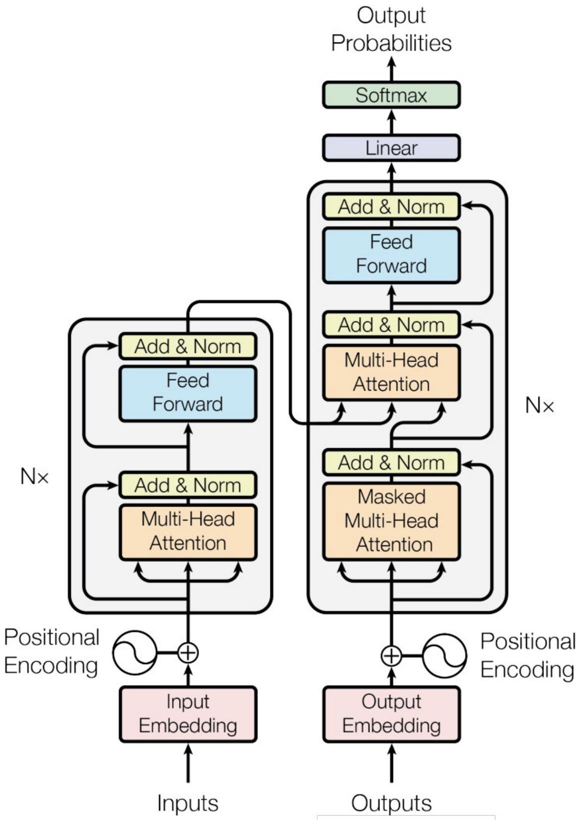
LoRA

Low-Rank Adaptation of Large Language Models

NNs = matrix multiplication

BERT

Encoder



GPT

Decoder

Rank of a matrix

- **Number of independent (column) vectors**

- **M =** $$\begin{matrix} 1 & 0 & 2 & 2 & 1 \\ -2 & 1 & -3 & -2 & 0 \\ 3 & -1 & 5 & 4 & 1 \end{matrix}$$

Rank of a matrix

- **Number of independent (column) vectors**

- **M =** $\begin{matrix} 1 & 0 & 2 & 2 & 1 \\ -2 & 1 & -3 & -2 & 0 \\ 3 & -1 & 5 & 4 & 1 \end{matrix}$

- **rank(M) = 2**

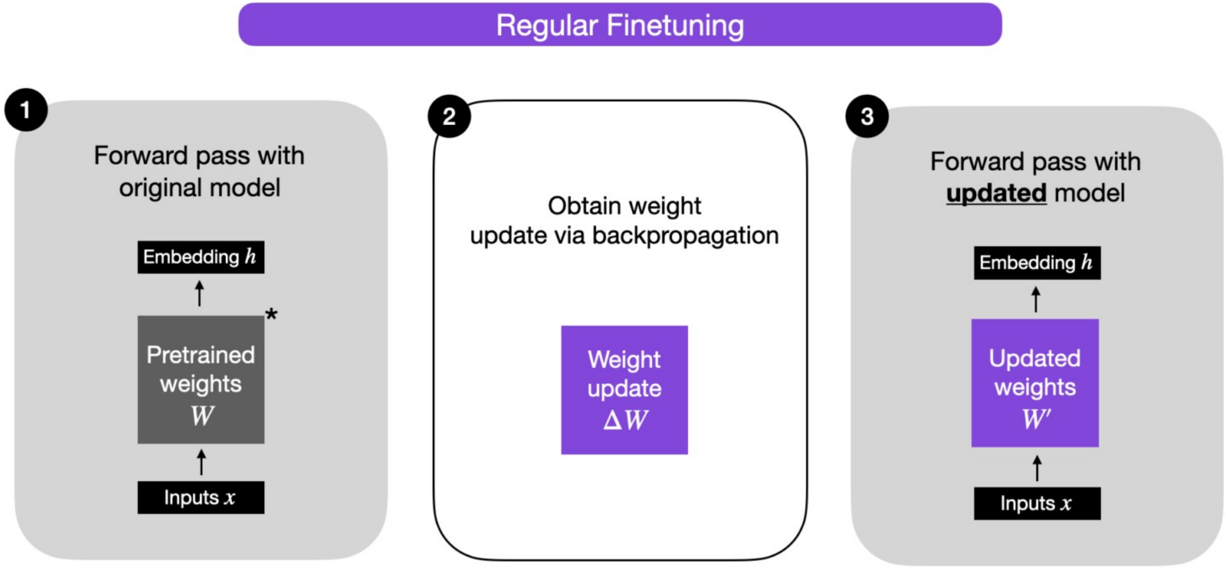
Rank of a matrix

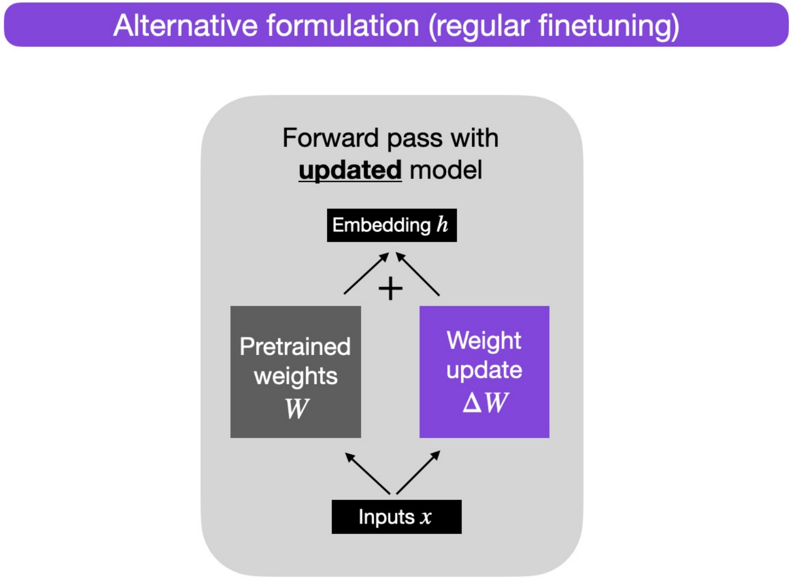
- **Number of independent (column) vectors**

- **M =** $\begin{matrix} 1 & 0 & 2 & 2 & 1 \\ -2 & 1 & -3 & -2 & 0 \\ 3 & -1 & 5 & 4 & 1 \end{matrix}$

- **rank(M) = 2**
- **Information content of M:** $\begin{matrix} 1 & 0 \\ -2 & 1 \\ 3 & -1 \end{matrix}$

Separating weight updates



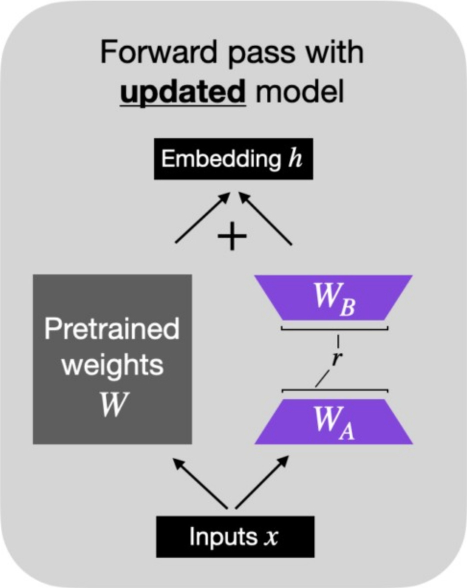


* The pretrained model could be any LLM, e.g., an encoder-style LLM (like BERT) or a generative decoder-style LLM (like GPT)

Putting it together

- **Observation: low rank in fine-tuned models**
- **Hypothesis: persists in separated weight matrices**
- **→ dimension of ΔW can be scaled down**
- **Decompose** $W_0 + \Delta W \in \mathbb{R}^{d \times k}$ **into** $W_0 + BA$ **where** $B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$

LoRA weights, W_A and W_B, represent ΔW



r « d and k

General benefits

- $\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t|x, y_{<t})\right) \rightarrow \max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})\right)$
- $|\Theta| \ll |\Phi_0|$ → **lower storage requirement and speed-up**
- **Converges to original fine-tuning by increasing rank**
- **Avoids inference latency: $W = W_0 + B \times A$**
- **Easy task switching: $W = W - B \times A$; $W = W + B' \times A'$**

LoRA training for transformers

- **4 matrices per attention layer: W_q, W_k, W_v, and W_o (ignores slicing through attention heads)**
- **No fine-tuning of feed-forward layers and layer norm**
- **Example: GPT-3**
  - 2/3 reduction of VRAM (1.2TB → 350GB)
  - 10000 times reduction of checkpoint size with $r = 4$ (350GB → 35MB)
  - 25% speedup during fine-tuning (32.5 tokens/s → 41.1 tokens/s)

It works!

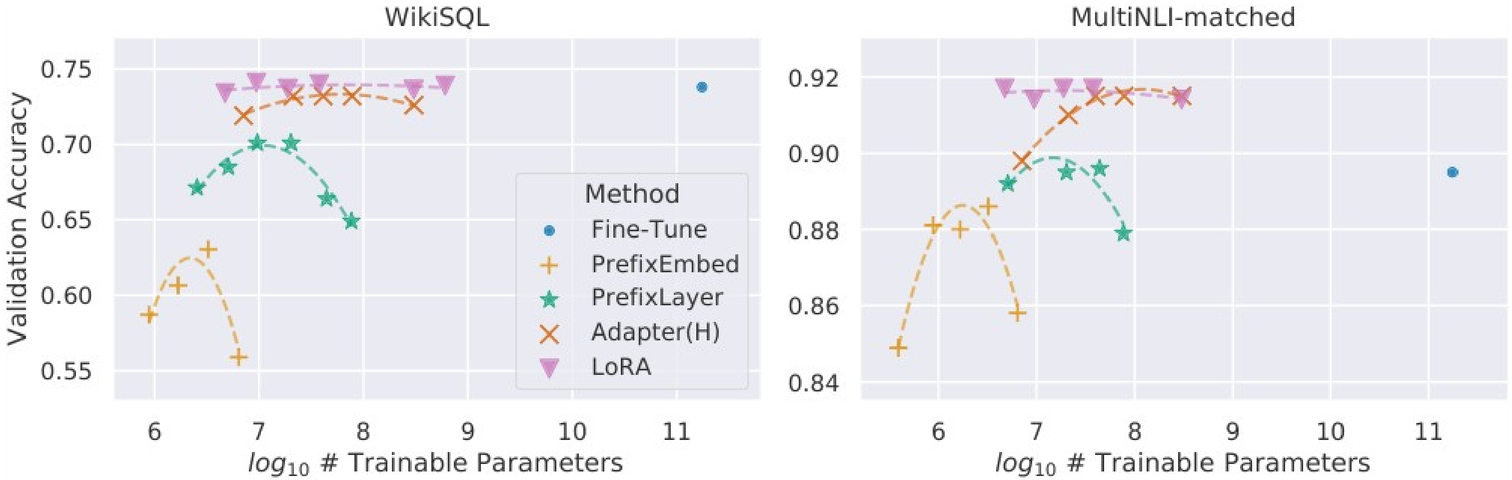
Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
$\mathrm{RoB}_{\mathrm{base}}$ (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
$\mathrm{RoB}_{\mathrm{base}}$ (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
$\mathrm{RoB}_{\mathrm{base}}$ ($\mathrm{Adpt}^{\mathrm{D}}$)*	0.3M	$87.1_{\pm.0}$	$94.2_{\pm.1}$	$88.5_{\pm1.1}$	$60.8_{\pm.4}$	$93.1_{\pm.1}$	$90.2_{\pm.0}$	$71.5_{\pm2.7}$	$89.7_{\pm.3}$	84.4
$\mathrm{RoB}_{\mathrm{base}}$ ($\mathrm{Adpt}^{\mathrm{D}}$)*	0.9M	$87.3_{\pm.1}$	$94.7_{\pm.3}$	$88.4_{\pm.1}$	$62.6_{\pm.9}$	$93.0_{\pm.2}$	$90.6_{\pm.0}$	$75.9_{\pm2.2}$	$90.3_{\pm.1}$	85.4
$\mathrm{RoB}_{\mathrm{base}}$ (LoRA)	0.3M	$87.5_{\pm.3}$	$\mathbf{95.1}_{\pm.2}$	$89.7_{\pm.7}$	$63.4_{\pm1.2}$	$\mathbf{93.3}_{\pm.3}$	$90.8_{\pm.1}$	$\mathbf{86.6}_{\pm.7}$	$\mathbf{91.5}_{\pm.2}$	**87.2**
$\mathrm{RoB}_{\mathrm{large}}$ (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
$\mathrm{RoB}_{\mathrm{large}}$ (LoRA)	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.9}_{\pm1.2}$	$\mathbf{68.2}_{\pm1.9}$	$\mathbf{94.9}_{\pm.3}$	$91.6_{\pm.1}$	$\mathbf{87.4}_{\pm2.5}$	$\mathbf{92.6}_{\pm.2}$	**89.0**
$\mathrm{RoB}_{\mathrm{large}}$ ($\mathrm{Adpt}^{\mathrm{P}}$)†	3.0M	$90.2_{\pm.3}$	$96.1_{\pm.3}$	$90.2_{\pm.7}$	$\mathbf{68.3}_{\pm1.0}$	$\mathbf{94.8}_{\pm.2}$	$\mathbf{91.9}_{\pm.1}$	$83.8_{\pm2.9}$	$92.1_{\pm.7}$	88.4
$\mathrm{RoB}_{\mathrm{large}}$ ($\mathrm{Adpt}^{\mathrm{P}}$)†	0.8M	$\mathbf{90.5}_{\pm.3}$	$\mathbf{96.6}_{\pm.2}$	$89.7_{\pm1.2}$	$67.8_{\pm2.5}$	$\mathbf{94.8}_{\pm.3}$	$91.7_{\pm.2}$	$80.1_{\pm2.9}$	$91.9_{\pm.4}$	87.9
$\mathrm{RoB}_{\mathrm{large}}$ ($\mathrm{Adpt}^{\mathrm{H}}$)†	6.0M	$89.9_{\pm.5}$	$96.2_{\pm.3}$	$88.7_{\pm2.9}$	$66.5_{\pm4.4}$	$94.7_{\pm.2}$	$92.1_{\pm.1}$	$83.4_{\pm1.1}$	$91.0_{\pm1.7}$	87.8
$\mathrm{RoB}_{\mathrm{large}}$ ($\mathrm{Adpt}^{\mathrm{H}}$)†	0.8M	$90.3_{\pm.3}$	$96.3_{\pm.5}$	$87.7_{\pm1.7}$	$66.3_{\pm2.0}$	$94.7_{\pm.2}$	$91.5_{\pm.1}$	$72.9_{\pm2.9}$	$91.5_{\pm.5}$	86.4
$\mathrm{RoB}_{\mathrm{large}}$ (LoRA)†	0.8M	$\mathbf{90.6}_{\pm.2}$	$96.2_{\pm.5}$	$\mathbf{90.2}_{\pm1.0}$	$68.2_{\pm1.9}$	$\mathbf{94.8}_{\pm.3}$	$91.6_{\pm.2}$	$\mathbf{85.2}_{\pm1.1}$	$\mathbf{92.3}_{\pm.5}$	**88.6**
$\mathrm{DeB}_{\mathrm{XXL}}$ (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
$\mathrm{DeB}_{\mathrm{XXL}}$ (LoRA)	4.7M	$\mathbf{91.9}_{\pm.2}$	$96.9_{\pm.2}$	$\mathbf{92.6}_{\pm.6}$	$\mathbf{72.4}_{\pm1.1}$	$\mathbf{96.0}_{\pm.1}$	$\mathbf{92.9}_{\pm.1}$	$\mathbf{94.9}_{\pm.4}$	$\mathbf{93.0}_{\pm.2}$	**91.3**

It works on larger models

Model & Method	# Trainable Parameters	E2E NLG Challenge				
		BLEU	NIST	MET	ROUGE-L	CIDEr
GPT-2 M (FT)*	354.92M	68.2	8.62	46.2	71.0	2.47
GPT-2 M (Adapter^{L})*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M (Adapter^{L})*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M (Adapter^{H})	11.09M	$67.3_{\pm.6}$	$8.50_{\pm.07}$	$46.0_{\pm.2}$	$70.7_{\pm.2}$	$2.44_{\pm.01}$
GPT-2 M (FT^{Top2})*	25.19M	68.1	8.59	46.0	70.8	2.41
GPT-2 M (PreLayer)*	0.35M	69.7	8.81	46.1	71.4	2.49
GPT-2 M (LoRA)	0.35M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.85}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{71.8}_{\pm.1}$	$\mathbf{2.53}_{\pm.02}$
GPT-2 L (FT)*	774.03M	68.5	8.78	46.0	69.9	2.45
GPT-2 L (Adapter^{L})	0.88M	$69.1_{\pm.1}$	$8.68_{\pm.03}$	$46.3_{\pm.0}$	$71.4_{\pm.2}$	$\mathbf{2.49}_{\pm.0}$
GPT-2 L (Adapter^{L})	23.00M	$68.9_{\pm.3}$	$8.70_{\pm.04}$	$46.1_{\pm.1}$	$71.3_{\pm.2}$	$2.45_{\pm.02}$
GPT-2 L (PreLayer)*	0.77M	70.3	8.85	46.2	71.7	2.47
GPT-2 L (LoRA)	0.77M	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

It works on even larger models

Model&Method	# Trainable Parameters	WikiSQL Acc. (%)	MNLI-m Acc. (%)	SAMSum R1/R2/RL
GPT-3 (FT)	175,255.8M	**73.8**	89.5	52.0/28.0/44.5
GPT-3 (BitFit)	14.2M	71.3	91.0	51.3/27.4/43.5
GPT-3 (PreEmbed)	3.2M	63.1	88.6	48.3/24.2/40.5
GPT-3 (PreLayer)	20.2M	70.1	89.5	50.8/27.3/43.5
GPT-3 (Adapter[H])	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 (Adapter[H])	40.1M	73.2	**91.5**	53.2/29.0/45.1
GPT-3 (LoRA)	4.7M	73.4	**91.7**	**53.8/29.8/45.9**
GPT-3 (LoRA)	37.7M	**74.0**	**91.6**	53.4/29.2/45.1



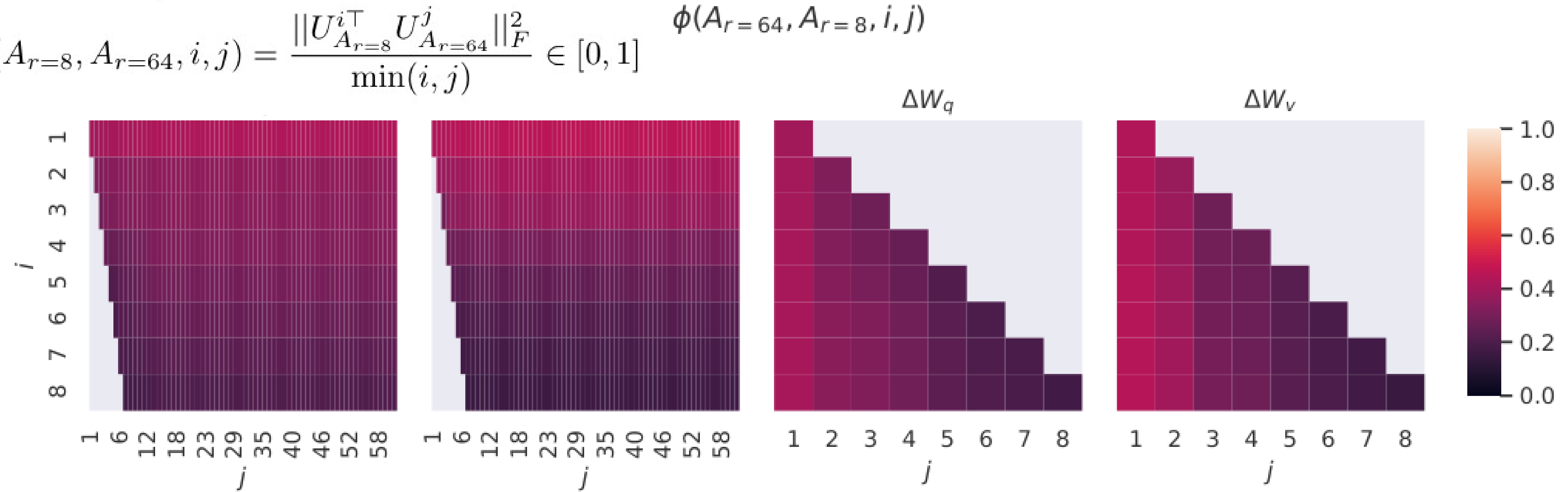
Optimal choice of trainable parameters

	# 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**

	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

How similar are the subspaces spanned by weight matrices of different ranks?

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



- **Similar result across different fine-tuned models, W_q has higher rank than W_v**

Digression: SVD and subspace similarity

- $M = \begin{matrix} 1 & 0 & 2 & 2 & 1 \\ -2 & 1 & -3 & -2 & 0 \\ 3 & -1 & 5 & 4 & 1 \end{matrix}$ $\quad A = \begin{matrix} 1 & 0 & 2 & 2 & 1 \\ -2 & 1 & -3 & -2 & 0 \end{matrix}$ $\quad A' = \begin{matrix} 1 & 0 & 2 & 2 & 1 \\ 2 & 0 & 4 & 4 & 2 \end{matrix}$

- SVD: $M = U \times \Sigma \times V$

- $U_M^1 \times U_A^1 \sim \{\{-1\}\}$
  $\rightarrow \phi(M, A, 1, 1) \sim 1$

  $U_M^1 \times U_A^1 \sim \{\{0.96\}\}$
  $\rightarrow \phi(M, A', 1, 1) \sim 0.92$

- $U_M^3 \times U_A^1 \sim \{\{-1\},\{0.02\},0\}$
  $\rightarrow \phi(M, A, 1, 3) \sim 1$

- $U_M^2 \times U_A^2 \sim \{\{-1,0\},\{0,1\}\}$
  $\rightarrow \phi(M, A, 2, 2) \sim 1$

  $U_M^2 \times U_A^2 \sim \{\{0.96,-0.2\},\{0.27,0.71\}\}$
  $\rightarrow \phi(M, A', 2, 2) \sim 1.54/2 = 0.76$

How does ΔW correlate with W?

- **Project W onto r-dimensional subspace of ΔW**

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

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.

How does ΔW correlate with W?

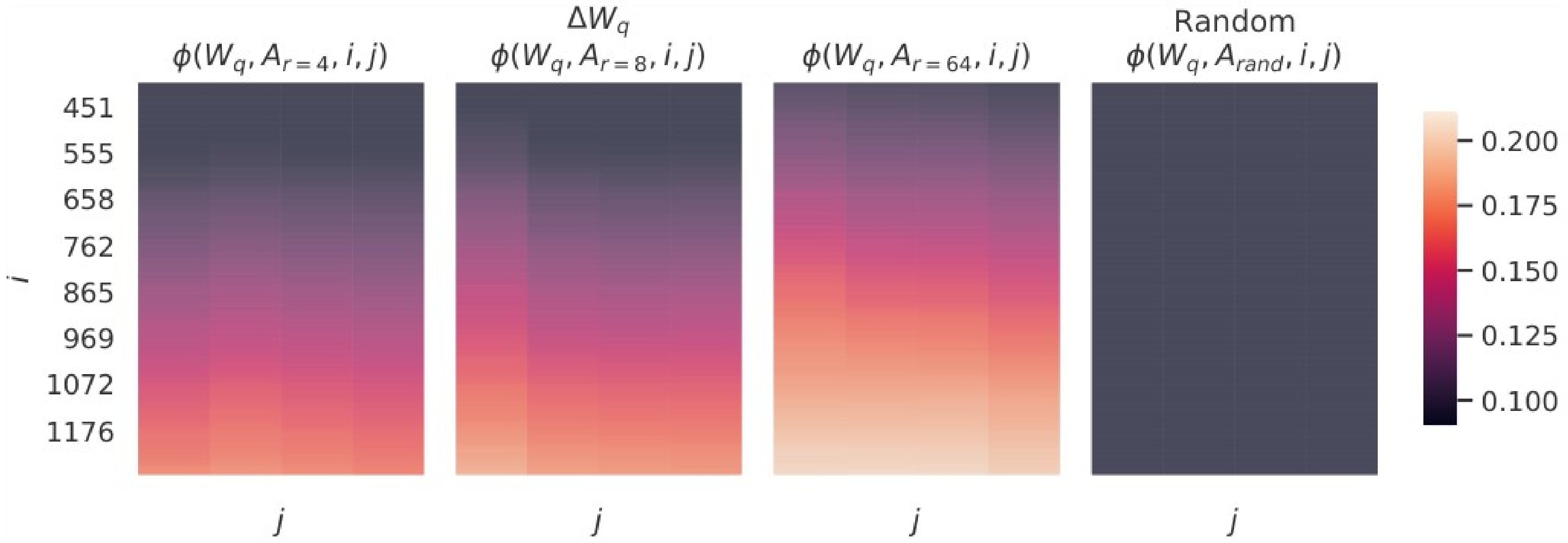


Figure 8: Normalized subspace similarity between the singular directions of W_q and those of ΔW_q with varying r and a random baseline. ΔW_q amplifies directions that are important but not emphasized in W. ΔW with a larger r tends to pick up more directions that are already emphasized in W.