ONE INITIALIZATION TO RULE THEM ALL: FINE-TUNING VIA EXPLAINED VARIANCE ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Foundation models (FMs) are pre-trained on large-scale datasets and then fine-tuned on a downstream task for a specific application. The most successful and most commonly used fine-tuning method is to update the pre-trained weights via a low-rank adaptation (LoRA). LoRA introduces new weight matrices that are usually initialized at random with a uniform rank distribution across model weights. Recent works focus on *weight-driven* initialization or learning of adaptive ranks during training. Both approaches have only been investigated in isolation, resulting in slow convergence or a uniform rank distribution, in turn leading to sub-optimal performance. We propose to enhance LoRA by initializing the new weights in a *data-driven* manner by computing singular value decomposition (SVD) on minibatches of activation vectors. Then, we initialize the LoRA matrices with the obtained right-singular vectors and re-distribute ranks among all weight matrices to explain the maximal amount of variance across layers. This results in our new method **E**xplained **V**ariance **A**daptation (EVA). We apply EVA to a variety of fine-tuning tasks ranging from language generation and understanding to image classification and reinforcement learning. EVA exhibits faster convergence than competitors and attains the highest average score across a multitude of tasks per domain while reducing the number of trainable parameters.

1 INTRODUCTION

Foundation models (Bommasani et al., 2021, FMs) are usually trained on large-scale data and then fine-tuned towards a particular downstream task. This training paradigm has led to significant advancements in the realm of language modeling (OpenAI, 2023; Touvron et al., 2023a; Reid et al., 2024), computer vision (Dehghani et al., 2023; Oquab et al., 2023), and reinforcement learning (Brohan et al., 2023; Zitkovich et al., 2023). With an increasing number of model parameters, the process of fine-tuning becomes prohibitively expensive. This results in the need for efficient alternatives to fine-tuning *all* parameters of the pre-trained model.

Parameter-efficient fine-tuning (PEFT) approaches are commonly used as an effective alternative to full fine-tuning (FFT). PEFT methods modify the pre-trained model by introducing a small number of new trainable parameters, while the pre-trained weights remain frozen. This leads to a substantial reduction in computational cost, both in terms of time and space. A particularly successful approach, LoRA (Hu et al., 2022), introduces new weights in the form of a low-rank decomposition for each weight matrix in the pre-trained model. After training, the new weights can be readily merged into the pre-trained weights without any additional inference latency. Recent research has explored various extensions to LoRA, such as different initialization schemes and adaptive rank allocation (see Table 1). Weight-driven initialization schemes are constrained to the information stored in the pre-trained weights. Further, adaptive rank allocation techniques usually optimize the ranks during the fine-tuning process which results in additional complexity for computing importance scores of ranks. Both approaches have merely been investigated in isolation thus far.

We propose a new method that extends LoRA with adaptive rank allocation and data-driven initialization by leveraging information from the downstream task. Our intuition is that the LoRA matrices should be initialized in a manner such that their linear subspace preserves the maximal amount of variance of the downstream data, i.e. they propagate the maximal amount of information through the model. The rationale behind this is that random initialization introduces wasteful update steps espe-

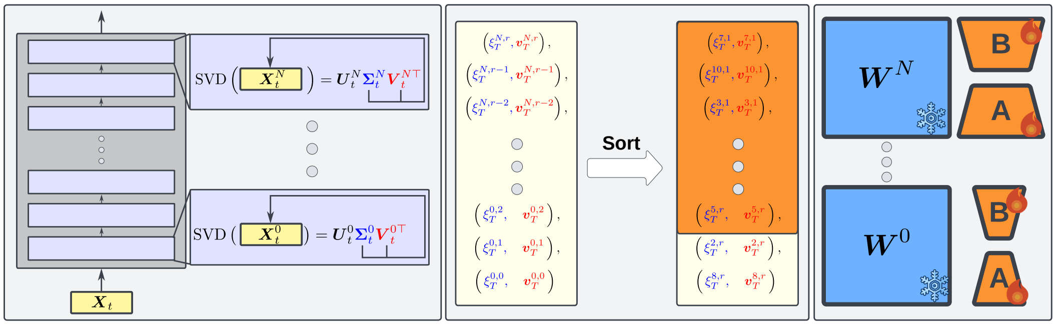

SVD $(X_t^N) = U_t^N \Sigma_t^N V_t^{N\top}$
SVD $(X_t^0) = U_t^0 \Sigma_t^0 V_t^{0\top}$
X_t
$(\xi_T^{N,r}, v_T^{N,r})$,
$(\xi_T^{N,r-1}, v_T^{N,r-1})$,
$(\xi_T^{N,r-2}, v_T^{N,r-2})$,
$(\xi_T^{0,2}, v_T^{0,2})$,
$(\xi_T^{0,1}, v_T^{0,1})$,
$(\xi_T^{0,0}, v_T^{0,0})$
Sort
$(\xi_T^{7,1}, v_T^{7,1})$,
$(\xi_T^{10,1}, v_T^{10,1})$,
$(\xi_T^{3,1}, v_T^{3,1})$,
$(\xi_T^{5,r}, v_T^{5,r})$,
$(\xi_T^{2,r}, v_T^{2,r})$,
$(\xi_T^{8,r}, v_T^{8,r})$
W^N
B
A
W^0
B
A

Figure 1: **Left:** We perform incremental SVD on activation vectors for the first T minibatches to obtain the right singular vectors. **Middle:** We sort all right-singular vectors according to their explained variance given by their respective singular values and only keep the top-k. **Right:** We allocate the top-k vectors as initialization for $\boldsymbol{A}$ and continue the standard LoRA fine-tuning procedure.

cially in the beginning of training (Meng et al., 2024). To identify the linear subspace that propagates the most amount of information, we employ incremental SVD (Ross et al., 2008) on activation vectors by passing minibatches of downstream data through the model. The right-singular vectors obtained by SVD represent the projection onto the principal components, and their corresponding singular values quantify each component's contribution to the total variance. This incremental approach allows us to efficiently process large datasets at constant memory cost. We initialize the downprojection of LoRA with the right-singular vectors to obtain an initialization that preserves the most variance of the downstream data. This results in a stronger gradient signal at the beginning of fine-tuning and thus, faster convergence. Given a fixed rank budget, we maximize the information propagated through the model, by sorting the right-singular vectors in descending order according to their singular values and allocate the top-k components to their respective weight matrices. This results in an adaptive rank allocation that can be computed at the beginning of training which allocates more complexity to weights where components explain less variance. We call the resulting method EVA, which is short for **E**xplained **V**ariance **A**daptation. Importantly, this procedure can be performed within the first few minibatches of LoRA fine-tuning without significant computational overhead.

We demonstrate the benefits of EVA on an array of downstream tasks, namely language generation and understanding, image classification, and reinforcement learning (RL). EVA consistently improves average performance across a multitude of tasks on each domain compared to LoRA and other recently proposed initialization or rank redistribution methods. For language generation, we fine-tune 7B-9B parameter language models on math and reasoning tasks, where EVA attains the highest average performance. Further, on a set of language understanding tasks, EVA improves the average performance compared to competitors. On image classification we fine-tune a pre-trained vision transformer (Dosovitskiy et al., 2021) on a set of 19 diverse tasks. We find that EVA attains the highest average score and improves over LoRA and established extensions thereof, with most gains on in-domain data. For our RL experiments we conduct fine-tuning on continuous control tasks and find that EVA significantly exceeds performance of LoRA and even exceeds performance of full fine-tuning (FFT) when combined with DoRA (Liu et al., 2024a). Finally, we conduct ablation studies on the different components of EVA. Our contributions are as follows:

- We propose a novel data-driven initialization scheme for LoRA by leveraging incremental SVD on minibatches of activation vectors.
- We propose a data-driven heuristic for adaptive rank allocation based on explained variance.
- We demonstrate the effectiveness of EVA across a variety of different domains.

2 RELATED WORK

LoRA (Hu et al., 2022) has sparked widespread interest in leveraging low-rank decompositions for fine-tuning due to its simplicity. Building on the success of LoRA, a number of other variants have been proposed (Kopiczko et al., 2024; Zi et al., 2023; Babakniya et al., 2023; Dettmers et al., 2023; Li et al., 2023; Nikdan et al., 2024; Liu et al., 2024a; Zhang et al., 2023a; Hayou et al., 2024; Chavan

Table 1: Comparison of EVA to existing initialization schemes for LoRA. Existing works either focus on weight initialization *or* adaptive rank allocation. EVA **combines** data-driven initialization with adaptive rank allocation to enhance convergence and downstream performance.

Method	Initialization	Adaptive ranks
LoRA (Hu et al., 2022)	Random	✗
AdaLoRA (Zhang et al., 2023a)	Random	✓
PiSSA (Meng et al., 2024)	Weight-driven	✗
OLoRA (Büyükakyüz, 2024)	Weight-driven	✗
LoRA-GA (Wang et al., 2024)	Data-driven	✗
EVA (Ours)	Data-driven	✓

et al., 2023). The most similar variants to EVA are AdaLoRA (Zhang et al., 2023a) and LoRA-GA (Wang et al., 2024). AdaLoRA adaptively alters the number of ranks for LoRA matrices during fine-tuning. Other more recent approaches learn gates to switch ranks on or off during fine-tuning (Liu et al., 2024b; Meo et al., 2024). In contrast, the data-driven initialization allows EVA to redistribute ranks for each LoRA matrix prior to fine-tuning. LoRA-GA is concurrent work that approximates the gradient of the original weight matrix via SVD, requiring computation of the gradients with respect to the original weights. Contrary, EVA initializes $\boldsymbol{A}$ via the right-singular vectors of minibatches of activation vectors, and is therefore less computationally expensive.

Initialization of LoRA matrices Common initialization schemes for neural networks (He et al., 2015; Glorot & Bengio, 2010) were designed to stabilize training of deep neural networks based on activation functions and depth. In the context of PEFT, Hu et al. (2022) and Liu et al. (2022) explored data-driven initialization by either pre-training on a different task first, or by unsupervised pre-training on the task at hand. Contrary, EVA does not require any gradient update steps, therefore it is much more efficient. Similarly, Nikdan et al. (2024) utilize a warm-up stage in LoRA fine-tuning, where gradients with respect to LoRA weights are used to initialize a sparse matrix for sparse adaptation (Sung et al., 2021) in combination with LoRA. Alternatively, Babakniya et al. (2023) initialize LoRA matrices using SVD on weight matrices obtained after a few steps of full fine-tuning for federated learning with heterogeneous data. Meng et al. (2024) use the main directions of the pre-trained weights to initialize the LoRA matrices. In contrast, EVA takes a data-driven approach to initialize the LoRA matrices. Similar initialization schemes were proposed for training deep networks from scratch (Mishkin & Matas, 2016; Krähenbühl et al., 2016).

Increasing efficiency of LoRA Several works have investigated how to increase efficiency of LoRA fine-tuning. Kopiczko et al. (2024) decrease the memory complexity by keeping both $\boldsymbol{A}$ and $\boldsymbol{B}$ frozen while merely training newly-introduced scaling vectors. This way, only random seeds for initializing $\boldsymbol{A}$ and $\boldsymbol{B}$ need to be stored. Another prominent approach is quantization (Dettmers et al., 2022), which has been successfully combined with LoRA (Dettmers et al., 2023). Other LoRA variants are compatible with quantization (Nikdan et al., 2024; Valipour et al., 2023; Meng et al., 2024). It has also been shown that initialization can improve fine-tuning quantized models (Li et al., 2023).

3 METHOD

We aim at initializing LoRA weights in a data-driven manner by leveraging data from the downstream task. Since EVA builds on LoRA (Hu et al., 2022), we first briefly explain LoRA in Section 3.1. Then, we explain the two essential steps conducted in EVA, namely (i), computing a data-driven initialization for the low-rank decomposition of LoRA matrices via SVD on activation vectors (Section 3.2), and (ii), adaptive assignment of ranks across all layers to maximize the explained variance throughout the pre-trained model (Section 3.3).

3.1 LOW-RANK ADAPTATION (LORA)

LoRA adds new trainable weights which are computed via an outer product of low-rank matrices (Hu et al., 2022). This is motivated by the low intrinsic dimensionality of language models (Aghajanyan

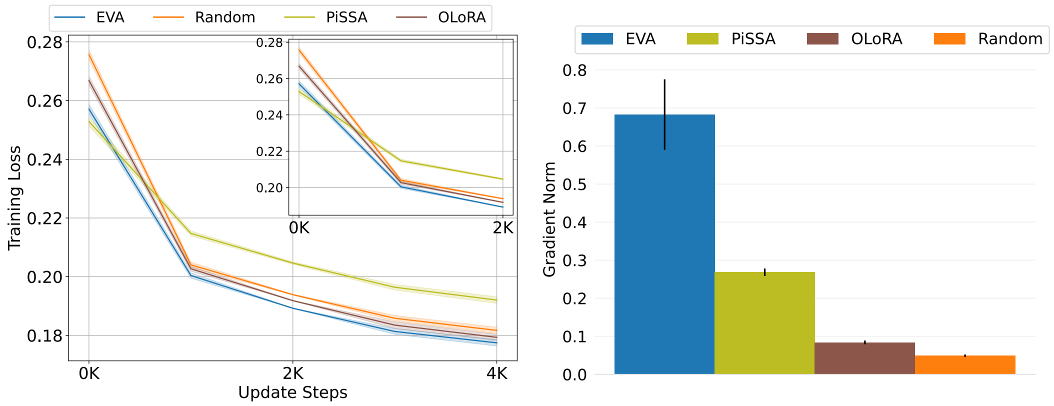


Figure 2: **Left:** Training loss for fine-tuning Llama-3.1-8B on the MetaMathQA dataset. We compare EVA to other initialization methods OLoRA, PiSSA, and random initialization (LoRA). We show mean and standard deviation across three random seeds. **Right:** Mean and standard deviation of gradient norm at the beginning of training for EVA, PiSSA, OLoRA and Random initialization of LoRA matrices. EVA exhibits significantly larger gradient norm leading to faster convergence.

et al., 2021) and relies on the assumption that the gradients during fine-tuning are also of low rank (Gur-Ari et al., 2018; Zhang et al., 2023b; Gauch et al., 2022). Let $\boldsymbol{x} \in \mathbb{R}^{d\times 1}$ be the input to a pre-trained weight matrix $\boldsymbol{W} \in \mathbb{R}^{k\times d}$. Then, LoRA introduces new weight matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ as a low-rank decomposition $\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{A}\boldsymbol{x}$, where $\boldsymbol{B} \in \mathbb{R}^{k\times r}$ and $\boldsymbol{A} \in \mathbb{R}^{r\times d}$. The rank r is a hyperparameter with $r \ll k$. During fine-tuning, $\boldsymbol{W}$ remains frozen while $\boldsymbol{A}$ and $\boldsymbol{B}$ are updated. Usually, $\boldsymbol{B}$ is initialized with zeros and $\boldsymbol{A}$ at random, such that fine-tuning starts from the pre-trained model. Additionally, a hyperparamter α is used to scale $\boldsymbol{B}\boldsymbol{A}\boldsymbol{x}$ by $\frac{\alpha}{r}$.

3.2 DATA-DRIVEN INITIALIZATION OF LOW-RANK ADAPTATION

Our aim is to obtain an effective initialization for $\boldsymbol{A}$ to find a linear subspace that preserves the most information of the downstream task, i.e. that explains the most variance. To this end, we perform SVD on batches of activation vectors $\boldsymbol{X} \in \mathbb{R}^{b\times d}$ to obtain the right-singular vectors, which constitute the directions that capture most of the variance (see Figure 1, left). More formally, we collect batches of activations $\boldsymbol{X}^i$ for N pre-trained weight matrices $\boldsymbol{W}^i \in \{\boldsymbol{W}^0, ..., \boldsymbol{W}^N\}$ that are selected for fine-tuning. Subsequently, we compute the SVD on each $\boldsymbol{X}^i$ to obtain the right-singular vectors $\boldsymbol{v}^i_{j,:}$ and their respective singular values σ^i_j as

$$\boldsymbol{X}^i = \boldsymbol{U}^i\boldsymbol{\Sigma}^i\boldsymbol{V}^{i\top} \approx \sum_{j=1}^{k} \boldsymbol{u}^i_{:,j}\sigma^i_j\boldsymbol{v}^i_{j,:}. \tag{1}$$

Here, $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left- and right-singular values, respectively, and $\boldsymbol{\Sigma}$ is a diagonal matrix containing the singular values. Note that in practice we compute only the top-k components and not the full SVD using truncated SVD (Halko et al., 2011). Generally, the stacked right-singular vectors $\boldsymbol{V}^i_{:r,:}$ are equivalent to the principal components of the covariance matrix of $\boldsymbol{X}^i$ (see proof in Appendix H). Therefore, the approximation of $\boldsymbol{V}^i_{:r,:}$ is a projection onto the principal components that explains the maximal amount of variance. By leveraging this projection as initialization for $\boldsymbol{A}$ we ensure that the maximum amount of information is preserved in the low dimensional subspace. Naively, we could simply collect batches of activations and stack them into a single matrix and perform SVD. However, this results in excessive memory overhead as we usually deal with large datasets and models.

To reduce the memory requirements, we incrementally update $\boldsymbol{V}^i_{:r,:}$ as proposed in Ross et al. (2008) whch is based on the sequential Karhunen-Loeve algorithm (Levy & Lindenbaum, 2000). For simplicity, let us assume that $\boldsymbol{A} = \boldsymbol{X}_0^{i\top}$ and $\boldsymbol{B} = \boldsymbol{X}_1^{i\top}$ are two batches of activations for weight matrix $\boldsymbol{W}^i$ obtained by passing two subsequent batches of the downstream data through the model. The aim is now to compute the SVD of the concatenated activation matrix $\begin{bmatrix}\boldsymbol{A}\boldsymbol{B}\end{bmatrix} = \boldsymbol{U}'\boldsymbol{\Sigma}'\boldsymbol{V}'^{\top}$ in constant memory. Further, We obtain $\boldsymbol{A} = \boldsymbol{U}_t \boldsymbol{\Sigma}_t \boldsymbol{V}_t^\top$ via SVD. Now let $\tilde{\boldsymbol{B}}$ be the component of $\boldsymbol{B}$ that is orthogonal to $\boldsymbol{U}$, which can be obtained via QR-decompositon or via $\tilde{\boldsymbol{B}} = \mathrm{orth}(\boldsymbol{B} - \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{B})$, where $\mathrm{orth}(\cdot)$ performs orthogonalization. Then the SVD of the concatenated activation matrix can be expressed in partitioned form as

$$[\boldsymbol{A}\boldsymbol{B}] = \left[\boldsymbol{U}\tilde{\boldsymbol{B}}\right] \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{U}^\top \boldsymbol{B} \\ \boldsymbol{0} & \tilde{\boldsymbol{B}}^\top \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{V}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}. \tag{2}$$

By setting $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{U}^\top \boldsymbol{B} \\ \boldsymbol{0} & \tilde{\boldsymbol{B}}\boldsymbol{B} \end{bmatrix}$, we can obtain SVD of the concatenated activation matrix by performing SVD on $\boldsymbol{R}$,$\boldsymbol{R} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^\top$, which is constant in time and memory as we only need to compute $\boldsymbol{U}'$ and $\boldsymbol{\Sigma}'$, which do not scale with the number of data samples. Hence, we perform

$$[\boldsymbol{A}; \boldsymbol{B}] = \left(\left[\boldsymbol{U}; \tilde{\boldsymbol{B}}\right] \tilde{\boldsymbol{U}}\right) \tilde{\boldsymbol{\Sigma}} \left(\tilde{\boldsymbol{V}}^\top \begin{bmatrix} \boldsymbol{V}^\top & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}\right), \tag{3}$$

and subsequently obtain $\boldsymbol{U}' = \left[\boldsymbol{U}\tilde{\boldsymbol{B}}\right] \tilde{\boldsymbol{U}}$ and $\boldsymbol{\Sigma}' = \tilde{\boldsymbol{\Sigma}}$. Additionally, Ross et al. (2008) adds a correction factor to account for the changing mean and variance of the data. The supporting proof that the covariance matrix of the original data matrix is equal to the covariance matrix of the concatenated matrix up to a constant factor is given in Ross et al. (2008). In this example, the left-singular values $\boldsymbol{U}$ do not scale with the number of samples. However, in our case we have $\boldsymbol{A} = \boldsymbol{X}_t^i$ and $\boldsymbol{B} = \boldsymbol{X}_{t+1}^i$, i.e. transposed data matrices, therefore it is the right-singular values $\boldsymbol{V}$ that do not depend on the number of samples and can be incrementally updated in constant time and memory. After each update step in the incremental SVD we check whether $\boldsymbol{V}^i$ has converged via cosine similarity, i.e. $\mathrm{cossim}(\boldsymbol{v}_{j,:}^{i,t-1}, \boldsymbol{v}_{j,:}^{i,t}) \geq \tau \quad \forall \quad 1 \leq j \leq r$. Then, we initialize $\boldsymbol{A}^i = \boldsymbol{V}_{:r,:}^i$ and stop computing incremental SVD for inputs to $\boldsymbol{W}^i$. We continue this procedure until all $\boldsymbol{V}_{:r,:}^i$ have converged. We illustrate the full incremental SVD procedure on a sequence of data batches in Algorithm 2 and discuss complexity of this procedure in Appendix F.

3.3 ADAPTIVE RANK ALLOCATION

Algorithm 1 Fine-tuning via EVA

```
Input: FM $\psi(\cdot)$, $\rho$, rank $r$, dataset $\mathcal{D}$
while not all_converged($\psi$) do
    $\boldsymbol{X} \leftarrow \psi$(next($\mathcal{D}$))    ▷ get activations
    $\boldsymbol{V}_{\text{new}}, \boldsymbol{\xi} \leftarrow$ Incremental-SVD($\boldsymbol{X}, \rho r$)
    if isclose($\boldsymbol{V}_{\text{old}}, \boldsymbol{v}_{\text{new}}$) then
        wrap_and_initialize($\boldsymbol{W}_j, \boldsymbol{V}_{\text{new}}$)
    end if
    $\boldsymbol{V}_{old} \leftarrow \boldsymbol{V}_{new}$
end while
redistribute_ranks($\psi, \boldsymbol{\xi}, \boldsymbol{V}_{\text{new}}$)
lora_finetune($\psi, \boldsymbol{X}$)
```

The singular values provide an estimate of the amount of variance each component in $\boldsymbol{V}_{:r,:}^i$ explains. Leveraging this, we can redistribute ranks across weight matrices of the pre-trained model such that the maximum amount of variance is explained. This can be done by allocating more ranks to layers that propagate more information, i.e., explain more variance. The variance explained by each component in $\boldsymbol{V}_{:r,:}^i$ is given by their explained variance ratio

$$\xi_j^i = \frac{\sigma_j^{i^2}}{(M-1)||\boldsymbol{\sigma}^i||_1}, \tag{4}$$

where $||\cdot||_1$ denotes the ℓ_1 norm, $\boldsymbol{\sigma}^i$ is a vector containing all r singular values, and M is the total number of samples used for the incremental SVD. We sort the components $\boldsymbol{v}_{j,:}^i$ for each weight matrix in descending order according to their explained variance ratio ξ_j^i (see Figure 1, middle). Then, we assign the top-k components to their respective pre-trained weights, which results in adaptive rank allocation (see Figure 1, right). Additionally, we introduce a hyperparameter $\rho \in [1, \infty)$ which controls the uniformity of the rank distribution. ρ determines the number of ranks that we compute during SVD and increasing ρ allows for an increasingly heterogeneous rank distribution. Further, ρ controls the maximum number of ranks a weight matrix can receive. For each $\boldsymbol{W}^i$ we compute $r\rho$ components, i.e., we assign $k = r\rho$ in Equation (1), resulting in $Nr\rho$ components in total. For the redistribution we only use the top-l, with $l = Nr$, components according to their explained variance ratio ξ_j^i. Thus, setting $\rho = 1$, results in a uniform rank distribution as in LoRA, but initialized according to EVA. Therefore, ρ provides us with the means to change the rank distribution in a controlled manner prior to fine-tuning at the initialization stage. In practice we found that the redistribution converges for values of $\rho > 2$ (see Appendix G). Finally, we initialize $\boldsymbol{B}$ with zeros and perform standard LoRA fine-tuning. In Algorithm 1 we provide pseudocode for EVA.

Table 2: Comparison of LoRA and DoRA to different initialization and rank re-distribution methods on NLG tasks. We report average performance across three seeds and respective standard deviation in Table 14. EVA+DoRA and EVA consistently attain the highest average performance across all tasks.

Model	Method	BoolQ	PIQA	SIQA	HellaSwag	Winogrande	ARC-e	ARC-c	OBQA	Avg.
Llama-2-7B	LoRA	67.2	83.9	82.0	94.7	84.0	87.8	74.1	84.0	82.2
	AdaLoRA	**74.8**	82.2	80.5	93.3	79.4	86.1	71.1	80.6	81.0
	PiSSA	62.6	84.8	81.2	94.5	84.8	87.8	74.8	85.4	82.0
	OLoRA	68.7	84.8	82.2	95.0	85.0	88.1	74.9	85.2	82.9
	LoRA-GA	69.0	**85.6**	82.3	95.0	85.0	88.7	75.9	85.8	83.4
	EVA	68.3	85.3	**82.9**	**95.2**	**85.2**	88.6	75.8	86.3	83.4
	DoRA	68.3	85.1	82.2	94.9	84.3	88.7	74.8	86.3	83.1
	EVA+DoRA	73.5	85.3	82.4	**95.2**	84.8	**88.9**	**76.0**	**87.3**	**84.2**
Llama-3.1-8B	LoRA	85.7	90.3	83.0	96.9	88.4	94.2	84.8	90.1	89.2
	AdaLoRA	83.9	89.5	81.7	96.2	86.3	93.7	82.7	86.8	87.6
	PiSSA	72.9	87.3	81.6	95.3	87.8	91.7	81.2	87.6	85.7
	OLoRA	86.0	90.4	**83.9**	97.0	88.6	**94.5**	84.7	90.3	89.4
	LoRA-GA	83.7	89.7	83.1	96.7	88.8	94.2	85.3	90.4	89.0
	EVA	85.3	90.4	83.4	97.0	89.0	94.4	**86.0**	90.3	89.5
	DoRA	**86.2**	**90.8**	83.4	96.9	88.6	94.3	84.9	89.4	89.3
	EVA+DoRA	85.8	**90.8**	**83.9**	**97.1**	**89.2**	94.4	85.9	**90.5**	**89.7**
Gemma-2-9B	LoRA	88.3	92.9	85.2	97.8	92.3	97.2	89.9	94.4	92.2
	AdaLoRA	87.3	91.8	84.6	97.3	91.3	97.0	90.0	92.6	91.5
	PiSSA	81.4	90.0	82.5	95.5	89.0	93.6	83.5	90.8	88.3
	OLoRA	87.7	92.5	85.2	97.5	92.5	96.6	88.7	93.7	91.8
	LoRA-GA	87.3	92.1	84.5	97.4	**93.2**	96.4	89.2	94.3	91.8
	EVA	**88.6**	93.0	**85.3**	**97.9**	92.8	**97.5**	**90.5**	94.5	**92.5**
	DoRA	88.3	92.6	84.9	97.7	92.2	97.1	89.9	94.5	92.1
	EVA+DoRA	**88.6**	**93.1**	85.1	**97.9**	92.5	97.3	89.6	**94.8**	92.4

4 EXPERIMENTS

First, we elaborate on implementation details of EVA in Section 4.1. Then, we show results for fine-tuning large language models (LLMs) on math and reasoning tasks in Section 4.2 and language understanding tasks in Section 4.3. Further we show results for image classification in Section 4.4 and decision making tasks in Section 4.5. Finally, in Section 4.6 we demonstrate that the computational overhead induced by EVA over LoRA is negligible and that incremental SVD converges and is invariant to batch order and batch size.

4.1 IMPLEMENTATION DETAILS

We follow the standard LoRA training procedure from Hu et al. (2022). Similar to Kalajdzievski (2023), we found LoRA training to be very sensitive to the scaling parameter α. Therefore, we set $\alpha = 1$ for all our experiments as we found this to be the most stable setting and only tuned the learning rate. We apply EVA to pre-trained weights only, i.e., we do not initialize newly introduced classifier heads. Following Zhang et al. (2023a), we apply LoRA adapters to all pre-trained weight matrices except for the embedding layer. For EVA we always search over $\rho \in \{1, 2\}$ to cover both uniform uniform and adaptive rank allocation and report the best score. For $\rho = 2$ we also set $\alpha = \alpha\frac{r_{new}}{r_{old}}$ to preserve the same scaling factor as set initially. All models we used for fine-tuning are publicly available on the huggingface hub (Wolf et al., 2020). For the implementation of baselines we leverage the widely used PEFT library (Mangrulkar et al., 2022). Across experiments we highlight the highest scores in boldface and underline the second-highest.

4.2 LANGUAGE GENERATION

We fine-tune three different LLMs, namely Llama-2-7B (Touvron et al., 2023b), Llama-3.1-8B (Dubey et al., 2024), and Gemma-2-9B (Rivière et al., 2024) on common sense and math reasoning benchmarks. For common sense reasoning we follow Liu et al. (2024a) and amalgamate a training set consisting of BoolQ (Christopher et al., 2019), PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2020), ARC-e and ARC-c (Clark et al., 2018) and OpenBookQA (Mihaylov et al., 2018). We apply all methods listed in Table 1 to all three models and additionally add a comparison to DoRA (Liu et al., 2024a) and EVA+DoRA, which combines EVA with DoRA. We train all methods with rank $r = 16$ and a learning rate of $5\text{e}-4$ for three random seeds. Further details on the fine-tuning settings can be found in Appendix B. We present our results in Table 2. Across models $\rho = 2$ yields the highest performance while it also notably decreases the number of trainable parameters compared to all LoRA-based methods (see Table 11 in Appendix B). For a comparison to EVA with uniform rank distribution see Table 10 in Appendix B. Moreover, we conduct experiments where we add rank-stabilization (Kalajdzievski, 2023), different learning rates for $\boldsymbol{A}$ and $\boldsymbol{B}$, or different values for α in Table 9 in Appendix B. Additionally, we provide results for leveraging the components that explain the *least* amount of variance in Table 12, which results in worse performance compared to EVA. Finally, EVA as well as EVA+DoRA are consistently among the best performing methods on all individual tasks. This highlights the effectiveness of EVA's data-driven initialization and rank allocation.

For the math fine-tuning experiments, we fine-tune all models on the MetaMathQA dataset (Yu et al., 2024) for one epoch with the same hyperparameters as for the common sense reasoning tasks and report the results in Table 3. We observe that EVA attains the highest performance on the GSM8K dataset for Gemma-2-9B using $\rho = 2$. For Llama-2-7B and Llama-3.1-8B the best performing method is EVA+DoRA using $\rho = 1$ closely followed by EVA. On MATH, EVA+DoRA performs best for Llama-2-7B with $\rho = 1$, while EVA attains the highest score for Llama-3.1-8B with $\rho = 1$ and Gemma-2-9B with $\rho = 2$. These results indicates that the performance of adaptive rank allocation depends on the selected model. We further analyze the resulting rank distributions for different values of ρ for Llama-2-7B and their effect on downstream performance in Appendix G. Finally, we provide additional results for Llama-2-7B on code fine-tuning tasks in Appendix B.

Table 3: Comparison of EVA to other initialization and adaptive rank methods on GSM8K and MATH datasets. We report mean and standard deviation across three random seeds.

Model	Method	GSM8K	MATH
Llama-2-7B	LoRA	$59.7_{\pm.8}$	$10.9_{\pm.2}$
	AdaLoRA	$56.9_{\pm.4}$	$9.6_{\pm.2}$
	PiSSA	$61.1_{\pm.3}$	$12.6_{\pm.4}$
	OLoRA	$60.7_{\pm.5}$	$11.8_{\pm.3}$
	LoRA-GA	$60.2_{\pm.6}$	$11.7_{\pm.4}$
	EVA	$61.9_{\pm.5}$	$13.1_{\pm.3}$
	DoRA	$59.8_{\pm.5}$	$11.5_{\pm.2}$
	EVA+DoRA	$\mathbf{62.5}_{\pm.8}$	$\mathbf{13.4}_{\pm.01}$
Llama-3.1-8B	LoRA	$78.3_{\pm.6}$	$30.1_{\pm.5}$
	AdaLoRA	$76.9_{\pm.2}$	$28.9_{\pm.7}$
	PiSSA	$78.8_{\pm.2}$	$29.5_{\pm.5}$
	OLoRA	$78.0_{\pm.1}$	$31.0_{\pm.7}$
	LoRA-GA	$78.8_{\pm.1}$	$30.0_{\pm.1}$
	EVA	$78.8_{\pm.3}$	$\mathbf{31.2}_{\pm.3}$
	DoRA	$77.9_{\pm.1}$	$30.2_{\pm.5}$
	EVA+DoRA	$\mathbf{79.1}_{\pm.5}$	$30.8_{\pm.4}$
Gemma-2-9B	LoRA	$83.4_{\pm.9}$	$40.7_{\pm.2}$
	AdaLoRA	$83.5_{\pm.5}$	$41.1_{\pm.4}$
	PiSSA	$79.8_{\pm.5}$	$34.9_{\pm.2}$
	OLoRA	$82.2_{\pm.2}$	$39.4_{\pm.6}$
	LoRA-GA	$82.8_{\pm.9}$	$40.4_{\pm.4}$
	EVA	$\mathbf{83.6}_{\pm.8}$	$\mathbf{41.5}_{\pm.3}$
	DoRA	$82.5_{\pm.6}$	$39.7_{\pm.4}$
	EVA+DoRA	$82.9_{\pm.3}$	$40.0_{\pm.6}$

4.3 LANGUAGE UNDERSTANDING

We train RoBERTa$_{\text{Large}}$ (Liu et al., 2019) and DeBERTav3$_{\text{Base}}$ (He et al., 2023) on the GLUE benchmark (Wang et al., 2019). The GLUE benchmark comprises eight downstream tasks, such as natural language inference, or sentiment analysis. Additionally to learning rate, we also search over different ranks within a maximal rank budget ($r \leq 16$). For further details about datasets, implementation, or hyperparameters, we refer to Appendix C. We also add FFT as a baseline, but neglect EVA+DoRA due to time constraints and report Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for the remaining tasks in Table 4. EVA ($\rho = 2$) attains the highest average score across all tasks for both RoBERTa$_{\text{Large}}$ and DeBERTav3$_{\text{Base}}$. Interestingly, DoRA usually only slightly improves over LoRA on low resource tasks (RTE, MRPC), while performing worse in high resource tasks (MNLI, QNLI, QQP, SST2). We also compare LoRA to EVA in Table 17 in Appendix C for different rank budgets, where EVA consistently improves over LoRA. We visualize resulting rank distribution patterns for different GLUE tasks in Appendix C. More ranks are assigned to higher layers of the query, key, and value projections in the self-attention, while the remaining weights often receive less ranks. This is a consistent pattern for both, DeBERTav3$_{\text{Base}}$ and RoBERTa$_{\text{Large}}$ and in line with the reduced number of trainable parameters for larger models.

Table 4: Comparison of all methods for RoBERTa$_{\text{Large}}$ (top) and DeBERTav3$_{\text{Base}}$ (bottom) on GLUE tasks. We report mean and standard deviation of Matthew's correlation for CoLA, Pearson correlation for STS-B, matched accuracy for MNLI, and accuracy for remaining tasks. For CoLA, RTE, MRPC, and STS-B we average over five seeds and for the remaining tasks over three seeds.

Method	MNLI	QNLI	QQP	SST2	CoLA	MRPC	RTE	STS-B	Avg
FFT	90.2	94.7	**92.2**	**96.4**	68.0	90.9	86.6	92.4	88.93
LoRA	90.7$_{\pm.1}$	94.8$_{\pm.1}$	92.0$_{\pm.0}$	96.2$_{\pm.3}$	69.1$_{\pm.5}$	91.1$_{\pm.6}$	88.1$_{\pm1.1}$	92.3$_{\pm.1}$	89.29
AdaLoRA	90.5$_{\pm.1}$	94.8$_{\pm.2}$	90.6$_{\pm.1}$	96.1$_{\pm.2}$	68.2$_{\pm.7}$	90.7$_{\pm.6}$	84.4$_{\pm.9}$	91.8$_{\pm.1}$	88.39
PiSSA	90.1$_{\pm.1}$	94.7$_{\pm.0}$	91.0$_{\pm.0}$	96.1$_{\pm.2}$	68.7$_{\pm1.3}$	90.4$_{\pm.6}$	87.6$_{\pm.5}$	92.5$_{\pm.3}$	88.89
OLoRA	**90.9**$_{\pm.1}$	**95.0**$_{\pm.1}$	92.0$_{\pm.2}$	96.3$_{\pm.3}$	69.0$_{\pm1.5}$	91.0$_{\pm1.0}$	87.9$_{\pm1.2}$	92.4$_{\pm.1}$	89.32
EVA	90.8$_{\pm.1}$	**95.0**$_{\pm.2}$	92.1$_{\pm.1}$	96.2$_{\pm.1}$	**69.5**$_{\pm1.4}$	**91.4**$_{\pm.8}$	**88.8**$_{\pm1.2}$	**92.6**$_{\pm.1}$	**89.55**
DoRA	89.5$_{\pm.1}$	94.6$_{\pm.1}$	89.9$_{\pm.1}$	96.1$_{\pm.1}$	69.3$_{\pm.8}$	91.0$_{\pm.6}$	88.4$_{\pm1.2}$	92.4$_{\pm.1}$	88.90
FFT	90.1	94.0	92.4	95.6	69.2	89.5	83.8	91.6	88.28
LoRA	90.5$_{\pm.1}$	94.3$_{\pm.1}$	92.4$_{\pm.1}$	95.2$_{\pm.3}$	72.0$_{\pm1.3}$	91.4$_{\pm.7}$	88.9$_{\pm.5}$	91.7$_{\pm.1}$	89.64
AdaLoRA	**90.8**	**94.6**	92.2	96.1	71.5	90.7	88.1	91.8	89.46
PiSSA	90.1$_{\pm.3}$	94.1$_{\pm.1}$	91.8$_{\pm.1}$	95.8$_{\pm.1}$	**72.7**$_{\pm1.7}$	90.9$_{\pm.6}$	86.5$_{\pm1.2}$	91.6$_{\pm.2}$	89.19
OLoRA	90.5$_{\pm.1}$	94.4$_{\pm.1}$	**92.6**$_{\pm.1}$	**96.2**$_{\pm.2}$	72.0$_{\pm1.0}$	91.6$_{\pm.7}$	89.1$_{\pm.9}$	**92.0**$_{\pm.2}$	89.80
EVA	90.6$_{\pm.1}$	94.4$_{\pm.1}$	92.4$_{\pm.04}$	**96.2**$_{\pm.2}$	72.5$_{\pm1.3}$	91.8$_{\pm.6}$	**89.4**$_{\pm.7}$	**92.0**$_{\pm.2}$	**89.91**
DoRA	89.0$_{\pm.2}$	94.1$_{\pm.1}$	88.0$_{\pm.1}$	94.6$_{\pm.4}$	70.3$_{\pm.5}$	**91.9**$_{\pm.6}$	87.8$_{\pm.7}$	91.8$_{\pm.1}$	88.44

Table 5: Fine-tuning DINOv2-g/14 on the VTAB-1K benchmark. Best average performance is highlighted in boldface. We report average accuracy across five seeds.

	Natural							Specialized				Structured								
	Cifar100	Caltech101	DTD	Flower102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	Average
FFT	73.1	89.7	78.4	99.7	92.2	89.5	55.5	74.8	95.0	88.2	70.5	93.6	64.2	**63.6**	68.8	92.0	**64.3**	**50.2**	**56.8**	76.8
LoRA	85.9	92.2	82.2	99.7	94.5	64.1	**63.6**	**88.8**	**97.0**	**92.6**	76.6	**97.7**	65.3	62.1	83.6	90.6	63.0	37.1	52.3	78.4
AdaLoRA	85.4	92.5	81.4	99.7	95.2	90.5	62.2	87.1	96.4	91.2	76.6	94.4	64.4	60.3	83.7	85.4	61.0	32.9	46.0	78.2
PiSSA	85.5	93.6	**82.3**	99.7	94.6	92.8	62.3	87.1	96.6	91.9	76.3	95.0	**66.3**	63.2	**84.9**	90.5	60.1	36.3	48.6	79.4
OLoRA	85.5	93.0	82.1	99.7	95.1	78.3	62.1	86.7	96.3	91.9	**76.8**	94.3	66.0	62.4	71.3	89.0	60.9	34.3	49.5	77.6
EVA	85.6	**93.9**	82.2	99.7	**95.9**	93.2	**63.6**	86.8	96.6	92.3	76.1	96.1	65.1	61.1	83.3	91.4	61.6	35.0	55.0	**79.7**
DoRA	85.9	92.7	82.1	99.7	95.2	34.4	61.4	88.6	96.8	92.4	**76.8**	97.6	65.4	62.7	84.4	43.2	63.1	37.8	52.6	74.4
EVA+DoRA	**86.2**	92.1	81.9	99.7	94.9	**93.8**	62.4	88.3	96.6	**92.6**	76.7	97.2	65.5	54.1	83.7	**93.3**	62.3	37.5	54.5	79.6

4.4 IMAGE CLASSIFICATION

We investigate the efficacy of EVA on the VTAB-1K (Zhai et al., 2019) benchmark, which has been widely used to evaluate PEFT methods. VTAB-1K comprises 19 image classification tasks that are divided into natural images, specialized images (medical images and remote sensing), and structured images (e.g. orientation prediction, depth estimation or object counting). We fine-tune a DINOv2-g/14 model (Oquab et al., 2023) that consists of around 1.1B parameters. For implementation details and hyperparameters see Appendix D. Our results are shown in Table 5 and we additionally report error bars in Table 20. EVA and EVA+DoRA with ($\rho = 2$) attain the best and second-best average accuracy across all tasks, respectively. Interestingly, EVA mainly improves over competitors on the natural tasks, i.e. in-domain datasets. LoRA performs best on the specialized tasks and full fine-tuning (FFT) performs best on the structured task. However, both LoRA and FFT perform worse on the remaining tasks, leading to a worse average score compared to EVA and EVA+DoRA.

4.5 DECISION MAKING

We follow the single task fine-tuning experiments in Schmied et al. (2024) and fine-tune a Decision Transformer (Chen et al., 2021a, DT) on the Meta-World benchmark suite (Yu et al., 2020). Meta-

Table 6: Results for single task fine-tuning experiments on the Meta-World benchmark. We report mean success rates and standard error across three seeds for every task.

	faucet-close	hammer	handle-press	peg-unplug	push-back	push	push-wall	shelf-place	stick-pull	window-close	Average
FFT	$1.0_{\pm .0}$	$\underline{0.97}_{\pm .03}$	$1.0_{\pm .0}$	$\underline{0.77}_{\pm .05}$	$\underline{0.87}_{\pm .05}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$\underline{0.63}_{\pm .03}$	$1.0_{\pm .0}$	0.92
LoRA	$1.0_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$0.6_{\pm .05}$	$0.63_{\pm .1}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$0.4_{\pm .09}$	$1.0_{\pm .0}$	0.86
AdaLoRA	$1.0_{\pm .0}$	$\underline{0.97}_{\pm .03}$	$1.0_{\pm .0}$	$0.4_{\pm .09}$	$0.57_{\pm .1}$	$\underline{0.97}_{\pm .03}$	$\underline{0.97}_{\pm .03}$	$1.0_{\pm .0}$	$0.13_{\pm .07}$	$1.0_{\pm .0}$	0.80
PiSSA	$1.0_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$0.43_{\pm 0.11}$	$0.57_{\pm 0.03}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$0.53_{\pm 0.1}$	$1.0_{\pm .0}$	0.85
OLoRA	$1.0_{\pm .0}$	$\underline{0.97}_{\pm 0.03}$	$1.0_{\pm .0}$	$0.57_{\pm 0.1}$	$0.63_{\pm 0.03}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$0.6_{\pm 0.12}$	$1.0_{\pm .0}$	0.88
EVA	$1.0_{\pm .0}$	$\underline{0.97}_{\pm .03}$	$1.0_{\pm .0}$	$0.63_{\pm .03}$	$0.77_{\pm .05}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$\underline{0.63}_{\pm .07}$	$1.0_{\pm .0}$	0.90
DoRA	$1.0_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$0.6_{\pm 1.2}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$\mathbf{0.67}_{\pm \mathbf{1.5}}$	$1.0_{\pm .0}$	$\underline{0.93}$
EVA+DoRA	$1.0_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$\mathbf{0.8}_{\pm \mathbf{.08}}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$\mathbf{1.0}_{\pm .0}$	$1.0_{\pm .0}$	$\underline{0.63}_{\pm .03}$	$1.0_{\pm .0}$	**0.94**

World consists of a diverse set of 50 tasks for robotic manipulation, such as object manipulation, grasping, or pushing buttons. We split Meta-World according to Wolczyk et al. (2021) into 40 pre-training tasks (MT40) and 10 fine-tuning tasks (CW10). We pre-train a 12 M parameter DT on MT40 and fine-tune it on the CW10 holdout tasks. We report success rates and standard errors for each task of CW10 in Table 6. We observe that EVA significantly reduces that gap between LoRA and FFT. Furthermore, DoRA performs particularly well in this experiment and exceeds FFT performance. Finally, our EVA+DoRA even improves upon DoRA and attains the best average performance across all tasks. We report results for different rank budgets in Table 22, as well as implementation details and hyperparameters in Appendix E.

4.6 SVD CONVERGENCE ANALYSIS

The data-driven initialization of EVA relies on incremental SVD on minibatches of activations in the initial training stage. In Figure 3, left, we show that this process converges for Llama-2-7B on MetaMathQA for different minibatch sizes. Using a minibatch size of 4 the computation for EVA's initialization lasts for approximately 80 seconds, which corresponds to around 90 minibatches. For a batch size of 32 the computation of the SVD components takes around 500 seconds. In Figure 3, right, we additionally show, that the main components obtained via SVD mostly remain consistent across different batch orders for a batch size of 4, again for Llama-2-7B on MetaMathQA. To this end, we plot cosine similarity between components obtained via incremental SVD after rank redistribution. These results indicate that these models exhibit certain activation patterns that remain consistent across different batch orders which lead to a robust initialization for EVA. We also show that the components for different batch sizes converge to mostly the same final initialization in Appendix F.

5 DISCUSSION AND LIMITATIONS

Alternative data-driven initialization schemes. We also investigated alternative data driven initialization schemes. Such alternatives include, but are not limited to, Kernel-PCA (Schölkopf et al., 1997) or Linear Discriminant Analysis (Fisher, 1936, LDA). While Kernel-PCA can account for non-linearities in the data, it scales with the number of datapoints. In our setting we deal with sequences, therefore the number of datapoints grows fast, making Kernel-PCA impractical. LDA projects the data onto a subspace that maximizes linear separability between classes. Such an initialization scheme may be particularly interesting for classification tasks like GLUE or VTAB-1K.

Additional latency of SVD. EVA leads to performance improvements over LoRA, but introduces additional latency in the beginning of training for computing the data-driven initialization. In Table 23 we demonstrate that this process constitutes merely 0.2% of the actual training time for Llama-2-7B on MetaMathQA. Further, in Appendix F we also show that this process is mostly invariant to the batch size, meaning that smaller batch sizes may be used for the SVD computation, resulting in

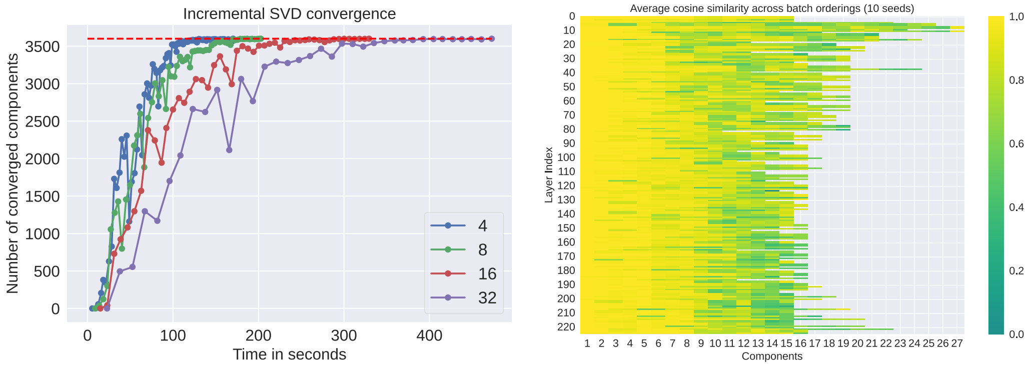


Figure 3: **Left:** Time in seconds until convergence of incremental SVD components for different batch sizes for Llama-2-7B on the MetaMathQA dataset. The dashed line indicates the total number of components. **Right:** Average cosine similarity between SVD components across 10 random seeds for permuting the batch order. The first 10 components remain mostly consistent across all permutations. While the remaining components vary, they strongly correlate with each other.

additional speedup. Since, the SVD computation does not require backpropagation and storing of optimizer states there is no overhead with respect to memory.

Effect of rank redistribution. Our experiments on language understanding tasks indicate that the effect of rank redistribution strongly depends on the downstream task, i.e. all models benefit from the redistribution on the common sense reasoning tasks, whereas for the math tasks a uniform rank distribution appears to perform best. In our experiments on language understanding and image classification, adaptive ranks performed best, while on decision making uniform ranks performed best. Generally the performance gap between the two is not big and since rank redistribution also leads to fewer trainable parameters we recommend to use it by default.

Reproducibility. We provide the source code along with the submission (see Appendix A) to ensure reproducibility. Further, to make EVA more accessible to the community, we will integrate it into the widely used PEFT library (Mangrulkar et al., 2022).

6 CONCLUSION AND BROADER IMPACT

We propose a novel method named Explained Variance Adaptation (EVA), extending the widely used LoRA with data-driven initialization and rank re-distribution. We initialize LoRA matrices in a data-driven manner by performing SVD on minibatches of activation vectors. Further, we re-distribute ranks across weight matrices according to the amount of variance they explain. In this regard, we also introduce a hyperparameter that allows for a controlled investigation of different rank distributions. Thereby, in EVA we bind the benefits of adaptive rank allocation and data-driven initialization, resulting in one initialization to rule them all. We demonstrate performance gains of EVA over LoRA and initialization schemes thereof on a variety of domains, ranging from language to vision and RL. Our results demonstrate that EVA variants consistently reach the highest average performance on a wide range of tasks across all domains.

We believe that EVA sheds a novel view on LoRA fine-tuning, where initialization of the newly introduced weights is guided by the downstream data. As we have shown, this can boost performance on a wide variety of domains. We believe that EVA can have a significant impact on future research on fine-tuning of foundation models, because it inherits all benefits of LoRA while improving performance at no significant additional cost. In the future, we aim at investigating the effect of rank redistribution on other initialization schemes and quantization, as well as alternative data-driven initialization schemes in more detail.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pp. 7319–7328. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.568.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H. Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. Slora: Federated parameter efficient fine-tuning of language models. *CoRR*, abs/2308.06522, 2023. doi: 10.48550/ARXIV.2308.06522.

Charles Beattie, Joel Z. Leibo, Denis Teplyashin, Tom Ward, Marcus Wainwright, Heinrich Küttler, Andrew Lefrancq, Simon Green, Víctor Valdés, Amir Sadik, Julian Schrittwieser, Keith Anderson, Sarah York, Max Cant, Adam Cain, Adrian Bolton, Stephen Gaffney, Helen King, Demis Hassabis, Shane Legg, and Stig Petersen. Deepmind lab. *CoRR*, abs/1612.03801, 2016.

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

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J. Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S. Ryoo, Grecia Salazar, Pannag R. Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong T. Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: robotics transformer for real-world control at scale. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.025.

Kerim Büyükakyüz. Olora: Orthonormal low-rank adaptation of large language models. *CoRR*, abs/2406.01775, 2024. doi: 10.48550/ARXIV.2406.01775.

Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. Algorithms for computing the sample variance: Analysis and recommendations. *The American Statistician*, 37(3):242–247, 1983. ISSN 00031305, 15372731.

Arnav Chavan, Zhuang Liu, Deepak K. Gupta, Eric P. Xing, and Zhiqiang Shen. One-for-all: Generalized lora for parameter-efficient fine-tuning. *CoRR*, abs/2306.07967, 2023. doi: 10.48550/ARXIV.2306.07967.

L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021a.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021b.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proc. IEEE*, 105(10):1865–1883, 2017. doi: 10.1109/JPROC.2017.2675998.

Clark Christopher, Lee Kenton, Chang Ming-Wei, Kwiatkowski Tom, Collins Michael, and Toutanova Kristina. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *NAACL*, 2019.

Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 3606–3613. IEEE Computer Society, 2014. doi: 10.1109/CVPR.2014.461.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah J. Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 7480–7512. PMLR, 2023.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30318–30332. Curran Associates, Inc., 2022.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783.

Li Fei-Fei, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, 2006. doi: 10.1109/TPAMI.2006.79.

Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals Eugenics*, 7: 179–188, 1936.

Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. doi: 10.1080/14786440109462720.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024.

Martin Gauch, Maximilian Beck, Thomas Adler, Dmytro Kotsur, Stefan Fiel, Hamid Eghbal-zadeh, Johannes Brandstetter, Johannes Kofler, Markus Holzleitner, Werner Zellinger, Daniel Klotz, Sepp Hochreiter, and Sebastian Lehner. Few-shot learning by dimensionality reduction in gradient space. In Sarath Chandar, Razvan Pascanu, and Doina Precup (eds.), *Conference on Lifelong Learning Agents, CoLLAs 2022, 22-24 August 2022, McGill University, Montréal, Québec, Canada*, volume 199 of *Proceedings of Machine Learning Research*, pp. 1043–1064. PMLR, 2022.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *Int. J. Robotics Res.*, 32(11):1231–1237, 2013. doi: 10.1177/0278364913491297.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pp. 249–256. JMLR.org, 2010.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient descent happens in a tiny subspace. *CoRR*, abs/1812.04754, 2018.

Nathan Halko, Per-Gunnar Martinsson, and Joel A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2): 217–288, 2011. doi: 10.1137/090771806.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pp. 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens.*, 12(7):2217–2226, 2019. doi: 10.1109/JSTARS.2019.2918242.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Zhiqiang Hu, Lei Wang, Yihuai Lan, Wanyu Xu, Ee-Peng Lim, Lidong Bing, Xing Xu, Soujanya Poria, and Roy Lee. LLM-adapters: An adapter family for parameter-efficient fine-tuning of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5254–5276, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.319.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 1988–1997. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.215.

Kaggle and EyePacs. Kaggle diabetic retinopathy detection, July 2015.

Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *CoRR*, abs/2312.03732, 2023. doi: 10.48550/ARXIV.2312.03732.

Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. ELoRA: Efficient low-rank adaptation with random matrices. In *The Twelfth International Conference on Learning Representations*, 2024.

Philipp Krähenbühl, Carl Doersch, Jeff Donahue, and Trevor Darrell. Data-dependent initializations of convolutional neural networks. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *CoRR*, pp. 32–33, 2009.

Yann LeCun, Fu Jie Huang, and Léon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004), with CD-ROM, 27 June - 2 July 2004, Washington, DC, USA*, pp. 97–104. IEEE Computer Society, 2004. doi: 10.1109/CVPR.2004.144.

Avraham Levy and Michael Lindenbaum. Sequential karhunen-loeve basis extraction and its application to images. *IEEE Trans. Image Process.*, 9(8):1371–1374, 2000. doi: 10.1109/83.855432.

Yixiao Li, Yifan Yu, Chen Liang, Pengcheng He, Nikos Karampatziakis, Weizhu Chen, and Tuo Zhao. Loftq: Lora-fine-tuning-aware quantization for large language models. *CoRR*, abs/2310.08659, 2023. doi: 10.48550/ARXIV.2310.08659.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. *CoRR*, abs/2402.09353, 2024a. doi: 10.48550/ARXIV.2402.09353.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

Zequan Liu, Jiawen Lyn, Wei Zhu, Xing Tian, and Yvette Graham. Alora: Allocating low-rank adaptation for fine-tuning large language models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pp. 622–641. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.NAACL-LONG.35.

Ilya Loshchilov and Frank Hutter. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101, 2017.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods, 2022.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. Pissa: Principal singular values and singular vectors adaptation of large language models, 2024.

Cristian Meo, Ksenia Sycheva, Anirudh Goyal, and Justin Dauwels. Bayesian-lora: Lora based parameter efficient fine-tuning using optimal quantization levels and rank values trough differentiable bayesian gates. *CoRR*, abs/2406.13046, 2024. doi: 10.48550/ARXIV.2406.13046.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

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

Dmytro Mishkin and Jiri Matas. All you need is a good init. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Mahdi Nikdan, Soroush Tabesh, and Dan Alistarh. Rosa: Accurate parameter-efficient fine-tuning via robust adaptation. *CoRR*, abs/2401.04679, 2024. doi: 10.48550/ARXIV.2401.04679.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Sixth Indian Conference on Computer Vision, Graphics & Image Processing, ICVGIP 2008, Bhubaneswar, India, 16-19 December 2008*, pp. 722–729. IEEE Computer Society, 2008. doi: 10.1109/ICVGIP.2008.47.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *CoRR*, abs/2304.07193, 2023. doi: 10.48550/ARXIV.2304.07193.

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pp. 3498–3505. IEEE Computer Society, 2012. doi: 10.1109/CVPR.2012.6248092.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *CoRR*, 2019.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy P. Lillicrap, Jean-Baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew M. Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, and et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *CoRR*, abs/2403.05530, 2024. doi: 10.48550/ARXIV.2403.05530.

Morgane Rivière, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozinska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucinska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju-yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjösund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, and Lilly McNealus. Gemma 2: Improving open language models at a practical size. *CoRR*, abs/2408.00118, 2024. doi: 10.48550/ARXIV.2408.00118.

David A. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *Int. J. Comput. Vis.*, 77(1-3):125–141, 2008. doi: 10.1007/S11263-007-0075-7.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial*

Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pp. 8732–8740. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6399.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. *CoRR*, abs/1904.09728, 2019.

Thomas Schmied, Markus Hofmarcher, Fabian Paischer, Razvan Pascanu, and Sepp Hochreiter. Learning to modulate pre-trained models in rl. *Advances in Neural Information Processing Systems*, 36, 2024.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In Wulfram Gerstner, Alain Germond, Martin Hasler, and Jean-Daniel Nicoud (eds.), *Artificial Neural Networks — ICANN'97*, pp. 583–588, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-69620-9.

Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse masks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 24193–24205, 2021.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023a. doi: 10.48550/ARXIV.2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023b. doi: 10.48550/ARXIV.2307.09288.

Mojtaba Valipour, Mehdi Rezagholizadeh, Ivan Kobyzev, and Ali Ghodsi. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pp. 3266–3279. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EACL-MAIN.239.

Bastiaan S. Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In Alejandro F. Frangi, Julia A. Schnabel, Christos Davatzikos, Carlos Alberola-López, and Gabor Fichtinger (eds.), *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, volume 11071 of *Lecture Notes in Computer Science*, pp. 210–218. Springer, 2018. doi: 10.1007/978-3-030-00934-2_24.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th*

International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. *CoRR*, abs/2407.05000, 2024. doi: 10.48550/ARXIV.2407.05000.

Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Maciej Wolczyk, Michal Zajkac, Razvan Pascanu, Lukasz Kuciński, and Piotr Miloś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6.

Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, pp. 3485–3492. IEEE Computer Society, 2010. doi: 10.1109/CVPR.2010.5539970.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

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

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, André Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. The visual task adaptation benchmark. *CoRR*, abs/1910.04867, 2019.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive budget allocation for parameter-efficient fine-tuning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023a.

Zhong Zhang, Bang Liu, and Junming Shao. Fine-tuning happens in tiny subspaces: Exploring intrinsic task-specific subspaces of pre-trained language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1701–1713, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.95.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *https://arxiv.org/abs/2402.14658*, 2024.

Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices. *CoRR*, abs/2309.02411, 2023. doi: 10.48550/ARXIV.2309.02411.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong T. Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. RT-2: vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Conference on Robot Learning, CoRL 2023, 6-9 November 2023, Atlanta, GA, USA*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 2023.

SUPPLEMENTARY MATERIAL

Anonymous authors
Paper under double-blind review

CONTENTS

A Reproducibility Statement **21**

B Natural language generation **21**
- B.1 Implementation details . . . 21
- B.2 Hyperparameter search . . . 21
- B.3 Additional results . . . 23

C Natural language understanding **25**
- C.1 Dataset Statistics . . . 25
- C.2 Implementation Details . . . 25
- C.3 Hyperparameter search . . . 25
- C.4 Additional results . . . 28

D Image Classification **33**
- D.1 Dataset statistics . . . 33
- D.2 Implementation details . . . 33
- D.3 Hyperparameter search . . . 33
- D.4 Additional results . . . 34

E Decision Making **34**
- E.1 Dataset statistics . . . 34
- E.2 Implementation details . . . 34
- E.3 Hyperparameter search . . . 36
- E.4 Additional results . . . 36

F Incremental SVD convergence analysis **37**
- F.1 Complexity . . . 37
- F.2 Batch Size invariance . . . 37
- F.3 Excluding ignored tokens for SVD . . . 38
- F.4 Efficiency of EVA initialization . . . 38

G Rank re-distribution analysis **38**

H Relation between SVD and PCA **39**

I Ablation Studies **41**

A REPRODUCIBILITY STATEMENT

We provide the source code to reproduce all our experiments in the supplementary material as a zip archive. The archive contains two sub-directories named NLU and NLG, which can be used to reproduce the results on language understanding and generation. For image classification and decision making experiments we used custom implementations which we will open-source as well. Both code directories contain instructions how to install the environment and how to execute all the parameter searches and obtain our results. Additionally, we provide a package that contains implementations for EVA along with different LoRA variants, such as DoRA, and ELoRA in the NLU code directory. We will release a unified codebase upon publication and also integrate EVA into the widely used PEFT library (Mangrulkar et al., 2022).

B NATURAL LANGUAGE GENERATION

We follow the experiments conducted in Hu et al. (2023) and fine-tune Llama-2-7B, Llama-3.1-8B and Gemma-2-9B on 8 common sense reasoning tasks with qa style prompts. We keep the original prompt templates unchanged aside from two minor modifications: For BoolQ we prepend the the passage field before the question and for WinoGrande we add a line "Answer format: ..." analogous to the other prompts. As done by Hu et al. (2023) as well as Liu et al. (2024a) we perform joint finetuning on all 8 tasks. We furthermore evaluate the pre-trained models mentioned above on the mathematical reasoning tasks GSM8K (Cobbe et al., 2021) and Math (Yu et al., 2024) after finetuning on MetaMathQA (Yu et al., 2024) as done in Meng et al. (2024). We keep the original prompt template for finetuning and evaluation. For all datasets we run finetuning for one epoch.

B.1 IMPLEMENTATION DETAILS

For finetuning our code base leverages peft implementations of adapter methods LoRA, AdaLoRA, PiSSA, OLoRA and DoRA. The initialization step for EVA is a custom implementation but for finetuning we can reformulate EVA as a LoRA adapter leveraging the rank_pattern argument of peft.LoraConfig. For evaluation we leverage scripts provided by the MetaMath github repository (Yu et al., 2024) for math reasoning tasks. For common sense reasoning we make use of the lm evaluation harness project (Gao et al., 2024) and define custom tasks using the finetuning prompts. For the SVD computation for joint finetuning on the common sense reasoning tasks we experiment with random and stratified sampling of examples from the 8 tasks and do not notice a difference in performance. All training and evaluation runs for Llama-2-7B were done on 4 A100 GPUs. Runs for Llama-3.1-8B and Gemma-2-9B utilized two different nodes, one with 4 A100 GPUs and one with 4 H200 GPUs.

Table 8: hyperparameters for finetuning on common sense reasoning and math reasoning

Training	
Optimizer	AdamW
Weight Decay	0.0
Lora Dropout	0.0
Batch Size	32
#Epoch	1
LR Schedule	Linear
Warmup ratio	0.03
Label Smooth	0.0
Learning Rate	5e-4
LoRA Dim	16
LoRA α	1
Batch Size SVD (EVA)	16
τ	0.99
Inference	
Beam Size	1.0
Length Penalty	1.0
repetition penalty	1.0

B.2 HYPERPARAMETER SEARCH

The reported results on language generation tasks in Table 2 and Table 3 are the best setting based on a grid search over different learning rates. We apply adapters to all linear layers including the language modelling head. Furthermore we set $\alpha = 1$ for all our experiments. We use AdamW with weight decay and a linear learning rate schedule with warm-up. We train for 1 epoch and use the final checkpoint for evaluation. All hyperparameters are summarized in Table 8

Table 7: Prompt templates with examples (red) used for finetuning on common sense and math reasoning tasks.

Dataset	Fine-tuning Data Template
BoolQ	Passage: Drinking in public – Drinking in public is most commonly accepted. After reading this passage, please answer the following question with true or false, question: can you drink on the street in china Answer format: true/false the correct answer is true
PIQA	Please choose the correct solution to the question: When boiling butter, when it's ready, you can Solution1: Pour it onto a plate Solution2: Pour it into a jar Answer format: solution 1/solution2 the correct answer is solution2
SIQA	Please choose the correct answer to the question: Carson relocated somewhere new. How would you describe Carson? Answer1: mobile Answer2: anxious Answer3: lonely Answer format: answer1/answer2/answer3 the correct answer is answer1
HellaSwag	Please choose the correct ending to complete the given sentence: Playing drums: People are standing behind large drums. A man Ending1: is playing a bag pipe. Ending2: starts to play around the drums. Ending3: begins playing a drum set. Ending4: begins playing the drums. Answer format: ending1/ending2/ending3/ending4 the correct answer is ending4
WinoGrande	Please choose the correct answer to fill in the blank to complete the given sentence: Ian volunteered to eat Dennis's menudo after already having a bowl because _ despised eating intestine. Option1: Ian Option2: Dennis Answer format: option1/option2 the correct answer is option2
ARC-e & ARC-c	Please choose the correct answer to the question: Which factor will most likely cause a person to develop a fever? Answer1: a leg muscle relaxing after exercise Answer2: a bacterial population in the bloodstream Answer3: several viral particles on the skin Answer4: carbohydrates being digested in the stomach Answer format: answer1/answer2/answer3/answer4 the correct answer is answer2
OBQA	Please choose the correct answer to the question: The sun is responsible for Answer1: puppies learning new tricks Answer2: children growing up and getting old Answer3: flowers wilting in a vase Answer4: plants sprouting, blooming and wilting Answer format: answer1/answer2/answer3/answer4 the correct answer is answer4
MetaMathQA	Below is an instruction that describes a task. Write a response that appropriately completes the request. ### Instruction: What is the value of the cosine of 90 degrees? ### Response: s $\\boxed{0}$.The answer is: 0

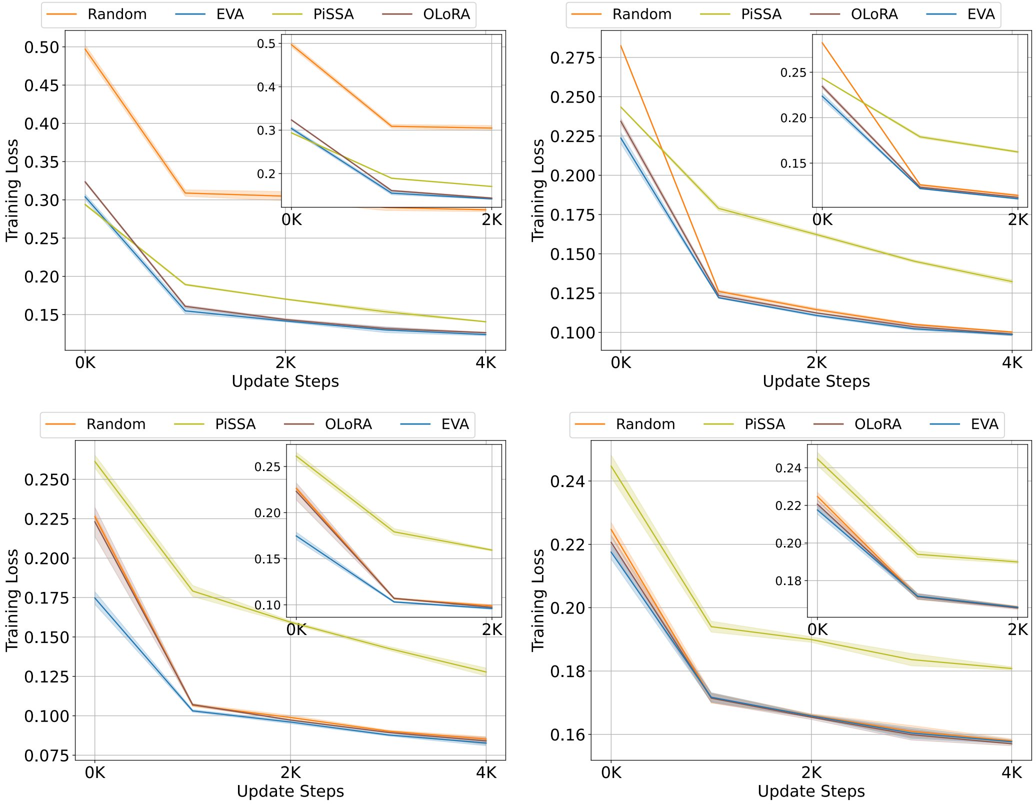


Figure 4: Loss curves for Llama-2-7B on common sense reasoning (top left), Llama-3.1-8B on common sense reasoning (top right), Gemma-2-9B on common sense reasoning (bottom right), and Gemma-2-9B on MetaMathQA. EVA consistently converges the fastest among all competitors.

B.3 ADDITIONAL RESULTS

We present additional loss curves for Llama-2-7B, Llama-3.1-8B, and Gemma-2-9B on the common sense and math reasoning tasks. in Figure 4. We find that EVA converges the fastest for all the different models on the different tasks.

Another experiment we conduct is to apply recently proposed changes to the scaling factor and learning rate. In Table 9 we show results for changing the scaling factor to $\alpha = \frac{2r}{\sqrt{r}}$ which results in rank stabilization (Kalajdzievski, 2023). Further, we present results for the regular setting $\alpha = 2r$ as proposed in Hu et al. (2022). Finally, we also show different learning rates for the two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ as proposed by Hayou et al. (2024). We make the following observations:

1. The standard setting $\alpha = 2r$ from Hu et al. (2022) leads to the worst performance
2. Rank stabilization via $\alpha = \frac{2r}{\sqrt{r}}$ significantly improves the performance of both LoRA and EVA
3. Different learning rates for $\boldsymbol{A}$ and $\boldsymbol{B}$ did not improve the results

To provide a comprehensive comparison about the effect of rank re-distribution, we compare uniform ranks ($\rho = 1$) to adaptive ranks ($\rho = 2$) on the common sense and math reasoning tasks in Table 10. We find that adaptive ranks consistently improves performance for Gemma-2-9B. For Llama-2-7B and Llama-3.1-8B we observe improvements on the common sense reasoning tasks only, while uniform ranks perform better on the math fine-tuning tasks.

In Table 11 we show the number of trainable parameters for EVA ($\rho = 2$) compared to LoRA on the common sense and math reasoning tasks. We find that after rank redistribution, EVA leads to improved performance while reducing the parameter count by approximately 1M. The reason for this is that parameters are usually re-distributed from higher dimensional projections to lower dimensional ones, i.e. from non-attention weights to attention weights. This results in improved performance while reducing the parameter count.

Finally, to verify our intuition that the LoRA matrix $\boldsymbol{A}$ should be initialized with the projection onto the components that explain the most variance, we compare its performance to initializing EVA with the components that explain the *least* amount of variance. We call this method EVA-minor and present results for it in Table 12. To implement EVA-minor, we sample 20 minibatches of data and perform truncated SVD on those and select the resulting minor components. This incurs substantial additional cost, as we must compute all components, whereas for EVA we only approximate the components that explain the most variance. Hence, incremental SVD is not beneficial in this case anymore and it is also not practical as obtaining the initialization takes hours instead of seconds for EVA. Moreover, our data-driven heuristic for adaptive rank allocation is not applicable to this case anymore, therefore we consider uniform ranks. Finally, we find that EVA consistently improves over EVA-minor, highlighting the importance of initializing EVA with the major components, i.e. the ones the explain the most variance.

Table 9: Comparison of EVA to LoRA using recently proposed advancements, such as rank stabilized scaling (Kalajdzievski, 2023) or different learning rates for $\boldsymbol{B}$ and $\boldsymbol{A}$ (Hayou et al., 2024), as well as the originally proposed scaling from Hu et al. (2022).

Adaptation	**Method**	**BoolQ**	**PIQA**	**SIQA**	**HellaSwag**	**Winogrande**	**ARC-e**	**ARC-c**	**OBQA**	**Avg.**
LoRA+	LoRA	64.5	84.7	81.6	94.4	83.8	87.3	73.9	85.5	82.0
	EVA	68.6	85.0	81.2	94.2	84.7	87.4	73.5	84.1	82.3
rsLoRA	LoRA	71.5	85.3	82.5	95.2	84.5	89.0	75.8	86.8	83.8
	EVA	75.5	86.1	82.7	95.4	86.1	89.3	76.3	86.3	84.7
$\alpha = 32$	LoRA	77.9	82.1	80.1	93.2	79.8	86.3	71.5	79.3	81.3
	EVA	68.6	84.9	82.2	94.6	84.1	87.8	74.7	84.4	82.7

Table 10: Comparison of EVA with rank redistribution ($\rho = 2$) and without rank redistribution ($\rho = 1$) for Llama-2-7B, Llama-3.1-8B, and Gemma-2-9B on common sense reasoning and math fine-tuning. Rank re-distribution works well for Gemma-2-9B and for Llama-2-7B and Llama-3.1-8B on the common sense reasoning tasks.

Model	ρ	**Common sense**	**GSM8K**	**MATH**
Llama-2-7B	1	83.4	61.9	13.1
	2	83.4	61.0	12.5
Llama-3.1-8B	1	89.4	78.8	31.2
	2	89.5	78.3	30.8
Gemma-2-9B	1	92.4	83.6	41.3
	2	92.5	83.6	41.5

In addition we also fine-tune Llama-2-7B on the Code-Feedback dataset Zheng et al. (2024) consisting of multi-turn conversations between user and AI Assistant. Due to limited computational resources and the long sequence lengths of the examples in this dataset we do not fine-tune Llama-3.1-8B and Gemma-2-9B or any DoRA variants. We evaluate the fine-tuned checkpoints on four coding benchmarks: MBPP Austin et al. (2021), HumanEval Chen et al. (2021b), MBPP+ and HumanEval+ Liu et al. (2023). The results are presented in Table 13. EVA shows the best performance on MBPP and MBPP+ while also exhibiting good performance on HumanEval and HumanEval+. On the latter two datasets, PiSSA is the best performing method. For finetuning we use a maximum sequence length of 2028 with right-side truncation. For decoding we set the temperature to 0.2 and top_p to 0.7

Table 11: Comparison of number of trainable parameters between LoRA-based methods and EVA on the math and common sense reasoning tasks. Common sense reasoning is an average over eight tasks. #Trainable represents the number of trainable parameters. EVA consistently improves performance while decreasing the number of trainable parameters.

Model	**Method**	**#Trainable**	**Common sense**	**GSM8K**	**MATH**
Llama-2-7B	LoRA	18.3M	82.2	59.7	10.9
	EVA	17.3M	83.4	61.9	13.1
Llama-3.1-8B	LoRA	20M	89.2	78.3	30.1
	EVA	18.9M	89.5	78.8	31.2
Gemma-2-9B	LoRA	24.5M	92.2	83.4	40.7
	EVA	23.1M	92.5	83.6	41.5

Table 12: Comparison of EVA to EVA-minor, which leverages components that explain the *least* amount of variance for initialization of $\boldsymbol{A}$, on the common sense reasoning tasks.

Method	**BoolQ**	**PIQA**	**SIQA**	**HellaSwag**	**Winogrande**	**ARC-e**	**ARC-c**	**OBQA**	**Avg.**
EVA	68.6	85.0	81.2	94.2	84.7	87.4	73.5	84.1	82.3
EVA-minor	64.0	83.4	81.5	94.3	82.0	87.3	73.0	81.6	80.9

In Table 14 we report the standard deviation across three seeds from the results in Table 2. For Llama-3.1-8B and Gemma-2-9B EVA has the smallest average standard deviation across tasks. For Llama-2-7B the standard the variance of EVA is only slightly above average in comparison to other methods, mainly due to the high standard deviation on the BoolQ dataset.

C NATURAL LANGUAGE UNDERSTANDING

C.1 DATASET STATISTICS

The dataset statistics for each task in the GLUE benchmark (Wang et al., 2019) are shown in Table 15. Generally, GLUE contains four low-resource datasets (RTE, MRPC, STS-B, and CoLA) and four high resource datasets (SST-2, QNLI, QQP, MNLI). While CoLA and SST-2 rely on single sentence classification, STS-B evaluates for similarity and the remaining tasks are based on pairwise text classification.

C.2 IMPLEMENTATION DETAILS

We base our implementation on the codebase of LoRA[1]. For these experiments, we initially precompute our initialization prior to the fine-tuning stage and store it as a checkpoint. However, we also provide the possibility to directly compute the initialization during the fine-tuning stage, as done for our experiments on VTAB-1k and Meta-World. By default, we always offload the computation of the initial checkpoint to CPU to save VRAM. We ran all our experiments on nodes with four A100 GPUs and used PyTorch's data-distributed parallel functionality (Paszke et al., 2019). Runtimes ranges from as little as 10 minutes per run for smaller datasets (RTE, STS-B) to around 15 hours for the largest datasets (QQP, MNLI).

C.3 HYPERPARAMETER SEARCH

For LoRA and EVA, we search over the number of ranks $r \in \{2, 4, 6, 8\}$ and different learning rates $\eta \in \{1e-3, 4e-4, 1e-4\}$ for RoBERTa$_{\text{Large}}$ and $\eta \in \{4e-3, 1e-3, 4e-4\}$ for DeBERTav3$_{\text{Base}}$. We report the best hyperparameter settings for both, RoBERTa$_{\text{Large}}$ and DeBERTav3$_{\text{Base}}$ for LoRA and EVA in Table 16. For AdaLoRA, we search over the same ranks and always start initial ranks

---

[1] `https://github.com/microsoft/LoRA`

Table 13: Comparison of EVA to other initialization and rank re-distribution schemes on code fine-tuning datasets. We report mean and standard deviation across three random seeds.

Method	MBPP	HumanEval	MBPP+	HumanEval+
LoRA	$22.2_{\pm 1.1}$	$18.9_{\pm 0.6}$	$30.7_{\pm 1.1}$	$18.9_{\pm 0.6}$
AdaLoRA	$21.5_{\pm 0.2}$	$17.1_{\pm 0.0}$	$29.4_{\pm 0.7}$	$17.1_{\pm 0.0}$
PiSSA	$22.8_{\pm 1.2}$	$\mathbf{19.9}_{\pm 0.9}$	$30.8_{\pm 0.7}$	$\mathbf{19.9}_{\pm 0.9}$
OLoRA	$22.3_{\pm 0.6}$	$18.9_{\pm 0.0}$	$32.4_{\pm 0.4}$	$18.9_{\pm 0.0}$
EVA	$\mathbf{22.9}_{\pm 0.7}$	$18.9_{\pm 1.2}$	$\mathbf{32.6}_{\pm 0.6}$	$18.9_{\pm 1.2}$

Table 14: Standard deviation across three seeds on common sense reasoning tasks.

Model	Method	BoolQ	PIQA	SIQA	HellaSwag	Winogrande	ARC-e	ARC-c	OBQA
Llama-2-7B	LoRA	1.498	0.252	0.233	0.102	0.658	0.072	0.489	0.822
	AdaLoRA	1.315	0.251	0.182	0.098	0.392	0.362	0.106	0.899
	PiSSA	0.358	0.294	0.138	0.096	0.298	0.386	0.494	1.117
	OLoRA	4.938	0.190	0.524	0.062	0.652	0.339	0.672	0.660
	LoRA-GA	10.573	0.416	1.049	0.115	0.344	0.170	0.560	0.721
	EVA	7.974	0.137	1.054	0.101	0.810	0.526	0.421	0.577
	DoRA	2.599	0.290	0.483	0.113	0.244	0.215	0.489	0.525
	EVA+DoRA	5.281	0.273	0.293	0.034	0.853	0.110	0.494	0.249
Llama-3.1-8B	LoRA	0.472	0.194	0.419	0.070	0.197	0.052	0.563	0.189
	AdaLoRA	0.510	0.044	0.261	0.040	0.392	0.201	0.804	0.748
	PiSSA	6.516	0.373	0.603	0.195	0.707	0.325	0.245	0.589
	OLoRA	0.298	0.245	0.397	0.057	0.451	0.173	0.329	0.189
	LoRA-GA	0.539	0.237	0.695	0.115	0.592	0.135	0.729	0.800
	EVA	0.353	0.031	0.194	0.046	0.209	0.292	0.178	0.808
	DoRA	0.225	0.112	0.315	0.014	0.260	0.119	0.698	0.000
	EVA+DoRA	0.225	0.168	0.121	0.117	0.392	0.105	0.175	0.249
Gemma-2-9B	LoRA	0.095	0.277	0.386	0.062	0.324	0.072	0.070	0.589
	AdaLoRA	0.088	0.353	0.217	0.033	0.098	0.209	0.106	0.432
	PiSSA	2.761	0.286	0.214	0.109	0.621	0.447	0.121	0.163
	OLoRA	0.066	0.451	0.501	0.099	0.501	0.267	0.448	0.573
	LoRA-GA	0.662	0.463	0.252	0.072	0.526	0.129	0.617	1.026
	EVA	0.275	0.136	0.111	0.094	0.260	0.119	0.040	0.249
	DoRA	0.189	0.420	0.301	0.074	0.419	0.091	0.000	0.499
	EVA+DoRA	0.132	0.296	0.490	0.070	0.037	0.150	0.715	0.340

with $r + 4$ that are then redistributed during training. For BOFT we sweep over different combinations of block sizes $b \in \{2, 4, 8, 16\}$ which determine the number of multiplicative matrices. Additionally, for both, AdaLoRA and BOFT, we search over the same learning rates as for the other LoRA variants. Further, we introduce hyperparameters that result in additional speed-up of our initialization, namely a threshold τ that considers components as converged, and a threshold δ that stops computation of the initialization when a certain percentage of components have converged. By default, we set $\tau = 0.99$ and $\delta = 1$, i.e. we only stop when all components are converged, and they are almost exactly the same. These parameters provide additional leeway to speed up the initialization stage of EVA.

We have explored the sensitivity of LoRA to different initialization schemes and found that, similar to other prominent initialization schemes (He et al., 2015; Glorot & Bengio, 2010), scale plays an important role along with directions. Originally, (Hu et al., 2022) propose to set $\alpha = 2r$, however, we found that this parameter is quite sensitive as also shown in (Kalajdzievski, 2023). Similarly, different ranks lead to very different results on different downstream tasks. Therefore, we suggest to always search over more ranks and choose the best performing one if the required compute budget is available. We also experimented with different learning rates for the $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices as proposed in (Hayou et al., 2024), however, this did not result in consistent improvements. Instead, we found

Table 15: GLUE benchmark suite statistics and evaluation metric for each corpus sorted by the number of examples in the training set.

Corpus	#Train	#Dev	#Test	Metric
RTE	2.5 k	276	3 k	Accuracy
MRPC	3.7 k	408	1.7 k	Accuracy
STS-B	7 k	1.5 k	1.4 k	Pearson correlation
CoLA	8.5 k	1 k	1 k	Matthew's correlation
SST-2	67 k	872	1.8 k	Accuracy
QNLI	108 k	5.7 k	5.7 k	Accuracy
QQP	364 k	40 k	391 k	Accuracy
MNLI	393 k	20 k	20 k	Accuracy

Table 16: The best hyperparameters RoBERTa$_{\text{Large}}$and DeBERTav3$_{\text{Base}}$that were found via gridsearch for each task of the GLUE benchmark.

Method	Dataset	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
	Optimizer	AdamW							
	Warmup Ratio	0.06							
	LR Schedule	Linear							
RoBERTa$_{\text{Large}}$ LoRA	Batch Size	8	16	8	8	8	8	16	8
	# Epochs	10	10	20	20	10	20	20	10
	LoRA rank	2	8	8	4	8	4	2	2
	Learning rate	4e-4	1e-3	4e-4	1e-3	1e-3	1e-3	1e-3	4e-4
	LoRA α	1							
	Max Seq. Len.	512							
	DDP GPUs	4							
RoBERTa$_{\text{Large}}$ EVA	Batch Size	8	16	8	8	8	8	16	8
	# Epochs	10	10	20	20	10	20	20	10
	LoRA rank	2	2	4	2	16	8	4	4
	Learning rate	4e-4	1e-3	4e-4	1e-3	4e-4	1e-3	1e-3	1e-3
	LoRA α	1							
	Max Seq. Len.	512							
	DDP GPUs	4							
DeBERTav3$_{\text{Base}}$ LoRA	Batch Size	32	32	16	32	64	32	32	16
	# Epochs	30	60	30	80	25	25	80	40
	LoRA rank	8	4	4	8	16	4	4	8
	Learning rate	4e-4	1e-3	4e-3	4e-3	4e-3	4e-3	4e-3	4e-3
	LoRA α	1							
	Max Seq. Len.	512							
	DDP GPUs	4							
DeBERTav3$_{\text{Base}}$ EVA	Batch Size	32	32	16	32	64	32	32	16
	# Epochs	30	60	30	80	25	25	80	40
	LoRA rank	8	2	4	8	16	4	2	2
	Learning rate	4e-4	4e-4	4e-3	4e-3	4e-3	4e-3	4e-3	4e-3
	LoRA α	1							
	Max Seq. Len.	512							
	DDP GPUs	4							

that learning rates for LoRA-style training can be surprisingly high ($4e-3$ for DeBERTav3$_{\text{Base}}$), while for larger models the learning rate needs to be approximately a magnitude smaller. A simple recipe that worked consistently well, was setting $\alpha = 1$, which results in a similar scaling factor as in Kalajdzievski (2023), and searching over a set of small learning rates for larger models and higher

Table 17: Comparison of LoRA to EVA using RoBERTa$_{\text{Large}}$ on all tasks from GLUE for equal rank budgets. Mean and standard deviation of Matthew's correlation for CoLA, pearson correlation for STS-B, and accuracy for remaining datasets on the development set across 5 seeds are shown.

Method	CoLA	MRPC	RTE	STS-B	MNLI	QNLI	QQP	SST-2	Avg
LoRA$_{r=2}$	68.0$_{\pm 1.4}$	90.9$_{\pm .8}$	88.1$_{\pm 1.1}$	92.3$_{\pm .1}$	91.9$_{\pm .1}$	94.8$_{\pm .3}$	90.6$_{\pm .1}$	96.1$_{\pm .1}$	89.09
EVA$_{r=2}$	69.1$_{\pm 1.4}$	90.8$_{\pm .5}$	88.2$_{\pm .7}$	92.5$_{\pm .1}$	90.8$_{\pm .1}$	94.9$_{\pm .1}$	91.9$_{\pm .1}$	96.2$_{\pm .1}$	89.30
LoRA$_{r=4}$	69.1$_{\pm .5}$	90.7$_{\pm .7}$	86.9$_{\pm .2}$	92.3$_{\pm .1}$	90.6$_{\pm .1}$	94.7$_{\pm .2}$	92.0$_{\pm .0}$	96.0$_{\pm .1}$	89.04
EVA$_{r=4}$	69.5$_{\pm 1.4}$	91.4$_{\pm .8}$	88.8$_{\pm 1.3}$	92.6$_{\pm .1}$	90.7$_{\pm .0}$	94.9$_{\pm .1}$	91.8$_{\pm .0}$	96.1$_{\pm .1}$	89.48
LoRA$_{r=8}$	68.8$_{\pm 1.0}$	91.1$_{\pm .6}$	87.1$_{0.7}$	92.2$_{\pm .2}$	90.6$_{\pm .2}$	94.8$_{\pm .1}$	91.8$_{\pm .0}$	96.2$_{\pm .3}$	89.08
EVA$_{r=8}$	69.0$_{\pm 1.4}$	91.1$_{\pm .4}$	88.4$_{\pm .6}$	92.6$_{\pm .3}$	90.6$_{\pm .1}$	94.9$_{\pm .1}$	92.1$_{\pm .1}$	96.1$_{\pm .2}$	89.35
LoRA$_{r=16}$	68.4$_{\pm 1.0}$	90.5$_{\pm .5}$	88.0$_{\pm .5}$	92.3$_{\pm .1}$	90.6$_{\pm .1}$	94.8$_{\pm .1}$	91.9$_{\pm .1}$	96.1$_{\pm .1}$	89.08
EVA$_{r=16}$	69.1$_{\pm .8}$	91.2$_{\pm .8}$	88.0$_{\pm .5}$	92.6$_{\pm .2}$	90.7$_{\pm .0}$	95.0$_{\pm .2}$	91.8$_{\pm .0}$	96.2$_{\pm .1}$	89.33

Table 18: Comparison of LoRA to EVA, EVA-Raw, and EVA-Max for RoBERTa$_{\text{Large}}$on the GLUE tasks CoLA, MRPC, RTE, and STS-B. We report mean and standard deviation of Matthew's correlation for CoLA, pearson correlation for STS-B, matched accuracy for MNLI, and accuracy for remaining tasks across 5 seeds.

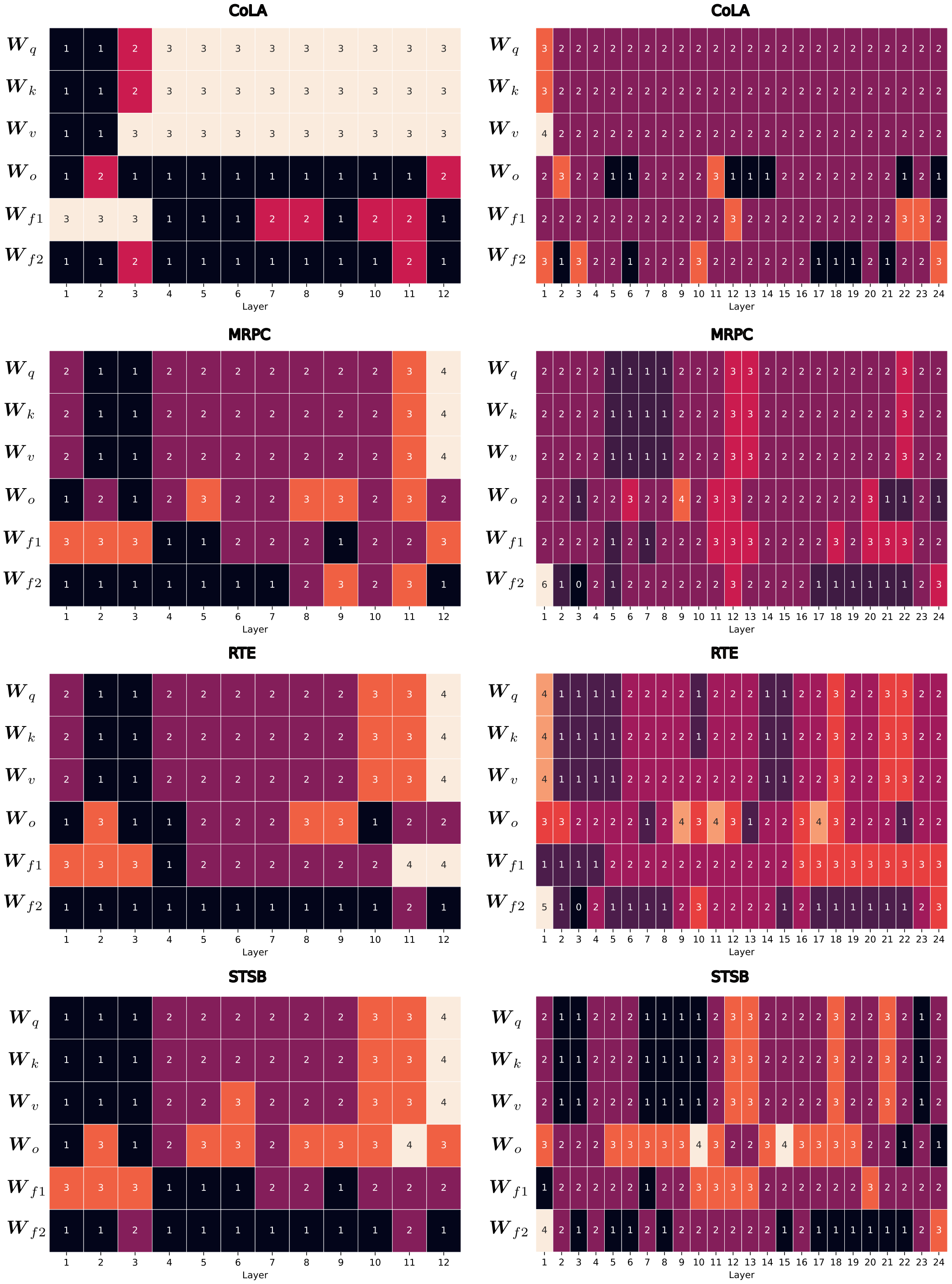
Method	CoLA	MRPC	RTE	STS-B	Avg
LoRA	69.1$_{\pm .5}$	91.1$_{\pm 0.6}$	88.1$_{\pm 1.1}$	92.3$_{\pm 0.1}$	85.2
EVA	**69.5$_{\pm 1.4}$**	**91.4$_{\pm 0.8}$**	**88.8$_{\pm 1.2}$**	**92.6$_{\pm 0.1}$**	**85.6**
EVA-Raw	69.4$_{\pm 1.1}$	91.0$_{\pm 0.9}$	88.2$_{\pm 0.3}$	92.5$_{\pm 0.2}$	85.3
EVA-Max	69.1$_{\pm 0.5}$	91.2$_{\pm 0.5}$	88.4$_{\pm 1.2}$	92.5$_{\pm 0.2}$	85.3

learning rates for smaller ones. For EVA, the only tunable hyperparameter is the rank budget, which we recommend to tune along with the fine-tuning learning rate.

C.4 ADDITIONAL RESULTS

We report additional results for EVA compared to LoRA for different rank budgets in Table 17. We find that EVA consistently outperforms LoRA for different rank budgets. This demonstrates the effectiveness of EVA among different compute budgets. Further, we show additional rank redistributions for the CoLA, MRPC, RTE, and STSB tasks for different for $r = 2$ (Figure 5), $r = 4$ (Figure 6), $r = 8$ (Figure 7), and $r = 16$ (Figure 8) for both, RoBERTa$_{\text{Large}}$ and DeBERTav3$_{\text{Base}}$. The distributions for the different models show different patterns. For DeBERTav3$_{\text{Base}}$ the higher attention layers usually receive more ranks than lower ones. For CoLA, there is also a high number of ranks in the very first layer. For RoBERTa$_{\text{Large}}$ it seems to be the opposite, as the very first layers consistently receive more ranks compared to later layers. There is also a notable difference across tasks for both models, which demonstrates the flexibility of EVA to allocate ranks dependent on the downstream task. Interestingly, for a higher initial rank ($r = 16$), the redistribution for DeBERTav3$_{\text{Base}}$ puts more emphasis on fine-tuning the self-attention specific weight matrices. This is not true for RoBERTa$_{\text{Large}}$, as $\boldsymbol{W}_{f1}$ also receives plenty of ranks across all tasks. Overall, the rank redistribution incurs different fine-tuning paradigms depending on the task and the initial rank.

Additionally, we show results for different rank redistributions that we obtain by using alternative measures for explained variance. Specifically, we compare EVA to using, (i), the raw eigenvalues (EVA-Raw), and (ii), normalizing by the maximum eigenvalue (EVA-Max). We report results for RoBERTa$_{\text{Large}}$ on four of the GLUE tasks, namely CoLA, RTE, MRPC, and STS-B in Table 18. Our results show that while EVA-Raw and EVA-Max slightly improve upon LoRA, they perform worse on average than EVA.



CoLA

	1	2	3	4	5	6	7	8	9	10	11	12
W_q	1	1	2	3	3	3	3	3	3	3	3	3
W_k	1	1	2	3	3	3	3	3	3	3	3	3
W_v	1	1	3	3	3	3	3	3	3	3	3	3
W_o	1	2	1	1	1	1	1	1	1	1	1	2
W_{f1}	3	3	3	1	1	1	2	2	1	2	2	1
W_{f2}	1	1	2	1	1	1	1	1	1	1	2	1

Layer

CoLA

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
W_q	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
W_k	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
W_v	4	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
W_o	2	3	2	2	1	1	2	2	2	2	3	1	1	1	2	2	2	2	2	2	2	1	2	1
W_{f1}	2	2	2	2	2	2	2	2	2	2	2	3	2	2	2	2	2	2	2	2	2	3	3	2
W_{f2}	3	1	3	2	2	1	2	2	2	3	2	2	2	2	2	2	1	1	1	2	1	2	2	3

Layer

MRPC

	1	2	3	4	5	6	7	8	9	10	11	12
W_q	2	1	1	2	2	2	2	2	2	2	3	4
W_k	2	1	1	2	2	2	2	2	2	2	3	4
W_v	2	1	1	2	2	2	2	2	2	2	3	4
W_o	1	2	1	2	3	2	2	3	3	2	3	2
W_{f1}	3	3	3	1	1	2	2	2	1	2	2	3
W_{f2}	1	1	1	1	1	1	1	2	3	2	3	1

Layer

MRPC

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
W_q	2	2	2	2	1	1	1	1	2	2	2	3	3	2	2	2	2	2	2	2	2	3	2	2
W_k	2	2	2	2	1	1	1	1	2	2	2	3	3	2	2	2	2	2	2	2	2	3	2	2
W_v	2	2	2	2	1	1	1	1	2	2	2	3	3	2	2	2	2	2	2	2	2	3	2	2
W_o	2	2	1	2	2	3	2	2	4	2	3	3	2	2	2	2	2	2	2	3	1	1	2	1
W_{f1}	2	2	2	2	1	2	1	2	2	2	3	3	3	2	2	2	2	3	2	3	3	3	2	2
W_{f2}	6	1	0	2	1	2	2	2	2	2	2	3	2	2	2	2	1	1	1	1	1	1	2	3

Layer

RTE

	1	2	3	4	5	6	7	8	9	10	11	12
W_q	2	1	1	2	2	2	2	2	2	3	3	4
W_k	2	1	1	2	2	2	2	2	2	3	3	4
W_v	2	1	1	2	2	2	2	2	2	3	3	4
W_o	1	3	1	1	2	2	2	3	3	1	2	2
W_{f1}	3	3	3	1	2	2	2	2	2	2	4	4
W_{f2}	1	1	1	1	1	1	1	1	1	1	2	1

Layer

RTE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
W_q	4	1	1	1	1	2	2	2	2	1	2	2	2	1	1	2	2	3	2	2	3	3	2	2
W_k	4	1	1	1	1	2	2	2	2	1	2	2	2	1	1	2	2	3	2	2	3	3	2	2
W_v	4	1	1	1	1	2	2	2	2	2	2	2	2	1	1	2	2	3	2	2	3	3	2	2
W_o	3	3	2	2	2	2	1	2	4	3	4	3	1	2	2	3	4	3	2	2	2	1	2	2
W_{f1}	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3
W_{f2}	5	1	0	2	1	1	1	1	2	3	2	2	2	2	1	2	1	1	1	1	1	1	2	3

Layer

STSB

	1	2	3	4	5	6	7	8	9	10	11	12
W_q	1	1	1	2	2	2	2	2	2	3	3	4
W_k	1	1	1	2	2	2	2	2	2	3	3	4
W_v	1	1	1	2	2	3	2	2	2	3	3	4
W_o	1	3	1	2	3	3	2	3	3	3	4	3
W_{f1}	3	3	3	1	1	1	2	2	1	2	2	2
W_{f2}	1	1	2	1	1	1	1	1	1	1	2	1

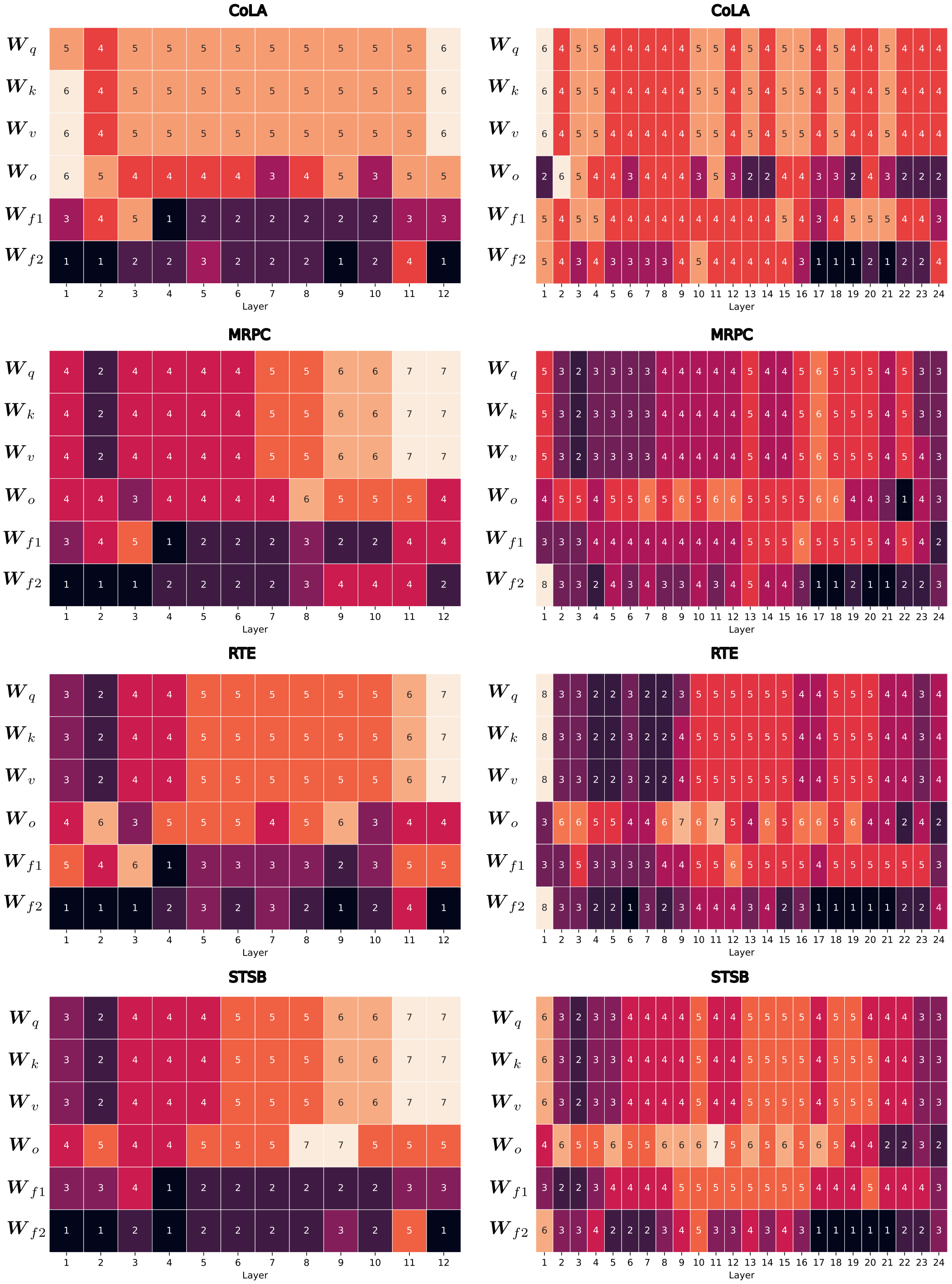
Layer

STSB

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
W_q	2	1	1	2	2	2	1	1	1	1	2	3	3	2	2	2	2	3	2	2	3	2	1	2
W_k	2	1	1	2	2	2	1	1	1	1	2	3	3	2	2	2	2	3	2	2	3	2	1	2
W_v	2	1	1	2	2	2	1	1	1	1	2	3	3	2	2	2	2	3	2	2	3	2	1	2
W_o	3	2	2	2	3	3	3	3	3	4	3	2	2	3	4	3	3	3	3	2	2	1	2	1
W_{f1}	1	2	2	2	2	2	1	2	2	3	3	3	3	2	2	2	2	2	2	3	2	2	2	2
W_{f2}	4	2	1	2	1	1	2	1	2	2	2	2	2	2	1	2	1	1	1	1	1	1	2	3

Layer

Figure 5: Rank distribution after initialization with EVA on four tasks of the GLUE benchmark (CoLA, MRPC, RTE, STSB) for DeBERTav3$_{\text{Base}}$ (left) and RoBERTa$_{\text{Large}}$ (right) with initial rank $r = 2$.



CoLA

	1	2	3	4	5	6	7	8	9	10	11	12
W_q	5	4	5	5	5	5	5	5	5	5	5	6
W_k	6	4	5	5	5	5	5	5	5	5	5	6
W_v	6	4	5	5	5	5	5	5	5	5	5	6
W_o	6	5	4	4	4	4	3	4	5	3	5	5
W_{f1}	3	4	5	1	2	2	2	2	2	2	3	3
W_{f2}	1	1	2	2	3	2	2	2	1	2	4	1

Layer

CoLA

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
W_q	6	4	5	5	4	4	4	4	4	5	5	4	5	4	5	5	4	5	4	4	5	4	4	4
W_k	6	4	5	5	4	4	4	4	4	5	5	4	5	4	5	5	4	5	4	4	5	4	4	4
W_v	6	4	5	5	4	4	4	4	4	5	5	4	5	4	5	5	4	5	4	4	5	4	4	4
W_o	2	6	5	4	4	3	4	4	4	3	5	3	2	2	4	4	3	3	2	4	3	2	2	2
W_{f1}	5	4	5	5	4	4	4	4	4	4	4	4	4	4	5	4	3	4	5	5	5	4	4	3
W_{f2}	5	4	3	4	3	3	3	3	4	5	4	4	4	4	4	3	1	1	1	2	1	2	2	4

Layer

MRPC

	1	2	3	4	5	6	7	8	9	10	11	12
W_q	4	2	4	4	4	4	5	5	6	6	7	7
W_k	4	2	4	4	4	4	5	5	6	6	7	7
W_v	4	2	4	4	4	4	5	5	6	6	7	7
W_o	4	4	3	4	4	4	4	6	5	5	5	4
W_{f1}	3	4	5	1	2	2	2	3	2	2	4	4
W_{f2}	1	1	1	2	2	2	2	3	4	4	4	2

Layer

MRPC

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
W_q	5	3	2	3	3	3	3	4	4	4	4	4	5	4	4	5	6	5	5	5	4	5	3	3
W_k	5	3	2	3	3	3	3	4	4	4	4	4	5	4	4	5	6	5	5	5	4	5	3	3
W_v	5	3	2	3	3	3	3	4	4	4	4	4	5	4	4	5	6	5	5	5	4	5	4	3
W_o	4	5	5	4	5	5	6	5	6	5	6	6	5	5	5	5	6	6	4	4	3	1	4	3
W_{f1}	3	3	3	4	4	4	4	4	4	4	4	4	5	5	5	6	5	5	5	5	4	5	4	2
W_{f2}	8	3	3	2	4	3	4	3	3	4	3	4	5	4	4	3	1	1	2	1	1	2	2	3

Layer

RTE

	1	2	3	4	5	6	7	8	9	10	11	12
W_q	3	2	4	4	5	5	5	5	5	5	6	7
W_k	3	2	4	4	5	5	5	5	5	5	6	7
W_v	3	2	4	4	5	5	5	5	5	5	6	7
W_o	4	6	3	5	5	5	4	5	6	3	4	4
W_{f1}	5	4	6	1	3	3	3	3	2	3	5	5
W_{f2}	1	1	1	2	3	2	3	2	1	2	4	1

Layer

RTE

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
W_q	8	3	3	2	2	3	2	2	3	5	5	5	5	5	5	4	4	5	5	5	4	4	3	4
W_k	8	3	3	2	2	3	2	2	4	5	5	5	5	5	5	4	4	5	5	5	4	4	3	4
W_v	8	3	3	2	2	3	2	2	4	5	5	5	5	5	5	4	4	5	5	5	4	4	3	4
W_o	3	6	6	5	5	4	4	6	7	6	7	5	4	6	5	6	6	5	6	4	4	2	4	2
W_{f1}	3	3	5	3	3	3	3	4	4	5	5	6	5	5	5	5	4	5	5	5	5	5	5	3
W_{f2}	8	3	3	2	2	1	3	2	3	4	4	4	3	4	2	3	1	1	1	1	1	2	2	4

Layer

STSB

	1	2	3	4	5	6	7	8	9	10	11	12
W_q	3	2	4	4	4	5	5	5	6	6	7	7
W_k	3	2	4	4	4	5	5	5	6	6	7	7
W_v	3	2	4	4	4	5	5	5	6	6	7	7
W_o	4	5	4	4	5	5	5	7	7	5	5	5
W_{f1}	3	3	4	1	2	2	2	2	2	2	3	3
W_{f2}	1	1	2	1	2	2	2	2	3	2	5	1

Layer

STSB

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
W_q	6	3	2	3	3	4	4	4	4	5	4	4	5	5	5	5	4	5	5	4	4	4	3	3
W_k	6	3	2	3	3	4	4	4	4	5	4	4	5	5	5	5	4	5	5	5	4	4	3	3
W_v	6	3	2	3	3	4	4	4	4	5	4	4	5	5	5	5	4	5	5	5	4	4	3	3
W_o	4	6	5	5	6	5	5	6	6	6	7	5	6	5	6	5	6	5	4	4	2	2	3	2
W_{f1}	3	2	2	3	4	4	4	4	5	5	5	5	5	5	5	5	4	4	4	5	4	4	4	3
W_{f2}	6	3	3	4	2	2	2	3	4	5	3	3	4	3	4	3	1	1	1	1	1	2	2	3

Layer

Figure 6: Rank distribution after initialization with EVA on four tasks of the GLUE benchmark (CoLA, MRPC, RTE, STSB) for DeBERTav3$_{\text{Base}}$ (left) and RoBERTa$_{\text{Large}}$ (right) with initial rank $r = 4$.

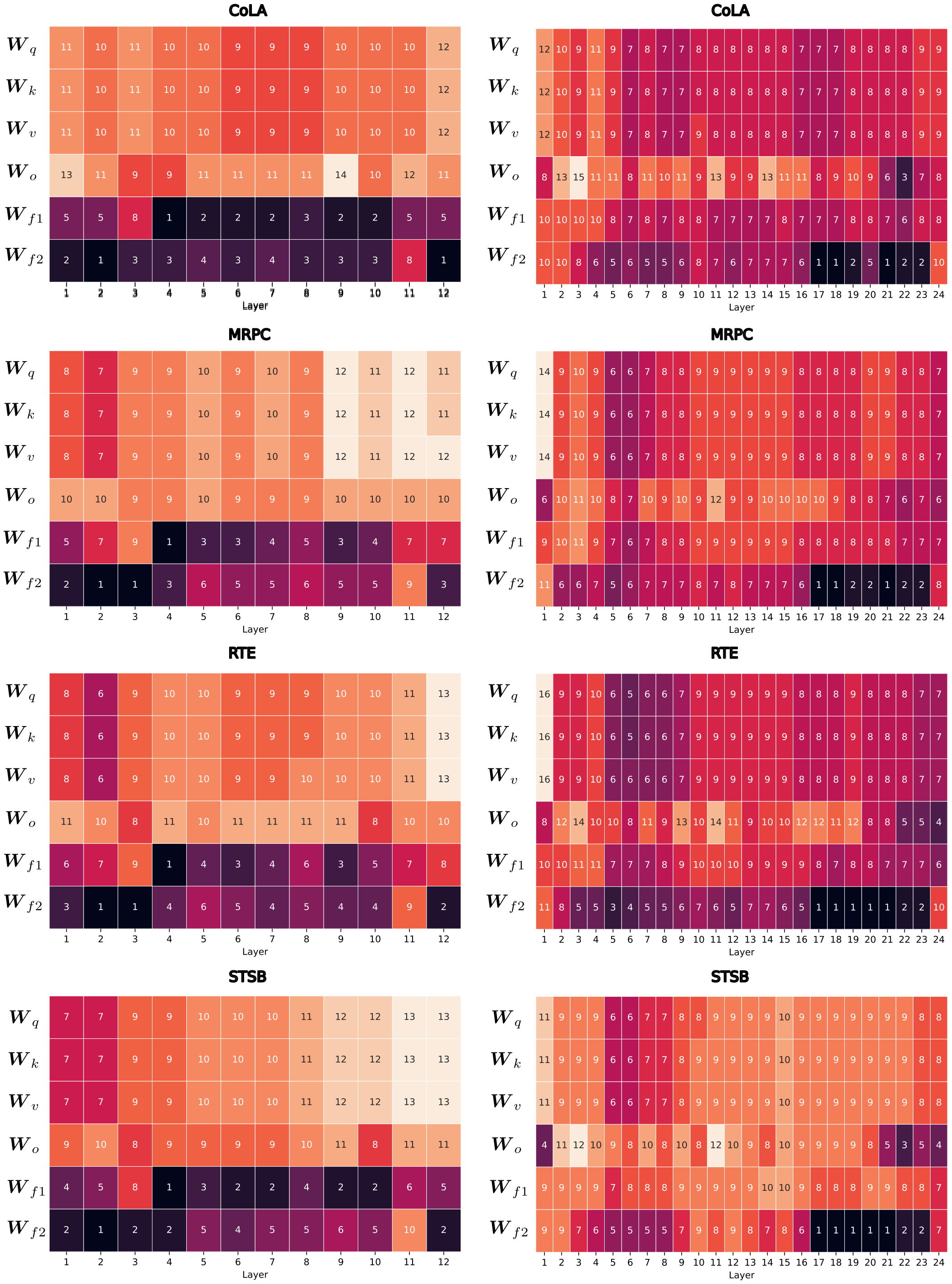


Figure 7: Rank distribution after initialization with EVA on four tasks of the GLUE benchmark (CoLA, MRPC, RTE, STSB) for DeBERTav3$_{\text{Base}}$ (left) and RoBERTa$_{\text{Large}}$ (right) with initial rank $r = 8$.

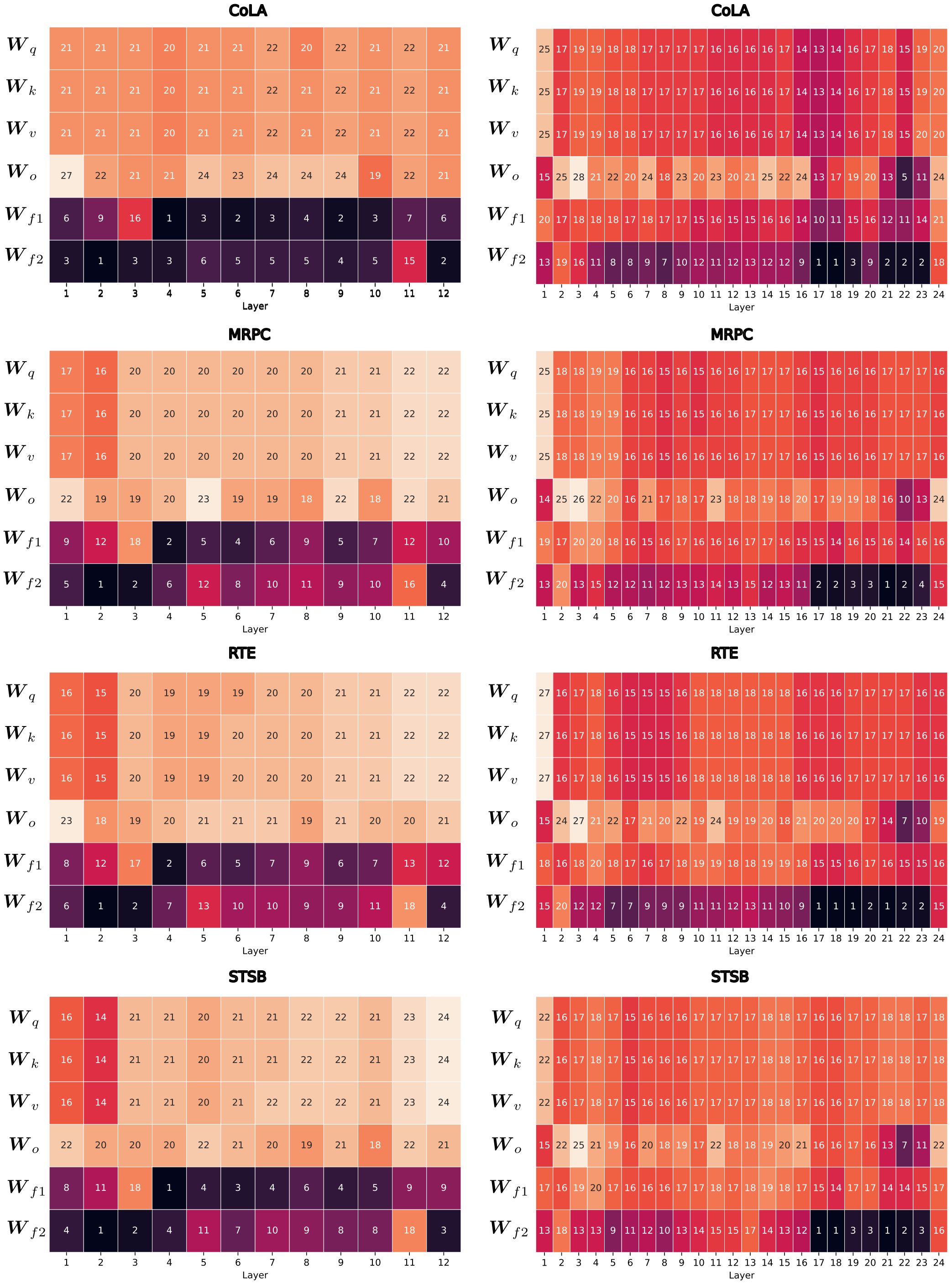


Figure 8: Rank distribution after initialization with EVA on four tasks of the GLUE benchmark (CoLA, MRPC, RTE, STSB) for DeBERTav3$_{\text{Base}}$ (left) and RoBERTa$_{\text{Large}}$ (right) with initial rank $r = 16$.

D IMAGE CLASSIFICATION

D.1 DATASET STATISTICS

The VTAB-1K benchmark consists of 19 datasets, each containing a subset of 1000 examples of their respective samples. We summarize the dataset statistics for each dataset in Table 19. While the original train sizes of the datasets vary drastically, the 1K subset provides equal datasets across tasks. The number of classes also varies from as little as two to almost 400.

Table 19: Category, train size and classes of the VTAB-1K dataset.

Category	**Dataset**	**Train size**	**Classes**
Natural	Caltech101 (Fei-Fei et al., 2006)	3060	102
Natural	CIFAR-100 (Krizhevsky, 2009)	50000	100
Natural	DTD (Cimpoi et al., 2014)	3760	47
Natural	Flowers102 (Nilsback & Zisserman, 2008)	2040	102
Natural	Pets (Parkhi et al., 2012)	3680	37
Natural	Sun397 (Xiao et al., 2010)	87003	397
Natural	SVHN (Netzer et al., 2011)	73257	10
Specialized	EuroSAT (Helber et al., 2019)	21600	10
Specialized	Resisc45 (Cheng et al., 2017)	25200	45
Specialized	Patch Camelyon (Veeling et al., 2018)	294912	2
Specialized	Retinopathy (Kaggle & EyePacs, 2015)	46032	5
Structured	Clevr/count (Johnson et al., 2017)	70000	8
Structured	Clevr/distance (Johnson et al., 2017)	70000	6
Structured	dSprites/location (Matthey et al., 2017)	663552	16
Structured	dSprites/orientation (Matthey et al., 2017)	663552	16
Structured	SmallNORB/azimuth (LeCun et al., 2004)	36450	18
Structured	SmallNORB/elevation (LeCun et al., 2004)	36450	9
Structured	DMLab (Beattie et al., 2016)	88178	6
Structured	KITTI/distance (Geiger et al., 2013)	5711	4

D.2 IMPLEMENTATION DETAILS

We implemented a custom pipeline to fine-tune DINOv2-L/14 on VTAB-1K that supports LoRA, DoRA and EVA. To train AdaLora, PiSSA and OLoRA, we integrate their implementation from the `peft` library (Mangrulkar et al., 2022) into our pipeline. This pipeline is designed to be highly parallelizable and to be executed on individual GPUs. A single evaluation run of a L/14 model (all 19 datasets with hyperparameter tuning and evaluation) takes roughly 160 A100 GPU-hours but can be easily parallelized. A g/14 run takes roughly 140 H100 GPU-hours. A single evaluation run consists of 1140 hyperparameter tuning runs (19 datasets * 5 learning rates * 4 ranks * 3 seeds) and 95 evaluation runs (19 datasets * 5 seeds). Details to hyperparameter tuning are described below.

We use the original DINOv2 models (Oquab et al., 2023) and train a classification head on top of the [CLS] token, where we initialize the classification head weights with a normal distribution with $\sigma = $ 2e-5 and bias with zeros. We train the classification head, LoRA matrices and biases. Images are resized to 224×224 resolution with bi-cubic interpolation and normalized with the per-channel mean and variance of ImageNet. We train all models in bfloat16 precision using the AdamW optimizer with a weight decay of 0.05 for 30 epochs. We use a cosine learning rate schedule with a linear warm-up for the first 3 epochs. Batch size is set to 64 where we use gradient accumulation if the batchsize does not fit into GPU memory. Full fine-tuning uses a layer-wise lr decay (Clark et al., 2020) of 0.75.

D.3 HYPERPARAMETER SEARCH

We first fine-tune on the 800 train samples of VTAB-1K datasets to find the best learning rate for the task. We sweep over `learning_rate` $\in \{2.5\text{e-}3, 1\text{e-}3, 7.5\text{e-}4, 5\text{e-}4, 2.5\text{e-}4\}$ and `rank` $\in \{2, 4, 8, 16\}$ and average the accuracy on the 200 validation samples over 3 different seeds to choose

Table 20: Standard deviations for the VTAB-1K results (Table 5) over 5 seeds.

	Natural							Specialized				Structured								
	Cifar100	Caltech101	DTD	Flower102	Pets	SVHN	Sun397	Camelyon	EuroSAT	Resisc45	Retinopathy	Clevr-Count	Clevr-Dist	DMLab	KITTI-Dist	dSpr-Loc	dSpr-Ori	sNORB-Azim	sNORB-Ele	Average
FFT	1.5	1.1	1.6	0.0	0.4	1.2	0.9	14.9	0.4	0.6	2.7	1.7	0.9	1.2	23.6	0.5	0.4	1.6	1.9	3.0
LoRA	0.2	0.4	0.2	0.0	0.3	36.4	**0.1**	0.5	0.3	0.1	0.4	**0.2**	0.3	0.5	1.2	0.4	0.4	0.7	0.4	2.3
AdaLoRA	**0.0**	**0.2**	0.4	0.0	0.1	0.4	**0.1**	0.3	0.3	0.2	0.3	0.3	0.2	0.3	0.8	0.8	0.3	0.3	0.4	**0.3**
PiSSA	0.2	0.4	0.3	0.0	0.2	0.5	0.2	0.7	0.2	**0.1**	0.4	0.3	0.4	**0.2**	0.7	**0.3**	0.5	0.4	0.5	0.3
OLoRA	0.3	0.3	0.4	0.0	0.3	29.4	0.1	0.3	**0.1**	0.2	**0.2**	0.5	**0.1**	0.3	24.6	0.3	0.4	0.3	0.8	3.1
EVA	0.2	0.5	**0.2**	0.0	**0.1**	**0.3**	**0.1**	**0.3**	0.2	0.3	0.4	0.5	0.3	0.6	0.6	0.5	0.5	**0.2**	0.5	0.3
DoRA	0.1	0.2	0.5	0.0	0.2	29.7	0.4	0.7	0.1	0.2	0.4	0.4	0.3	0.3	**0.6**	36.2	0.5	0.3	**0.3**	3.8
EVA+DoRA	0.2	1.3	0.6	0.0	0.3	0.5	0.3	0.4	0.2	0.3	0.3	0.4	0.4	12.8	1.3	2.5	**0.3**	0.6	0.6	1.2

the best learning rate and rank for each dataset. For evaluation, we train on the union of train and validation set using 5 different seeds and report the average accuracy on the test set.

D.4 ADDITIONAL RESULTS

To complement our main results in Table 5, we report the respective standard deviations in Table 20.

E DECISION MAKING

E.1 DATASET STATISTICS

Meta-World (Yu et al., 2020) is an established benchmark in RL for multi-task continuous control. The benchmark consists of 50 challenging robotics tasks simulated using a Sawyer robotic arm in the MuJoCo physics engine (Todorov et al., 2012). All 50 tasks in Meta-World share the same underlying robotic arm. Therefore, all tasks share a common state (39-dimensional continuous vector) and action-space (6-dimensional). The reward functions in Meta-World are dense and based on the distance of the robotic arm to the goal location or objects. All episodes last for 200 environment interactions.

For our experiments on Meta-World, we leverage the datasets released by Schmied et al. (2024). We follow Wołczyk et al. (2021) and Schmied et al. (2024), and split the 50 tasks into 40 pre-training tasks (MT40) and 10 fine-tuning tasks (CW10). The CW10 tasks are:

`hammer-v2`, `push-wall-v2`, `faucet-close-v2`, `push-back-v2`, `stick-pull-v2`, `stick-pull-v2`, `handle-press-side-v2`, `push-v2`, `shelf-place-v2`, `window-close-v2`, and `peg-unplug-side-v2`.

The datasets contain 2M transitions for every of the 50 tasks, amounting to 80M transitions (320M tokens) across all training tasks. The average success rate and rewards across all MT40 tasks are 84% and 1414.62, respectively. We list the statistics per task in Table 21.

E.2 IMPLEMENTATION DETAILS

We implemented our pipeline that supports training for Meta-World on top of the code-base provided by Schmied et al. (2024). Our custom implementation supports training LoRA, DoRA and EVA. Furthermore, we leverage the `peft` library (Mangrulkar et al., 2022) to train the remaining methods.

Table 21: Dataset statistics for all MT40 tasks from Schmied et al. (2024).

Task	$\|\mathcal{S}\|$	$\|\mathcal{A}\|$	**Success Rate**	**Reward**
`assembly-v2`	39	4	0.0	1206.9
`basketball-v2`	39	4	0.9	1375.95
`bin-picking-v2`	39	4	0.0	474.81
`box-close-v2`	39	4	0.0	759.15
`button-press-topdown-v2`	39	4	1.0	1299.24
`button-press-topdown-wall-v2`	39	4	1.0	1296.16
`button-press-v2`	39	4	1.0	1430.44
`button-press-wall-v2`	39	4	1.0	1508.16
`coffee-button-v2`	39	4	1.0	1499.17
`coffee-pull-v2`	39	4	1.0	1313.88
`coffee-push-v2`	39	4	0.6	508.14
`dial-turn-v2`	39	4	0.8	1674.29
`disassemble-v2`	39	4	1.0	1396.55
`door-close-v2`	39	4	1.0	1535.4
`door-lock-v2`	39	4	1.0	1712.65
`door-open-v2`	39	4	1.0	1544.32
`door-unlock-v2`	39	4	1.0	1733.64
`drawer-close-v2`	39	4	1.0	1845.92
`drawer-open-v2`	39	4	1.0	1710.65
`faucet-open-v2`	39	4	0.9	1727.98
`hand-insert-v2`	39	4	1.0	1607.17
`handle-press-v2`	39	4	1.0	1854.79
`handle-pull-side-v2`	39	4	1.0	1613.72
`handle-pull-v2`	39	4	1.0	1581.75
`lever-pull-v2`	39	4	1.0	1449.05
`peg-insert-side-v2`	39	4	1.0	1545.19
`pick-out-of-hole-v2`	39	4	1.0	1435.64
`pick-place-v2`	39	4	0.0	6.59
`pick-place-wall-v2`	39	4	0.1	702.59
`plate-slide-back-side-v2`	39	4	1.0	1766.24
`plate-slide-back-v2`	39	4	1.0	1773.56
`plate-slide-side-v2`	39	4	1.0	1663.35
`plate-slide-v2`	39	4	1.0	1667.35
`reach-v2`	39	4	1.0	1858.99
`reach-wall-v2`	39	4	1.0	1831.14
`soccer-v2`	39	4	0.4	445.84
`stick-push-v2`	39	4	1.0	1470.71
`sweep-into-v2`	39	4	1.0	1761.69
`sweep-v2`	39	4	1.0	1458.35
`window-open-v2`	39	4	1.0	1537.59
Average	-	-	0.84 ± 0.34	1414.62 ± 439.39

For our experiments on Meta-World, we use a GPT2-like network architecture (Radford et al., 2019) with 4 Transformer layers, 8 heads, and hidden dimension of 512 resulting in 16M parameters. We use a context of 50 time steps, which amounts to a sequence length of 200, as each timestep contains states, actions, rewards and RTGs. We embed states, actions, rewards and return-to-gos (RTGs) using separate linear embedding layers per modality, as proposed by Chen et al. (2021a). We train with a batch size of 128 using a constant learning rate of $1e^{-4}$, 4000 linear warm-up steps followed by a cosine decay to $1e^{-6}$, using the AdamW optimizer (Loshchilov & Hutter, 2017). We employ gradient clipping of 0.25, weight decay of 0.01, and a dropout rate of 0.2. Our DT implementation employs global position embedding. For every task, we set the target return to the maximum return achieved in the respective training datasets, as proposed by (Schmied et al., 2024). Furthermore, we employ mixed-precision (Micikevicius et al., 2017) and flash-attention (Dao, 2023) to speed-up training.

We first **pre-train** a DT on all MT40 tasks (80M transitions) for 1M updates via next-action prediction by minimizing the mean-squared error. The resulting pre-trained model attains an average success rate of 80% across all MT40 tasks. Then we **fine-tune** the DT on each of the CW10 down-stream tasks for 100K updates with the same set of hyperparameters as used for pre-training. We run all our experiments on a public research cluster with 4xA100-40GB GPU nodes. A single fine-tuning run with EVA for one task takes roughly 1 hour on one A100.

E.3 HYPERPARAMETER SEARCH

In line with previous experiments, we tune the rank for LoRA, DoRA, AdaLora and EVA, $\texttt{rank} \in \{2, 4, 8, 16\}$. Further, we sweep over the same learning rates as for the GLUE tasks.

E.4 ADDITIONAL RESULTS

In Table 22, we show the full comparison for all methods on CW10. EVA+DoRA consistently outperforms all competitors for the different rank budgets.

Table 22: Rank-wise comparison for all methods on CW10. We fine-tune a 12M DT on 10 tasks individually and report the mean success rates/rewards (± standard error) for every task.

Method	Rank	faucet-close	hammer	handle-press-side	peg-unplug-side	push-back	push	push-wall	shelf-place	stick-pull	window-close	Average
FFT	-	$0.97_{\pm0.03}$	$0.93_{\pm0.03}$	$1.0_{\pm0.0}$	$0.6_{\pm0.05}$	$0.7_{\pm0.12}$	$1.0_{\pm0.0}$	$0.93_{\pm0.03}$	$1.0_{\pm0.0}$	$0.57_{\pm0.07}$	$1.0_{\pm0.0}$	$0.87_{\pm0.03}$
LoRA	2	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.6_{\pm0.05}$	$0.57_{\pm0.07}$	$0.97_{\pm0.03}$	$0.93_{\pm0.03}$	$1.0_{\pm0.0}$	$0.37_{\pm0.1}$	$1.\pm0.0$	$0.84_{\pm0.04}$
	4	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.47_{\pm0.12}$	$0.63_{\pm0.1}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.23_{\pm0.12}$	$1.0_{\pm0.0}$	$0.83_{\pm0.05}$
	8	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.43_{\pm0.05}$	$0.4_{\pm0.09}$	$0.97_{\pm0.03}$	$0.93_{\pm0.03}$	$1.0_{\pm0.0}$	$0.23_{\pm0.12}$	$1.0_{\pm0.0}$	$0.79_{\pm0.06}$
	16	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.43_{\pm0.03}$	$0.47_{\pm0.03}$	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.4_{\pm0.09}$	$1.0_{\pm0.0}$	$0.82_{\pm0.05}$
DoRA	2	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.57_{\pm0.05}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.33_{\pm0.11}$	$1.0_{\pm0.0}$	$0.89_{\pm0.04}$
	4	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.6_{\pm0.12}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.43_{\pm0.12}$	$1.0_{\pm0.0}$	$0.9_{\pm0.04}$
	8	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.47_{\pm0.12}$	$0.93_{\pm0.05}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.57_{\pm0.15}$	$1.0_{\pm0.0}$	$0.9_{\pm0.04}$
	16	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.57_{\pm0.12}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.67_{\pm0.15}$	$1.0_{\pm0.0}$	$0.92_{\pm0.03}$
AdaLoRA	2	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.37_{\pm0.05}$	$0.37_{\pm0.05}$	$0.93_{\pm0.05}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.13_{\pm0.07}$	$1.0_{\pm0.0}$	$0.77_{\pm0.06}$
	4	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.37_{\pm0.07}$	$0.57_{\pm0.1}$	$0.97_{\pm0.03}$	$0.9_{\pm0.08}$	$1.0_{\pm0.0}$	$0.13_{\pm0.07}$	$1.0_{\pm0.0}$	$0.79_{\pm0.06}$
	8	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.3_{\pm0.05}$	$0.57_{\pm0.14}$	$0.93_{\pm0.03}$	$0.87_{\pm0.07}$	$1.0_{\pm0.0}$	$0.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.76_{\pm0.06}$
	16	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.4_{\pm0.09}$	$0.57_{\pm0.12}$	$0.97_{\pm0.03}$	$0.93_{\pm0.05}$	$1.0_{\pm0.0}$	$0.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.78_{\pm0.06}$
OLoRA	2	$1.0_{\pm0.0}$	$0.9_{\pm0.05}$	$1.0_{\pm0.0}$	$0.47_{\pm0.03}$	$0.33_{\pm0.03}$	$0.97_{\pm0.03}$	$0.97_{0.03}$	$1.0_{\pm0.0}$	$0.27_{\pm0.11}$	$1.0_{\pm0.0}$	$0.79_{\pm0.05}$
	4	$1.0_{\pm0.0}$	$0.9_{\pm0.05}$	$1.0_{\pm0.0}$	$0.43_{\pm0.03}$	$0.63_{\pm0.12}$	$1.0_{\pm0.0}$	$1.0_{0.0}$	$1.0_{\pm0.0}$	$0.6_{\pm0.12}$	$1.0_{\pm0.0}$	$0.86_{\pm0.04}$
	8	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.57_{\pm0.1}$	$0.5_{\pm0.08}$	$1.0_{\pm0.0}$	$1.0_{0.0}$	$1.0_{\pm0.0}$	$0.53_{\pm0.14}$	$1.0_{\pm0.0}$	$0.86_{\pm0.04}$
	16	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.4_{\pm0.05}$	$0.63_{\pm0.03}$	$1.0_{\pm0.0}$	$1.0_{0.0}$	$1.0_{\pm0.0}$	$0.43_{\pm0.05}$	$1.0_{\pm0.0}$	$0.84_{\pm0.04}$
PiSSA	2	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.43_{\pm0.11}$	$0.53_{\pm0.07}$	$0.97_{\pm0.03}$	$0.9_{0.08}$	$1.0_{\pm0.0}$	$0.33_{\pm0.17}$	$1.0_{\pm0.0}$	$0.81_{\pm0.05}$
	4	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.37_{\pm0.07}$	$0.7_{\pm0.05}$	$0.97_{\pm0.03}$	$1.0_{0.0}$	$1.0_{\pm0.0}$	$0.07_{\pm0.05}$	$1.0_{\pm0.0}$	$0.81_{\pm0.06}$
	8	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.3_{\pm0.0}$	$0.57_{\pm0.03}$	$0.97_{\pm0.03}$	$1.0_{0.0}$	$1.0_{\pm0.0}$	$0.53_{\pm0.1}$	$1.0_{\pm0.0}$	$0.83_{\pm0.05}$
	16	$1.0_{\pm0.0}$	$0.93_{\pm0.03}$	$1.0_{\pm0.0}$	$0.33_{\pm0.12}$	$0.47_{\pm0.03}$	$1.0_{\pm0.0}$	$0.97_{0.03}$	$1.0_{\pm0.0}$	$0.47_{\pm0.11}$	$1.0_{\pm0.0}$	$0.82_{\pm0.05}$
EVA	2	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.43_{\pm0.07}$	$0.77_{\pm0.05}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.63_{\pm0.07}$	$1.0_{\pm0.0}$	$0.88_{\pm0.04}$
	4	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.43_{\pm0.05}$	$0.47_{\pm0.12}$	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.23_{\pm0.05}$	$1.0_{\pm0.0}$	$0.81_{\pm0.05}$
	8	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.63_{\pm0.03}$	$0.7_{\pm0.08}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.23_{\pm0.03}$	$1.0_{\pm0.0}$	$0.85_{\pm0.05}$
	16	$1.0_{\pm0.0}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$0.53_{\pm0.03}$	$0.77_{\pm0.07}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.83_{\pm0.06}$
EVA + DoRA	2	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.8_{\pm0.08}$	$0.97_{\pm0.03}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.43_{\pm0.12}$	$1.0_{\pm0.0}$	$0.92_{\pm0.03}$
	4	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.8_{\pm0.05}$	$0.93_{\pm0.03}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.63_{\pm0.03}$	$1.0_{\pm0.0}$	$0.94_{\pm0.02}$
	8	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.63_{\pm0.19}$	$0.87_{\pm0.07}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.57_{\pm0.03}$	$1.0_{\pm0.0}$	$0.91_{\pm0.04}$
	16	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.67_{\pm0.2}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$1.0_{\pm0.0}$	$0.5_{\pm0.16}$	$1.0_{\pm0.0}$	$0.92_{\pm0.04}$

F INCREMENTAL SVD CONVERGENCE ANALYSIS

For completeness we first present the vanilla incremental SVD algorithm introduced by Ross et al. (2008) that we leverage in this work in Algorithm 2. As this algorithm incrementally updates the $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ components, we need to keep track of changing mean and variance estimates. For the mean this is trivial, but the computation of running variances can introduce numerical instabilities. To counteract this, usually the *young and cramer update* is employed (Chan et al., 1983).

Algorithm 2 Incremental SVD algorithm from Ross et al. (2008)

```
Input: Sequence of data batches $\{\boldsymbol{A}^0, \dots, \boldsymbol{A}^T\}$, truncated SVD $\mathrm{SVD}(\cdot)$, orthogonalization function
    $\mathrm{orth}(\cdot)$, running variance update function $\mathrm{young\_cramer\_update}(\cdot,\cdot)$
$\bar{\boldsymbol{m}}^0 \leftarrow \frac{1}{b}\sum_{i=0}^{b} \boldsymbol{A}_{:,i}$, $\boldsymbol{\sigma}^0 \leftarrow \frac{\sum_{i=0}^{b}(\boldsymbol{A}_{:,i} - \bar{\boldsymbol{m}}^0)^2}{b-1}$    ▷ initialize incremental mean/variance
$\boldsymbol{U}_0\boldsymbol{\Sigma_0}\boldsymbol{V}^\top \leftarrow \mathrm{SVD}(\boldsymbol{A}^0 - \bar{\boldsymbol{a}}^0)$    ▷ Perform initial SVD on $\boldsymbol{A}$ to get initial components
for $i$ in $1, \dots, T$ do
    $\bar{\boldsymbol{a}}^i \leftarrow \frac{1}{b}\sum_b \boldsymbol{A}^i_{:,i}$, $\bar{\boldsymbol{m}}^i \leftarrow \bar{\boldsymbol{m}}^i + \frac{\boldsymbol{a}^i - \bar{\boldsymbol{m}}^{i-1}}{b(i+1)}$    ▷ compute mean vectors
    $\boldsymbol{\sigma}^i \leftarrow \mathrm{young\_cramer\_update}(\boldsymbol{\sigma}^{i-1}, \boldsymbol{A}^i)$    ▷ Update running variance
    $\hat{\boldsymbol{A}}^i \leftarrow \left[\boldsymbol{A}^i - \bar{\boldsymbol{a}}^i; \sqrt{\frac{b(i+1)}{2b}}\left(\bar{\boldsymbol{m}}^i - \bar{\boldsymbol{a}}^i\right)\right]$    ▷ concatenate mean correction factor
    $\tilde{\boldsymbol{A}}^i \leftarrow \mathrm{orth}(\hat{\boldsymbol{A}}^i - \boldsymbol{U}_{i-1}\boldsymbol{U}_{i-1}^\top\hat{\boldsymbol{A}}^i)$    ▷ Obtain orthogonal component to $\boldsymbol{U}$
    $\boldsymbol{R} = \begin{bmatrix} \boldsymbol{\Sigma_{i-1}} & \boldsymbol{U}_{i-1}{\top}\hat{\boldsymbol{A}}^i \\ \boldsymbol{0} & \tilde{\boldsymbol{A}}^i\hat{\boldsymbol{A}}^i \end{bmatrix}$    ▷ Define matrix $\boldsymbol{R}$
    $\tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Sigma}}\tilde{\boldsymbol{V}}^\top \leftarrow \mathrm{SVD}(\boldsymbol{R})$    ▷ Perform SVD on $\boldsymbol{R}$
    $\boldsymbol{U}_i \leftarrow \left[\boldsymbol{U}_{i-1}; \tilde{\boldsymbol{A}}^i\right]\tilde{\boldsymbol{U}}$, $\boldsymbol{\Sigma}_i \leftarrow \tilde{\boldsymbol{\Sigma}}$    ▷ Update SVD components
end for
```

In the following sections we analyze the behavior of this algorithm under different conditions, i.e. different batch sizes, etc.

F.1 COMPLEXITY

The computation of SVD introduces computational overhead in the initial training stage. Since we do not require gradient computation or storing of optimizer states, there is no overhead in terms of memory. SVD has a time complexity of $\mathcal{O}(\min(b^2 d, bd^2))$ which can be reduced to $\mathcal{O}(k^2 b)$ for $k << d$ by randomly choosing k columns from $\boldsymbol{X}$ as introduced in Halko et al. (2011). Let T be the number of minibatches until all components are converged for N weight matrices, then the time complexity is $\mathcal{O}(NTk^2 b)$. In other words, the complexity scales linearly with the number of weight matrices and the number of minibatches. To speed up the computation of SVD, we provide an implementation that runs entirely on GPU.

F.2 BATCH SIZE INVARIANCE

We conduct an analysis on the convergence of the components obtained via SVD. Specifically, we investigate the difference in components according to cosine similarity across different batch sizes. Previously we have seen that the components obtained across different batch orderings are heavily correlated. In Figure 9 we visualize the cosine similarities between the SVD components for different batch sizes, namely 4, 8, 16, and 32 for Llama-2-7B on the MetaMathQA dataset. We observe that the components correlate strongly and remain mostly invariant to the batch size. This indicates that smaller batch sizes may be used for obtaining the initialization which results in less computational overhead. In the case of Llama-2-7B on MetaMathQA, this means that we can use a batch size of 4 since it induces a computational overhead of around 100 seconds. Afterwards we can continue the fine-tuning process with a larger batch size.

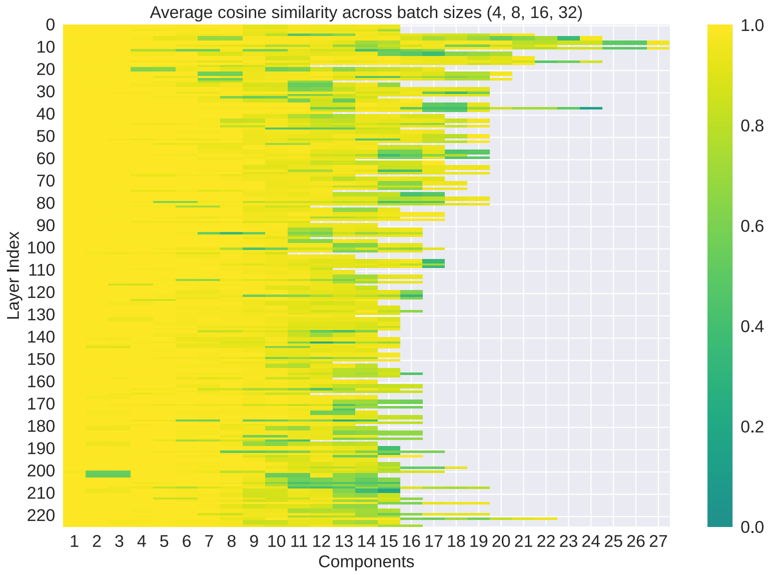


Figure 9: Average cosine similarity between components obtained via SVD on minibatches of activation vectors across different batch sizes. The components strongly correlate indicating that the SVD computation is mostly invariant to the batch size and returns mostly the same components.

F.3 EXCLUDING IGNORED TOKENS FOR SVD

For some datasets we notice that masking out tokens for the SVD computation which are ignored for the loss calculation during finetuning can be advantageous. This can however result in a significant reduction of the effective batch size for SVD if the number of completion tokens is small. An example where this is the case in our experiments are the common sense reasoning tasks which have long prompts but completion tokens are only one word per sample. This setting can lead to cases were SVD does not converge for lower batch sizes. We therefore do not mask out the prompt tokens in our experiments. Another setting where masking ignored tokens can be advantageous are multi-turn conversation where the model is only trained on the assistant tokens. To achieve the results in Table 13 we mask out user tokens together with the prompt for the SVD computation.

F.4 EFFICIENCY OF EVA INITIALIZATION

We investigate the efficacy of the incremental SVD for obtaining a data-driven initialization to LoRA-GA (Wang et al., 2024), another concurrent work on data-driven initialization. LoRA-GA performs SVD on the full gradient matrix to obtain a lower dimensional subspace approximation and initializes $\boldsymbol{A}$ and $\boldsymbol{B}$ accordingly. In Table 23 we show the wall clock time required for LoRA-GA and EVA as a fraction of the total training time. We observe that EVA takes up only 0.7% of the training time for initialization, while LoRA-GA takes approximately 4.8%. This demonstrates the EVA is approximately seven times faster than LoRA-GA while achieving better performance. Furthermore, EVA is even faster than PiSSA even though PiSSA is weight-driven. Finally, even though EVA is slightly slower than OLoRA, it attains a better performance vs complexity trade-off as it outperforms OLoRA on average on all our experiments.

G RANK RE-DISTRIBUTION ANALYSIS

To illuminate the rank re-distribution process, we visualize the resulting ranks for each weight matrix after SVD for Llama-2-7B on the MetaMathQA dataset for different values of ρ. Setting $\rho = 1$ results in a uniform rank distribution as in standard LoRA. However, setting $\rho > 1$ alters the number of ranks per weight matrix. In Figure 10 we visualize the number of ranks assigned to each weight matrix for different values of $\rho > 1$ and in Figure 11 we visualize the corresponding deltas. Both

Table 23: Time in minutes required for computing initialization of LoRA-GA, PiSSA and EVA as % of total training time for Llama-2-7B on a single A100 GPU fine-tuned on the common sense reasoning tasks presented in Table 2. Training time is averaged across two runs for one epoch. For LoRA-GA we use the default number of steps (64). For EVA we report efficiency across different batch sizes.

Initialization	**Method**	**Initialization**	**Training**	**% of Training**
Weight-driven	PiSSA	7.43	482.67	1.5
	OLoRA	0.3	482.67	0.1
Data-driven	LoRA-GA	11.7	482.67	2.4
	$\text{EVA}_{\text{bs}=16}$	3.3	482.67	0.7
	$\text{EVA}_{\text{bs}=8}$	1.38	482.67	0.3
	$\text{EVA}_{\text{bs}=4}$	1.17	482.67	0.2

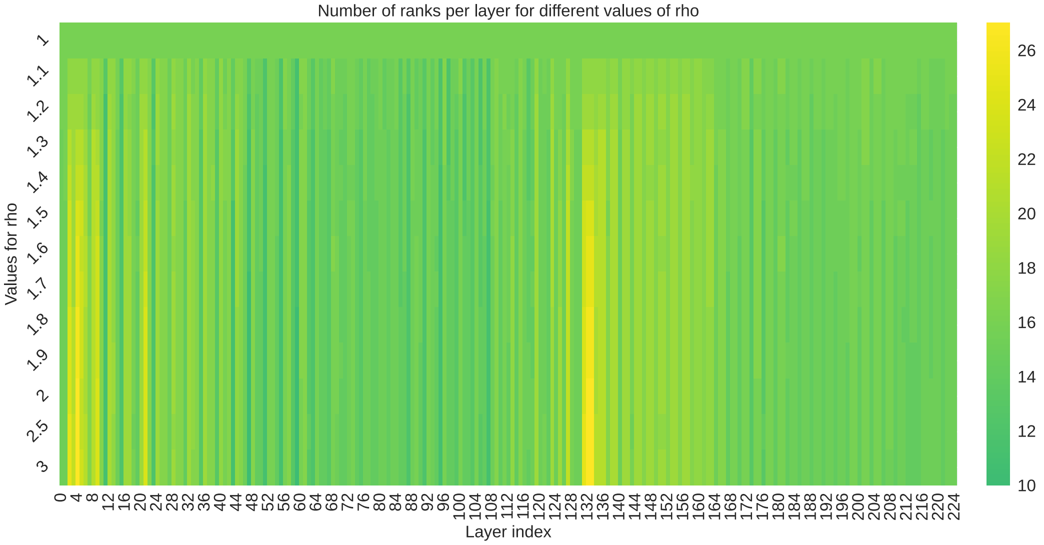


Figure 10: The resulting rank allocation per weight matrix in each layer for Llama-2-7B on the MetaMathQA dataset with different values of ρ. The first row represents a uniform distribution where each weight matrix receives the same rank $r = 16$. The most change occurs for $\rho < 1.5$. The re-distribution converges for larger values of ρ.

visualizations clearly illustrate that the most change occurs for values of $\rho < 1.5$. Setting ρ to higher values results in less and less change. Interestingly, some ranks still change when going from $\rho = 2.5$ to $\rho = 3$. Finally, we conduct hyperparameter search in which we search over different values of $\rho \in \{1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2, 2.5, 3\}$. We report the results in Figure 12. We find that for Llama-2-7B on MetaMathQA a uniform distribution performs favorably. The second-best performance is shared by $\rho = 1.5$ and $\rho = 2$. Therefore, we always search for $\rho = 1$ and $\rho = 2$ for all our remaining experiments when we apply EVA and select the best performing one.

H RELATION BETWEEN SVD AND PCA

PCA (F.R.S., 1901) is a commonly used tool to decompose a matrix of datasamples $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ into its principal components, i.e. the directions that explain the most variance in the data. The principal components allow projection onto a lower dimensional manifold by preserving the maximal amount

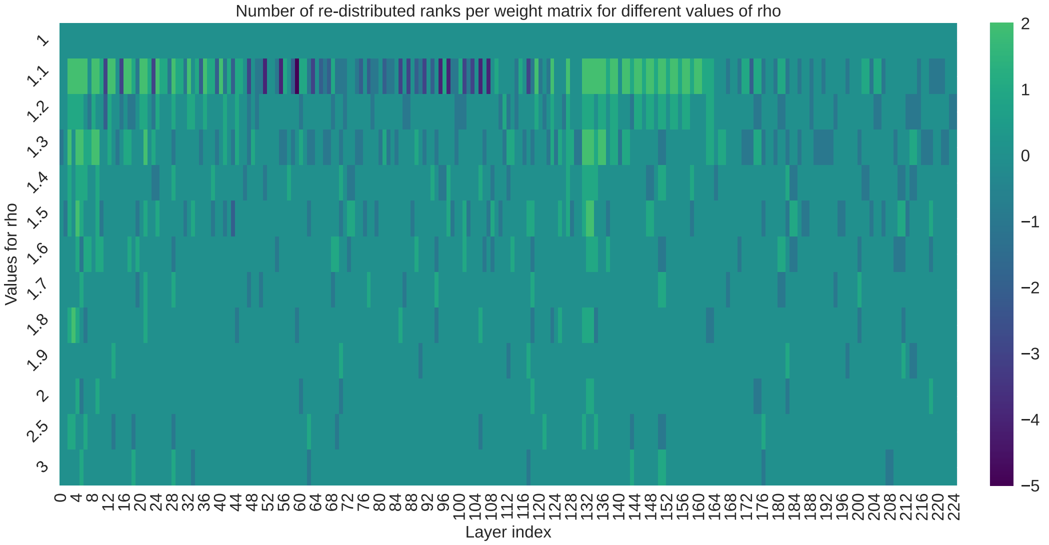


Figure 11: Deltas between rank distributions per weight matrix in each layer for Llama-2-7B on the MetaMathQA dataset with different values of ρ. The first row represents a uniform distribution where each weight matrix receives the same rank $r = 16$. The most change occurs in the range $\rho \in [1, 1.5]$. Larger values of ρ do not induce additional significant changes to the rank distribution.

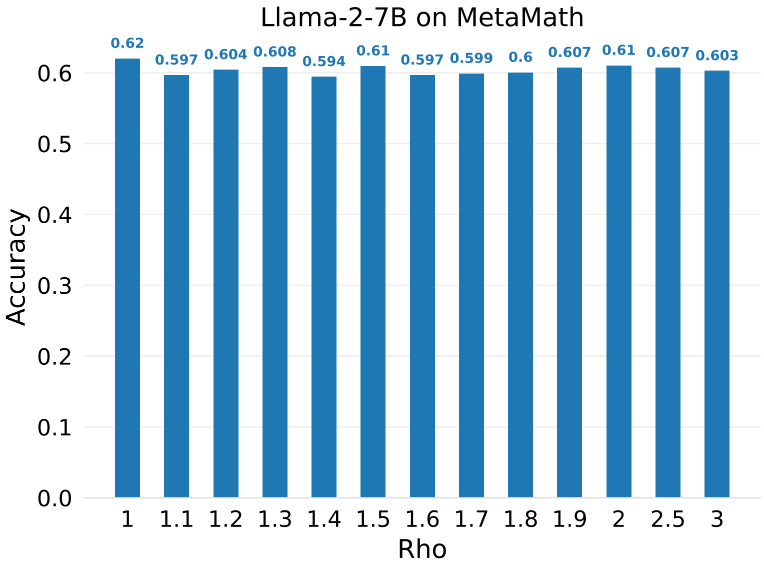


Figure 12: Accuracy for different values of ρ when fine-tuning Llama-2-7B on the MetaMathQA dataset.

of variance. To this end, PCA first computes the sample covariance matrix

$$\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{A}^\top \boldsymbol{A}, \tag{5}$$

where we assume that $\boldsymbol{A}$ is centered. To obtain the *principal directions* of $\boldsymbol{S}$, we perform eigenvalue decomposition as

$$\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top, \tag{6}$$

where $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$ and eigenvalues are sorted in descending order, i.e. $\lambda_1 \geq \lambda_2 \geq \lambda_n$. The matrix $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ is a matrix of eigenvectors where each column is being referred to as a *principal direction* of $\boldsymbol{S}$. To project $\boldsymbol{A}$ onto a lower dimensional manifold that explains the most variance we can take the top-k principal directions $\boldsymbol{V}_{:,:k}$ and perform $\boldsymbol{AV}$.

PCA is in practice often implemented in the form of SVD as there are efficient approximations thereof (Halko et al., 2011). As mentioned in Equation (1), SVD decomposes the matrix $\boldsymbol{A}$ into

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{7}$$

where $\boldsymbol{U} \in \mathbb{R}^{m \times n}$ is a unitary matrix, $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ is a diagonal matrix of singular values $\boldsymbol{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_n)$, and the columns of $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ are called the right singular vectors.

Now we can establish the equivalence between the principal directions obtained by PCA and the right-singular vectors of SVD by substituting $\boldsymbol{A}$ with the right hand side of Equation (7) as

$$\boldsymbol{S} = \frac{1}{n-1}\boldsymbol{A}^\top \boldsymbol{A} = \frac{1}{n-1}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{V}\hat{\boldsymbol{\Sigma}}\boldsymbol{V}^\top. \tag{8}$$

Here, we absorb the factor $\frac{1}{n-1}$ into $\hat{\boldsymbol{\Sigma}}$. Therefore, the right-singular vectors $\boldsymbol{V}$ are the principal directions and $\boldsymbol{\Sigma}\boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{\Sigma} = \boldsymbol{\Sigma}$ as $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}$ because $\boldsymbol{U}$ is real.

I ABLATION STUDIES

Finally, we conduct ablation studies on EVA to investigate important factors that contribute to its performance. Specifically, we investigate the impact of scale and directions. To this end, we use the VTAB-1K dataset because it comprises a diverse set of tasks and allows for a systematic investigation on in-domain data (natural), and out-of-distribution data (specialized and structured). We report results for our ablation studies in Table 24 and explain the different settings in the following paragraphs.

Effect of scale. To investigate the effect of scale on the initialization, we add a setting which uses whitening (EVA-whiten). Whitening scales the initialization by the reciprocal of their eigenvalues, which alters scale, but preserves directions. We found that whitening can significantly improve performance on structured (out-of-distribution) tasks even leading to a slightly higher average score than EVA. This indicates that scale is especially important for structured data. However, EVA-whiten experiences a slight performance drop on natural and specialized tasks.

Effect of directions. To address the importance of the directions of the components, we randomly permute its rows (EVA-perm). This preserves scale while corrupting directions and ℓ_2 norm of $\boldsymbol{A}$. Additionally, we add a setting where we randomly rotate $\boldsymbol{A}$ (EVA-rot), which preserves ℓ_2 norm, but alters directions. We find that altering directions leads to a performance drop on the structured tasks, while changing ℓ_2 norm leads to a drop on the natural tasks. Both, EVA-perm and EVA-rot lead to worse average performance across all tasks compared to EVA.

Table 24: Group-wise averages for DINOv2-G/14 ablation studies on the VTAB-1K benchmark.

Method	Nat.	Spec.	Struct.	All
LoRA	83.2	**88.8**	69.0	78.4
LoRA-redist	87.3	88.0	68.2	79.4
EVA-whiten	87.5	87.5	**69.1**	**79.8**
EVA-rot	**87.7**	88.0	68.2	79.6
EVA-perm	87.4	87.8	68.3	79.5
EVA	**87.7**	87.9	68.6	79.7

Effect of rank redistribution. We conduct an experiment in which we randomly initialize $\boldsymbol{A}$ after performing rank redistribution (LoRA-redist). This setting gives insights on the effect of the redistribution and whether its benefits are bound to EVA. The redistribution has a positive effect on LoRA on the natural tasks, but a negative effect on both structured and specialized tasks. This illustrates that rank redistribution is most beneficial in combination with EVA's initialization of $\boldsymbol{A}$.

Generally, we can say that EVA performs particularly well on natural images and whitening can enhance its performance on out-of-distribution images. The decisive factor with respect to this improvement seems to be a controlled change in the scale of the initialization induced by the singular values. Therefore, by changing the scale in a controlled manner we can make EVA more compatible for different kinds of data. The results for EVA-perm confirm that the scale is the decisive factor for initialization.