

to maximize downstream performance? 2) Is the “optimal” adaptation matrix ΔW *really rank-deficient*? If so, what is a good rank to use in practice? 3) What is the connection between ΔW and W ? Does ΔW highly correlate with W ? How large is ΔW comparing to W ?

We believe that our answers to question (2) and (3) shed light on the fundamental principles of using pre-trained language models for downstream tasks, which is a critical topic in NLP.

7.1 WHICH WEIGHT MATRICES IN TRANSFORMER SHOULD WE APPLY LoRA TO?

Given a limited parameter budget, which types of weights should we adapt with LoRA to obtain the best performance on downstream tasks? As mentioned in Section 4.2, we only consider weight matrices in the self-attention module. We set a parameter budget of 18M (roughly 35MB if stored in FP16) on GPT-3 175B, which corresponds to $r = 8$ if we adapt one type of attention weights or $r = 4$ if we adapt two types, for all 96 layers. The result is presented in Table 5.

	# of Trainable Parameters = 18M						
Weight Type	W_q	W_k	W_v	W_o	W_q, W_k	W_q, W_v	W_q, W_k, W_v, W_o
Rank r	8	8	8	8	4	4	2
WikiSQL ($\pm 0.5\%$)	70.4	70.0	73.0	73.2	71.4	73.7	73.7
MultiNLI ($\pm 0.1\%$)	91.0	90.8	91.0	91.3	91.3	91.3	91.7

Table 5: Validation accuracy on WikiSQL and MultiNLI after applying LoRA to different types of attention weights in GPT-3, given the same number of trainable parameters. Adapting both W_q and W_v gives the best performance overall. We find the standard deviation across random seeds to be consistent for a given dataset, which we report in the first column.

Note that putting all the parameters in ΔW_q or ΔW_k results in significantly lower performance, while adapting both W_q and W_v yields the best result. This suggests that even a rank of four captures enough information in ΔW such that it is preferable to adapt more weight matrices than adapting a single type of weights with a larger rank.

7.2 WHAT IS THE OPTIMAL RANK r FOR LoRA?

We turn our attention to the effect of rank r on model performance. We adapt $\{W_q, W_v\}$, $\{W_q, W_k, W_v, W_o\}$, and just W_q for a comparison.

	Weight Type	$r = 1$	$r = 2$	$r = 4$	$r = 8$	$r = 64$
WikiSQL ($\pm 0.5\%$)	W_q	68.8	69.6	70.5	70.4	70.0
	W_q, W_v	73.4	73.3	73.7	73.8	73.5
	W_q, W_k, W_v, W_o	74.1	73.7	74.0	74.0	73.9
MultiNLI ($\pm 0.1\%$)	W_q	90.7	90.9	91.1	90.7	90.7
	W_q, W_v	91.3	91.4	91.3	91.6	91.4
	W_q, W_k, W_v, W_o	91.2	91.7	91.7	91.5	91.4

Table 6: Validation accuracy on WikiSQL and MultiNLI with different rank r . To our surprise, a rank as small as one suffices for adapting both W_q and W_v on these datasets while training W_q alone needs a larger r . We conduct a similar experiment on GPT-2 in Section H.2.

Table 6 shows that, surprisingly, LoRA already performs competitively with a very small r (more so for $\{W_q, W_v\}$ than just W_q). This suggests the update matrix ΔW could have a very small “intrinsic rank”.⁶ To further support this finding, we check the overlap of the subspaces learned by different choices of r and by different random seeds. We argue that increasing r does not cover a more meaningful subspace, which suggests that a low-rank adaptation matrix is sufficient.

⁶However, we do not expect a small r to work for every task or dataset. Consider the following thought experiment: if the downstream task were in a different language than the one used for pre-training, retraining the entire model (similar to LoRA with $r = d_{model}$) could certainly outperform LoRA with a small r .

Subspace similarity between different r . Given $A_{r=8}$ and $A_{r=64}$ which are the learned adaptation matrices with rank $r = 8$ and 64 using the *same pre-trained model*, we perform singular value decomposition and obtain the right-singular unitary matrices $U_{A_{r=8}}$ and $U_{A_{r=64}}$.⁷ We hope to answer: how much of the subspace spanned by the top i singular vectors in $U_{A_{r=8}}$ (for $1 \leq i \leq 8$) is contained in the subspace spanned by top j singular vectors of $U_{A_{r=64}}$ (for $1 \leq j \leq 64$)? We measure this quantity with a normalized subspace similarity based on the Grassmann distance (See Appendix G for a more formal discussion)

$$\phi(A_{r=8}, A_{r=64}, i, j) = \frac{\|U_{A_{r=8}}^{i\top} U_{A_{r=64}}^j\|_F^2}{\min(i, j)} \in [0, 1] \quad (4)$$

where $U_{A_{r=8}}^i$ represents the columns of $U_{A_{r=8}}$ corresponding to the top- i singular vectors.

$\phi(\cdot)$ has a range of $[0, 1]$, where 1 represents a complete overlap of subspaces and 0 a complete separation. See Figure 3 for how ϕ changes as we vary i and j . We only look at the 48th layer (out of 96) due to space constraint, but the conclusion holds for other layers as well, as shown in Section H.1.

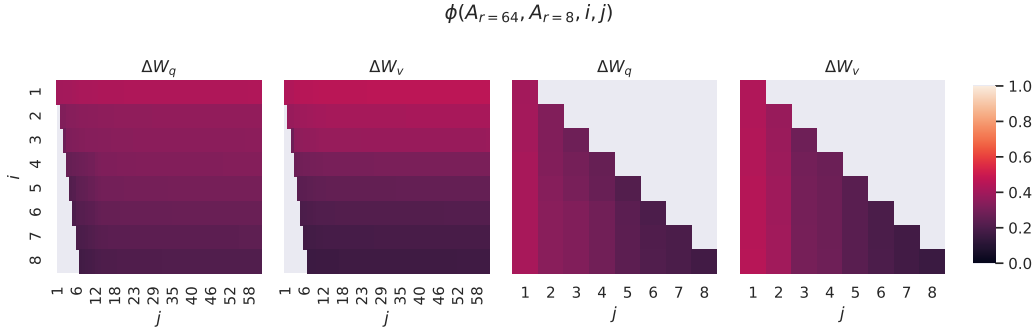


Figure 3: Subspace similarity between column vectors of $A_{r=8}$ and $A_{r=64}$ for both ΔW_q and ΔW_v . The third and the fourth figures zoom in on the lower-left triangle in the first two figures. The top directions in $r = 8$ are included in $r = 64$, and vice versa.

We make an *important observation* from Figure 3.

Directions corresponding to the top singular vector overlap significantly between $A_{r=8}$ and $A_{r=64}$, while others do not. Specifically, ΔW_v (resp. ΔW_q) of $A_{r=8}$ and ΔW_v (resp. ΔW_q) of $A_{r=64}$ share a subspace of dimension 1 with normalized similarity > 0.5 , providing an explanation of why $r = 1$ performs quite well in our downstream tasks for GPT-3.

Since both $A_{r=8}$ and $A_{r=64}$ are learned using the same pre-trained model, Figure 3 indicates that the top singular-vector directions of $A_{r=8}$ and $A_{r=64}$ are the most useful, while other directions potentially contain mostly random noises accumulated during training. Hence, the adaptation matrix can indeed have a very low rank.

Subspace similarity between different random seeds. We further confirm this by plotting the normalized subspace similarity between two randomly seeded runs with $r = 64$, shown in Figure 4. ΔW_q appears to have a higher “intrinsic rank” than ΔW_v , since more common singular value directions are learned by both runs for ΔW_q , which is in line with our empirical observation in Table 6. As a comparison, we also plot two random Gaussian matrices, which do not share any common singular value directions with each other.

7.3 HOW DOES THE ADAPTATION MATRIX ΔW COMPARE TO W ?

We further investigate the relationship between ΔW and W . In particular, does ΔW highly correlate with W ? (Or mathematically, is ΔW mostly contained in the top singular directions of W ?) Also,

⁷Note that a similar analysis can be carried out with B and the left-singular unitary matrices – we stick with A for our experiments.

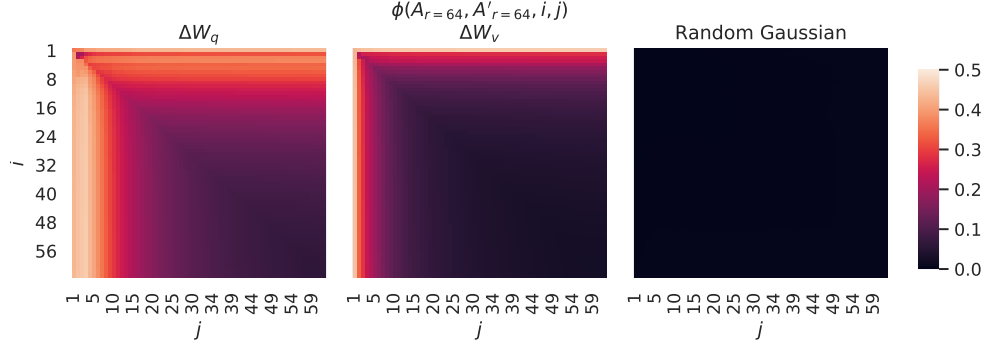


Figure 4: **Left and Middle:** Normalized subspace similarity between the column vectors of $A_{r=64}$ from two random seeds, for both ΔW_q and ΔW_v in the 48-th layer. **Right:** the same heat-map between the column vectors of two random Gaussian matrices. See Section H.1 for other layers.

how “large” is ΔW comparing to its corresponding directions in W ? This can shed light on the underlying mechanism for adapting pre-trained language models.

To answer these questions, we project W onto the r -dimensional subspace of ΔW by computing $U^\top W V^\top$, with U/V being the left/right singular-vector matrix of ΔW . Then, we compare the Frobenius norm between $\|U^\top W V^\top\|_F$ and $\|W\|_F$. As a comparison, we also compute $\|U^\top W V^\top\|_F$ by replacing U, V with the top r singular vectors of W or a random matrix.

	ΔW_q	$r = 4$ W_q	Random	ΔW_q	$r = 64$ W_q	Random
$\ U^\top W_q V^\top\ _F =$	0.32	21.67	0.02	1.90	37.71	0.33
$\ W_q\ _F = 61.95$	$\ \Delta W_q\ _F = 6.91$			$\ \Delta W_q\ _F = 3.57$		

Table 7: The Frobenius norm of $U^\top W_q V^\top$ where U and V are the left/right top r singular vector directions of either (1) ΔW_q , (2) W_q , or (3) a random matrix. The weight matrices are taken from the 48th layer of GPT-3.

We draw *several conclusions* from Table 7. First, ΔW has a stronger correlation with W compared to a random matrix, indicating that ΔW amplifies some features that are already in W . Second, instead of repeating the top singular directions of W , ΔW only *amplifies directions that are not emphasized in W* . Third, the amplification factor is rather huge: $21.5 \approx 6.91/0.32$ for $r = 4$. See Section H.4 for why $r = 64$ has a smaller amplification factor. We also provide a visualization in Section H.3 for how the correlation changes as we include more top singular directions from W_q . This suggests that the low-rank adaptation matrix potentially *amplifies the important features for specific downstream tasks that were learned but not emphasized in the general pre-training model*.

8 CONCLUSION AND FUTURE WORK

Fine-tuning enormous language models is prohibitively expensive in terms of the hardware required and the storage/switching cost for hosting independent instances for different tasks. We propose LoRA, an efficient adaptation strategy that neither introduces inference latency nor reduces input sequence length while retaining high model quality. Importantly, it allows for quick task-switching when deployed as a service by sharing the vast majority of the model parameters. While we focused on Transformer language models, the proposed principles are generally applicable to any neural networks with dense layers.

There are many directions for future works. 1) LoRA can be combined with other efficient adaptation methods, potentially providing orthogonal improvement. 2) The mechanism behind fine-tuning or LoRA is far from clear – how are features learned during pre-training transformed to do well on downstream tasks? We believe that LoRA makes it more tractable to answer this than full fine-