One Initialization to Rule them All: Fine-tuning via Explained Variance Adaptation

Fabian Paischer[1*] **Lukas Hauzenberger**[1*] **Thomas Schmied**[1]
Benedikt Alkin[1,3] **Marc Peter Deisenroth**[2] **Sepp Hochreiter**[1,3]
[1] ELLIS Unit, LIT AI Lab, Institute for Machine Learning, JKU Linz, Austria
[2] University College London
[3] NXAI GmbH, Linz, Austria
paischer@ml.jku.at

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 modulate the pre-trained weights via a low-rank adaptation (LoRA) of newly introduced weights. These weight matrices are usually initialized at random with the same rank for each layer across the FM, which results in suboptimal performance. We propose to enhance LoRA by initializing the new weights in a data-driven manner, by computing singular value decomposition on activation vectors. Then, we initialize the new LoRA matrices with the obtained right-singular vectors. Finally, we re-distribute the ranks among layers to explain the maximal amount of variance across all layers. This assignment results in an adaptive allocation of ranks per weight matrix, and inherits all benefits of LoRA. We apply our new method, **E**xplained **V**ariance **A**daptation (EVA), to a variety of fine-tuning tasks comprising language understanding and generation, image classification, and reinforcement learning. EVA consistently attains the highest average score across a multitude of tasks per domain.

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., 2023; 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 a commonly used as an effective alternative to full fine-tuning (FFT). They usually inject a small fraction of new trainable weights into the pre-trained model. During fine-tuning, the pre-trained weights remain frozen and only the new weights are updated. This substantially reduces the computational cost in terms of both, time and space dimensions. 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. While LoRA yields strong performance compared to full fine-tuning, current approaches

[*]Equal contribution

4th NeurIPS Efficient Natural Language and Speech Processing Workshop (ENLSP-IV 2024).

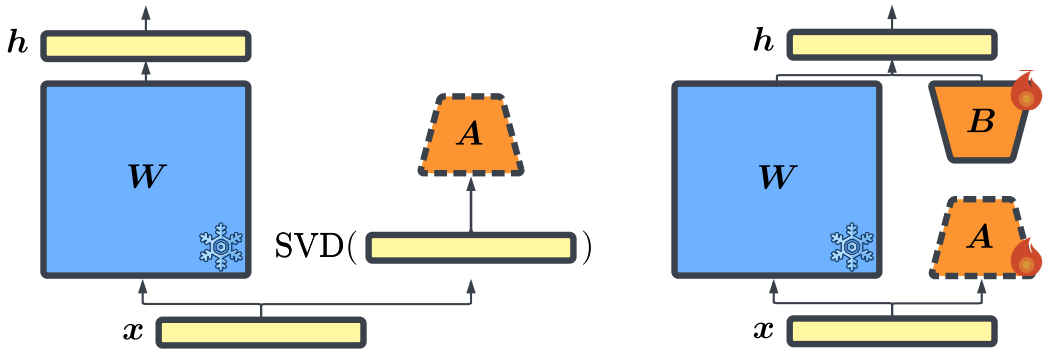

Figure 1: **Left:** EVA performs singular value decomposition on activation vectors for the first few mini-batches to obtain a suitable initialization for the LoRA matrix $\boldsymbol{A}$. **Right:** After initializing $\boldsymbol{A}$, we allocate ranks to maximize the explained variance throughout the model and continue the standard LoRA fine-tuning procedure, where $\boldsymbol{W}$ is kept frozen and only $\boldsymbol{A}$ and $\boldsymbol{B}$ are trained.

either initialize the LoRA weights according to statistics of the pre-trained weights (Meng et al., 2024) or at random (Hu et al., 2022; Zhang et al., 2023a).

We propose a new data-driven initialization of LoRA weights by leveraging information from the downstream task at hand. Certain activation patterns of FMs have been shown to be crucial for model performance (Sun et al., 2024). Therefore, we aim at leveraging activations computed on the downstream data for initialization of LoRA weights. To this end, we propagate a few mini-batches of the fine-tuning data through the model and compute the singular value decomposition (SVD) on activation vectors to obtain right-singular vectors. We leverage this projection to initialize the down-projection in LoRA. Further, We sort all ranks according to their explained variance and only assign those that maximize it for a given rank budget. This results in an effective initialization of LoRA matrices, that (i) is data-driven by leveraging information from the downstream task, and (ii) allocates ranks to pre-trained weights to maximize the explained variance throughout the model. 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 mini-batches during LoRA fine-tuning without significant computational overhead.

We demonstrate the benefits of EVA on an array of downstream tasks, namely language generation, 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 PEFT methods.. On language understanding tasks, EVA exhibits average performance gains compared to LoRA on several tasks of the GLUE benchmark (Wang et al., 2019). 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 again attains higher average scores than competitors, exhibiting most improvements on out-of-distribution 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., 2024). Finally, we conduct ablation studies to demonstrate that the combination of direction and scale provided by EVA leads to the best performance.

2 Method

EVAaims at initializing LoRA weights in a data-driven manner by leveraging data from the downstream task. Since EVAbuilds on low-rank decomposition of weight matrices as in LoRA (Hu et al., 2022), we first briefly explain LoRA in Section 2.1. In Section 2.2, we describe how we obtain an effective initialization for the low-rank decomposition of LoRA matrices via SVD on activation vectors. This enables an adaptive assignment of ranks across all layers to maximize the explained variance throughout the pre-trained model, which we explain in more detail in Section 2.3.

2.1 Low-Rank Adaptation (LoRA)

LoRA adds new trainable weights that 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 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). In the following, we explain LoRA in more detail. Let $\boldsymbol{x} \in \mathbb{R}^{d \times 1}$ be a an activation vector that serves as input to a pre-trained weight matrix $\boldsymbol{W} \in \mathbb{R}^{k \times d}$. LoRA introduces new weight matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ as a low-rank decomposition

$$\boldsymbol{h} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{A}\boldsymbol{x}\,, \tag{1}$$

where $\boldsymbol{B} \in \mathbb{R}^{k \times r}$ and $\boldsymbol{A} \in \mathbb{R}^{r \times d}$. The rank r is a hyperparameter and $r \ll k$. During fine-tuning, $\boldsymbol{W}$ remains frozen and only $\boldsymbol{A}$ and $\boldsymbol{B}$ are updated. Usually $\boldsymbol{B}$ is initialized with zeros, such that the initial forward pass at the beginning of fine-tuning is not altered. $\boldsymbol{A}$ is usually initialized at random following a Gaussian distribution. Additionally, Hu et al. (2022) introduce a constant scaling factor α which is used to scale $\boldsymbol{B}\boldsymbol{A}\boldsymbol{x}$ by $\frac{\alpha}{r}$. In Hu et al. (2022), this scaling factor is set to $\alpha = 2r$.

2.2 Data-driven Initialization of Low-Rank Adaptation

Our aim is to find an effective initialization for the low-rank matrix $\boldsymbol{A}$ in a data-driven manner to maximize performance on the downstream task. To this end, we perform SVD on batches of activation vectors $\boldsymbol{X} \in \mathbb{R}^{b \times d}$ to obtain the right-singular values, which constitute the directions that capture most of the variance. Therefore, we devote the initial training phase to During the initial training phase, we propagate mini-batches of data through the model and incrementally compute SVD on activation vectors. More formally, we collect activations $\boldsymbol{X}$ for each weight matrix $\boldsymbol{W}$ of the pre-trained model. Subsequently, we compute SVD on these vectors to obtain singular values σ^i as

$$\boldsymbol{X} = \sum_{j=1}^{r} \boldsymbol{u}_{:,j}\sigma_j\boldsymbol{v}_{j,:}. \tag{2}$$

Importantly, we compute the SVD incrementally on each mini-batch of the fine-tuning data and update $\boldsymbol{v}_{:r,:}$ after each forward pass through the model. After every mini-batch we check whether $\boldsymbol{v}_{:r,:}$ has converged. To this end, we measure the cosine similarity between subsequent computations of $\boldsymbol{v}_{:r,:}$ and determine convergence based on a threshold τ. If the right-singular values have converged, i.e. $\text{cossim}(\boldsymbol{v}_{j,:}^{t-1}, \boldsymbol{v}_{j,:}^{t}) \geq \tau$, we initialize $\boldsymbol{A}^i = \boldsymbol{v}_{:r,:}$ and do not compute SVD for the corresponding weight matrix anymore. We continue this procedure until $\boldsymbol{v}_{:r,:}$ has converged for all weight matrices.

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 $\mathcal{O}(\min(b^2d, bd^2))$ which can be reduced to $\mathcal{O}(k^2b)$ for $k << d$ by randomly choosing k columns from $\boldsymbol{X}$ as introduced in Halko et al. (2011). Let l be the number of weight matrices for which we compute SVD and T be the number of minibatches until all components are converged, then the time complexity is $\mathcal{O}(lTk^2b)$. In other words, the complexity scales linearly with the number of weight matrices and the number of minibatches. To further speed up the computation of SVD, we provide an implementation that runs entirely on GPU.

Algorithm 1 Fine-tuning via EVA

Input: FM $\psi(\cdot)$, ρ, rank r, dataset $\mathcal{D}$

while `not all_converged`(ψ) **do**
  $\boldsymbol{X} \leftarrow \psi($`next`$(\mathcal{D}))$ ▷ get activations
  $\boldsymbol{V}_{\text{new}}, \boldsymbol{\xi} \leftarrow \text{SVD}(\boldsymbol{X}, \rho r)$
  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`$(\psi, \boldsymbol{X})$

2.3 Adaptive Rank Allocation

The singular values obtained by SVD provide an estimate of the variance that is explained by their components. Leveraging this information, 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

Table 1: Comparison of LoRA to EVA and other state-of-the-art PEFT methods for RoBERTa$_{\text{Large}}$(top) and DeBERTav3$_{\text{Base}}$(bottom) on all 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 across 5 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$_{\pm.1}$	89.13
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
BOFT	90.1$_{\pm.2}$	94.4$_{\pm.2}$	91.2$_{\pm.0}$	96.1$_{\pm.1}$	68.4$_{\pm.9}$	90.0$_{\pm.5}$	86.5$_{\pm.6}$	92.5$_{\pm.0}$	88.65
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
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**
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**$_{\pm.1}$	95.9$_{\pm.3}$	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
BOFT	90.3	94.2	92.1	**96.4**	**73.0**	**92.4**	88.8	91.9	89.89
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
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**

allocating more ranks to layers that propagate more information, i.e., explain more variance. More formally, the variance explained by each component in $\boldsymbol{v}^i_{j,:}$ is given by their explained variance ratio

$$\xi^i_j = \frac{{\sigma^i_j}^2}{(M-1)||\boldsymbol{\sigma}^i||_1}, \tag{3}$$

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 computation. Next, we sort the components $\boldsymbol{v}^i_{j,:}$ for each weight matrix in descending order according to their explained variance ratio ξ^i_j. Finally, we assign ranks to pre-trained weights until we reach a certain rank budget.

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. That is, for each $\boldsymbol{W}^i$ we compute $r\rho$ components initially meaning we obtain $Nr\rho$ components in total. For the redistribution we only use the top Nr components according to their explained variance ratio ξ^i_j. 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, as opposed to learning it throughout the training process as done in prior works (Zhang et al., 2023a; Valipour et al., 2023; Meo et al., 2024). In practice we found that the redistribution converges for values of $\rho > 2$. Finally, we initialize $\boldsymbol{B}$ with zeros and perform the standard LoRA fine-tuning, as recommended in Hayou et al. (2024a). In Algorithm 1 we provide pseudocode for EVA.

3 Experiments

First, we elaborate on implementation details of EVA in Section 3.1. Then, we show results for language understanding, image classification, decision making and language generation tasks in Section 3.2, Section 3.3, and Section 3.4, Section 3.5, respectively. Finally, in Section 3.6 and Section 3.7 we investigate convergence properties of our data-driven initialization and report results on ablation studies.

3.1 Implementation Details

We follow the standard LoRA training procedure from Hu et al. (2022) and run hyperparameter searches on the number of ranks and the learning rate. 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, unless mentioned otherwise, because this appeared to be the most stable setting. We use a batch size of 4 for the initial training phase to compute the initialization of EVA with $\rho = 2r$. We only apply our initialization to pre-trained weights, i.e., we do not initialize newly introduced classifier heads. Following Zhang et al. (2023a), we always apply LoRA to all pre-trained

weight matrices. 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 libary (Mangrulkar et al., 2022).

3.2 Language Understanding

For the language understanding benchmarks, 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 dataset comprises eight downstream tasks, such as natural language inference, or sentiment analysis. We compare EVA to LoRA (Hu et al., 2022), DoRA (Liu et al., 2024), AdaLoRA (Zhang et al., 2023a), and BOFT (Liu et al., 2023). Most prior works merely compare performance for a certain rank budget (Zhang et al., 2023a; Liu et al., 2023). We hypothesize that different tasks require different parameter budgets to maximize performance. Therefore, we search over different rank budgets for LoRA methods and the number of sparse matrices for BOFT. Further, we search over the learning rate and always report the best performing setting. We include variance estimates for all methods we trained ourselves. For further details about datasets, implementation, or hyperparameter settings, we refer the reader to Appendix B. We report Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for the remaining tasks. We report our results in Table 1. For RoBERTa$_{\text{Large}}$EVA consistently achieves the highest average scores across all tasks and attains statistically significant improvements over LoRA on STSB, QNLI, and QQP. 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). Overall, when including the rank budget as a hyperparameter, LoRA yields better performance than AdaLoRA, BOFT, and DoRA. We also add a comparison of LoRA to EVA in Table 8 in Appendix B where we show that EVA consistently improves over LoRA for different rank budgets. For DeBERTav3$_{\text{Base}}$, EVA again attains the highest average performance and significantly improves performance over LoRA on STS-B.

We visualize the resulting rank redistribution of EVA for DeBERTav3$_{\text{Base}}$on the GLUE task RTE with $r = 4$ in Figure 2. More ranks are assigned to higher layers of the query ($\boldsymbol{W}_q$), key ($\boldsymbol{W}_k$), and value ($\boldsymbol{W}_v$) projections in the self-attention, while the attention output ($\boldsymbol{W}_o$) and the feed-forward layers ($\boldsymbol{W}_{f1}$,$\boldsymbol{W}_{f2}$) are often assigned a lower number of ranks. We show additional re-distribution patterns for different rank budgets in Appendix B. A pattern that is common among both, DeBERTav3$_{\text{Base}}$and RoBERTa$_{\text{Large}}$is that $\boldsymbol{W}_{f2}$ often only receives a single rank. This means that the first component already explains most of the variance of the respective layer. This pattern corroborates findings of Sun et al. (2024) that LMs often exhibit massive activations on single neurons which encode implicit biases.

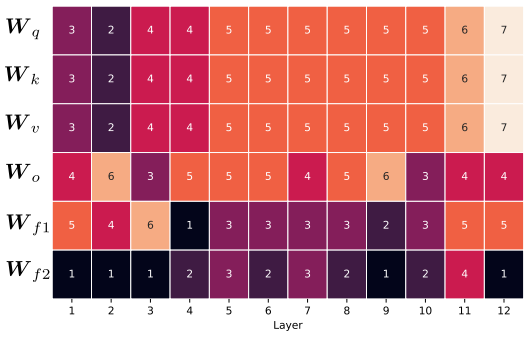

Figure 2: Rank redistribution in EVA for DeBERTav3$_{\text{Base}}$on the GLUE task RTE with initial rank $r = 4$.

3.3 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). For our experiments, we fine-tune the commonly used DINOv2-L/14 model (Oquab et al., 2023) and compare EVA to LoRA (Hu et al., 2022), DoRA (Liu et al., 2024), AdaLoRA (Zhang et al., 2023a), and BOFT (Liu et al., 2023).

Our results in Table 2 demonstrate that EVA again exhibits the highest average score across all tasks and among all competitors. We report error bars for all methods we trained ourselves in Table 13 in Appendix D.4, as well as a comparison for different rank budgets (Table 12). EVA leads to significant improvements over LoRA on both, natural images (Cifar100, Caltech101) and structured images (KITTI-Dist, sNORB-Ele). The highest improvement of EVA over LoRA (+4.9% on sNORB-Ele)

Table 2: Fine-tuning DINOv2-L/14 on the VTAB-1K benchmark. Best average performance is highlighted in boldface. We report average accuracy across five seeds, highlight the best performance in boldface and underline the second best.

	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	67.6	91.7	77.9	99.7	93.7	**92.8**	52.3	88.1	96.1	90.9	77.2	67.2	59.8	58.1	82.8	83.6	**62.0**	36.9	39.4	74.6
LoRA	82.8	93.2	**82.5**	99.7	**95.2**	92.4	62.8	87.8	96.6	**92.0**	74.2	**96.4**	65.2	63.5	82.6	91.7	61.2	36.7	48.7	79.2
AdaLoRA	82.8	93.8	81.8	99.7	94.9	90.4	61.6	88.1	96.5	91.5	75.5	94.6	64.7	62.2	83.5	89.5	60.8	35.9	46.6	78.7
BOFT	82.6	94.1	82.1	99.7	94.8	**92.8**	62.1	87.6	96.3	91.8	73.5	92.3	**65.7**	**62.9**	82.4	88.6	61.2	**38.9**	47.9	78.8
EVA	83.3	95.4	81.8	99.7	95.0	92.2	63.0	87.1	96.5	91.1	74.3	95.5	65.6	62.8	**84.6**	90.5	61.6	36.5	**53.6**	**79.5**
DoRA	**83.5**	**97.5**	81.3	99.7	**95.2**	91.3	61.7	**92.7**	**97.3**	89.3	**77.7**	91.8	65.5	56.5	84.2	90.0	61.3	26.5	45.2	78.3
EVA+DoRA	83.0	93.2	82.1	99.7	94.9	92.6	**63.1**	88.7	96.7	**92.0**	73.3	94.5	**65.7**	60.4	82.4	**92.5**	61.0	37.6	52.4	79.3

is significantly higher than the largest performance drop (-1.2% on dSpr-Loc). Further, most PEFT methods outperform FFT on the VTAB-1K benchmark. This is consistent with our results on the GLUE benchmark, where PEFT methods consistently outperformed FFT on smaller datasets. DoRA also significantly improves over LoRA on some tasks (Cifar100, Caltech101, Camelyon, Retinopathy, and KITTI-Dist), however, it also performs significantly worse on a variety of tasks (sNORB-Azim, DTD, SVHN, Resisc45, Clevr-Count, DMLab, dSpri-Loc), therefore yields worse performance on average. We observe similar behavior for both BOFT and AdaLoRA.

3.4 Decision Making

We follow the single task fine-tuning experiments in Schmied et al. (2024) and fine-tune a Decision Transformer (Chen et al., 2021, DT) on the Meta-World benchmark suite (Yu et al., 2020). Meta-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 afterwards fine-tune it on the CW10 holdout tasks. We evaluate EVA against LoRA, AdaLoRA, DoRA, and FFT, excluding BOFT due to compatibility issues with our custom layers and report success rates and standard errors for each task of CW10 in Table 3. As in Schmied et al. (2024), we observe that FFT significantly outperforms LoRA. However, EVA reduces that gap and achieves a performance close to FFT. Furthermore, DoRA significantly improves upon both, LoRA and EVA and reaches an even higher performance than FFT. Therefore, we add an additional setting to investigate whether initializing DoRA with EVA can further advance performance (EVA+DoRA). Indeed, we observe that EVA+DoRA leads to the best average performance across all tasks. These results demonstrate that EVA can also improve other LoRA variants, such as DoRA. We report results for different rank budgets in Table 15 in Appendix E.

3.5 Language Generation

We follow the experiments conducted in Hu et al. (2022) and fine-tune GPT2 (Radford et al., 2019) medium and large on the E2E dataset (Novikova et al., 2017). The dataset aims at evaluating the generation capabilities of language models. Given a set of restaurant-specific attributes and their corresponding values, the model aims to generate natural and coherent utterances. This requires content selection, as not all of the attributes are useful to construct meaningful natural language descriptions. The generated texts are then evaluated by measuring how close they are to human generated utterances. Due to resource constraints we only compare EVA to FFT, LoRA, DoRA and AdaLoRA. Since the official implementation for BOFT does not support the custom layers of GPT2, we do not include it. We report BLEU scores (Papineni et al., 2002) for both model sizes and all methods in Table 4. EVA attains the highest scores for both model sizes, closely followed by DoRA for GPT2-medium and AdaLoRA for GPT2-large. All evaluated PEFT methods outperformed FFT, potentially due to their inherent regularization effects.

Table 3: Results for single task fine-tuning experiments on the Meta-World benchmark (Yu et al., 2020). We report mean success rates and standard error across three seeds for every task, highlight the best performance in boldface and underline the second best.

	faucet-close	hammer	handle-press-side	peg-unplug-side	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
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	$1.0_{\pm.0}$	$\mathbf{1.0}_{\pm.0}$	$1.0_{\pm.0}$	$0.6_{\pm1.2}$	$\mathbf{1.0}_{\pm.0}$	$\mathbf{1.0}_{\pm.0}$	$\mathbf{1.0}_{\pm.0}$	$1.0_{\pm.0}$	$\mathbf{0.67}_{\pm\mathbf{1.5}}$	$1.0_{\pm.0}$	$\underline{0.93}$
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**

3.6 SVD Convergence Analysis

The data-driven initialization of EVA relies on incremental SVD on minibatches of activations. We conduct this procedure in the initial training phase. In Figure 3, left, we show that this process converges for RoBERTa$_{\text{Large}}$ on the GLUE task STS-B 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 112 minibatches. Increasing the batch size results in more computational overhead. However, even for larger batch sizes, such as 64, the initialization only takes around 180 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. Here, we measure cosine similarity between components obtained via incremental SVD after rank redistribution for different minibatch orders for all layers of RoBERTa$_{\text{Large}}$. This indicates that these models exhibit certain activation patterns that remain consistent across different batch orders which lead to a robust initialization for EVA.

Table 4: BLEU scores for GPT2-medium and GPT2-large on the E2E dataset. We report mean and standard deviation across three seeds, highlight best performance in boldface and underline second best.

Method	GPT2-medium	GPT2-large
FFT	68.20	68.50
LoRA	$69.6_{\pm.3}$	$69.3_{\pm.3}$
DoRA	$\underline{69.7}_{\pm.2}$	$69.2_{\pm.8}$
AdaLoRA	$68.8_{\pm.6}$	$\underline{69.5}_{\pm.4}$
EVA	$\mathbf{69.8}_{\pm.2}$	$\mathbf{69.6}_{\pm.4}$

3.7 Ablation Studies

We conduct ablation studies on EVA to investigate the importance of its constituents. In particular, we investigate the impact of scale and directions of our initialization. For this line of experiments, we use the VTAB-1K dataset because it comprises a diverse set of tasks and allows for a systematic investigation. We report results for our ablation studies in Table 5 and explain the different settings in the following paragraphs.

Effect of the scale To address the effect of scale on the initialization, we add a setting which uses whitening (EVA-whiten). This scales the initialization by the reciprocal of their eigenvalues, while preserving its directions. We found that whitening can significantly improve performance on structured vision tasks. This indicates that scale is especially important for out-of-distribution datasets. However, on the remaining groups it leads to a decrease in performance.

Effect of directions To address the importance of the direction of our initialization, we randomly permute its rows (EVA-perm). This has the effect, that the scale of is preserved while the directions and ℓ_2 norm of $\boldsymbol{A}$ are altered. Additionally, we add a setting where we randomly rotate $\boldsymbol{A}$ (EVA-rot),

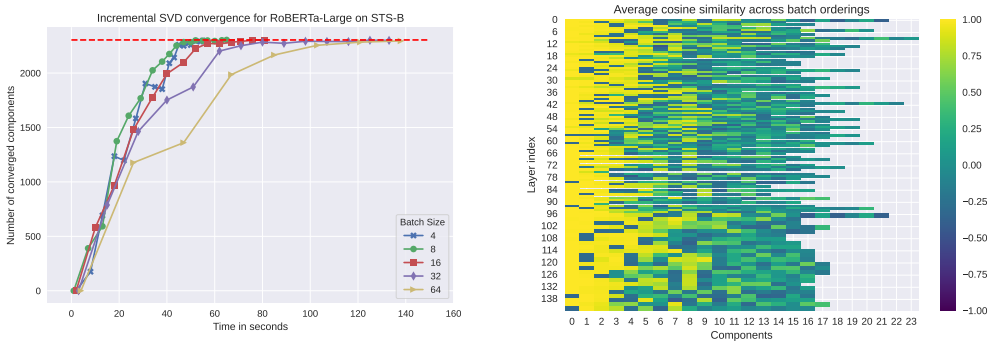

Figure 3: **Left:** Time in seconds until convergence of incremental SVD components for different batch sizes for RoBERTa$_{\text{Large}}$on the GLUE task STSB. The dashed line indicates the total number of components. **Right:** Average cosine similarity between SVD components across 10 random seeds for permuting the batch order. The first 10 components remain consistent across all permutations. While the remaining components vary, they strongly correlate as their cosine similarity is around 0.5.

which preserves ℓ_2 norm, but alters directions. EVA-perm leads to worse performance on natural and structured tasks compared to EVA. Surprisingly, it still outperforms LoRA on average across all tasks. Similarly, EVA-rot outperforms LoRA on average, but experiences drops in performance for natural and structured tasks. These results indicate that the directions of $\boldsymbol{A}$ are particularly important for both natural and specialized tasks.

Effect of rank redistribution We conduct an experiment in which we randomly initialize $\boldsymbol{A}$ after performing rank redistribution (LoRA-redist). While this setting is not practical, it gives insights on the effect of the redistribution and whether its benefits are bound to EVA. Further, we add a setting where we do not perform the rank redistribution, but still use our initialization (EVA-no-redist). The redistribution has a positive effect on LoRA on the structured tasks, but a negative effect on both natural and specialized tasks, leading to an average performance worse than LoRA. However, EVA-no-redist leads to an improvement for both, natural and structured tasks. This illustrates that rank redistribution is mostly beneficial in combination with EVA's initialization of $\boldsymbol{A}$.

4 Conclusion and Future Work

Table 5: Group-wise averages for DINOv2-L/14 ablation studies on the VTAB-1K benchmark.

Method	Nat.	Spec.	Struct.	All
LoRA	86.9	**87.6**	68.2	79.2
LoRA-redist	86.7	87.2	68.4	79.1
EVA-whiten	86.5	87.3	**69.0**	79.3
EVA-rot	87.0	87.4	68.6	79.4
EVA-perm	86.8	87.5	68.7	79.3
EVA-no-redist	87.0	87.3	68.7	79.4
EVA	**87.2**	**87.6**	68.7	**79.5**

We propose a novel PEFT method named Explained Variance Adaptation (EVA). EVA initializes LoRA matrices in a data-driven manner by leveraging the fine-tuning dataset. To this end, we compute SVD on activation vectors of mini-batches during the initial training stage. Further, we re-distribute ranks across the entire model according to maximize the amount of variance they explain. Thereby, in EVA we bind the benefits of adaptive rank allocation per weight matrix and effective data-driven initialization, resulting in one initialization to rule them all. Our experiments demonstrated performance gains of EVA over state-of-the-art PEFT methods on language understanding, language generation, image classification and decision making tasks. In the future we aim at applying EVA to large-scale models and investigate the effect of EVA on convergence properties and quantization. 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 yielding significant improvements at almost no additional cost.

Acknowledgments and Disclosure of Funding

We acknowledge EuroHPC Joint Undertaking for awarding us access to Vega at IZUM, Slovenia, Karolina at IT4Innovations, Czech Republic, MeluXina at LuxProvide, Luxembourg, Leonardo at CINECA, Italy, MareNostrum5 at BSC, Spain. The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. We thank the projects Medical Cognitive Computing Center (MC3), INCONTROL-RL (FFG-881064), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for GranularFlow (FFG-871302), EPILEPSIA (FFG-892171), AIRI FG 9-N (FWF-36284, FWF-36235), AI4GreenHeatingGrids (FFG- 899943), INTEGRATE (FFG-892418), ELISE (H2020-ICT-2019-3 ID: 951847), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01). We thank NXAI GmbH, Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, UCB Biopharma SRL, Merck Healthcare KGaA, Verbund AG, GLS (Univ. Waterloo), Software Competence Center Hagenberg GmbH, Borealis AG, TÜV Austria, Frauscher Sensonic, TRUMPF and the NVIDIA Corporation. Fabian Paischer acknowledges travel support from ELISE (GA no 951847)

References

Aghajanyan, A., Gupta, S., and Zettlemoyer, L. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Zong, C., Xia, F., Li, W., and Navigli, R. (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.

Babakniya, S., Elkordy, A. R., Ezzeldin, Y. H., Liu, Q., Song, K., El-Khamy, M., and Avestimehr, S. Slora: Federated parameter efficient fine-tuning of language models. *CoRR*, abs/2308.06522, 2023. doi: 10.48550/ARXIV.2308.06522.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. Deepmind lab. *CoRR*, abs/1612.03801, 2016.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R. B., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N. S., Chen, A. S., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N. D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M. S., Krishna, R., Kuditipudi, R., and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M. S., Salazar, G., Sanketi, P. R., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H. T., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. RT-1: robotics transformer for real-world control at scale. In Bekris, K. E., Hauser, K., Herbert, S. L., and Yu, J. (eds.), *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023. doi: 10.15607/RSS.2023.XIX.025.

Chavan, A., Liu, Z., Gupta, D. K., Xing, E. P., and Shen, Z. One-for-all: Generalized lora for parameter-efficient fine-tuning. *CoRR*, abs/2306.07967, 2023. doi: 10.48550/ARXIV.2306.07967.

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

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

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. 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.

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

Dehghani, M., Djolonga, J., Mustafa, B., Padlewski, P., Heek, J., Gilmer, J., Steiner, A. P., Caron, M., Geirhos, R., Alabdulmohsin, I., Jenatton, R., Beyer, L., Tschannen, M., Arnab, A., Wang, X., Ruiz, C. R., Minderer, M., Puigcerver, J., Evci, U., Kumar, M., van Steenkiste, S., Elsayed, G. F., Mahendran, A., Yu, F., Oliver, A., Huot, F., Bastings, J., Collier, M., Gritsenko, A. A., Birodkar, V., Vasconcelos, C. N., Tay, Y., Mensink, T., Kolesnikov, A., Pavetic, F., Tran, D., Kipf, T., Lucic, M., Zhai, X., Keysers, D., Harmsen, J. J., and Houlsby, N. Scaling vision transformers to 22 billion parameters. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (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.

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

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (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.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. 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.

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

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

Gauch, M., Beck, M., Adler, T., Kotsur, D., Fiel, S., Eghbal-zadeh, H., Brandstetter, J., Kofler, J., Holzleitner, M., Zellinger, W., Klotz, D., Hochreiter, S., and Lehner, S. Few-shot learning by dimensionality reduction in gradient space. In Chandar, S., Pascanu, R., and Precup, D. (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.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. Vision meets robotics: The KITTI dataset. *Int. J. Robotics Res.*, 32(11):1231–1237, 2013. doi: 10.1177/0278364913491297.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, D. M. (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.

Gur-Ari, G., Roberts, D. A., and Dyer, E. Gradient descent happens in a tiny subspace. *CoRR*, abs/1812.04754, 2018.

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

Hao, Y., Cao, Y., and Mou, L. Flora: Low-rank adapters are secretly gradient compressors. *CoRR*, abs/2402.03293, 2024. doi: 10.48550/ARXIV.2402.03293.

Hayou, S., Ghosh, N., and Yu, B. The impact of initialization on lora finetuning dynamics, 2024a.

Hayou, S., Ghosh, N., and Yu, B. Lora+: Efficient low rank adaptation of large models, 2024b.

He, K., Zhang, X., Ren, S., and Sun, J. 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.

He, P., Gao, J., and Chen, W. 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.

Helber, P., Bischke, B., Dengel, A., and Borth, D. 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.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. 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.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. B. 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.

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

Kopiczko, D. J., Blankevoort, T., and Asano, Y. M. ELoRA: Efficient low-rank adaptation with random matrices. In *The Twelfth International Conference on Learning Representations*, 2024.

Krähenbühl, P., Doersch, C., Donahue, J., and Darrell, T. Data-dependent initializations of convolutional neural networks. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

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

LeCun, Y., Huang, F. J., and Bottou, L. 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.

Li, Y., Yu, Y., Liang, C., He, P., Karampatziakis, N., Chen, W., and Zhao, T. Loftq: Lora-fine-tuning-aware quantization for large language models. *CoRR*, abs/2310.08659, 2023. doi: 10.48550/ARXIV.2310.08659.

Lialin, V., Shivagunde, N., Muckatira, S., and Rumshisky, A. Stack more layers differently: High-rank training through low-rank updates. *CoRR*, abs/2307.05695, 2023. doi: 10.48550/ARXIV.2307.05695.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (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.

Liu, S., Wang, C., Yin, H., Molchanov, P., Wang, Y. F., Cheng, K., and Chen, M. Dora: Weight-decomposed low-rank adaptation. *CoRR*, abs/2402.09353, 2024. doi: 10.48550/ARXIV.2402.09353.

Liu, W., Qiu, Z., Feng, Y., Xiu, Y., Xue, Y., Yu, L., Feng, H., Liu, Z., Heo, J., Peng, S., Wen, Y., Black, M. J., Weller, A., and Schölkopf, B. Parameter-efficient orthogonal finetuning via butterfly factorization. *CoRR*, abs/2311.06243, 2023. doi: 10.48550/ARXIV.2311.06243.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

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

Mangrulkar, S., Gugger, S., Debut, L., Belkada, Y., Paul, S., and Bossan, B. Peft: State-of-the-art parameter-efficient fine-tuning methods, 2022.

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

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

Meo, C., Sycheva, K., Goyal, A., and Dauwels, J. 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.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

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

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

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

Nilsback, M. and Zisserman, A. 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.

Novikova, J., Dusek, O., and Rieser, V. The E2E dataset: New challenges for end-to-end generation. In Jokinen, K., Stede, M., DeVault, D., and Louis, A. (eds.), *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pp. 201–206. Association for Computational Linguistics, 2017. doi: 10.18653/V1/W17-5525.

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

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P., Li, S., Misra, I., Rabbat, M. G., Sharma, V., Synnaeve, G., Xu, H., Jégou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. Dinov2: Learning robust visual features without supervision. *CoRR*, abs/2304.07193, 2023. doi: 10.48550/ARXIV.2304.07193.

Papineni, K., Roukos, S., Ward, T., and Zhu, W. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pp. 311–318. ACL, 2002. doi: 10.3115/1073083.1073135.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. 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.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E. Z., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (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.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *CoRR*, 2019.

Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T. P., Alayrac, J., Soricut, R., Lazaridou, A., Firat, O., Schrittwieser, J., Antonoglou, I., Anil, R., Borgeaud, S., Dai, A. M., Millican, K., Dyer, E., Glaese, M., Sottiaux, T., Lee, B., Viola, F., Reynolds, M., Xu, Y., Molloy, J., Chen, J., Isard, M., Barham, P., Hennigan, T., McIlroy, R., Johnson, M., Schalkwyk, J., Collins, E., Rutherford, E., Moreira, E., Ayoub, K., Goel, M., Meyer, C., Thornton, G., Yang, Z., Michalewski, H., Abbas, Z., Schucher, N., Anand, A., Ives, R., Keeling, J., Lenc, K., Haykal, S., Shakeri, S., Shyam, P., Chowdhery, A., Ring, R., Spencer, S., Sezener, E., 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.

Schmied, T., Hofmarcher, M., Paischer, F., Pascanu, R., and Hochreiter, S. Learning to modulate pre-trained models in rl. *Advances in Neural Information Processing Systems*, 36, 2024.

Schölkopf, B., Smola, A., and Müller, K.-R. Kernel principal component analysis. In Gerstner, W., Germond, A., Hasler, M., and Nicoud, J.-D. (eds.), *Artificial Neural Networks — ICANN'97*, pp. 583–588, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-69620-9.

Sun, M., Chen, X., Kolter, J. Z., and Liu, Z. Massive activations in large language models. In *First Conference on Language Modeling*, 2024.

Sung, Y., Nair, V., and Raffel, C. Training neural networks with fixed sparse masks. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (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.

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

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971, 2023. doi: 10.48550/ARXIV.2302.13971.

Valipour, M., Rezagholizadeh, M., Kobyzev, I., and Ghodsi, A. Dylora: Parameter-efficient tuning of pre-trained models using dynamic search-free low-rank adaptation. In Vlachos, A. and Augenstein, I. (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.

Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., and Welling, M. Rotation equivariant cnns for digital pathology. In Frangi, A. F., Schnabel, J. A., Davatzikos, C., Alberola-López, C., and Fichtinger, G. (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.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. 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.

Wołczyk, M., Zając, M., Pascanu, R., Kuciński, Ł., and Miłoś, P. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Wolczyk, M., Zajkac, M., Pascanu, R., Kuciński, L., and Miloś, P. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. 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.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. 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.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., Beyer, L., Bachem, O., Tschannen, M., Michalski, M., Bousquet, O., Gelly, S., and Houlsby, N. The visual task adaptation benchmark. *CoRR*, abs/1910.04867, 2019.

Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., and Zhao, T. 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.

Zhang, Z., Liu, B., and Shao, J. Fine-tuning happens in tiny subspaces: Exploring intrinsic task-specific subspaces of pre-trained language models. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (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.

Zhao, J., Zhang, Z., Chen, B., Wang, Z., Anandkumar, A., and Tian, Y. Galore: Memory-efficient LLM training by gradient low-rank projection. *CoRR*, abs/2403.03507, 2024. doi: 10.48550/ARXIV.2403.03507.

Zi, B., Qi, X., Wang, L., Wang, J., Wong, K., and Zhang, L. 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.

Zitkovich, B., Yu, T., Xu, S., Xu, P., Xiao, T., Xia, F., Wu, J., Wohlhart, P., Welker, S., Wahid, A., Vuong, Q., Vanhoucke, V., Tran, H. T., Soricut, R., Singh, A., Singh, J., Sermanet, P., Sanketi, P. R., Salazar, G., Ryoo, M. S., Reymann, K., Rao, K., Pertsch, K., Mordatch, I., Michalewski, H., Lu, Y., Levine, S., Lee, L., Lee, T. E., Leal, I., Kuang, Y., Kalashnikov, D., Julian, R., Joshi, N. J., Irpan, A., Ichter, B., Hsu, J., Herzog, A., Hausman, K., Gopalakrishnan, K., Fu, C., Florence, P., Finn, C., Dubey, K. A., Driess, D., Ding, T., Choromanski, K. M., Chen, X., Chebotar, Y., Carbajal, J., Brown, N., Brohan, A., Arenas, M. G., and Han, K. RT-2: vision-language-action models transfer web knowledge to robotic control. In Tan, J., Toussaint, M., and Darvish, K. (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

Contents

A Reproducibility Statement

We provide the source code to reproduce all our experiments in the supplementary material as a zip archive. The code contains instructions how to install the environment and how to execute all the parameter searches that we conducted. Additionally, we provide a package that contains implementations for EVA along with different LoRA variants, such as DoRA, and ELoRA. We will release our codebase upon publication and also integrate EVA into the widely used PEFT library (Mangrulkar et al., 2022).

B Natural language understanding

B.1 Dataset Statistics

The dataset statistics for each task in the GLUE benchmark (Wang et al., 2019) are shown in Table 6. 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.

Table 6: 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

B.2 Implementation Details

We base our implementation on the codebase of LoRA[1]. For these experiments, we initially pre-compute 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).

B.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 7. For AdaLoRA, we search over the same ranks and always start initial ranks 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) proposed 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., 2024b), however, this did not result in consistent improvements. Instead, we found 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 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.

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

Table 7: 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							

B.4 Additional results

We report additional results for EVA compared to LoRA for different rank budgets in Table 8. 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 4), $r = 4$ (Figure 5), $r = 8$ (Figure 6), and $r = 16$ (Figure 7) 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 9. Our

Table 8: 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 9: 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	$\mathbf{69.5_{\pm 1.4}}$	$\mathbf{91.4_{\pm 0.8}}$	$\mathbf{88.8_{\pm 1.2}}$	$\mathbf{92.6_{\pm 0.1}}$	**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

results show that while EVA-Raw and EVA-Max slighthly improve upon LoRA, they perform worse on average than EVA.

C Natural language generation

C.1 Dataset statistics

The aim of the E2E dataset (Novikova et al., 2017) is to evaluate the generation capabilities of language models. Each data point consists of a set of attributes and their assigned values, and cover common concepts of the restaurant sector. These are also referred to as meaning representations and consists of 3–8 attributes (slots), such as name, food or area, and their values. We show a sample from the dataset below:

```
{'human_reference': 'The Vaults pub near Café Adriatic has a 5 star rating.
Prices start at £30.',
 'meaning_representation': 'name[The Vaults], eatType[pub],
 priceRange[more than £30], customer rating[5 out of 5], near[Café Adriatic]'}
```

The dataset consists of 51.46 K samples and is split into a ratio of 76.5/8.5/15 percent for train, development and test splits, respectively. Further, the splits contain distinct meaning representations.

C.2 Implementation details

Similar to B, our implementation is built on top of the LoRA codebase in PyTorch. To run the AdaLoRA baseline, we use the implementation provided by the huggingface `peft` library. The DoRA baseline as well as EVA are custom implementations. Fine-tuning GPT2-medium and GPT2-large on e2e for five epochs takes several hours depending on the model size. For testing we run beam search on the test set with a batch size of one as batched generation had a negative impact on performance. Due to setting batch size 1 evaluation takes around 10 hours. All training runs were executed on single A100 GPUs.

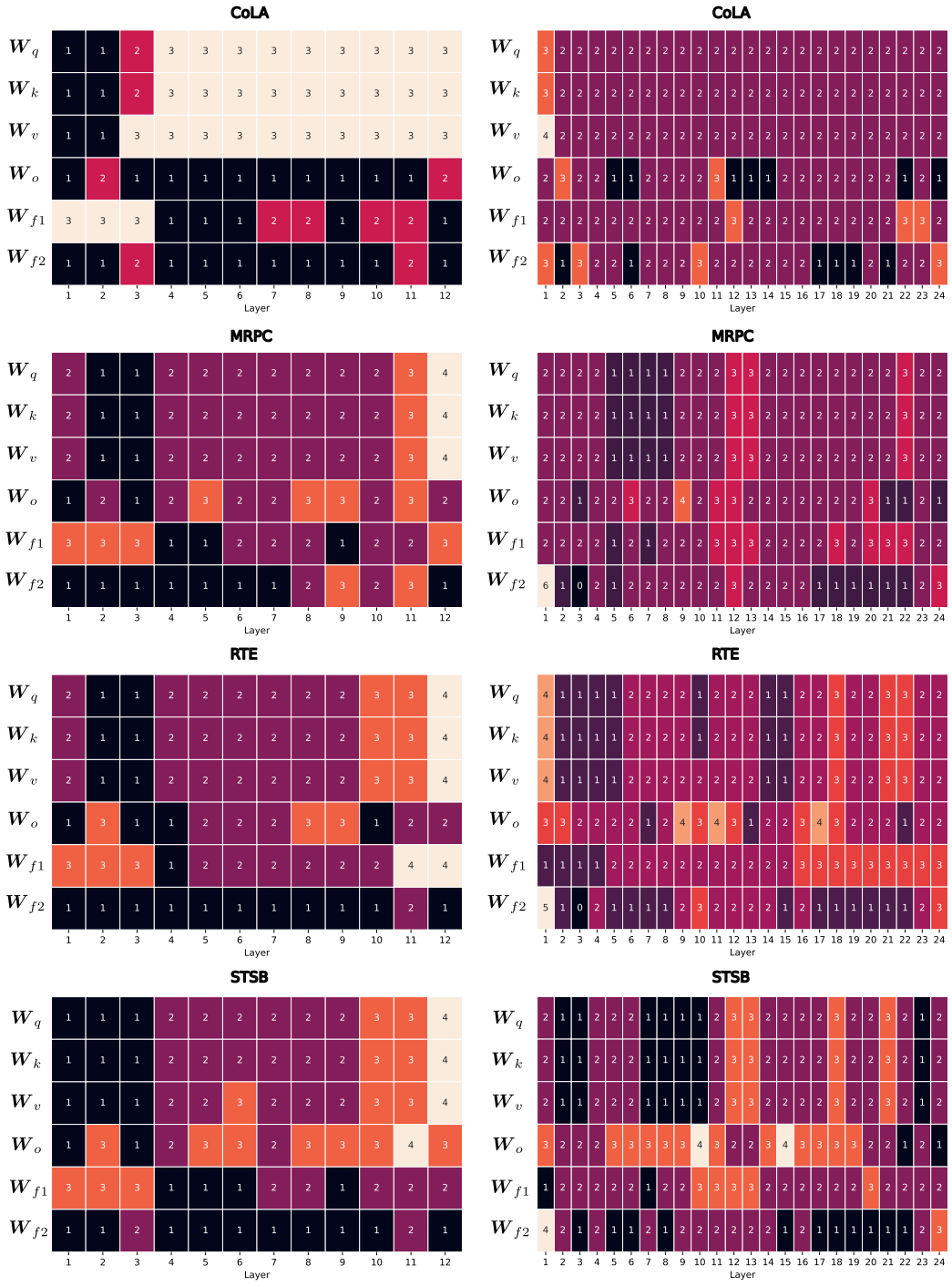

Figure 4: Rank distribution after initialization with EVA on four tasks of the GLUE benchmark (CoLA, MRPC, RTE, STSB) for DeBERTav3$_{\text{Base}}$(left) and RoBERTa$_{\text{Large}}$(right) with initial rank $r = 2$.

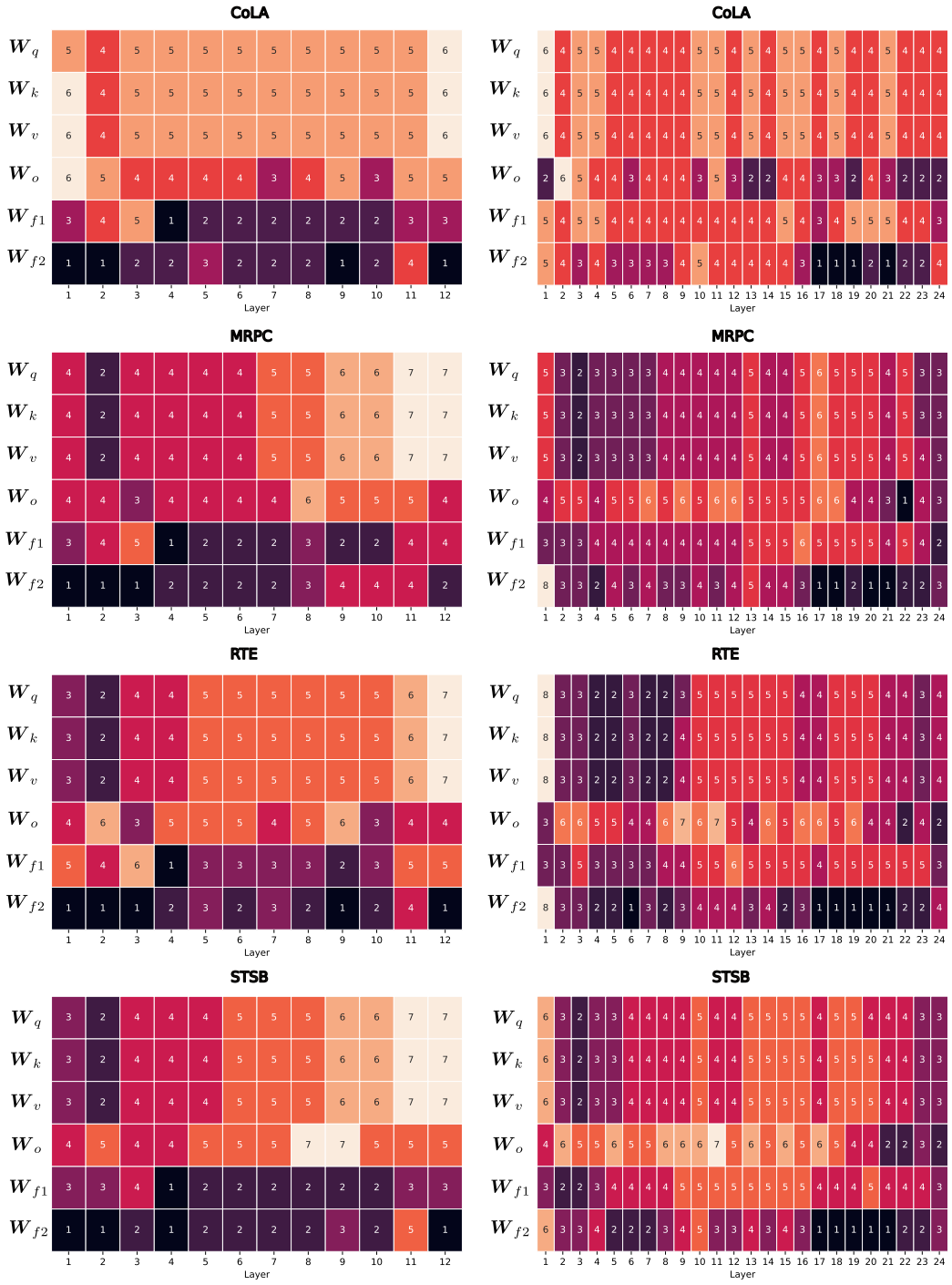

Figure 5: Rank distribution after initialization with EVA on four tasks of the GLUE benchmark (CoLA, MRPC, RTE, STSB) for DeBERTav3$_{\text{Base}}$(left) and RoBERTa$_{\text{Large}}$(right) with initial rank $r = 4$.

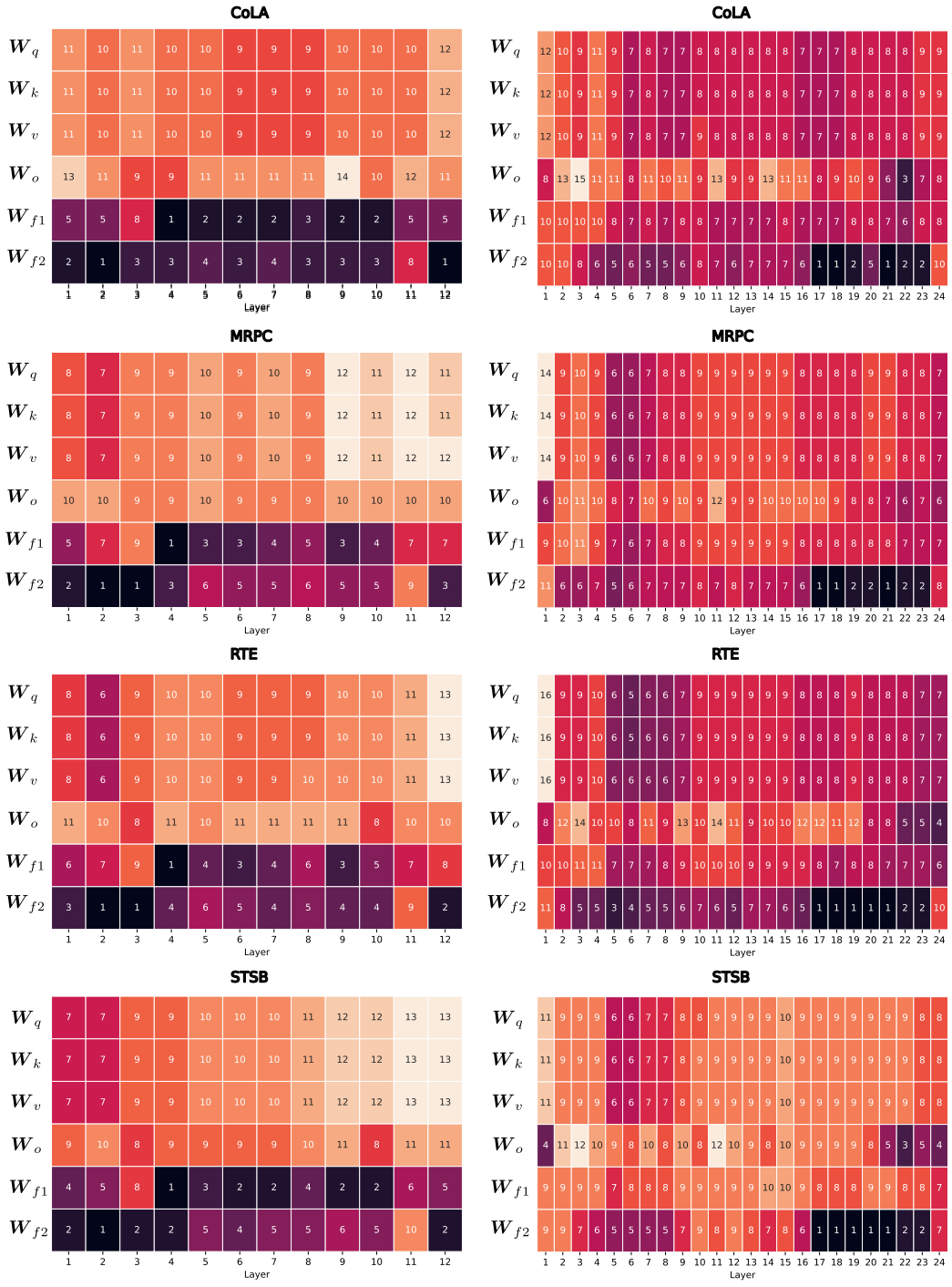

Figure 6: Rank distribution after initialization with EVA on four tasks of the GLUE benchmark (CoLA, MRPC, RTE, STSB) for DeBERTav3$_{\text{Base}}$(left) and RoBERTa$_{\text{Large}}$(right) with initial rank $r = 8$.

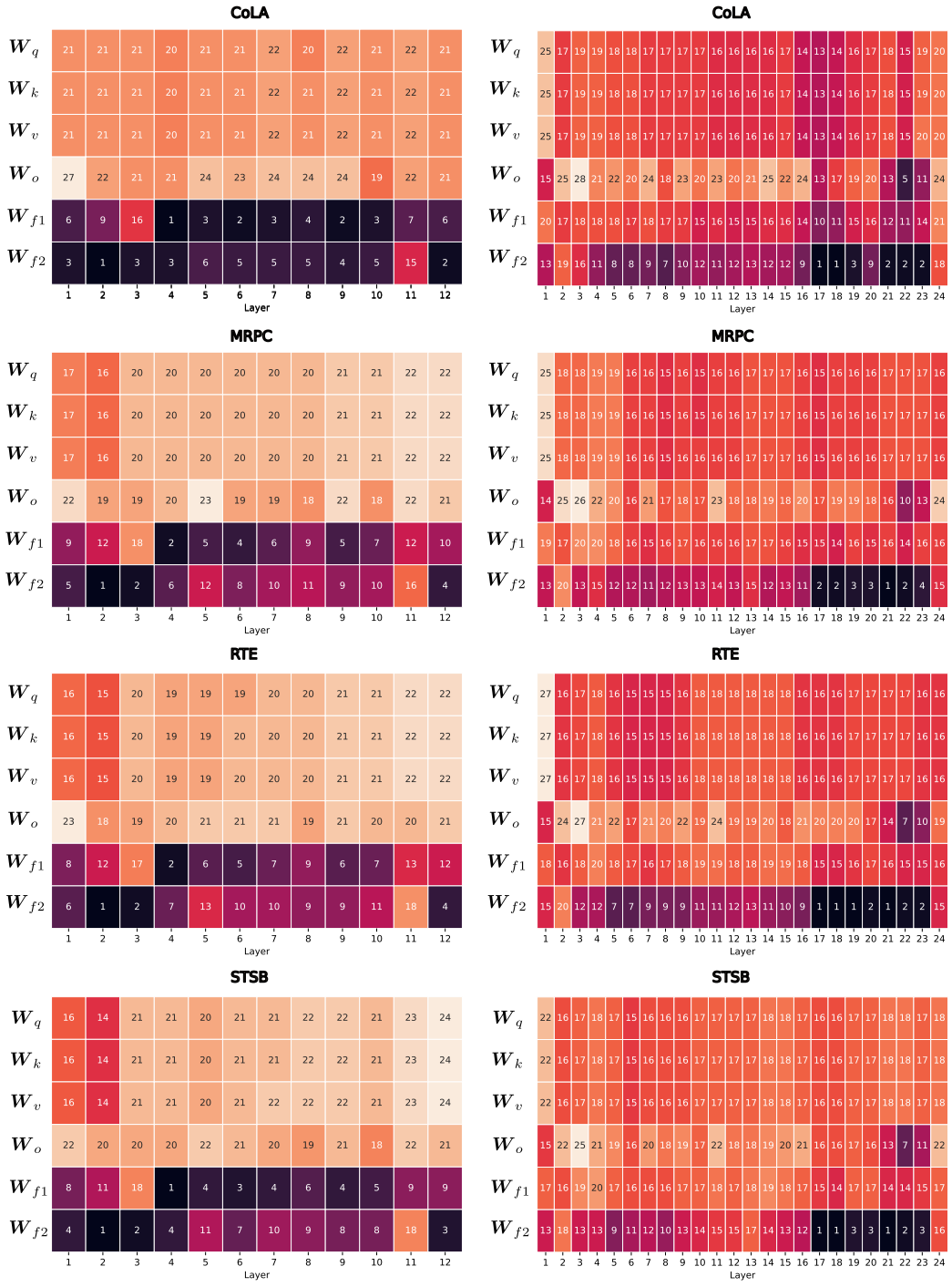

Figure 7: Rank distribution after initialization with EVA on four tasks of the GLUE benchmark (CoLA, MRPC, RTE, STSB) for DeBERTav3$_{\text{Base}}$(left) and RoBERTa$_{\text{Large}}$(right) with initial rank $r = 16$.

C.3 Hyperparameter search

For finetuning on E2E, we follow the hyperparameter settings used by Hu et al. (2022). The reported results in 4 are the best setting for each method based on a grid search over different learning rates. Further, as in Hu et al. (2022), we set $\alpha = 32$ for all our experiments. We use AdamW with weight decay and a linear learning rate schedule with warm-up. We train for a total of 5 epochs and use the best checkpoint for evaluation. All hyperparameters are summarized in 10

Table 10: hyperparameters for finetuning GPT2-medium and GPT2-large on the E2E dataset

Training	
Optimizer	AdamW
Weight Decay	0.01
Lora Dropout	0.0
Batch Size	8
#Epoch	5
LR Schedule	Linear
Warmup Steps	500
Label Smooth	0.1
Learning Rates	{1e-3, 4e-4, 2e-4}
LoRA Dims	{2, 4, 8, 16}
LoRA α	32
Inference	
Beam Size	5
Length Penalty	0.9
no repeat ngram size	4

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 11. 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.

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 BOFT and AdaLora, 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 A100-40GB GPUs. All VTAB-1K experiments were conducted on a public research cluster with 4xA100-40GB nodes. A single run (all 19 datasets with hyperparameter tuning and evaluation) takes roughly 160 GPU-hours but can be easily parallelized.

We use the original DINOv2-L/14 model 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.

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 rates $\{$2.5e-3, 1e-3, 7.5e-4, 5e-4, 2.5e-4$\}$ and average the accuracy on

Table 11: 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

the 200 validation samples over 3 different seeds to choose the best learning rate 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.

For each method, we additionally sweep over one method-specific hyperparameter. For LoRA, DoRA, AdaLoRa, and EVA we sweep over `rank` $\in \{2, 4, 8, 16\}$ and for BOFT we sweep over `block_size` $\in \{2, 4, 8, 16\}$.

D.4 Additional results

We provide a comparison between EVA and LoRA for different rank-budgets in Table 12. We find that EVA performs on-par or better on average across all tasks, demonstrating its effectiveness. Further, to complement our main results in Table 2, we report the respective standard deviations in Table 13.

Table 12: Average VTAB-1K test performances across 5 seeds using the best learning rate tuned on validation set performance.

	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
LoRA$_{r=2}$	83.5	93.2	82.1	99.7	95.1	92.5	62.8	87.8	96.6	92.0	72.5	96.3	63.9	63.5	82.6	89.3	60.9	39.0	48.7	79.1
LoRA$_{r=4}$	82.8	93.0	82.2	99.7	95.2	92.4	62.7	88.7	96.6	92.2	73.4	95.7	65.1	61.2	83.8	84.0	61.3	36.7	49.6	78.7
LoRA$_{r=8}$	83.8	93.0	82.5	99.7	95.0	92.4	62.9	87.2	96.6	92.2	74.2	96.4	65.2	61.1	83.0	91.7	61.2	39.5	44.1	79.0
LoRA$_{r=16}$	83.5	93.2	82.1	99.7	95.1	92.5	62.8	87.8	96.6	92.0	72.5	96.3	63.9	63.5	82.6	89.3	60.9	39.0	48.7	79.1
EVA$_{r=2}$	83.1	95.4	81.8	99.7	95.0	92.4	63.0	87.7	96.4	91.6	73.8	95.5	65.4	62.2	83.4	90.5	61.6	36.5	51.7	79.3
EVA$_{r=4}$	82.3	95.2	82.0	99.7	94.8	92.5	63.0	87.3	96.7	92.0	73.4	92.8	65.3	62.8	84.5	89.9	59.2	35.9	53.6	79.1
EVA$_{r=8}$	83.3	94.5	81.7	99.7	94.7	92.5	62.6	87.0	96.5	91.1	74.0	93.8	65.7	63.6	84.6	89.0	58.9	37.6	51.4	79.1
EVA$_{r=16}$	83.3	94.5	81.9	99.7	95.0	92.2	63.0	87.1	96.5	92.2	74.3	93.2	65.6	63.8	82.2	87.9	59.1	37.8	52.1	79.0

Table 13: Standard deviations for the VTAB-1K results (Table 2) 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

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 14.

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 AdaLora.

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. (2021). 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.

Table 14: Dataset statistics for all MT40 tasks from Schmied et al. (2024).

Task	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{A}\rvert$	**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

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

Table 15: Full comparison for all methods on CW10. We fine-tune a 12M DT on 10 tasks individually and report the mean success rates/rewards ($\pm$ 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	-	$0.97_{\pm 0.03}$	$0.93_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.6_{\pm 0.05}$	$0.7_{\pm 0.12}$	$1.0_{\pm 0.0}$	$0.93_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.57_{\pm 0.07}$	$1.0_{\pm 0.0}$	$0.87_{\pm 0.03}$
LoRA	2	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.6_{\pm 0.05}$	$0.57_{\pm 0.07}$	$0.97_{\pm 0.03}$	$0.93_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.37_{\pm 0.1}$	$1.\pm 0.0$	$0.84_{\pm 0.04}$
	4	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.47_{\pm 0.12}$	$0.63_{\pm 0.1}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.23_{\pm 0.12}$	$1.0_{\pm 0.0}$	$0.83_{\pm 0.05}$
	8	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.43_{\pm 0.05}$	$0.4_{\pm 0.09}$	$0.97_{\pm 0.03}$	$0.93_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.23_{\pm 0.12}$	$1.0_{\pm 0.0}$	$0.79_{\pm 0.06}$
	16	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.43_{\pm 0.03}$	$0.47_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.4_{\pm 0.09}$	$1.0_{\pm 0.0}$	$0.82_{\pm 0.05}$
DoRA	2	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.57_{\pm 0.05}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.33_{\pm 0.11}$	$1.0_{\pm 0.0}$	$0.89_{\pm 0.04}$
	4	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.6_{\pm 0.12}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.43_{\pm 0.12}$	$1.0_{\pm 0.0}$	$0.9_{\pm 0.04}$
	8	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.47_{\pm 0.12}$	$0.93_{\pm 0.05}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.57_{\pm 0.15}$	$1.0_{\pm 0.0}$	$0.9_{\pm 0.04}$
	16	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.57_{\pm 0.12}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.67_{\pm 0.15}$	$1.0_{\pm 0.0}$	$0.92_{\pm 0.03}$
AdaLoRA	2	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.37_{\pm 0.05}$	$0.37_{\pm 0.05}$	$0.93_{\pm 0.05}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.13_{\pm 0.07}$	$1.0_{\pm 0.0}$	$0.77_{\pm 0.06}$
	4	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.37_{\pm 0.07}$	$0.57_{\pm 0.1}$	$0.97_{\pm 0.03}$	$0.9_{\pm 0.08}$	$1.0_{\pm 0.0}$	$0.13_{\pm 0.07}$	$1.0_{\pm 0.0}$	$0.79_{\pm 0.06}$
	8	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.3_{\pm 0.05}$	$0.57_{\pm 0.14}$	$0.93_{\pm 0.03}$	$0.87_{\pm 0.07}$	$1.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.76_{\pm 0.06}$
	16	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.4_{\pm 0.09}$	$0.57_{\pm 0.12}$	$0.97_{\pm 0.03}$	$0.93_{\pm 0.05}$	$1.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.78_{\pm 0.06}$
EVA	2	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.43_{\pm 0.07}$	$0.77_{\pm 0.05}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.63_{\pm 0.07}$	$1.0_{\pm 0.0}$	$0.88_{\pm 0.04}$
	4	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.43_{\pm 0.05}$	$0.47_{\pm 0.12}$	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.23_{\pm 0.05}$	$1.0_{\pm 0.0}$	$0.81_{\pm 0.05}$
	8	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.63_{\pm 0.03}$	$0.7_{\pm 0.08}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.23_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.85_{\pm 0.05}$
	16	$1.0_{\pm 0.0}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.53_{\pm 0.03}$	$0.77_{\pm 0.07}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.83_{\pm 0.06}$
EVA + DoRA	2	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.8_{\pm 0.08}$	$0.97_{\pm 0.03}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.43_{\pm 0.12}$	$1.0_{\pm 0.0}$	$0.92_{\pm 0.03}$
	4	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.8_{\pm 0.05}$	$0.93_{\pm 0.03}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.63_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.94_{\pm 0.02}$
	8	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.63_{\pm 0.19}$	$0.87_{\pm 0.07}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.57_{\pm 0.03}$	$1.0_{\pm 0.0}$	$0.91_{\pm 0.04}$
	16	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.67_{\pm 0.2}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$1.0_{\pm 0.0}$	$0.5_{\pm 0.16}$	$1.0_{\pm 0.0}$	$0.92_{\pm 0.04}$

F Discussion

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), Linear Discriminant Analysis (Fisher, 1936, LDA). While Kernel-PCA can account for non-linearities in the data, it scales with the number of datapoints, i.e., in our setting we perform PCA on minibatches of sequences. Therefore, the number of datapoints grows fast with each minibatch, making Kernel-PCA infeasible. LDA projects the data onto a subspace that maximizes linear separability between classes. Such an initialization scheme is particularly interesting for classification tasks like GLUE or VTAB-1K. However, we observed on the GLUE tasks that the columns of the LDA projection matrix never converges during the initial computation phase.

Additional latency of SVD EVA leads to significant performance improvements over LoRA, but introduces additional latency in the beginning of training for computing the data-driven initialization. We found that this process consistently converges after a few minibatches across all tasks. Further, it

does not require backpropagation through the model compared to standard LoRA fine-tuning. While storing components requires slightly more memory, this can entirely be offloaded to CPU, and thus, does not result in additional GPU memory requirements. We found that in practice, there is no considerable difference between runtimes of LoRA and EVA.

What method performs well on which tasks? We conducted a broad range of experiments and found that EVA improves upon LoRA and competitors on tasks that are out-of-distribution. Further, it also resulted in significant improvements on some in-domain data. Throughout all of our experiments, we observed that EVA is the most stable method and consistently improves average scores across tasks versus other state-of-the-art PEFT methods. We also observed that DoRA can significantly improve upon LoRA as it did on the RL tasks and on certain datasets in VTAB-1K, and initializing DoRA with EVA leads to further improvements, especially on out-of-distribution tasks. Therefore, EVA advances the state-of-the-art among other PEFT competitors.

G Related Work

LoRA variants The advent of LoRA (Hu et al., 2022) has sparked widespread interest in leveraging low-rank decompositions for fine-tuning large models 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., 2024; Zhang et al., 2023a; Hayou et al., 2024b; Chavan et al., 2023). The most similar variants to EVA are AdaLoRA (Zhang et al., 2023a) and PiSSA (Meng et al., 2024). AdaLoRA adaptively allocates ranks for the introduced LoRA matrices during fine-tuning. In contrast, the data-driven initialization allows EVA to redistribute the ranks for each LoRA matrix at the beginning of training after the first few mini-batches. PiSSA initializes the LoRA matrix $\boldsymbol{A}$ via the top singular vectors of the pr-trained weight matrices. Contrary, EVA initializes $\boldsymbol{A}$ via the right singular-vectors of activation vectors and is therefore data-driven. Since EVA mostly constitutes an effective initialization, it can be readily plugged into most LoRA variants such as DoRA (Liu et al., 2024), or ELoRA (Kopiczko et al., 2024).

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, our initialization does not require any gradient update steps, therefore it is much more efficient. Similarly, Nikdan et al. (2024) uses a warm-up stage of LoRA fine-tuning and use gradients with respect to LoRA weights to initialize a sparse matrix for sparse adaptation (Sung et al., 2021) in combination with LoRA. Alternatively, Babakniya et al. (2023) initializes LoRA matrices with SVD on the original weight matrices after a few steps of full-finetuning for federated learning that usually comes with heterogeneous data for each user. Finally, 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, instead of relying on components of the pre-trained weights. Similar initialization schemes were proposed by Mishkin & Matas (2016); Krähenbühl et al. (2016) for training deep networks from scratch.

Increasing efficiency of LoRA Several works have investigated how to further break down the complexity of LoRA for fine-tuning FMs. Kopiczko et al. (2024) decrease the memory complexity of LoRA by initializing both $\boldsymbol{A}$ and $\boldsymbol{B}$ at random and keeping them 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 fruitful avenue is quantization (Dettmers et al., 2022), which has been successfully applied to LoRA matrices (Dettmers et al., 2023). More recent LoRA variants (Nikdan et al., 2024; Valipour et al., 2023) also provide quantized versions. It has also been shown that proper initialization for quantization results in improved fine-tuning performance (Li et al., 2023).

Alleviating the low-rank constraint The weight matrices introduced in LoRA are constrained by a low rank. Many recent works have investigated whether this constraint can be alleviated by merely considering gradient updates to be low-rank, while updating the full-rank weight matrices. To this end, Zi et al. (2023) leverage the delta between subsequent LoRA update steps to update the pre-trained weights. Hao et al. (2024) investigates the dynamics of LoRA and uses random matrices

to project the gradient into lower dimensional space. Similarly, Zhao et al. (2024) and Lialin et al. (2023) explored large scale pre-training by using low rank gradient updates. All of these works aim at keeping the original weights of full rank, however, it is not yet clear whether this is a necessary requirement for effective fine-tuning.