LoRA vs Full Fine-tuning: An Illusion of Equivalence

Reece Shuttleworth **Jacob Andreas** **Antonio Torralba** **Pratyusha Sharma**
MIT CSAIL
{rshuttle, jda, torralba, pratyusha}@mit.edu

Abstract

Fine-tuning is a crucial paradigm for adapting pre-trained large language models to downstream tasks. Recently, methods like Low-Rank Adaptation (LoRA) have been shown to effectively fine-tune LLMs with an extreme reduction in trainable parameters. But, *are their learned solutions really equivalent?* We study how LoRA and full-finetuning change pre-trained models by analyzing the model's weight matrices through the lens of their spectral properties. We find that LoRA and full fine-tuning yield weight matrices whose singular value decompositions exhibit very different structure: weight matrices trained with LoRA have new, high-ranking singular vectors, which we call *intruder dimensions*, while those trained with full fine-tuning do not. Further, we extend the finding that LoRA forgets less than full fine-tuning and find its forgetting is vastly localized to the intruder dimension – by causally intervening on the intruder dimensions by changing their associated singular values post-fine-tuning, we show that they cause forgetting. Moreover, scaling them down significantly improves modeling of the pre-training distribution with a minimal drop in downstream task performance. Given this, we should expect accumulating intruder dimensions to be harmful and lead to more forgetting. This will be amplified during continual learning because of sequentially fine-tuning, and we show that LoRA models do accumulate intruder dimensions here tend to perform worse in this setting, emphasizing the practicality of our findings.

1 Introduction

Adapting large, pre-trained models to downstream tasks via fine-tuning is a computation- and data-efficient way to create domain-specific models for a variety of tasks. The simplest approach is to fine-tune all parameters of the pre-trained model on downstream task data [Devlin et al., 2019, Ouyang et al., 2022]. However, as pre-trained models grow larger, full fine-tuning becomes increasingly challenging and expensive. Recently, parameter-efficient fine-tuning (PEFT) methods, especially low-rank adaptation (LoRA; Hu et al., 2021), have been shown to enable fine-tuning with only a fraction of the trainable parameters. **While LoRA can match full fine-tuning performance, are the solutions learned by the two methods similar?**

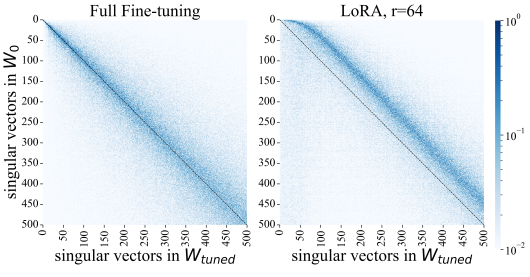


Figure 1: **LoRA and full fine-tuning update the parameter space differently.** Similarity matricies of pre- and post-fine-tuning singular vectors for LLaMA2-7B that characterize the spectral differences introduced during fine-tuning. Full fine-tuning retains most of the pre-training structure, while LoRA has a diagonal shift. Color shows cosine similarity.

While full fine-tuning treats every parameter as trainable, LoRA treats the learned update to a weight matrix as the product of two low-rank matrices [Hu et al., 2021]. While this parameterization is empirically effective, a principled explanation of

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

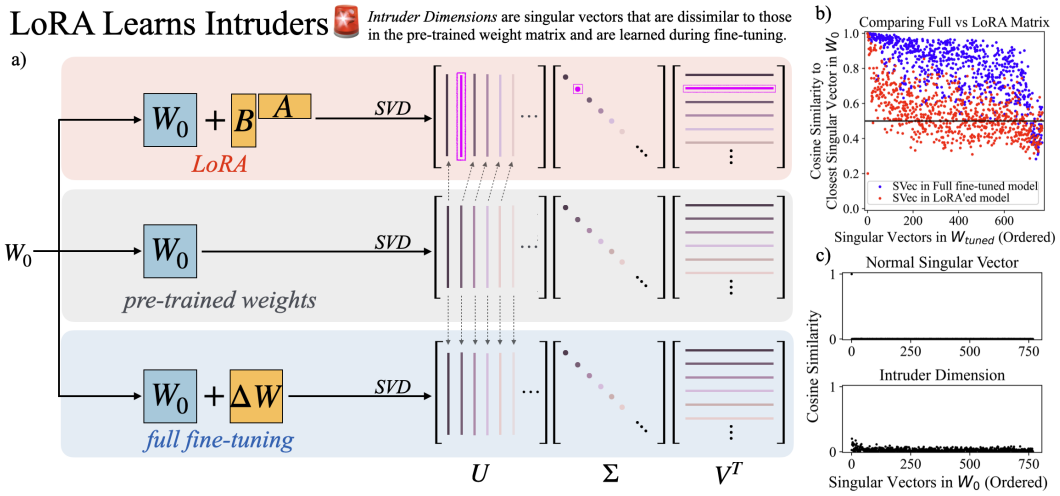


Figure 2: **Characterizing structural differences between solutions learnt by LoRA & Full Fine-tuning. a)** We measure the changes to the SVD of the pre-trained weights made during fine-tuning. We observe *intruder dimensions* introduced by LoRA in top ranking singular vectors but not by full fine-tuning. **b)** Comparing a matrix fine-tuned with full fine-tuning or LoRA. **c)** The intruder dimension shows near-zero absolute cosine similarity with all pre-trained singular vectors, in contrast to other singular vectors of the finetuned matrix.

the mechanism by which it matches the full fine-tuning performance has remained elusive. One explanation is offered by the *intrinsic dimension hypothesis* [Li et al., 2018, Aghajanyan et al., 2021], which posits that the update learned via fine-tuning has a low intrinsic rank, suggesting that LoRA might recover an approximately equivalent solution to full fine-tuning. However, prior work has observed differences in the ability of LoRA and full fine-tuning to independently change the angle and magnitude with which a neuron transforms its input [Liu et al., 2024]. Moreover, other work has also observed that LoRA has difficulty matching the performance of full fine-tuning on difficult tasks like code generation [Biderman et al., 2024, Zhuo et al., 2024] and long-form text generation [Ivison et al., 2023]. Therefore, it is unclear if these findings indicate a limit in LoRA's ability to fit to a specific downstream task, or if these methods learn inherently different solutions.

In this paper, we show that full fine-tuning and LoRA learn different solutions with characteristic differences in their spectral properties (as shown in Fig. 1 for LLaMA2-7B [Touvron et al., 2023b]) and that these spectral differences are causally related to different model behaviors. We observe:

1. **LoRA and full fine-tuning produce structurally different parameter updates, characterized by the existence of *intruder dimensions* in weight matrices tuned by LoRA**. Intruder dimensions are singular vectors with large associated singular values that are very dissimilar to the singular vectors in the pre-trained weight matrix. In contrast, fully fine-tuned models remain spectrally similar to the pre-trained model and do not contain intruder dimensions.

2. **LoRA forgets less than full fine-tuning...but not always.** We extend the findings of Biderman et al. [2024] that LoRA forgets less to the case *even* when there is equal fine-tuning performance between LoRA and full fine-tuning, showing that a difference in fit is not simply the cause of this finding but rather is inherent to these methods. However, this is not always the case: despite nearly identical fine-tuning task accuracies, we show that different selections of LoRA alpha and learning rate lead to starkly different generalization behaviors, even leading to LoRA forgetting more than full fine-tuning. We also find that models with the best generalization for each of these hyperparameter settings have the fewest intruder dimensions.

3. **Intruder dimensions cause forgetting of the pre-training distribution.** Scaling down the associated singular values of high-ranking intruder dimensions leads to a large drop in loss on the pre-training distribution (forgetting) but only a minimal drop in test performance. The drop in forgetting we observe when scaling down singular vectors is unique to intruder dimensions and indicates that they interfere with the pre-trained language modeling ability of these models. Given this finding, we should expect accumulating intruder dimensions to be harmful and lead to more forgetting. To amplify this accumulation and examine its effect, we fine-tune in a continual learning

setting (sequentially fine-tuning on many tasks) and show that LoRA models do indeed tend to forget more on previously learned tasks in this setting, providing additional support for our findings.

2 Background & Related Work

Methods for fine-tuning. Pre-trained language models offer a foundation for downstream applications, eliminating the need to train from scratch [Ouyang et al., 2022, Devlin et al., 2019]. Full fine-tuning, in which every parameter of a pre-trained model is updated, is commonly used [Devlin et al., 2019, Liu et al., 2019]. Low Rank Adaptation (LoRA; Hu et al., 2021), which represents the update to the weights as a product of two low-rank matrices, reduces computation and memory requirements relative to full fine-tuning. Past work has shown that LoRA matches full fine-tuning performance for tasks like sequence classification [Hu et al., 2021], instruction tuning [Dettmers et al., 2023, Ghosh et al., 2024], and chat [Dettmers et al., 2023]. Other work has shown a gap in performance on harder tasks like code generation [Biderman et al., 2024, Zhuo et al., 2024]. We focus our investigation on both cases to ensure our findings generalize to all use cases.

LoRA, formally. Given a pre-trained weight matrix $W_0 \in \mathbb{R}^{m \times n}$, full fine-tuning treats the learned matrix update as $\Delta W \in \mathbb{R}^{m \times n}$. Instead, LoRA decomposes ΔW into a product of two matrices such that $\Delta W = BA$, where $B \in \mathbb{R}^{m \times r}$, $A \in \mathbb{R}^{r \times n}$, and where the rank r is generally $r \ll min(m, n)$. During prediction,

$$Y = W_{tuned}X = (W_0 + \frac{\alpha}{r}BA)X \ .$$

B is initialized to zero, and A sampled from an isotropic Gaussian. All parameters in B and A are trained. From this we can see that while full fine-tuning has mn trainable parameters per weight matrix, LoRA only has $mr + rn$. See Appendix F for derivation of LoRA adapter gradients.

LoRA Variants. Many variations of LoRA exist. Methods improve LoRA's performance or memory-efficiency by initializing with the principal [Meng et al., 2024] or minor [Wang et al., 2024] components of the underlying weight matrix, training with quantization [Dettmers et al., 2023], adaptively allocating different ranks [Zhang et al., 2023], or sequentially training multiple LoRAs [Xia et al., 2024]. Other methods propose similar but alternative architectures [Liu et al., 2024, Kopiczko et al., 2024, Koohpayegani et al., 2024]. Other work has also proposed low rank manipulations to the activations instead of the weights [Wu et al., 2024]. Although the primary focus of our study is on the original LoRA setup [Hu et al., 2021], we also study a few LoRA variants (Appendix P). While we leave a rigorous analysis of all possible variants to future work, our preliminary experiments show that our findings generalize to several variants. Additionally, we also demonstrate the robustness of our findings across a range of LoRA hyperparameter settings (Appendices B.4, N, O).

Analysis of Solutions. The intrinsic dimension measure [Li et al., 2018] was used by Aghajanyan et al. [2021] to argue that the fine-tuning update for a pre-trained LLM has low intrinsic rank, explaining why only a small number of trainable parameters are necessary to reach 90% of full fine-tuning performance. This finding motivated Hu et al. [2021] to hypothesize that LoRA works because solutions of low intrinsic rank exist. But to our knowledge, no past work has compared the rank (or other properties of weight matrices) between LoRA and full-fine tuning on tasks where they are matched in performance. While Liu et al. [2024] showed that LoRA has difficulty changing directional and magnitude components of a neuron independently, it is unclear if this difference is due to an inability of LoRA to fit as well as full fine-tuning to the adaptation task.

Relation to Biderman et al. [2024]. Recent work comparing LoRA to full fine-tuning has found that LoRA forgets less when fine-tuned on math and code [Biderman et al., 2024] and more closely resembles the pre-trained model [Ghosh et al., 2024]. We extend the findings of Biderman et al. [2024] to the case when there is equal fine-tuning performance between LoRA and full fine-tuning, showing that a difference in fit to the fine-tuning task is not simply the cause of this finding but rather is inherent to these methods.

Singular Value Decomposition. The SVD decomposes a matrix $M \in \mathbb{R}^{m \times n}$ such that $M = U\Sigma V^T$, where $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ have orthonormal columns representing the singular vectors of M and $\Sigma \in \mathbb{R}^{m \times n}$ is a diagonal matrix containing the singular values of M. U and V^T represent rotations that matrix M performs, while Σ represents scaling along those axes. Singular vectors, ordered by singular values, reveal a matrix's most important axes of transformation.

3 Structural Differences

Inspired by Sharma et al. [2024]'s finding that the singular value decomposition (SVD, Klema and Laub, 1980) can be used to selectively prune singular vectors to improve model performance, this paper adopts the SVD of neural network parameters as a lens for understanding the changes that fine-tuning makes to pre-trained weights. Understanding how these dimensions change can give us insight into how a particular fine-tuning method changes the pre-trained model. In particular, we study how well singular vectors in weight matrices fine-tuned with LoRA or full fine-tuning map to singular vectors in the pre-trained weights (using cosine similarity).

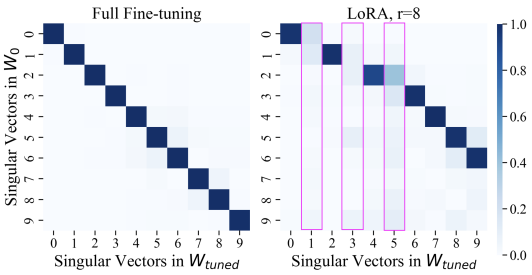


Figure 3: **LoRA and full fine-tuning learn distinct structural solutions.** LoRA introduces *intruder dimensions* (represented by outlined columns).

Visually, we observe in Fig. 2(b) that LoRA and full fine-tuning's singular vectors have very different similarities to the pre-trained singular vectors: singular vectors of models fine-tuned with LoRA appear to have, on average, much lower cosine similarity to pre-trained singular vectors in comparison to full fine-tuning. Interestingly, in LoRA fine-tuned models, we also observe the presence of high ranking singular vectors with very low cosine similarity to any pre-trained singular vector.[1] In Fig. 2(c), we show the difference between these vectors with low cosine similarity to the pre-trained singular vectors and normal singular vectors from the fine-tuned weights. This "new" dimension can be seen in Fig. 2(b) as the lone red dot in the bottom left corner. We name these "new" dimensions *intruder dimensions*, which we define formally as follows:

Definition 3.1. A singular vector y_j from the fine-tuned weight matrix W_{tuned} is an **intruder dimension** if and only if $\max_i(cos(y_j, x_i)) < \epsilon$, where ϵ is a similarity threshold and x_i are the singular vectors of W_0.

Examples of intruder dimensions may be seen in Fig. 3. Here, we plot the similarity matrix between the top 10 singular vectors (ranked by singular value) in the pre-trained and fine-tuned matrices. While full fine-tuning appears to have a clear one-to-one mapping, LoRA appears to have its mapping shifted by "blank" columns (outlined in magenta): these are intruder dimensions, with low cosine similarity to every pre-trained singular vector. A zoomed out version of this plot can be seen in Fig. 1, in which we see an off diagonal shift due to intruder dimensions.

Algorithm 1 Finding intruder dimensions.

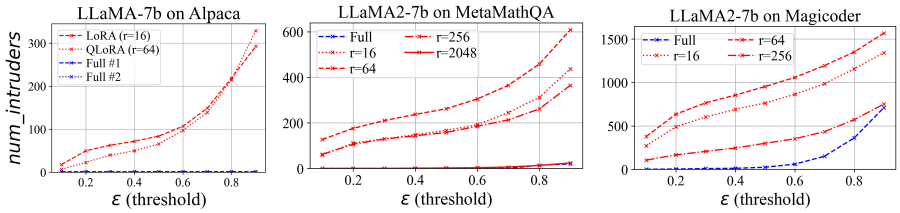
Require: Pre-trained weights W_0, fine-tuned weights W_t, cosine similarity threshold ϵ, # of fine-tuned singular vectors to examine k.

$[U_0, \Sigma_0, V_0^\top] \leftarrow \text{SVD}(W_0)$
$[U_t, \Sigma_t, V_t^\top] \leftarrow \text{SVD}(W_{\text{tuned}})$
$\text{n\_intruders} \leftarrow 0$
$n \leftarrow$ # of pre-trained singular vectors
for $j \leftarrow 1$ **to** k **do**
  if $\forall i \leq n : \cos\big(U_0[i], U_t[j]\big) < \epsilon$ **then**
    $\text{n\_intruders} \leftarrow \text{n\_intruders} + 1$
  end if
end for
return n_intruders

It is important to note that in the case of full fine-tuning, the singular vectors that map to a pre-trained singular vector with high cosine similarity also have similar singular values. From these initial measurements, it appears that LoRA and full fine-tuning have structural differences in the changes they make to the pre-trained weights: while full fine-tuning appears to make small changes to the existing singular vectors and singular values, LoRA introduces new singular vectors that have a large contribution to the norm of the updated parameter matrix.

Our Models. We study LLaMA2-7B [Touvron et al., 2023b] and RoBERTa-base [Liu et al., 2019]. RoBERTa-base is a pre-trained encoder-only language model and we fine-tune it on six different sequence classification tasks. See Appendix C.3 for fine-tuning details. LLaMA2-7B is a pre-trained decoder-only language model, and we study it when fine-tuned on either code or math. These checkpoints are provided by Biderman et al. [2024]. We also study LLaMA-7B [Touvron et al.,

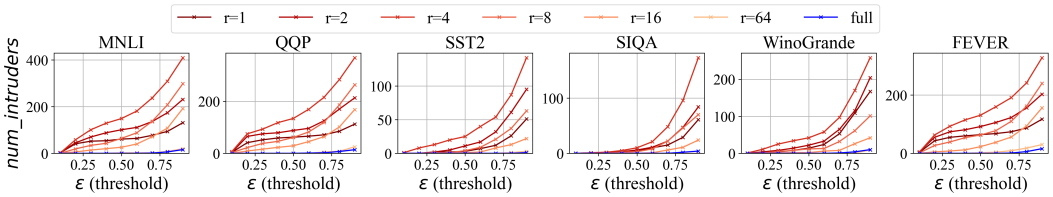
[1] Recall that in high dimensions, a vector can have low cosine similarity to a set of orthogonal vectors that span a space; see Appendix E for discussion.



(a) LLaMA-7B fine-tuned on Alpaca.

(b) LLaMA2-7B fine-tuned on MetaMathQA.

(c) LLaMA2-7B fine-tuned on Magicoder-Evol-Instruct.



(d) Number of intruder dimensions in RoBERTa models fine-tuned on 6 different tasks.

Figure 4: **LoRA has intruder dimensions, whereas full fine-tuning does not.** Here, we set $k = 10$ and measure the impact of ϵ on the number of intruder dimensions measured. LoRA introduces many intruder dimensions in the top 10 ranked singular vectors, while full fine-tuning does not. Numbers are reported are the sums across the entire model.

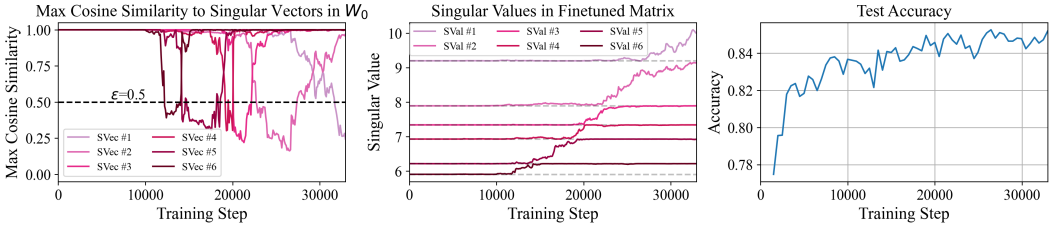


Figure 5: **Evolution of an intruder dimension across training steps.** *(Left)* Intruder dimensions, and their rank, in a LoRA fine-tuned weight matrix during fine-tuning. *(Middle)* Their associated singular values, which shows that the singular value associated with the intruder dimension increases. *(Right)* Test accuracy across training steps.

2023a] models fine-tuned on instruction following. See Appendix L for more details about these models. Importantly, these LLaMA models span math, code, and chat, which are considerably harder than sequence classification tasks. This ensures wide coverage of LoRA use cases.

Our Method. To calculate the number of intruder dimensions in a specific weight matrix, we use Algorithm. 1. In it, we first compute the SVD of both the pre-trained and resulting LoRA and full fine-tuned weights. Then, for each of the top k highest-ranking singular vectors, we measure its maximum cosine similarity with all of the pre-trained singular vectors. If this maximum cosine similarity is less than some threshold ϵ, we classify this singular vector as an intruder dimension. Note that both k, the number of fine-tuned singular vectors to examine, and ϵ, the cosine similarity threshold, are hyperparameters; we verify the robustness of our findings for a wide range of ϵ and k values in Fig. 4 and Fig. 15 respectively. To determine the number of intruder dimensions in a specific model, we run this algorithm for each weight matrix in the model and sum the total.

LoRA fine-tuned models contain high-ranking intruder dimensions while fully fine-tuned models do not. To characterize the differences in fine-tuning methods, we first evaluate the differences in the total number of intruder dimensions in the top 10 highest-ranking singular vectors ($k = 10$). We repeat this procedure for a range of ϵ values, our cosine similarity threshold. The results are presented in Fig. 4. For LLaMA2-7B, we find that models trained with LoRA contain intruder dimensions for ranks at least as high as $r \leq 256$. For RoBERTa, we consistently observe intruder dimensions for rank $r \leq 16$, even for low values of ϵ. Interestingly, we observe that fully fine-tuned models, for all model sizes, almost *never* contain intruder dimensions in their top 10 singular vectors, even for epsilon values of about 0.6 to 0.9. This means that full fine-tuning makes smaller changes to the same

set of high contribution pre-trained singular vectors, rather than introducing new singular vectors like LoRA. Importantly, the number of intruder dimensions appears to drop as rank increases past a certain threshold, suggesting that the low-rank nature, as well as the update rule of LoRA, induces them to occur. This is underscored by the $r = 2048$ case of LLaMA2-7B fine-tuned on math (Fig 4b), which does not have intruder dimensions and instead has a very similar curve to full fine-tuning. As rank increases past a threshold and LoRA begins to resemble a high rank update, intruder dimensions begin to disappear.

LoRA variants have intruder dimensions. We examine 4 other LoRA variants to ensure that our findings do not only apply to vanilla LoRA. We examine AdaLoRA [Zhang et al., 2023], LoRA+ [Hayou et al., 2024], PiSSA [Meng et al., 2024], and VeRA [Kopiczko et al., 2024]. In all of these cases, we find intruder dimensions with similar characteristics to vanilla LoRA (see Fig. 23). This shows our findings hold to other variants. For more discussion about these methods, see Appendix P.

Intruder dimensions are distributed across both high and low singular values. We examine the extent to which intruder dimensions exist throughout the entire weight matrix and how they are distributed. To do this, we hold ϵ fixed and measure the number of intruder dimensions while varying the proportion of the fine-tuned singular vectors that we examine (Appendix J, Fig. 15). Here, we can see that LoRA consistently has more intruder dimensions than full fine-tuning, regardless of what fraction of the singular values we examine. See Appendix J for more discussion.

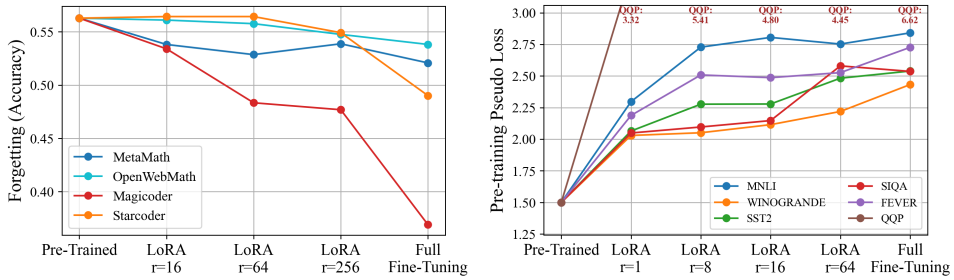
Intruder dimensions increase in magnitude and change in direction as fine-tuning progresses. To further understand how a particular intruder dimension is introduced during fine-tuning with LoRA, we measure the maximum cosine similarity between the top individual fine-tuned singular vectors and all the pre-trained singular vectors across many intermediate steps in the fine-tuning process, as seen in Fig. 5 (*left*). In parallel, we track changes in their associated singular values as seen in Fig. 5 (*middle*). As is evident from the graphs, intruder dimensions appear to gradually increase their "rank" (*left*) as their singular value is increased (*middle*) while simultaneously changing in direction too as training progresses.

Additional empirical observations. 1. We find that the random seed used by LoRA to initialize its adapters plays no role in the resulting structure (see Appendix O). **2.** We observe that the total number of intruder dimensions increases linearly with respect to the size of the fine-tuning dataset up to a certain point before saturating (Appendix K). **3.** We study the effective rank of these fine-tuning updates (Appendix I). However, we find that this measure does not suffice to explain the behavioral differences we observe in LoRA and full fine-tuning. Also, it is important to note that even if it had, its global nature would prevent the precise examinations we conduct in future sections, like in Fig. 8.

Experimental and theoretical justification for why intruder dimensions occur. It is important to note that intruder dimensions are an empirical observation of LoRA. We find that a variety of factors play a role in the introduction of intruder dimensions. In the next section, we present results that suggest that learning rate and LoRA's α contribute to intruder dimensions. In the appendix, we present findings that suggest that tuning the B matrix only leads to fewer intruder dimensions (B.5), and demonstrate how the addition of orthogonal vectors to the pre-trained weight matrix models the introduction of intruder dimensions well (B.2).

4 Model Differences: Forgetting and Out-of-Distribution Generalization

LoRA forgets less. We measure the change in out of distribution performance (forgetting) induced by fine-tuning. For LLaMA2-7B, we follow Biderman et al. [2024] and measure forgetting as the average score on Hellaswag [Zellers et al., 2019], WinoGrande [Sakaguchi et al., 2021], Arc-Challenge [Clark et al., 2018]. For RoBERTa-base, we measure its "pseudo-loss", which is analogous to language modelling loss for encoder-only models, as described by Salazar et al. [2020] on a sample of its pre-training dataset (as described by Liu et al. [2019]). Going forward, we refer to these values as "forgetting" and report them in Fig. 6. We observe that across all tasks, full fine-tuning forgets more of its pre-training language modeling ability in comparison to LoRA. Importantly, all our RoBERTa-base models fine-tune to equivalent accuracy on the downstream task (Table 2). This extends the finding that LoRA forgets less [Biderman et al., 2024] to the case where LoRA and full fine-tuning have equal fit, showing that LoRA forgetting less is not simply a function of it underfitting the fine-tuning task in comparison to full fine-tuning (like in Biderman et al. [2024]), but rather a characteristic of LoRA itself.



(a) LLaMA2-7B. (b) RoBERTa-base.

Figure 6: **LoRA forgets less, even with same fit to fine-tuning task.** For LLaMA2-7B, forgetting is measured on unrelated tasks, as described in Biderman et al. [2024]. For RoBERTa, Pseudo loss on a sample of its pre-training distribution measured as described by Salazar et al. [2020]. In both, LoRA forgets less than full fine-tuning.

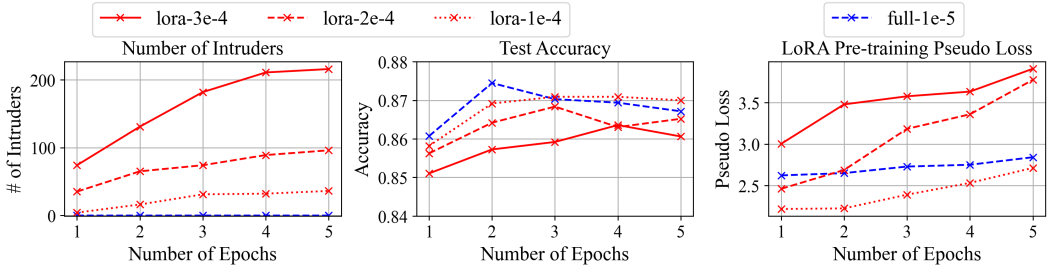
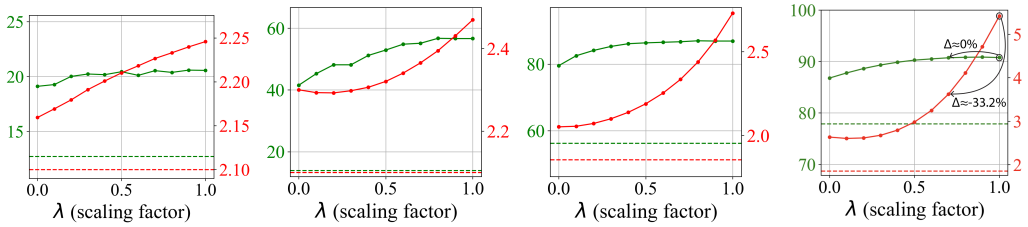


Figure 7: **As training progresses, models with growing amount of intruder dimensions continue to forget more, despite non-increasing test performance.** We also measure a strong correlation(ρ = 0.971, p-value $\ll 0.001$) between number of intruder dimensions and pre-training pseudo loss. Bigger learning rates lead to more intruder dimensions and forgetting.

LoRA α impacts generalization and intruder dimensions. For our experiments, we use the commonly used $\alpha = 2r$ [Biderman et al., 2024] as well as $\alpha = 8$ [Hu et al., 2021]. For both settings of α, models obtain equivalent performance on the target task (Tables 1 & 2). However, when $\alpha = 8$, all ranks of LoRA—even very large ones—exhibit intruder dimensions (Fig. 19a), have a much smaller effective rank than when $\alpha = 2r$ (Appendix I), and have much worse generalization (more forgetting, Fig. 20). Models trained with $\alpha = 2r$ have fewer intruder dimensions and generalize better. This provides additional evidence highlighting the importance of using $\alpha = 2r$ [Kalajdzievski, 2023, Biderman et al., 2024], particularly for higher ranks of LoRA.

An increase in intruder dimensions leads to an increase in forgetting. We do a learning rate sweep for RoBERTa-base with LoRA $r = 8$ on MNLI to observe its impact. Across epochs, we measure the number of intruder dimensions, test accuracy, and forgetting (pre-training loss). We report the results of these models and our baseline full fine-tuning model in Fig. 7. We observe that *across* and *within* training runs, as the number of intruder dimensions increase, forgetting (meaning worse generalization) also increases. Test accuracy has no such relation. Separately, we see that for large learning rates with many intruder dimensions, LoRA models *forget more* than full fine-tuning. This shows that while LoRA in general does forget less than full fine-tuning, it is not a guarantee.

Intruder dimensions strongly correlate with forgetting. When we measure the Spearman correlation between the number of intruder dimensions with forgetting in Fig. 7, we find an extremely strong fit($\rho = 0.971$, p-value $\ll 0.001$). When measuring the same for our LLaMA2-7B models across training epochs (Fig. 11), we still find a strong and still statistically significant relationship ($\rho = 0.59$, p-value $= 0.0006$). In contrast, when measuring the correlation between intruder dimensions and test accuracy, we find no statistically significant relationship: for RoBERTa, we measure $\rho = -0.3381$ & p-value $= 0.218$. For LLaMA2-7B, we measure $\rho = -0.3178$ & p-value $= 0.0869$. See Appendix G for more information. These results suggest that intruder dimensions are clearly linked with forgetting but *are not necessary for performance.* We examine this claim and whether this relationship is causal in the next section.



(a) LLaMA2-7B on code with r=16.

(b) LLaMA2-7B on math with r=16.

(c) RoBERTa-base on MNLI with r=8.

(d) RoBERTa-base on QQP with r=8.

Figure 8: **Scaling down intruder dimensions in fine-tuned models reduces forgetting but not performance.** We scale the top intruder dimension in each matrix such that $W = W_0 + \Delta W + (\lambda - 1)u_i\sigma_i v_i^T$. Lines represent forgetting (red) and learning (green). Dotted lines represent pre-trained baselines. **Axis Labels:** Green: Test Accuracy (%). Red: Pre-training Loss (Forgetting).

5 Intruder Dimensions Cause Forgetting

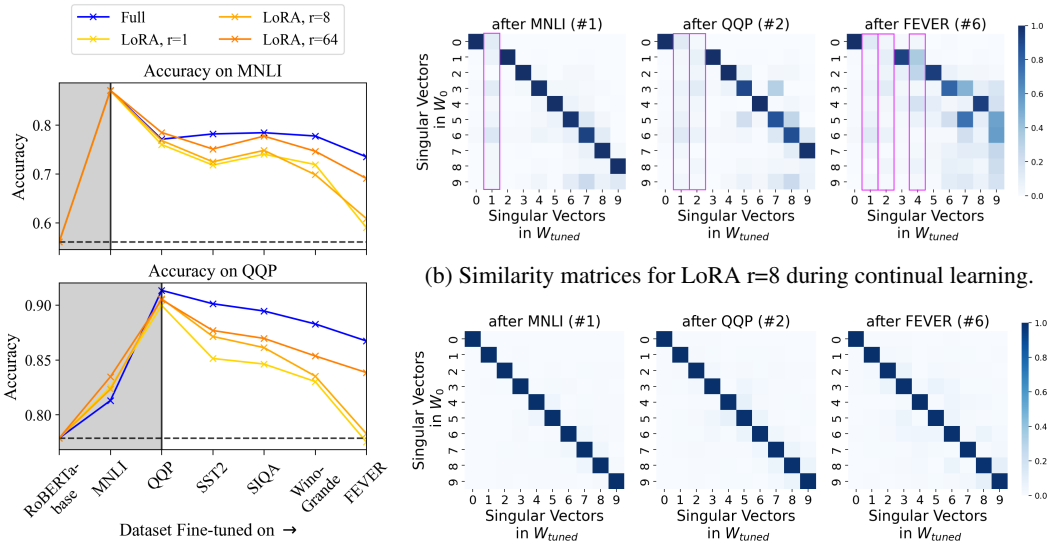
Previously, we observed that the number of intruder dimensions correlates strongly with forgetting. *Do intruder dimensions cause this forgetting?*

Scaling the magnitude of intruder dimensions. To test if intruder dimensions *cause* increased forgetting, we must intervene on intruder dimensions and see the impact. We do this by finding the highest ranked (by singular value) intruder dimension in each weight matrix and scale its contribution such that the new weight matrix is $W = W_0 + \Delta W + (\lambda - 1)u_i\sigma_i v_i^T$, where i is the index of the top intruder dimension ($\lambda = 0$ is removal and $\lambda = 1$ is no change). We sweep $\lambda's$ between 0 and 1, scale the intruder dimensions, and measure test accuracy and pre-training loss. For a comparison baseline, we select the neighbor of the intruder dimension to separately scale. See Fig. 8 for results (and Figs. 12& 13 for full results.)

Scaling down intruder dimensions reduces forgetting. In Fig. 8, we show that when we scale down the top intruder dimension of each weight matrix, we measure a significant reduction in forgetting (pre-training loss) while incurring a minimal drop in test accuracy. For all examples in Fig. 8, we observe that when using $\lambda = 0.7$ or 0.9, there is almost no impact on fine-tuning performance, while there is a large percentage drop in forgetting (See Tables 3&4). In one example for LLaMA2-7B fine-tuned on MetaMath with LoRA $r = 256$, we observe that scaling the top intruder dimension in each matrix with $\lambda = 0.3$ leads to a 0.1% drop in test accuracy and a 33.3% drop in the forgetting induced by fine-tuning. In another for RoBERTa-base fine-tuned on QQP, using $\lambda = 0.7$ leads to equivalent in test accuracy and a 33.2% reduction in the forgetting induced by fine-tuning. In certain scenarios, we even see test accuracy *improve* along with a drop in forgetting. If we instead increase their contribution ($\lambda > 1$), we observe more forgetting. Across the board, scaling down intruder dimensions seems to have little impact on test accuracy but a major impact on forgetting. This pattern is exclusive to intruder dimensions (Fig. 12): if we instead intervene on pre-trained singular vectors that have a similar singular value to the intruder dimension (ensuring similar contributions to matrix) and scale their magnitude down, we see that forgetting goes up. These results indicate that the forgetting observed in LoRA is caused by intruder dimensions interfering with the pre-trained language modeling capabilities. Moreover, the scale (singular value) of these intruder dimensions is not essential for the fine-tuning task performance. See Appendix H for further discussion.

Continual Learning Setup. To examine a practical example of how accumulating intruder dimensions, which cause forgetting, may impact performance, we study continual learning, since it requires learning and remembering across a range of tasks. To do this, we train RoBERTa sequentially on multiple tasks and measure performance as new tasks are learned. We use the same training recipe and datasets as before but now sequentially in the following dataset order: MNLI, QQP, SST-2, SIQA, WinoGrande, FEVER. After training on a certain dataset in the sequence, we merge the LoRA weights into the model and reinitialize the LoRA adapter before training on the next task. After training on a specific task, we test on all tasks by, for each task, separately retraining its classification head before testing on its test set. Results are shown in Fig. 9a.

Accumulating intruder dimensions hurts LoRA models during continual learning. In Fig. 9a, initially both LoRA and full fine-tuning train to equal performance (MNLI), which is consistent with



(a) Continual learning results.

(b) Similarity matrices for LoRA r=8 during continual learning.

(c) Similarity matrices for full fine-tuning during continual learning.

Figure 9: **Full fine-tuning is better than LoRA at continual learning because of accumulating intruder dimensions.** When sequentially training on six tasks, full fine-tuning retains performance better than LoRA. in Fig. 9a, horizontal dotted line indicates baseline pre-trained performance. Vertical solid line indicates when a specific dataset is fine-tuned on. Gray region represents performance before the model has been trained on that task. See Appendix M, Fig. 17 for more. In Figs. 9b&9c, we see that LoRA accumulates intruder dimensions across tasks and contributes to its degrading performance, whereas full fine-tuning does not.

our previous observations. However, we observe that all ranks of LoRA degrade much more rapidly than full fine-tuning. Low ranks of LoRA, which have the most intruder dimensions, degrade the most. We attribute this divergence from earlier results—where LoRA appeared to forget less—to the accumulation of intruder dimensions during continual learning, which drive forgetting. To show this, we visualize how intruders are added across tasks in Fig. 9b. Here, we see that each task adds its own intruder dimensions leading to a large amount of intruder dimensions upon the completion of the six task continual learning experiment. In contrast, in Fig. 9c, we see that full fine-tuning retains the pre-trained structure well justifying why full fine-tuning forgets less during continual learning.

Implications and prescriptions in fine-tuning. These findings suggest several implications and prescriptions during LoRA fine-tuning. We have shown that intruder dimensions drive forgetting, and therefore should be avoided when possible. Interestingly, this presents a data free model evaluation method to examine which model is most overfit to the fine-tuning task (forgotten the most): given two equally performing models on downstream test sets, you should select the one with fewer intruder dimensions. Intruder dimensions appear to be a necessary part of fine-tuning, but they can be mitigated. Further, these results show the danger of using LoRA during continual learning and justifies using many different adapters without combining them, like advocated in Sheng et al. [2024].

6 Conclusion

This paper describes the finding that LoRA and full fine-tuning update different parts of the parameter space resulting in distinct spectral properties: LoRA often introduces intruder dimensions—high-ranking singular vectors dissimilar to those in pre-trained weights. These structural differences persist across a series of ablations. Next, we find that models with fewer intruder dimensions exhibit better out-of-distribution generalization and forget less of the pretraining distribution. Last, we show that intruder dimensions *cause* increased forgetting: We show that reducing the magnitude of high ranking intruder dimensions leads to minimal changes in test performance but a large drop in pretraining loss. We show that this is particularly relevant during continual training: even though LoRA forgets less than full fine-tuning after training on one task, sequentially training leads to an accumulation of intruder dimensions that causes more forgetting than full-finetuning.

Acknowledgements

We would like to thank Jacob Portes and Dan Biderman for corresponding with us and releasing their LLaMA-2 7B checkpoints for us to use. This enabled us to study a more comprehensive range of models. We would also like to thank Leshem Chosen, Lucas Hennigen, Han Guo, Vighnesh Subramaniam, Valerio Pepe, and the entire Language & Intelligence lab for their helpful feedback on this work. This research was supported in part by the National Science Foundation under grant IIS-2238240.

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, August 2021. URL https://aclanthology.org/2021.acl-long.568.

Loubna Ben Allal, Niklas Muennighoff, Logesh Kumar Umapathi, Ben Lipkin, and Leandro von Werra. A framework for the evaluation of code generation models. https://github.com/bigcode-project/bigcode-evaluation-harness, 2022.

Dan Biderman, Jose Gonzalez Ortiz, Jacob Portes, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John P. Cunningham. LoRA Learns Less and Forgets Less. Transactions on Machine Learning Research, 2024. URL https://arxiv.org/abs/2405.09673.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient Finetuning of Quantized LLMs. In *Advances in Neural Information Processing Systems*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, June 2019. URL https://aclanthology.org/N19-1423.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Sreyan Ghosh, Chandra Kiran Reddy Evuru, Sonal Kumar, Ramaneswaran S, Deepali Aneja, Zeyu Jin, Ramani Duraiswami, and Dinesh Manocha. A Closer Look at the Limitations of Instruction Tuning. In *Proceedings of the 41st International Conference on Machine Learning*. International Conference on Machine Learning, 2024. URL https://arxiv.org/abs/2402.05119.

Aaron Gokaslan and Vanya Cohen. OpenWebText Corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A Generic News Crawler and Extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223, March 2017. doi: 10.5281/zenodo.4120316.

Yongchang Hao, Yanshuai Cao, and Lili Mou. Flora: Low-Rank Adapters Are Secretly Gradient Compressors. In *Proceedings of the 41st International Conference on Machine Learning*. International Conference on Machine Learning, 2024. URL https://arxiv.org/abs/2402.03293.

Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient Low Rank Adaptation of Large Models, 2024. URL https://arxiv.org/abs/2402.12354.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. International Conference on Learning Representations, 2021.

Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. Camels in a Changing Climate: Enhancing LM Adaptation with Tulu 2, 2023. URL https://arxiv.org/abs/2311.10702.

Damjan Kalajdzievski. A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA, 2023. URL https://arxiv.org/abs/2312.03732.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

V. Klema and A. Laub. The singular value decomposition: Its computation and some applications. *IEEE Transactions on Automatic Control*, 25(2):164–176, 1980. doi: 10.1109/TAC.1980.1102314.

Soroush Abbasi Koohpayegani, KL Navaneet, Parsa Nooralinejad, Soheil Kolouri, and Hamed Pirsiavash. NOLA: Compressing LoRA using Linear Combination of Random Basis. International Conference on Learning Representations, 2024. URL https://arxiv.org/abs/2310.02556.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. VeRA: Vector-based Random Matrix Adaptation. International Conference on Learning Representations, 2024. URL https://arxiv.org/abs/2310.11454.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the Intrinsic Dimension of Objective Landscapes. International Conference on Learning Representations, 2018. URL https://arxiv.org/abs/1804.08838.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you!, 2023. URL https://arxiv.org/abs/2305.06161.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-Decomposed Low-Rank Adaptation. In *Proceedings of the 41st International Conference on Machine Learning*. International Conference on Machine Learning, 2024. URL https://arxiv.org/abs/2402.09353.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019. URL https://arxiv.org/abs/1907.11692.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. PEFT: State-of-the-art Parameter-Efficient Fine-Tuning methods. https://github.com/huggingface/peft, 2022.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal Singular Values and Singular Vectors Adaptation of Large Language Models, 2024. URL https://arxiv.org/abs/2404.02948.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text, 2023. URL https://arxiv.org/abs/2310.06786.

Olivier Roy and Martin Vetterli. The effective rank: A measure of effective dimensionality. In *2007 15th European Signal Processing Conference*, pages 606–610, 2007.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. Masked Language Model Scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.240. URL http://dx.doi.org/10.18653/v1/2020.acl-main.240.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense Reasoning about Social Interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454.

Pratyusha Sharma, Jordan T. Ash, and Dipendra Misra. The Truth is in There: Improving Reasoning in Language Models with Layer-Selective Rank Reduction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ozX92bu8VA.

Ying Sheng, Shiyi Cao, Dacheng Li, Coleman Hooper, Nicholas Lee, Shuo Yang, Christopher Chou, Banghua Zhu, Lianmin Zheng, Kurt Keutzer, Joseph E. Gonzalez, and Ion Stoica. S-lora: Serving thousands of concurrent lora adapters, 2024. URL https://arxiv.org/abs/2311.03285.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1170.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, 2023a. URL https://arxiv.org/abs/2302.13971.

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

Trieu H. Trinh and Quoc V. Le. A Simple Method for Commonsense Reasoning, 2019. URL https://arxiv.org/abs/1806.02847.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

Hanqing Wang, Yixia Li, Shuo Wang, Guanhua Chen, and Yun Chen. Milora: Harnessing minor singular components for parameter-efficient llm finetuning, 2024. URL https://arxiv.org/abs/2406.09044.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering Code Generation with OSS-Instruct. In *Proceedings of the 41st International Conference on Machine Learning*. International Conference on Machine Learning, 2024. URL https://arxiv.org/abs/2312.02120.

Adina Williams, Nikita Nangia, and Samuel Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101. URL https://aclanthology.org/N18-1101.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. ReFT: Representation Finetuning for Language Models, 2024. URL https://arxiv.org/abs/2404.03592.

Wenhan Xia, Chengwei Qin, and Elad Hazan. Chain of LoRA: Efficient Fine-tuning of Language Models via Residual Learning. In *Proceedings of the 41st International Conference on Machine Learning*. International Conference on Machine Learning, 2024. URL https://arxiv.org/abs/2401.04151.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng YU, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=N8N0hgNDRt.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=lq62uWRJjiY.

Jiacheng Zhu, Kristjan Greenewald, Kimia Nadjahi, Haitz Sáez de Ocáriz Borde, Rickard Brüel Gabrielsson, Leshem Choshen, Marzyeh Ghassemi, Mikhail Yurochkin, and Justin Solomon. Asymmetry in Low-Rank Adapters of Foundation Models. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024. URL https://openreview.net/forum?id=PHrrbfrMEl.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Terry Yue Zhuo, Armel Zebaze, Nitchakarn Suppattarachai, Leandro von Werra, Harm de Vries, Qian Liu, and Niklas Muennighoff. Astraios: Parameter-Efficient Instruction Tuning Code Large Language Models, 2024. URL https://arxiv.org/abs/2401.00788.

A Limitations

While we gather a diverse set of model (across sizes and types) and task types, it is always possible for findings to fail to hold for even larger models or in different scenarios. However, we have no reason to believe this is the case. Further, we point out that we lack the compute required to be able to fine-tune larger models.

B Why do intruder dimensions exist & can we alleviate them?

Here, we discuss possible causes of intruder dimensions.

B.1 What do intruder dimensions do?

Conjecture: Intruder dimensions, as high-ranking singular vectors, contribute significantly to the norm and stability of the parameter matrix. In contrast to pre-trained singular vectors that are learned from large pre-training corpora, LoRA introduces intruder dimensions learned solely from the smaller dataset of the fine-tuning task, which overpower the pre-trained vectors, as seen in the experiments so far. This suggests that these intruder dimensions are very task specific. On the other hand, full fine-tuning, while adapting just as effectively to the fine-tuning task, retains the spectral properties of the pre-trained model effectively. Our experiments that scale down intruder dimensions provide evidence for the claim that intruder dimensions are specialized to the fine-tuning task, since we observe that increasing the norm of an intruder dimension (using $\lambda > 1$) leads to little change in adaptation task performance but leads to a significant increase in forgetting (pre-training loss).

B.2 Adding a random vector

Adding an random vector to a pre-trained matrix introduces an intruder dimension: To help provide intuition about how new singular vectors in the SVD can be added by LoRA, we examine mathematical conditions that lead to their creation. Certainly, when comparing SVD$(W + \lambda vv^T)$ and SVD(W), where W are the pre-trained weights in $\mathbb{R}^{n\times n}$, v is a randomly sampled vector in $\mathbb{R}^n$, and λ is a scalar value greater than the largest singular value of W, we expect this update to create an intruder dimension (as v is nearly orthogonal to the existing singular vectors w.h.p.).

B.3 Differences in update rule

As described in Appendix F, LoRA and full fine-tuning have characteristically different update rules, even for the same training examples. We highlight that LoRA has gradients projected into a low-rank space [Hao et al., 2024], leading to conditions similar to the toy example in section B.2 above.

B.4 Impact of learning rate on intruder dimensions.

We study the impact learning rate has on intruder dimensions by sweeping a range of learning rates while keeping all other hyperparameters fixed for LoRA $r = 8$ and fine-tune on MNLI. Across training epochs, we report the number of intruder dimensions, test accuracy, and pre-training loss (Fig. 7). Across models, we see they have similar test accuracies after 5 epochs but very different numbers of intruder dimensions. We note that small learning rates do not converge as fast as the ones we tested and have difficulty reaching the maximum performance that larger learning rates are able to reach. We see that as we increase learning rate, the number of intruder dimensions also increases (left, Fig. 7). This illustrates a tradeoff in the selection of learning rate of LoRA: picking a larger learning rate may lead to faster convergence and potentially better test accuracy with more intruder dimensions and drift in overall language modeling performance, while smaller learning rates may lead to less drift but potentially lower test accuracy.

Because of this experiment, one may dispute our findings with the claim that they are due to our specific selection of hyperparameters. Therefore, we find it important to note that we adopt hyperparameters from prior literature [Hu et al., 2021], default settings in common machine learning

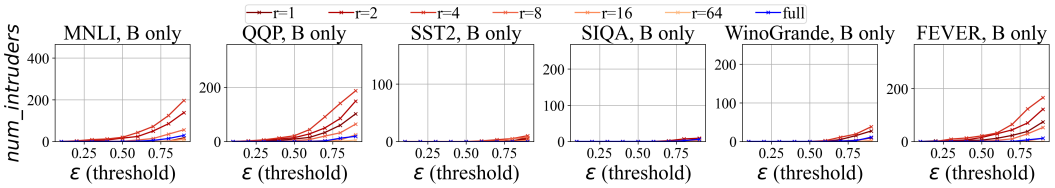


Figure 10: **Impact of only tuning B on the number of intruder dimensions.** We randomly initialize A such that it has singular values of 1, freeze it, and only train B. When we do this, we see a sharp reduction in high ranking intruder dimensions in comparison to those in normal LoRA (reported in Fig. 4d). Graphs for a specific dataset have the same range as Fig. 4d for easy comparison.

libraries[2], and also study externally trained open-sourced models. This means that our findings are a reflection of common practices and not due to a selection bias by us. Learning rates used by prior work were likely determined based on a variety of factors like speed of convergence and best resulting test accuracy and therefore selected large learning rates that still converge well but coincidentally result in intruder dimensions.

B.5 Matrix product parameterization of LoRA

Multiplying matrices together amplifies their spectral differences (their singular values) and in most cases leads to a lower effective rank. To test the impact of the product BA on the introduction of intruder dimensions, we randomly initialize A such that all its singular values are 1 and freeze it. We only tune B and keep the rest of our fine-tuning recipe the same. Comparing this with vanilla LoRA is fair because Zhu et al. [2024] found that tuning B is more impactful and important for generalization in comparison to A and Hao et al. [2024] showed that only tuning B effectively approximates LoRA. As we can see in Fig. 10, we see a sharp drop in the number of high ranking intruder dimensions when only tuning B in comparison to the vanilla LoRA case where we train A and B separately, as reported in Fig. 4. This suggests that the matrix product of LoRA is an important component in the introduction of intruder dimensions because of how it amplifies the spectral differences of B and A.

C Implementation Details

C.1 Evaluation details

We follow the precedence of Biderman et al. [2024] when evaluating LLaMA2-7B: When fine-tuned on code, we evaluate on HumanEval [Chen et al., 2021] using the bigcode-eval-harness [Ben Allal et al., 2022]. When fine-tuned on math, we evaluate on GSM8K [Cobbe et al., 2021] using the lm-eval-harness [Gao et al., 2024]. On both, we evaluate task forgetting by evaluating on Hellaswag, WinoGrande, and Arc-Challenge using the lm-eval-harness [Gao et al., 2024].

We measure language modeling loss for all our LLaMA2-7B models on a random sample of its pre-training data distribution, according to Touvron et al. [2023b]. We measure "pseudo-loss" for all our fine-tuned RoBERTa models on a random sample of the four datasets that RoBERTa used for pre-training(OpenWebText [Gokaslan and Cohen, 2019], CCNews [Hamborg et al., 2017], Stories [Trinh and Le, 2019], and bookcorpus [Zhu et al., 2015]) and weigh them proportionally to their contribution as described by Liu et al. [2019].

C.2 Compute Resources

All experiments were run on an internal, shared 8xA100-SXM4-80GB machine. All RoBERTa-base fine-tuning runs required a single A100 GPU. All evaluations and analyses also required a single A100 GPU. Many experiments were run sequentially due to need to share these computing resources. Due to these constraints, instead of fine-tuning our own LLaMA2-7B models, we use publicly released fine-tuned models. For more information on these models, see Section L. Each RoBERTa-base

[2]PEFT, the most popular LoRA library, use learning rates $\geq$ 1e-3 in their tutorials and states "With LoRA-like methods, you can afford to use a higher batch size and learning rate." [Mangrulkar et al., 2022].

fine-tune run takes at most 6 hours on a single GPU. Evaluating an arbitrary LLaMA2-7B model for both test accuracy and forgetting takes about 45 minutes on a single GPU.

C.3 RoBERTa fine-tuning details

We generally follow the procedure used by Hu et al. [2021]. For all models, we use a linear learning rate schedule with 0.06 linear warmup ratio and train for a maximum of 5 epochs with batch size 16. We use the Adam optimizer [Kingma and Ba, 2017] with no weight decay and a maximum sequence length of 512. We fine-tune all linear layers besides the embedding matrix. For full fine-tuning, we use a learning rate of 1e-5. For LoRA, we set $\alpha = 2r$, and train for all ranks in {1, 2, 4, 8, 16, 64}. We hold the "total learning rate of LoRA", which is $\alpha * \eta$, fixed as we sweep rank such that this product always equals 2.4e-3. We fine-tune these models to equivalent accuracy on their downstream task. We fine-tune on six sequence classification tasks: sentiment analysis [Socher et al., 2013], entailment [Williams et al., 2018], duplicate identification [Wang et al., 2019], fact verification [Thorne et al., 2018], and common sense reasoning [Sap et al., 2019, Sakaguchi et al., 2021].

D Model Accuracies

We report the accuracies that our RoBERTa models achieve in Table 1 and Table 2. Our main results are based on the models in Table 2.

Model	Type	MNLI	SST-2	QQP	WinoGrande	SIQA	FEVER
RoBERTa-base	Full	0.8745	0.9438	0.9152	0.6582	0.6499	0.6892
	r=1	0.8647	0.9358	0.9045	0.6251	0.672	0.6712
	r=2	0.8604	0.9415	0.9058	0.6172	0.6581	0.6673
	r=4	0.8607	0.9369	0.9079	0.6472	0.6505	0.6694
	r=8	0.8648	0.9438	0.9108	0.6417	0.6586	0.6582
	r=16	0.8604	0.9427	0.9095	0.6235	0.6853	0.663
	r=64	0.8671	0.9484	0.9117	0.6614	0.6638	0.6601

Table 1: Model accuracies on their given downstream task after fine-tuning for $\alpha = 8$.

Model	Type	MNLI	SST-2	QQP	WinoGrande	SIQA	FEVER
RoBERTa-base	Full	0.8745	0.9438	0.9152	0.6582	0.6499	0.6892
	r=1	0.8677	0.9415	0.9042	0.6275	0.6418	0.687
	r=2	0.869	0.945	0.9054	0.6551	0.6438	0.6822
	r=4	0.8698	0.9472	0.9089	0.6361	0.6602	0.6827
	r=8	0.8704	0.9472	0.9093	0.6346	0.6607	0.6928
	r=16	0.8739	0.9461	0.9093	0.6417	0.6571	0.6924
	r=64	0.8719	0.9472	0.9061	0.6212	0.6167	0.6864

Table 2: Model accuracies on their given downstream task after fine-tuning for $\alpha = 2r$. Our main results are based on these models.

E Cosine Similarity with Orthogonal Vectors that Span a Space

Here we demonstrate why it is possible for a vector to have low cosine similarity with every orthogonal vector that collectively span a space if the dimensionality of the vectors is high.

Minimizing the Maximum Cosine Similarity. Lets take $Z = \min_{v \in \mathbb{R}^n} \max_i cos(v, x_i)$, where v is an arbitrary vector and each vector x_i, which we collectively call X, make up an orthonormal basis that span the space. Z can be small in a high dimensional space.

2-D case. Assume $X = I$ without loss of generality. It is trivial to see that $Z = \frac{1}{\sqrt{2}}$, and is when $v = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$.

3-D case. Assume $X = I$ without loss of generality. $Z = \frac{1}{\sqrt{3}}$ when $v = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{bmatrix}$.

N-D case. In the N-D case, we can see, via induction, that $Z = \frac{1}{\sqrt{n}}$.

As we can see here, if n is large, the value of Z will be low, even though we are doing the cosine similarity of a vector with respect to a set of orthonormal vectors that span a space.

F Derivation of LoRA Adapter's Gradients

Our calculations were derived independently but follow a similar line to that of Hao et al. [2024].

Derivation for Full Fine-tuning. Full fine-tuning is structured such that

$$Y = W_{tuned}X = (W_0 + \Delta W)X,$$

where $X \in \mathbb{R}^{n \times b}$ are the inputs, $Y \in \mathbb{R}^{m \times b}$ are the outputs, $W_0 \in \mathbb{R}^{m \times n}$ are the pre-trained weights, and $\Delta W \in \mathbb{R}^{m \times n}$ is the fine-tuning update. Accordingly, $\frac{\partial L}{\partial \Delta W} = \frac{\partial L}{\partial Y} X^T$, and the update is

$$\Delta W_n = \Delta W_{n-1} - \eta \frac{\partial L}{\partial Y}_n X_n^T,$$

where η is the learning rate.

Derivation for LoRA. LoRA is structured such that

$$Y = W_{tuned}X = (W_0 + \frac{\alpha}{r} BA)X,$$

where $X \in \mathbb{R}^{n \times b}$ are the inputs, $Y \in \mathbb{R}^{m \times b}$ are the outputs, $W_0 \in \mathbb{R}^{m \times n}$ are the pre-trained weights, $B \in \mathbb{R}^{m \times r}$ is initialized to zero, $A \in \mathbb{R}^{r \times n}$ is randomly initialized, and α is a hyperparameter. Accordingly, $\frac{\partial L}{\partial B} = \frac{\alpha}{r} \frac{\partial L}{\partial Y} X^T A^T$ and $\frac{\partial L}{\partial A} = \frac{\alpha}{r} B^T \frac{\partial L}{\partial Y} X^T$. Therefore, their respective updates are

$$B_n = B_{n-1} - \eta \frac{\alpha}{r} \frac{\partial L}{\partial Y} X^T A^T$$

and

$$A_n = A_{n-1} - \eta \frac{\alpha}{r} B^T \frac{\partial L}{\partial Y} X^T,$$

where η is the learning rate.

Differences in First Step. During the very first step of training, given identical examples both full fine-tuning and LoRA have the same X and Y for each layer since B is initialized to zero. After the first step, full fine-tuning has a update matrix equal to

$$\Delta W_{full} = -\eta \frac{\partial L}{\partial Y} X^T.$$

In contrast, LoRA has an update matrix equal to

$$\Delta W_{lora} = (\frac{\alpha}{r})(B_0 - \eta \frac{\alpha}{r} \frac{\partial L}{\partial Y} X^T A_0^T)(A_0 - \eta \frac{\alpha}{r} B_0^T \frac{\partial L}{\partial Y} X^T).$$

Since $B_0 = 0$,

$$\Delta W_{lora} = (\frac{\alpha}{r})(-\eta \frac{\alpha}{r} \frac{\partial L}{\partial Y} X^T A_0^T)(A_0).$$

From this, we can see that the gradient steps are clearly different, even with the same training examples.

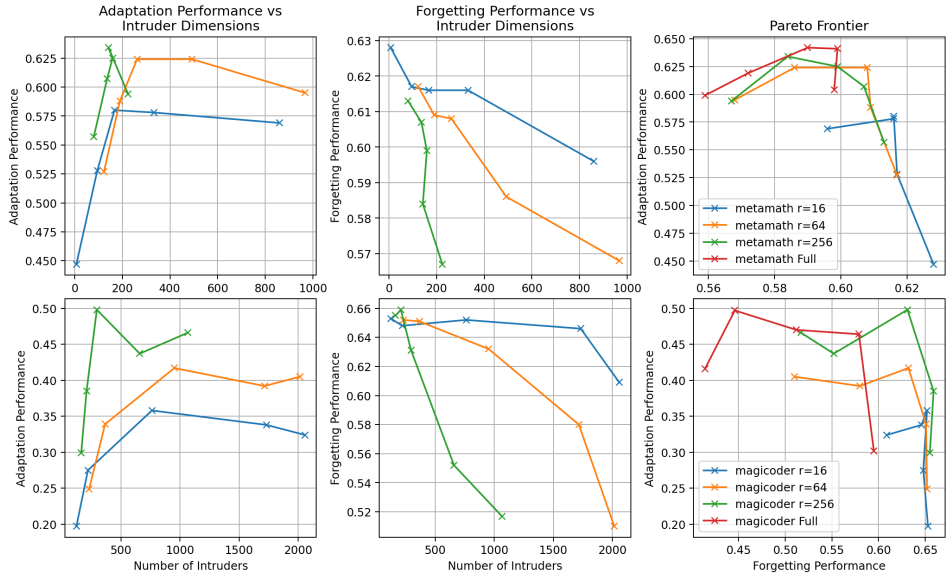


Figure 11: **For LLaMA2-7B, intruder dimensions correlate with forgetting.** Top row: MetaMath. Bottom row: Magicoder. We display intruder dimensions vs test accuracy and intruder dimensions vs forgetting.

G Intruder Dimensions Correlate with Forgetting

G.1 For RoBERTa

As mentioned in the main text, when measuring the Spearman correlation between the number of intruder dimensions and forgetting in Fig. 7 we find an extremely strong fit, with $\rho = 0.971$ and p-value $\ll 0.001$. This shows us that intruder dimensions strongly correlate with forgetting. In contrast, when we correlate intruder dimensions with performance, we find no such correlation: for RoBERTa, we measure $\rho = -0.3381$ with p-value $= 0.218$.

G.2 LLaMA2-7B

When measuring the Spearman correlation between number of intruder dimensions and forgetting for our LLaMA2-7B models across training epochs (Fig. 11 (*middle*)), we find a statistically significant relationship with $\rho = 0.59$ and p-value $= 0.0006$. When correlating intruder dimensions and test accuracy (Fig. 11 (*left*)), we instead measure $\rho = -0.3178$ with p-value $= 0.0869$. Again, we see that intruder dimensions correlates with forgetting.

H Intruder Dimensions Cause Forgetting (Scaling Experiments)

H.1 Performance Differences When Scaling Down Intruder Dimensions

H.1.1 RoBERTa

We report our findings for RoBERTa models fine-tuned on MNLI, QQP, and FEVER in Table 3. Remember that we scale down using the equation $W = W_0 + \Delta W + (\lambda - 1)u_i\sigma_i v_i^T$. Here, we see that scaling down intruder dimensions leads to a sharp drop in forgetting (pre-training loss) and a much smaller drop in test accuracy. Scaling down an intruder dimension by two ($\lambda = 0.5$) results always leads to less than a two percent drop in test accuracy but double digit percentage drops in forgetting. One particularly compelling example, as shown in Fig. 8d, shows how scaling down the top intruder dimensions with $\lambda = 0.7$ when fine-tuning on QQP and using LoRA r=8 result in essentially no (0.0%) drop in adaptation performance but a large (-33.2%) drop in forgetting. Our findings of the impact of scaling down intruder dimensions hold across three datasets and both LoRA r=1 and r=8, which were the two ranks that we found to have many intruder dimensions. Note that

this experiment is meaningless if a model has no intruder dimensions, since no singular vectors will be removed.

Task	LoRA Rank	$\lambda = 0.1$		$\lambda = 0.3$		$\lambda = 0.5$		$\lambda = 0.7$		$\lambda = 0.9$	
		TA	PTL	TA	PTL	TA	PTL	TA	PTL	TA	PTL
MNLI	r=1	-18.7	-13.3	-8.3	-14.5	-2.7	-13.7	-0.6	-11.0	0.0	-5.1
	r=8	-5.1	-24.7	-1.9	-23.1	-0.6	-19.8	-0.2	-14.5	0.0	-5.9
QQP	r=1	-8.6	-35.5	-4.4	-35.8	-1.6	-34.1	-0.4	-28.9	0.1	-14.6
	r=8	-3.3	-52.0	-1.6	-50.6	-0.6	-45.0	-0.0	-33.2	0.1	-13.0
FEVER	r=1	-11.1	-10.3	-4.2	-11.8	-0.4	-11.1	0.6	-8.7	0.6	-3.8
	r=8	-5.6	-14.6	-1.0	-15.3	0.7	-13.4	1.3	-9.7	0.6	-4.0

Table 3: **Impact of scaling RoBERTa-base's intruder dimensions on test accuracy (TA) and pre training loss (PTL).** Numbers reported are the percent change in test accuracy and percent reduction in forgetting induced by fine-tuning. **Scaling down intruder dimensions leads to less forgetting.** On RoBERTa. PTPL is Pre-training loss and TA is test accuracy. Both are reported as percent change with respect to the unchanged fine-tuned model. Scaling down intruder dimensions has large impact on forgetting but little impact on test accuracy.

H.1.2 LLaMA2-7B

Our finding that scaling down intruder dimensions leads to less forgetting but similar test accuracy holds to LLaMA2-7B. One particularly interesting example is for LLaMA2-7B fine-tuned on MetaMath with $r = 256$: when scaling the top intruder dimensions down with $\lambda = 0.5$, we see a large drop (-25.2%) in forgetting and an *increase* (+1.8%) in test accuracy.

Dataset	LoRA Rank	$\lambda = 0.3$		$\lambda = 0.5$		$\lambda = 0.7$		$\lambda = 0.9$	
		TA	PTL	TA	PTL	TA	PTL	TA	PTL
MetaMath	r=16	-15.0	-46.5	-6.6	-40.4	-2.7	-28.2	0.0	-10.5
	r=64	-5.8	-29.1	-4.4	-22.3	0.3	-14.2	0.3	-4.9
	r=256	-0.1	-33.3	1.8	-25.2	0.5	-15.5	-0.8	-5.2
Magicoder	r=16	-1.6	-37.5	-0.6	-24.4	-0.1	-13.2	0.1	-4.1
	r=64	-5.0	-12.7	-3.3	-8.3	0.3	-4.3	2.2	-1.4
	r=256	-4.3	-12.8	-1.3	-10.2	-0.9	-7.0	-1.6	-2.8

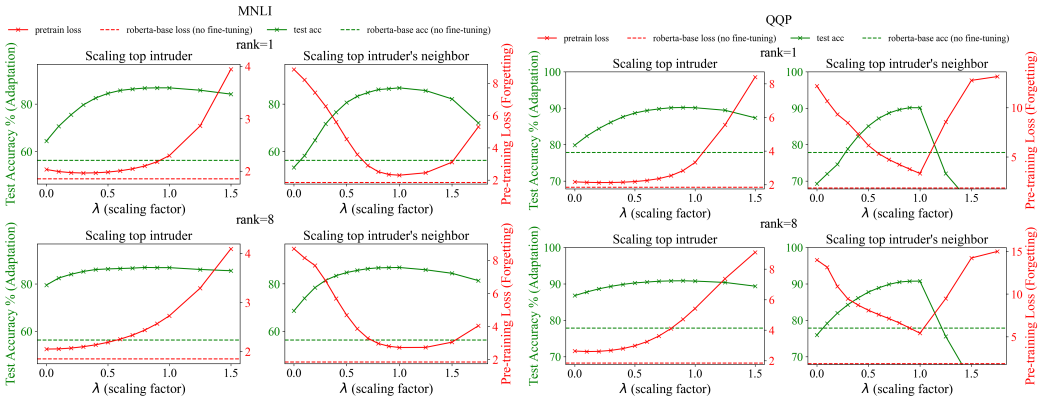
Table 4: **Impact of scaling LLaMA2-7B's intruder dimensions on test accuracy (TA) and pre training loss (PTL).** Numbers reported are the percent change in test accuracy and percent reduction in forgetting induced by fine-tuning.

H.2 Intruder Dimensions Cause Worse OOD Performance

As we discussed in section 5 of the main text, for LoRA models with main intruder dimensions ($r = 1$ and $r = 8$ in our experiments) we measure the impact of intruder dimensions by identifying and scaling the top intruder dimension in every weight matrix such that $W = W_0 + \Delta W + (\lambda - 1)u_i\sigma_i v_i^T$, where i is the index of the top intruder dimension (note that $\lambda = 0$ is removal, $\lambda = 1$ is no change, and $\lambda = 2$ doubles the intruder dimension). For our RoBERTa models fine-tuned on MNLI, QQP, and FEVER, we use $\lambda \in \{0, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0\}$ and for each measure the test accuracy and pre-training loss. For a comparison baseline, we select the neighbor of the intruder dimension to separately scale.

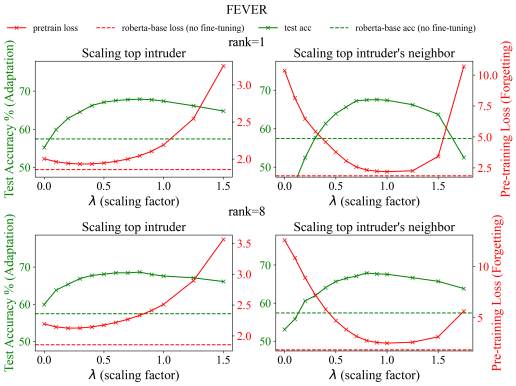
We report these results in Fig. 12a (MNLI), Fig. 12b (QQP), and Fig. 12c (FEVER). We find that when scaling down intruder dimensions ($\lambda < 1$), we observe a clear and significant drop in forgetting (pre-training loss) but a negligible drop in test accuracy (adaptation). When we instead scale up intruder dimensions ($\lambda > 1$), we observe that forgetting increases significantly. We observe that when scaling up top intruder dimensions by 50% ($\lambda = 1.5$), adaptation performance remains relatively flat with a large increase in forgetting, providing further evidence that intruder dimensions are task specific. These trends hold across all 6 models we study (3 datasets with two different LoRA ranks).

In contrast, when we scale a neighboring (pre-trained) singular vector of intruder dimensions instead, we observe starkly different behaviors. When scaling down ($\lambda < 1$) these pre-trained singular vectors,



(a) RoBERTa-base fine-tuned on MNLI.

(b) RoBERTa-base fine-tuned on QQP.



(c) RoBERTa-base fine-tuned on FEVER.

Figure 12: **Scaling RoBERTa-base's intruder dimensions.** We scale the top intruder dimension in each matrix by λ, a multiplicative constant, such that $W = W_0 + \Delta W + (\lambda - 1)u_i\sigma_i v_i^T$. Using $\lambda < 1$ leads to a large drop in pre-training loss while only slightly impacting the test accuracy. For Figs. 12a, 12b, and 12c, we also scale the intruder dimension's neighbor, which is a pre-trained singular vector. Changing these vectors negatively impacts both pre-training loss and test accuracy.

we observe we see that forgetting sharply increases and adaptation performance drops more sharply than when scaling intruders instead. When scaling up ($\lambda > 1$) these pre-trained singular vectors, we observe similar drops in forgetting and adaptation performance. This is likely because pre-trained singular vectors are well tuned for language modeling, and therefore any change to them will have negative downstream impacts on performance. This shows that are observations are not due to the robustness of the model to scaling down specific singular vectors, but rather the difference in contribution to model performance of intruder dimensions vs. pre-trained singular vectors.

Here, we clarify some possible points of confusion. We only scale the top intruder dimension in each weight matrix, if and only if an intruder exists in that matrix, so even when $\lambda = 0$ we do not recover the pre-trained model because not all weight matrices have intruder dimensions and it is possible for multiple intruder dimensions to exist in a weight matrix if LoRA $r > 1$. Furthermore, intruder dimensions are not perfectly orthogonal to the pre-trained singular vectors. Due to the orthogonality constraint imposed by the SVD, all singular vectors will be changed slightly in the matrix, so that even when we remove the intruder dimension, the resulting matrix will be slightly different. These reasons are why when $\lambda = 0$ our LoRA $r = 1$ models do not return to baseline performance.

These findings hold to LLaMA2-7B: in Fig 13, we see that scaling down intruder dimensions leads to much less forgetting but similar test accuracy.

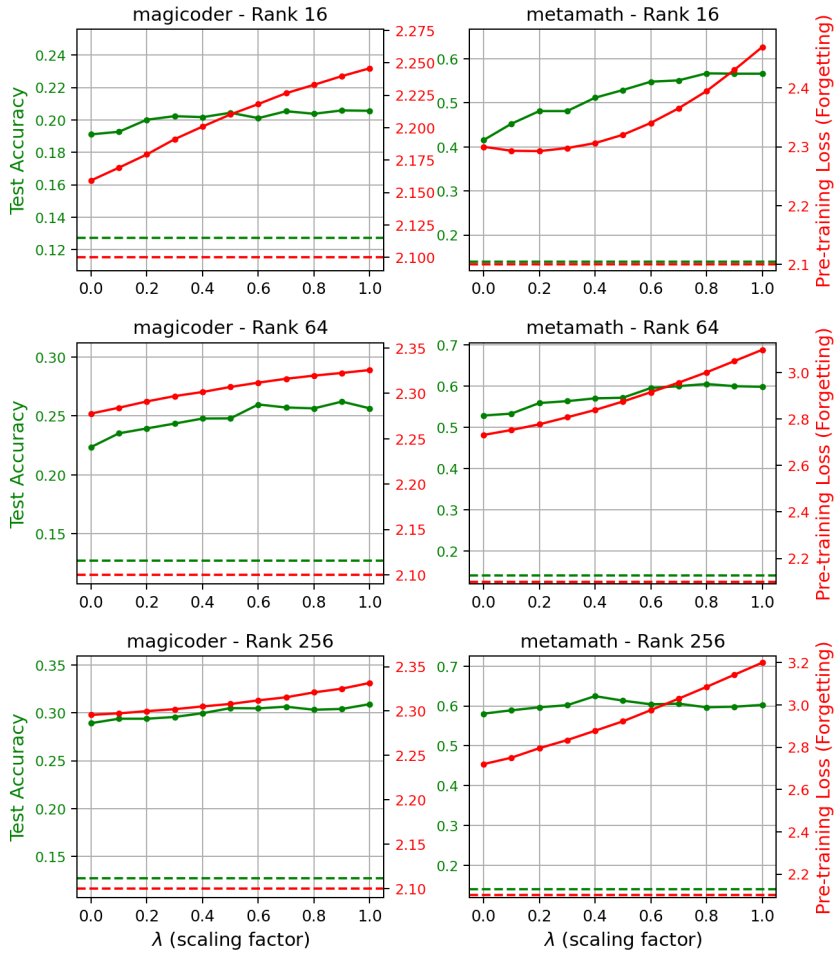
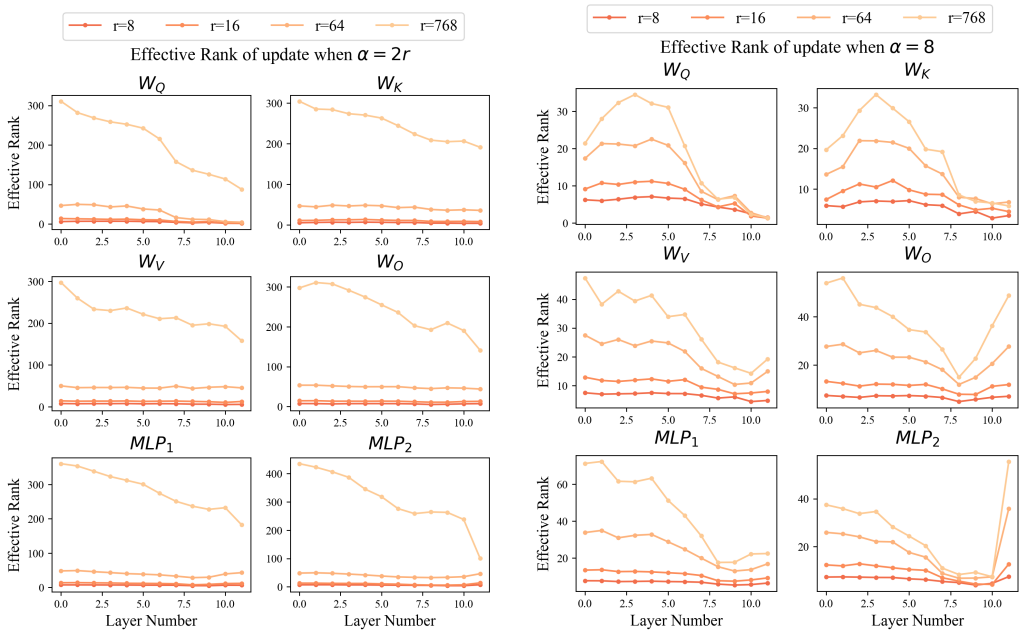


Figure 13: **Scaling down LLaMA2-7B's intruder dimensions leads to less forgetting and nearly equivalent test accuracy.**

I The Effective Rank of the Update Matrix Depends on Alpha

Kalajdzievski [2023] found that LoRA can have gradient collapse when rank is high if alpha is not set properly. Biderman et al. [2024] found that setting $\alpha = 2r$ is very important for the performance of high rank LoRA. In this section, we provide additional evidence of the importance of setting $\alpha = 2r$ and show that if α is held fixed, high rank LoRA converges to low rank solutions. To do this, for both $\alpha = 2r$ and $\alpha = 8$, we measure the effective rank [Roy and Vetterli, 2007] of the weight matrix updates for different LoRA ranks. Effective rank is a measure of the information density of a matrix and can be thought of as an estimation of the rank needed to capture the information held in the weight matrix. It is computed using the singular values of a matrix and we expect the LoRA rank to be the upper bound on what the effective rank can be: LoRA $r = 8$ should have an effective rank update of at most 8. We present the effective rank measurements in Fig. 14. In this plot, we find that when $\alpha = 2r$ (Fig. 14a), LoRA $r = 64$ and $r = 768$ have much higher effective rank than $r = 8$ and $r16$, with $r = 768$ appearing to always have effective rank above 100. In stark contrast, we see that when $\alpha = 8$ (Fig. 14b), high ranks of LoRA have much lower effective ranks, frequently even converging to the effective rank of much lower rank updates (like $r = 8$ and $r = 16$). For example, LoRA $r = 768$ has an effective rank that is consistently below 50 when $\alpha = 8$. We note that full fine-tuning has an effective rank update of above 400 consistently. These plots suggest that when α is kept fixed when scaling LoRA rank, the solutions are uable to take advantage of their higher expressability and instead *converge to low rank solutions*.

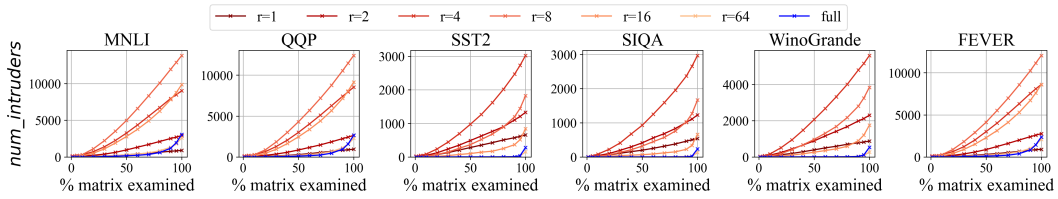


(a) Effective Rank of the LoRA update when $\alpha = 2r$. (b) Effective Rank of the LoRA update when $\alpha = 8$.

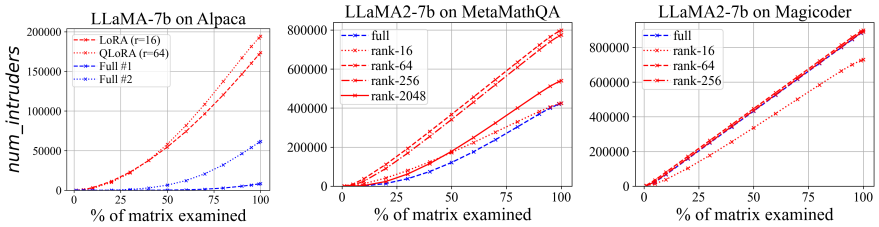
Figure 14: Effective rank of LoRA update matrices (ΔW) for RoBERTa fine-tuned on MNLI. We observe that when $\alpha = 2r$, higher ranks of LoRA ($r = 64, 768$) have much higher effective rank than the same ranks of LoRA but instead with $\alpha = 8$. Building on Kalajdzievski [2023], Biderman et al. [2024], this suggests that $\alpha = 2r$ is necessary for high ranks of LoRA to utilize their expressive capacity. Note: full fine-tuning consistently has updates with effective rank above 400.

J Impact of Matrix Percentage on Number of Intruder Dimensions

In this section, we examine the extent to which intruder dimensions exist throughout the entire weight matrix and how they are distributed. As described in the main text, we hold ϵ fixed as $\epsilon = 0.5$ and measure the number of intruder dimensions while varying the proportion of the fine-tuned singular vectors that we examine (this means varying our k parameter in Algorithm 1). Here, we can see that LoRA consistently has more intruder dimensions than full fine-tuning, regardless of what fraction of the singular values we examine. The only caveat to this is that, for some datasets, full fine-tuning passes LoRA $r = 1$ when examining the last 20% of the fine-tuned singular vectors. This is likely due to the limited expressivity of rank 1 updates and is interesting because it suggests that in this case, full fine-tuning may be changing lower-ranking singular vectors more than LoRA. One interesting contradiction to our findings is in Fig. 15d, which shows that full fine-tuning and LoRA appear to have very similar distributions of intruder dimensions within their matrix when fine-tuned on code. This is likely due to the large domain shift from natural language to coding tasks (Biderman et al. [2024] also make this observation of a large domain shift required for models fine-tuned on Magicoder [Wei et al., 2024]).



(a) Impact of the number of singular vectors in the fine-tuned matrix we examine, k, on the number of intruder dimensions for RoBERTa models fine-tuned on 6 different tasks. Here, we set $\epsilon = 0.5$.



(b) LLaMA-7B fine-tuned on Alpaca.

(c) LLaMA2-7B fine-tuned on MetaMathQA.

(d) LLaMA2-7B fine-tuned on Magicoder-Evol-Instruct.

Figure 15: **Impact of k, the number of fine-tuned singular vectors we examine, on the number of intruder dimensions.** We see that models fine-tuned with LoRA tend to have more intruder dimensions than full fine-tuning, regardless of the value of k used.

K Impact of Dataset Size On Intruder Dimensions

The total number of intruder dimensions increases proportionally to the size of the fine-tuning dataset. Using our training recipe (Appendix C.3), we fine-tuned models on data subsets of varying sizes. We trained RoBERTa-base on MNLI using LoRA with rank 1 and 8 (cases where we originally saw intruder dimensions) and measure the number of intruder dimensions along with the impact of ϵ and k (Fig. 16). For $r = 8$, as we train on more data, more intruder dimensions are introduced. Interestingly, however, LoRA with rank 1 appears to converge to similar amounts of intruder dimensions, regardless of the dataset size. This may be because of the limited expressivity of models with $r = 1$. This experiments suggest that with smaller datasets, fewer intruder dimensions may be introduced by LoRA.

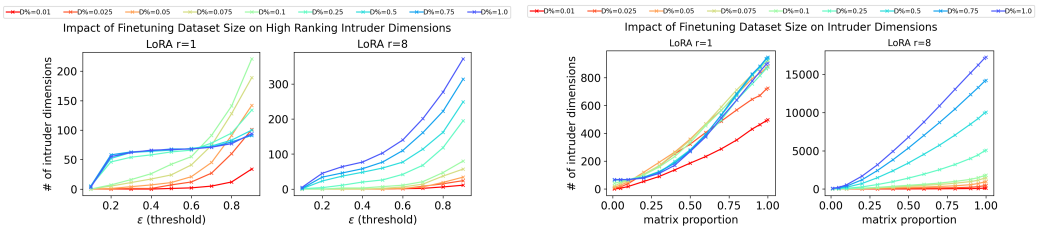


Figure 16: *(Left)* Impact of cosine similarity threshold, ϵ, on the number of intruder dimensions for LoRA models trained on different proportions of the MNLI dataset. *(Right)* Impact of the number of fine-tuned singular vectors we examine, k, on the number of intruder dimensions for LoRA models trained on different proportions of the MNLI dataset. We see that training on a larger proportion of the dataset increases the number of intruder dimensions in the model.

L LLaMA/LLaMA-2 Instruction Tuned Models

Our LLaMA-7B checkpoints were fine-tuned on the Alpaca [Taori et al., 2023] and consist of two fully fine-tuned models, one LoRA model with rank 16, and one QLoRA [Dettmers et al., 2023] model with rank 64. Our LLaMA2-7B checkpoints were fine-tuned on either code (IFT with Magicoder-Evol-Instruct-110K [Wei et al., 2024] or CPT with StarCoder [Li et al., 2023]) or math (IFT with MetaMathQA [Yu et al., 2024] or CPT with OpenWebMath [Paster et al., 2023]) and consist of one fully fine-tuned model and 3-4 LoRA'ed models of different ranks for each dataset and generously provided by Biderman et al. [2024]. In Fig. 4a, Full #1 refers to "PKU-Alignment/alpaca-7b-reproduced" and Full #2 refers to "chavinlo/alpaca-native".

Hugging Face Path	Base Model	IT Dataset
timdettmers/qlora-alpaca-7b	LLaMA-7b	Alpaca
tloen/alpaca-lora-7b	LLaMA-7b	Alpaca
PKU-Alignment/alpaca-7b-reproduced	LLaMA-7b	Alpaca
chavinlo/alpaca-native	LLaMA-7b	Alpaca
LoRA-TMLR-2024/magicoder-lora-rank-16-alpha-32	LLaMA2-7b	Magicoder
LoRA-TMLR-2024/magicoder-lora-rank-64-alpha-128	LLaMA2-7b	Magicoder
LoRA-TMLR-2024/magicoder-lora-rank-256-alpha-512	LLaMA2-7b	Magicoder
LoRA-TMLR-2024/magicoder-full-finetuning-lr-5e-05	LLaMA2-7b	Magicoder
LoRA-TMLR-2024/metamath-lora-rank-16-alpha-32	LLaMA2-7b	MetaMath
LoRA-TMLR-2024/metamath-lora-rank-64-alpha-128	LLaMA2-7b	MetaMath
LoRA-TMLR-2024/metamath-lora-rank-256-alpha-512	LLaMA2-7b	MetaMath
LoRA-TMLR-2024/metamath-full-finetuning-lr-1e-05	LLaMA2-7b	MetaMath
LoRA-TMLR-2024/starcoder-lora-rank-16-20B-tokens	LLaMA2-7b	StarCoder
LoRA-TMLR-2024/starcoder-lora-rank-64-20B-tokens	LLaMA2-7b	StarCoder
LoRA-TMLR-2024/starcoder-lora-rank-256-20B-tokens	LLaMA2-7b	StarCoder
LoRA-TMLR-2024/starcoder-full-finetuning-lr-1e-05-20B-token	LLaMA2-7b	StarCoder
LoRA-TMLR-2024/openwebmath-lora-rank-16-20B-tokens	LLaMA2-7b	OpenWebMath
LoRA-TMLR-2024/openwebmath-lora-rank-64-20B-tokens	LLaMA2-7b	OpenWebMath
LoRA-TMLR-2024/openwebmath-lora-rank-256-20B-tokens	LLaMA2-7b	OpenWebMath
LoRA-TMLR-2024/openwebmath-full-finetuning-lr-1e-05-20B-tokens	LLaMA2-7b	OpenWebMath

Table 5: Hugging Face model paths for LLaMA-7b/LLaMA2-7b IT models.

M Continual Learning

M.1 Performance during continual learning

As described in Section 4 in the main text, we train sequentially on 6 tasks and measure task performance on all of these tasks across tasks trained on (continual learning). We report the full graph of our findings in Fig. 17. In it, we find that when all our models are trained to similar accuracy, lower ranks of LoRA, which coincide with more intruder dimensions, forget more of their previously learned tasks than higher ranks of LoRA and full fine-tuning.

M.2 Similarity matrices during continual learning

After each continual learning dataset we fine-tune on, we measure the similarity matrix between the current model and the pre-trained model. In Fig. 18b, we observe that LoRA accumulates intruder dimensions across fine-tuning datasets. In contrast, in Fig. 18a we observe that the pre-trained structure of the model is retained well across fine-tuning datasets. These experiments suggest why LoRA appears to degrade faster during continual learning.

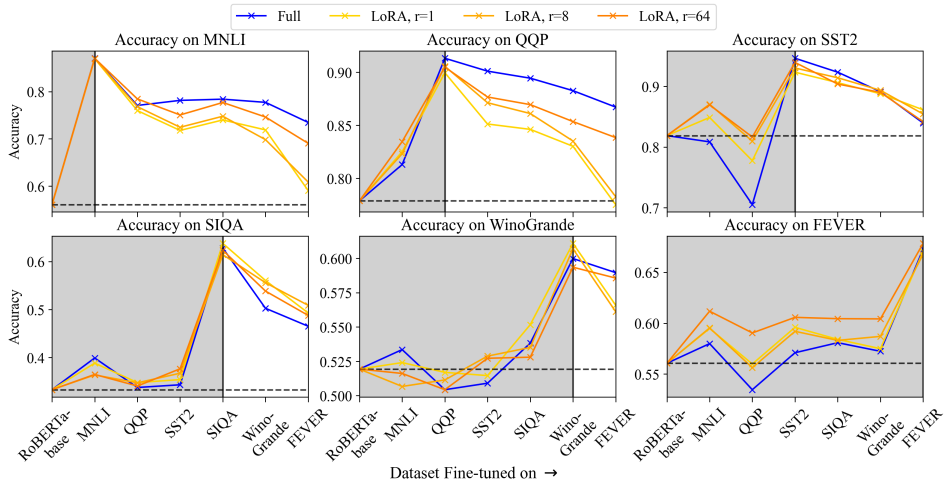


Figure 17: Full plot of Fig. 9a. Continual Learning performance of RoBERTa for full fine-tuning and LoRA. We sequentially train on six tasks, in order from left to right. Horizontal dotted line indicates baseline pre-trained performance. Vertical solid line indicates when a specific dataset is fine-tuned on. Gray region represents performance before the model has been trained on that task. We are interested in the differences in accuracies of these methods both right after training (at the vertical black line) and later (in the white region). We see that low ranks of LoRA forget previously learned tasks more.

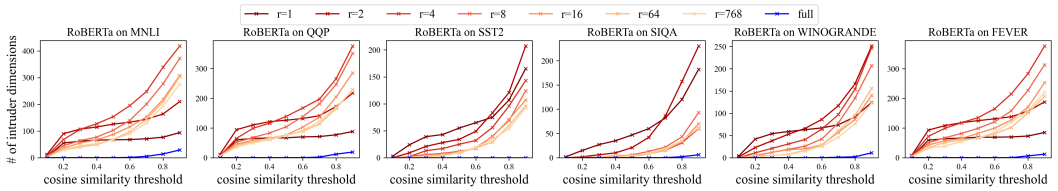
(a) Continual learning similarity matrices for full fine-tuning.

(b) Continual learning similarity matrices for LoRA.

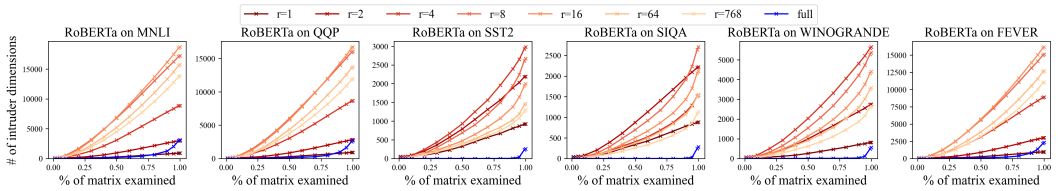
Figure 18: **LoRA accumulates intruder dimensions, while full fine-tuning does not.** The pre-trained structure of the model degrades across tasks trained on.

N Case Study: Setting Alpha=8 instead of Alpha=2r

Our main experiments were conducted with $\alpha = 2r$. However, Hu et al. [2021] instead set $\alpha = 8$ for RoBERTa-base. While not the recommended practice now, we explore what impact this selection has on our findings. We report our key plots in Fig. 19a, 19b, 20, 21, & 14b. In Fig. 19a & 19b we see that LoRA'd models with high rank have significantly more intruder dimensions in comparison to when $\alpha = 2r$. Interestingly, whereas when $\alpha = 2r$ LoRA models with ranks like 64 had no or very few intruder dimensions (see Fig. 4), they now have numerous intruder dimensions. These differences are corroborated by Fig. 14b, where we see that the learned solutions of LoRA have significantly lower effective rank in comparison to when $\alpha = 2r$. For example, we see in Fig. 14b that when LoRA has a rank of 768, the effective rank is never above 100. In contrast, we see in Fig. 14a that with the same rank of 768, LoRA always has an effective rank above 768. This suggests that when $\alpha = 8$, LoRA is converging to lower rank solutions than when $\alpha = 2r$. This supports the finding that setting $\alpha = 2r$ improves LoRA's performance when a high rank is used [Biderman et al., 2024, Kalajdzievski, 2023]. Behaviorally, we see in Fig. 21 that LoRA models with high rank have much more forgetting on previously learned tasks in comparison to full fine-tuning and LoRA when $\alpha = 2r$ is used ($\alpha = 2r$ results are in Fig. 17). Likewise, in Fig. 14b we see that when LoRA has high rank, it has much more forgetting on the pre-trained distribution in comparison to LoRA when $\alpha = 2r$.



(a) Number of intruder dimensions in RoBERTa models fine-tuned on 6 different tasks. Here, we set $k = 10$. We use the same conditions as in Fig. 4d but instead now set $\alpha = 8$ instead of $\alpha = 2r$.



(b) Impact of the number of singular vectors in the fine-tuned matrix we examine, k, on the number of intruder dimensions for RoBERTa models fine-tuned on 6 different tasks. Here, we set $\epsilon = 0.5$. We use the same conditions as in Fig. 15a but instead now set $\alpha = 8$ instead of $\alpha = 2r$.

Figure 19: We find that when $\alpha = 8$ instead of $\alpha = 2r$, our models have more intruder dimensions. *(Top)* Replication of Fig. 4d with $\alpha = 8$ instead of $\alpha = 2r$. *(Bottom)* Replication of Fig. 15a with $\alpha = 8$ instead of $\alpha = 2r$.

O Impact of Random Seeds

To ensure that random seed does not play a role in the number of intruder dimensions we observe in a model, we sample 5 different seeds and fine-tune RoBERTa-base on MNLI using the same methodology as in Fig. 4d. We find that the initialization has a negligible role on the number of intruder dimensions. This shows that our findings are not dependent on the random initialization of the LoRA modules.

P LoRA Variants

We focus significantly on the standard LoRA method proposed by Hu et al. [2021] in order to study it in depth. However, many variants of LoRA have been proposed recently. AdaLoRA [Zhang et al., 2023] adaptively allocates LoRA rank to different modules in order to ensure optimal allocation of trainable parameters to certain modules. LoRA+ [Hayou et al., 2024] sets different learning rates for the A and B modules in LoRA. PiSSA [Meng et al., 2024] initializes the A and B modules with

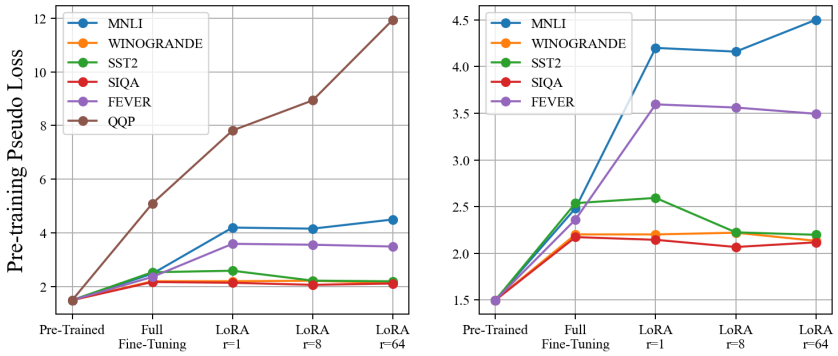


Figure 20: For $\alpha = 8$. RoBERTa's performance on its pre-training data distribution after fine-tuning on a particular task. We measure pseudo loss as described by Salazar et al. [2020]. We compare these results to when $\alpha = 2r$ (Fig. 6).

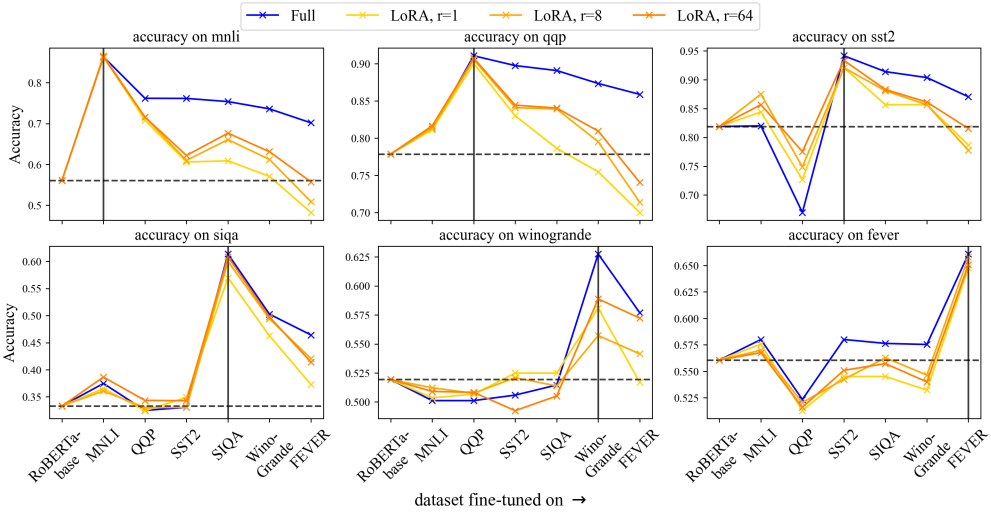


Figure 21: For $\alpha = 8$. RoBERTa's performance on six datasets during continual learning. We sequentially train on six tasks, in order from left to right. Horizontal dotted line indicates baseline pre-trained performance. Vertical solid line indicates when a specific dataset is fine-tuned on. We compare these results to when $\alpha = 2r$ (Fig. 17).

the top ranking singular vectors of the pre-trained weights. VeRA [Kopiczko et al., 2024] models LoRA as the product of two random matrices with trainable parameters doing elementwise operations on the resulting vectors. These variants may have important impacts on the presence of intruder dimensions. For example, PiSSA initializes with the singular vectors and therefore may have an easier time changing them, possibly leading to more intruder dimensions. In contrast, LoRA+ in effect lowers the learning rate, which we found to be important to introducing intruder dimensions, and may therefore reduce the number of intruder dimensions.

In order to examine if intruder dimensions are still relevant for these methods, we rerun our MNLI fine-tuning experiment with RoBERTa with each of these methods with default hyperparameters that they provide. These results are supplied in Fig. 23. Interestingly, we see that all the variations we examine have intruder dimensions. Some interesting observations include: LoRA+ and LoRA $r = 1$ appear to have nearly identical curves, suggesting they have very ismilar intruder dimension characteristics. We again see that with higher ranks ($r = 64$) these LoRA variants tend to have very few intruder dimensions. However, it does appear that methods that explicitly modify the singular vectors, like PiSSA, have many intruder dimensions. This makes sense since they are explicitly constructed to modify the singular vectors on the pre-trained model. These findings emphasize that

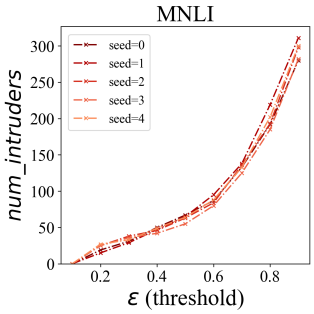


Figure 22: **Impact of Random Seeds on intruder dimensions.** We fine-tune RoBERTa-base across 5 random seeds and use our same methodology as in Fig. 4d. We find that the initialization has a negligible role on the number of intruder dimensions. This shows that our findings are not dependent on the random initialization of the LoRA modules.

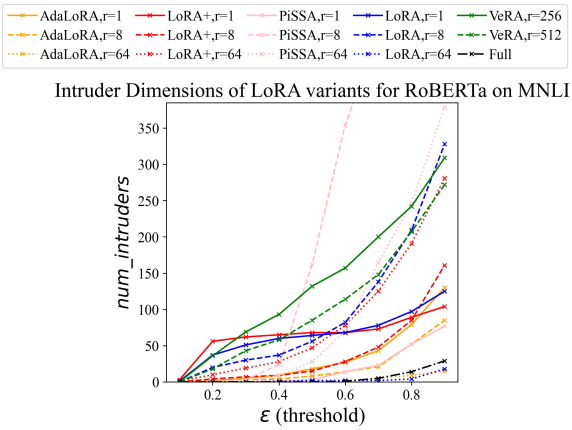


Figure 23: **Measuring LoRA variants for intruder dimensions.** $k = 10$. We compare variants of LoRA to normal LoRA (blue) and full fine-tuning (black). We find that the LoRA variants we examine still have intruder dimensions and shows that our findings are not just exclusive to normal LoRA.

intruder dimensions are not just an observed phenomenom in normal LoRA and suggests to future work the examination of LoRA variants.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Each claim is clearly stated. We callout with bold each claim in the introduction and takeaways are bolded in later sections to call out our key claims.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes. We discuss the limitations of our work throughout the paper. We also add a new section in the appendix to discuss limitations (Section A).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We have no theoretical results. We briefly justify results with calculations in the appendix(section F), but these are not central or important to our results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We release code to replicate our main results (code is not attached to avoid breaking double-blind review, link will be attached upon acceptance). This github repository contains exact commands needed to replicate our findings. Additionally, methodology for reproducing experiments is located in the appendix. All our data is publicly available.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release code to replicate our main results (code is not attached to avoid breaking double-blind review, link will be attached upon acceptance). This github repository contains exact commands needed to replicate our findings. Additionally, methodology for reproducing experiments is located in the appendix. All our data is publicly available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All details for replicating our training and evaluation methods are provided in the appendix. This includes datasets, models, hyperparameters, optimizer, etc. See section C.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars for most experiments are not used because of the computational requirements for these experiments, making it infeasible to, for example, perform more than one training run. Elsewhere, analyses are deterministic and do not need error bars. In addition, we do report statistical significance tests for our correlations of intruder dimensions and forgetting and intruder dimensions and test accuracy.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: All of this information is provided in a subsection of section C in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: this paper conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: This paper does not have a true societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper poses no such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite every single model (Section L) and codebase/dataset (Section C) we use. While difficult to find licenses for these resources, they are all open source.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We don't release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: No LLM usage.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.