (Qwen et al., 2025), and configure rank settings of 512, 256, and 1024 for these models, respectively. The CommonSenseQA (Talmor et al., 2019) dataset is used for training all models, and OpenCompass (Contributors, 2023) is employed as the evaluation framework. Following the approach of Hu et al. (2022), OSoRA is applied to the query and value projection matrices in each self-attention module. The optimal learning rates, training epochs, and other hyperparameters were determined through systematic tuning, with detailed configurations available in Table 7.

As demonstrated in Table 1, OSoRA achieves competitive performance across all evaluated models. For Qwen1.5-7B, OSoRA achieves the highest average score (72.08%) among all methods, outperforming LoRA (71.83%), DoRA (71.73%), PiSSA (72.05%), and VeRA (71.50%). OSoRA excels particularly on BoolQ (84.31%), PIQA (78.84%), ARC_e (88.54%), ARC_c (78.64%), and OBQA (76.20%), achieving the best scores among all methods. For LLaMA2-13B, OSoRA shows strong performance on BoolQ (74.10%), SIQA (54.76%), and HellaSwag (55.13%), while for Qwen2.5-32B, it performs best on HellaSwag (92.76%) and OBQA (91.40%). These results are notable given OSoRA's significantly reduced parameter count compared to other methods.

4.2 Mathematics

For the mathematical task, we follow the experimental setup from Meng et al. (2024) and fine-tune the Mistral-7B Instruct v0.3 (Jiang et al., 2023) and LLaMA3-8B Instruct (Grattafiori et al., 2024) models. The training set is the MetaMathQA dataset (Yu et al., 2024) and the evaluation framework is also the OpenCompass (Contributors, 2023). The hyperparameters are detailed in Table 8.

As shown in Table 2, OSoRA demonstrates superior performance on the mathematical task across both models. For Mistral-7B v0.3, OSoRA achieves the highest scores on both MATH (Hendrycks et al., 2021) (12.10%) and GSM8K (Cobbe et al., 2021) (54.81%), outperforming the next best method PiSSA by 0.14% and 1.82% respectively. Similarly, for LLaMA3-8B, OSoRA attains the best results with 27.36% on MATH and 78.85% on GSM8K, surpassing LoRA by 0.20% and 5.39% respectively. The average performance gain of OSoRA over other methods is particularly notable (33.46% for Mistral-7B and 53.11% for

Model	Method	MATH	GSM8K	Avg.
Mistral-7B v0.3	LoRA	11.68	51.40	31.54
	DoRA	11.78	51.55	31.67
	PiSSA	11.96	52.99	32.48
	VeRA	10.70	49.20	29.95
	OSoRA	**12.10**	**54.81**	**33.46**
LLaMA3-8B	LoRA	27.16	73.46	50.31
	DoRA	26.60	73.39	50.00
	PiSSA	26.38	74.45	50.42
	VeRA	24.24	75.59	49.92
	OSoRA	**27.36**	**78.85**	**53.11**

Table 2: Accuracy comparison of Mistral-7B v0.3 and LLaMA3-8B with different PEFT methods on MATH and GSM8K benchmarks. The table shows percentage scores for each method, with OSoRA achieving the highest performance on both benchmarks across both models. Results are based on 4-shot evaluation, with the best scores in each category highlighted in bold.

LLaMA3-8B), while requiring significantly fewer trainable parameters compared to alternative approaches.

4.3 Robustness of Different rank settings

This section explores the impact of various rank configurations on OSoRA, VeRA and LoRA by adjusting r within the set {64, 128, 256, 512} for OSoRA and VeRA, and {2, 4, 8, 16, 32, 64} for LoRA, respectively. The performance of the fine-tuned models was assessed on GPQA (Rein et al., 2024) benchmark and the accuracy of the Qwen2-7B model on the GPQA Diamond task is reported. The learning rate is set to $2e^{-5}$ for LoRA, 0.005 for VeRA and OSoRA. Additionally, the batch size is set to 1 for all methods, training for 1 epoch with a warmup rate of 0.03, cosine learning rate schedule.

As shown in Table 3 and Figure 1, we observe that OSoRA demonstrates more stable performance across different rank settings compared to LoRA and VeRA. While LoRA achieves its peak performance at $r = 32$ (36.87%), its accuracy fluctuates significantly across different ranks. VeRA shows similar inconsistency, with its best performance at $r = 256$ (33.84%). In contrast, OSoRA maintains relatively consistent performance across lower ranks and achieves its highest accuracy at $r = 512$ (35.86%). Figure 1 further illustrates that as rank increases, LoRA's parameter count grows exponentially, whereas both VeRA and OSoRA maintain a more efficient linear growth in parameter count. This demonstrates that OSoRA offers a better balance between performance and parameter efficiency, particularly at higher rank settings.

r	LoRA	VeRA	OSoRA
2	29.80	-	-
4	28.28	-	-
8	30.30	-	-
16	28.79	-	-
32	36.87	-	-
64	32.83	31.31	31.82
128	-	27.78	30.81
256	-	33.84	31.31
512	-	29.80	35.86

Table 3: Accuracy comparison of Qwen2-7B model with different PEFT methods (LoRA, VeRA, and OSoRA) across various rank settings on the GPQA Diamond task. The results show how different rank values affect model performance.

4.4 Ablation Study

Impact of Training Individual Components (S_r or O) The importance of jointly training both components S_r and O in Equation (4) is first examined. In this analysis, two simplified variants are considered: one where only S_r is trained while O remains fixed as an all-ones vector, and another where only O is trained while S_r remains fixed at the initial singular values derived from the decomposition of W_0. The experimental setup from Section 4.2 is maintained.

The results of our ablation study on mathematical tasks (MATH and GSM8K) using the Mistral-7B v0.3 model are presented in Figure 3. Three variants are compared: standard OSoRA (where both S_r and O are trained), OSoRA* (where S_r is fixed and only O is trained), and OSoRA** (where O is fixed and only S_r is trained). It is clearly demonstrated by the results that superior performance is yielded by jointly training both components compared to when either component is trained individually. On the MATH benchmark, 12.1% accuracy is achieved by standard OSoRA, by which OSoRA* (10.08%) and OSoRA** (9.02%) are significantly outperformed. Similarly, on GSM8K, 54.81% accuracy is reached by standard OSoRA, compared to 49.05% for OSoRA* and 44.73% for OSoRA**.

Notably, a more pronounced performance drop is observed when O is fixed (OSoRA**), by which

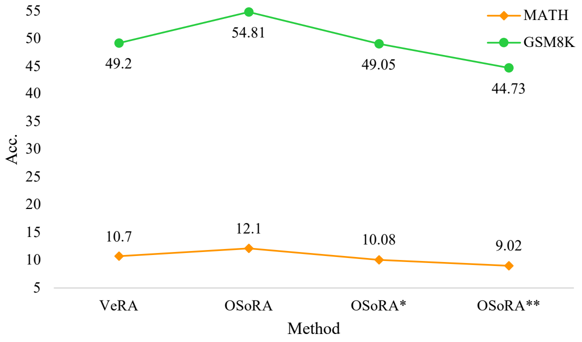

Figure 3: Ablation study on the impact of training different components in OSoRA. The figure compares accuracy on mathematical tasks (MATH and GSM8K) across three variants: standard OSoRA with both S_r and O trained, OSoRA* with only O trained (fixed S_r), and OSoRA** with only S_r trained (fixed O). The results highlight that joint training of both components achieves the best performance, while fixing the output dimension vector O leads to the largest degradation in model accuracy.

it is suggested that a particularly crucial role in the adaptation process is played by the output dimension scaling vector. This finding is aligned with the theoretical understanding that fine-grained control over how the model's output dimensions are adjusted during adaptation is provided by O. Meanwhile, the importance of different principal components is modulated by the singular value vector S_r, by which optimal performance is also essentially enabled.

Our design choice to jointly train both components is validated by these results, as complementary aspects of the adaptation process are captured by them that cannot be fully realized when either component is trained in isolation.

Impact of Gaussian Distribution Initialization for Vector O In this experiment, the impact of initializing the learnable vector O with $\mathcal{Gaussian}$ distribution (denoted as OSoRA$_\mathcal{G}$) instead of ones in Equation (4) is investigated.

The results of this comparison on the mathematical tasks using LLaMA3-8B are presented in Table 4. It is revealed by the findings that notably worse performance is led to by initializing O with a $\mathcal{Gaussian}$ distribution (OSoRA$_\mathcal{G}$) compared to the standard ones initialization used in OSoRA. Specifically, only 24.12% accuracy on MATH and 73.62% on GSM8K are achieved by OSoRA$_\mathcal{G}$, compared to OSoRA's 27.36% and 78.85%, respectively. A significant performance drop of 3.24% on MATH and 5.23% on GSM8K is represented by this.

Model	MATH	GSM8K	Avg.
VeRA	24.24	75.59	49.92
OSoRA	27.36	78.85	53.11
OSoRA$_\mathcal{G}$	24.12	73.62	48.87

Table 4: Accuracy comparison of OSoRA and OSoRA$_\mathcal{G}$ on the MATH and GSM8K tasks. The table shows that OSoRA achieves better performance than OSoRA$_\mathcal{G}$ on both tasks, with a 3.24% higher accuracy on MATH (27.36% vs. 24.12%) and 5.23% higher on GSM8K (78.85% vs. 73.62%), resulting in a 4.24% higher average score (53.11% vs. 48.87%).

Interestingly, comparable performance to VeRA is shown by OSoRA$_\mathcal{G}$ (24.12% vs. 24.24% on MATH and 73.62% vs. 75.59% on GSM8K), by which it is suggested that a crucial role in OSoRA's effectiveness is played by the initialization strategy. A more stable starting point for adaptation is provided by the all-ones initialization, by which the pretrained weights' singular vectors can be leveraged more effectively from the beginning of training.

Exploring Input-Dimension Vector Adaptation: OSoRA$_k$ In this experiment, OSoRA$_k$ is introduced as a variant of OSoRA where the learnable vector $O \in \mathbb{R}^d$ (output dimension) in Equation (4) is replaced with $O \in \mathbb{R}^k$ (input dimension). The formulation can be expressed as:

$$y = W_0' x + U_r \underline{\Lambda_{S_r}} V_r^\top \underline{\Lambda_O} x \tag{11}$$

where $\Lambda_O \in \mathbb{R}^{k \times k}$ is a diagonal matrix constructed from the learnable vector $O \in \mathbb{R}^k$.

Following the experimental setup described in Section 4.2, OSoRA$_k$ is evaluated against the original OSoRA on both MATH and GSM8K benchmarks. The comparative results across different models are presented in Table 5. It is indicated by the findings that similar performance levels are achieved by both variants. On Mistral-7B v0.3, a slight advantage on MATH (12.10% vs. 11.98%) is demonstrated by OSoRA, while marginally better performance on GSM8K (55.88% vs. 54.81%) is shown by OSoRA$_k$. The pattern is found to be consistent with LLaMA3-8B, where a slight edge on MATH (27.36% vs. 27.34%) is maintained by OSoRA and a minimal advantage on GSM8K (78.92% vs. 78.85%) is shown by OSoRA$_k$. Notably, approximately 50% more trainable parameters (294,912 vs. 196,608) are required

Method	Params	MATH	GSM8K
Mistral-7B v0.3			
OSoRA	196,608	**12.10**	54.81
OSoRA$_k$	294,912	11.98	**55.88**
LLaMA3-8B			
OSoRA	196,608	**27.36**	78.85
OSoRA$_k$	294,912	27.34	**78.92**

Table 5: Accuracy comparison of OSoRA and OSoRA$_k$ on the MATH and GSM8K tasks. The table shows that OSoRA$_k$ achieves the comparable performance as OSoRA but with more parameters.

Method	Params	MATH	GSM8K
Mistral-7B v0.3			
DoRA	6,979,584	11.78	51.55
OSoRA	196,608	12.10	54.81
OSoRA + DoRA	360,448	**12.36**	**55.50**
LLaMA3-8B			
DoRA	6,979,584	26.60	73.39
OSoRA	196,608	**27.36**	78.85
OSoRA + DoRA	360,448	27.12	**79.08**

Table 6: Accuracy comparison of DoRA, OSoRA, and their combination (OSoRA + DoRA) on the MATH and GSM8K tasks. The table shows that combining OSoRA with DoRA can further improve performance while maintaining parameter efficiency.

by OSoRA$_k$, by which it is suggested that superior parameter efficiency is provided by the original OSoRA formulation while competitive performance is maintained.

Integrate OSoRA with DoRA The integration of OSoRA with DoRA is explored to investigate potential performance improvements from combining these PEFT methods. Weight updates are decomposed into magnitude and direction components by DoRA, while singular values with frozen singular vectors are optimized by OSoRA. The complementary strengths of both methods are leveraged through this combination.

The integration of OSoRA with DoRA can be formulated as:

$$y = \|W_0\|_c \frac{W_0' x + \underline{\Lambda_O} U_r \underline{\Lambda_{S_r}} V_r^\top x}{\|W_0' + \underline{\Lambda_O} U_r \underline{\Lambda_{S_r}} V_r^\top\|_c} \tag{12}$$

where $\|\cdot\|_c$ denotes the vector-wise norm of a matrix across each column vector, similar to DoRA's approach. OSoRA's parameter efficiency is maintained while DoRA's magnitude-direction decomposition benefits are gained through this formulation.

The combined approach is evaluated on the Mathematical task using the experimental setup described in Section 4.2. Comparative results across different models are presented in Table 6. Performance enhancement is indicated by integrating OSoRA with DoRA. The best results on both MATH (12.36%) and GSM8K (55.50%) are achieved by the combined approach for Mistral-7B v0.3, outperforming both individual methods. For LLaMA3-8B, while better MATH performance is shown by OSoRA alone, the highest GSM8K score (79.08%) is achieved by the combined approach. Only 360,448 trainable parameters are required by the combined approach, which is significantly fewer than DoRA's 6,979,584 parameters, by which OSoRA's parameter efficiency advantage is maintained while performance is potentially improved.

5 Conclusion

In this paper, we introduced OSoRA, a novel PEFT method that performs SVD to adapt LLMs with minimal trainable parameters. Our approach combines the strengths of existing PEFT methods while addressing their limitations. By initializing with the top singular vectors of pretrained weights and training only singular values and scaling vectors, OSoRA achieves superior performance across various tasks while maintaining parameter efficiency.

Our extensive experiments demonstrate that OSoRA consistently outperforms state-of-the-art PEFT methods including LoRA, DoRA, PiSSA, and VeRA across common sense reasoning and mathematical tasks. The method's effectiveness is particularly notable on complex tasks like MATH and GSM8K, where it achieves comparable or better results with orders of magnitude fewer parameters than competing approaches.

We also explored variations of OSoRA, including OSoRA$_k$ with additional trainable parameters and integration with DoRA, showing the flexibility and extensibility of our approach. These results highlight the potential of informed initialization strategies in PEFT and contribute to making LLM fine-tuning more accessible and efficient, potentially enabling fine-tuning of increasingly large models on limited computational resources without sacrificing performance.

6 Limitations

Despite OSoRA's promising results, it faces several key limitations. The method requires computing SVD of pretrained weight matrices, introducing computational overhead that may challenge its use with extremely large models. Additionally, by operating within a fixed subspace defined by top singular vectors, OSoRA may struggle with tasks requiring significant departures from pretrained capabilities.

The performance heavily relies on appropriate rank selection - too small fails to capture important variations, while too large wastes computation. Unlike VeRA which can use any rank, OSoRA is constrained by the weight matrix dimensions. Our experiments also focused mainly on decoder-only models, leaving its effectiveness on other architectures like encoder-decoder or multimodal systems largely unexplored.

There are also concerns about potential overfitting on smaller datasets due to the concentrated adaptation in singular values and scaling vectors. Finally, integrating OSoRA with other PEFT methods introduces complexity in implementation and tuning that requires further investigation. These limitations point to important directions for future research and improvement.

References

Asma Ben Abacha, Wen-wai Yim, Yujuan Fu, Zhaoyi Sun, Meliha Yetisgen, Fei Xia, and Thomas Lin. 2025. MEDEC: A Benchmark for Medical Error Detection and Correction in Clinical Notes. *arXiv preprint arXiv:2412.19260*.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2019. PIQA: Reasoning about Physical Commonsense in Natural Language. *arXiv preprint arXiv:1911.11641*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, and Amanda Askell. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

OpenCompass Contributors. 2023. OpenCompass: A Universal Evaluation Platform for Foundation Models. `https://github.com/open-compass/opencompass`.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. *arXiv preprint arXiv:2305.14314*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, and Alex Vaughan. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. In *Thirty-Fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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

Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, and Lucile Saulnier. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Dawid J Kopiczko, Tijmen Blankevoort, and Yuki M Asano. 2024. VeRA: Vector-Based Random Matrix Adaptation. In *International Conference on Learning Representations*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. 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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. 2024. DoRA: Weight-Decomposed Low-Rank Adaptation. In *Forty-First International Conference on Machine Learning*, pages 32100–32121. PMLR.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland. Association for Computational Linguistics.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models. In *Advances in Neural Information Processing Systems*, volume 37, pages 121038–121072. Curran Associates, Inc.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a Suit of Armor Conduct Electricity? A New Dataset for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, and Fei Huang. 2025. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. GPQA: A Graduate-Level Google-Proof Q&A Benchmark. In *First Conference on Language Modeling*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense reasoning about social interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Qwen Team. 2024. Introducing Qwen1.5.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, and Shruti Bhosale. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. 2024. Metamath: Bootstrap Your Own Mathematical Questions for Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive Budget Allocation for Parameter- Efficient Fine-Tuning. In *International Conference on Learning Representations*.

A Common Sense Reasoning Hyper-parameters

This section details the hyper-parameter configurations used in our Common Sense Reasoning experiments, including learning rates, batch sizes, and rank settings across different model architectures.

B Mathematics Hyper-parameters

This section details the hyper-parameter configurations used in our mathematical experiments and corresponding ablation studies.

Model	**Method**	r	η	**Batch**†
LLaMA2-13B	LoRA	16	2e-5	20
	DoRA			
	PiSSA			
	VeRA	512	3e-3	
	OSoRA			
Qwen1.5-7B	LoRA	16	2e-5	20
	DoRA			
	PiSSA			
	VeRA	256	5e-2	16
	OSoRA		3e-3	20
Qwen2.5-32B	LoRA	16	2e-5	20
	DoRA			
	PiSSA			
	VeRA	1024	8e-4	16
	OSoRA			

Table 7: Hyper-parameters used for Common Sense Reasoning experiments. All methods were trained for 1 epoch with a warmup rate of 0.03, cosine learning rate schedule, and maximum sequence length of 512. †Batch represents the effective batch size (product of *batch size* and *gradient accumulation steps*).

Model	**Method**	r	η	**Batch**†
Mistral-7B v0.3	LoRA	16	2e-5	128
	DoRA			
	PiSSA			
	VeRA	512	5e-3	
	OSoRA		2e-5	
LLaMA3-8B	LoRA	16	2e-5	128
	DoRA			
	PiSSA			
	VeRA	512	5e-3	
	OSoRA		2e-5	

Table 8: Hyper-parameters used for Mathematical experiments. All methods were trained for 1 epoch with a warmup rate of 0.03, cosine learning rate schedule, and maximum sequence length of 512. †Batch represents the effective batch size (product of *batch size* and *gradient accumulation steps*).