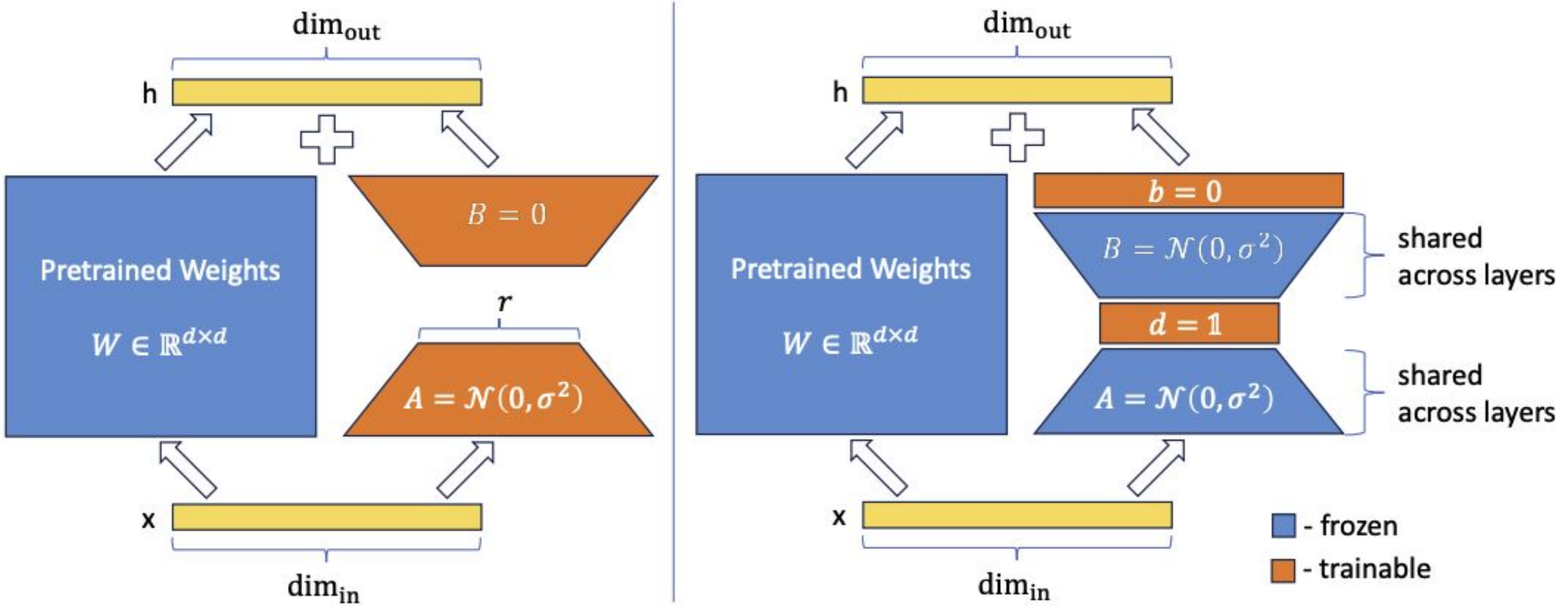
Low Rank methods

Aksenov Yaroslav

VeRA: Vector-based Random Matrix Adaptation

$$h = W_0 x + \Delta W x = W_0 x + \underline{\Lambda_b} B \underline{\Lambda_d} A x$$

A, B – random initialized matrices
b, d – diagonal vectors



VeRA: Vector-based Random Matrix Adaptation

Table 1: Theoretical memory required to store trained VeRA and LoRA weights for $\text{RoBERTa}_{\text{base}}$, $\text{RoBERTa}_{\text{large}}$ and GPT-3 models. We assume that LoRA and VeRA methods are applied on query and key layers of each transformer block.

		LoRA		VeRA	
	Rank	# Trainable Parameters	Required Bytes	# Trainable Parameters	Required Bytes
BASE	1	36.8K	144KB	18.4K	72KB
	16	589.8K	2MB	18.8K	74KB
	256	9437.1K	36MB	24.5K	96KB
LARGE	1	98.3K	384KB	49.2K	192KB
	16	1572.8K	6MB	49.5K	195KB
	256	25165.8K	96MB	61.4K	240KB
GPT-3	1	4.7M	18MB	2.4M	9.1MB
	16	75.5M	288MB	2.8M	10.5MB
	256	1207.9M	4.6GB	8.7M	33MB

VeRA: Vector-based Random Matrix Adaptation

GLUE — General Language Understanding Evaluation benchmark

	Method	# Trainable Parameters	SST-2	MRPC	CoLA	QNLI	RTE	STS-B	Avg.
BASE	FT	125M	94.8	90.2	63.6	92.8	78.7	91.2	85.2
	BitFit	0.1M	93.7	**92.7**	62.0	91.8	81.5	90.8	85.4
	Adpt^{D}	0.3M	$94.2_{\pm 0.1}$	$88.5_{\pm 1.1}$	$60.8_{\pm 0.4}$	$93.1_{\pm 0.1}$	$71.5_{\pm 2.7}$	$89.7_{\pm 0.3}$	83.0
	Adpt^{D}	0.9M	$94.7_{\pm 0.3}$	$88.4_{\pm 0.1}$	$62.6_{\pm 0.9}$	$93.0_{\pm 0.2}$	$75.9_{\pm 2.2}$	$90.3_{\pm 0.1}$	84.2
	LoRA	0.3M	$\mathbf{95.1}_{\pm 0.2}$	$89.7_{\pm 0.7}$	$63.4_{\pm 1.2}$	$\mathbf{93.3}_{\pm 0.3}$	$\mathbf{86.6}_{\pm 0.7}$	$\mathbf{91.5}_{\pm 0.2}$	**86.6**
	VeRA	**0.043M**	$94.6_{\pm 0.1}$	$89.5_{\pm 0.5}$	$\mathbf{65.6}_{\pm 0.8}$	$91.8_{\pm 0.2}$	$78.7_{\pm 0.7}$	$90.7_{\pm 0.2}$	85.2
LARGE	Adpt^{P}	3M	$96.1_{\pm 0.3}$	$90.2_{\pm 0.7}$	$\mathbf{68.3}_{\pm 1.0}$	$\mathbf{94.8}_{\pm 0.2}$	$83.8_{\pm 2.9}$	$92.1_{\pm 0.7}$	87.6
	Adpt^{P}	0.8M	$\mathbf{96.6}_{\pm 0.2}$	$89.7_{\pm 1.2}$	$67.8_{\pm 2.5}$	$\mathbf{94.8}_{\pm 0.3}$	$80.1_{\pm 2.9}$	$91.9_{\pm 0.4}$	86.8
	Adpt^{H}	6M	$96.2_{\pm 0.3}$	$88.7_{\pm 2.9}$	$66.5_{\pm 4.4}$	$94.7_{\pm 0.2}$	$83.4_{\pm 1.1}$	$91.0_{\pm 1.7}$	86.8
	Adpt^{H}	0.8M	$96.3_{\pm 0.5}$	$87.7_{\pm 1.7}$	$66.3_{\pm 2.0}$	$94.7_{\pm 0.2}$	$72.9_{\pm 2.9}$	$91.5_{\pm 0.5}$	84.9
	LoRA-FA	3.7M	96.0	90.0	68.0	94.4	86.1	92.0	87.7
	LoRA	0.8M	$96.2_{\pm 0.5}$	$90.2_{\pm 1.0}$	$68.2_{\pm 1.9}$	$\mathbf{94.8}_{\pm 0.3}$	$85.2_{\pm 1.1}$	$\mathbf{92.3}_{\pm 0.5}$	**87.8**
	VeRA	**0.061M**	$96.1_{\pm 0.1}$	$\mathbf{90.9}_{\pm 0.7}$	$68.0_{\pm 0.8}$	$94.4_{\pm 0.2}$	$\mathbf{85.9}_{\pm 0.7}$	$91.7_{\pm 0.8}$	**87.8**

VeRA: Vector-based Random Matrix Adaptation

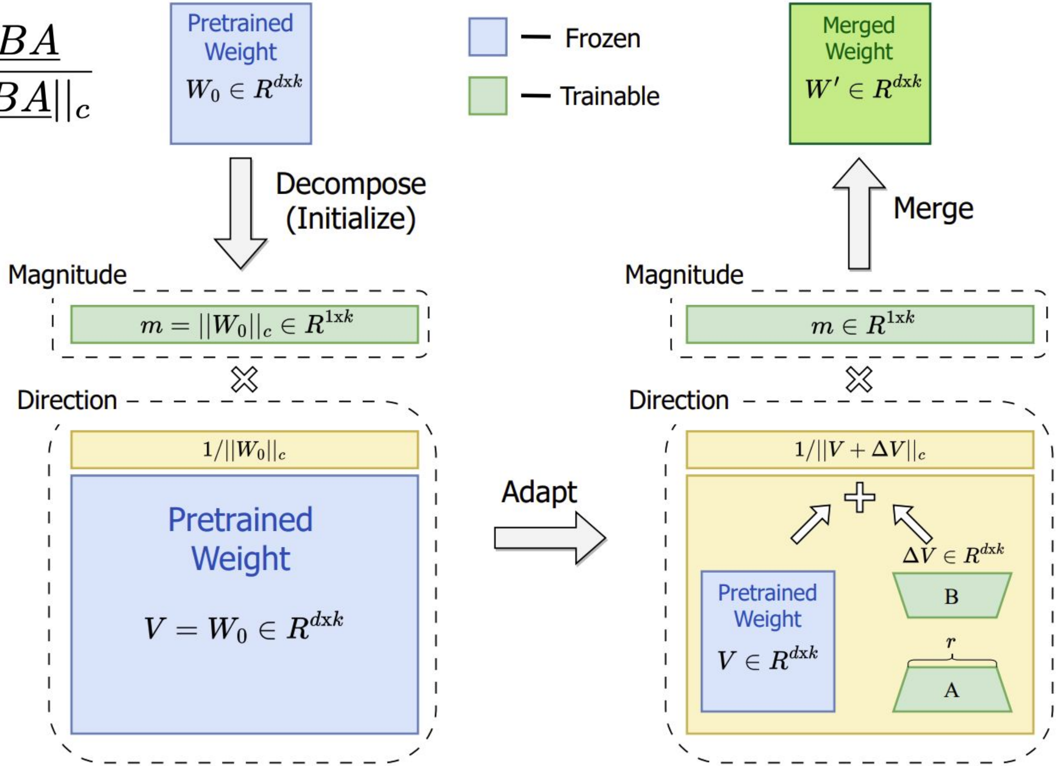
E2E dataset

	Method	# Trainable Parameters	BLEU	NIST	METEOR	ROUGE-L	CIDEr
MEDIUM	FT[1]	354.92M	68.2	8.62	46.2	71.0	2.47
	Adpt[L1]	0.37M	66.3	8.41	45.0	69.8	2.40
	Adpt[L1]	11.09M	68.9	8.71	46.1	71.3	2.47
	Adpt[H1]	11.09M	67.3	8.50	46.0	70.7	2.44
	DyLoRA[2]	0.39M	69.2	8.75	46.3	70.8	2.46
	AdaLoRA[3]	0.38M	68.2	8.58	44.1	70.7	2.35
	LoRA	0.35M	68.9	8.69	46.4	71.3	**2.51**
	VeRA	**0.098M**	**70.1**	**8.81**	**46.6**	**71.5**	2.50
LARGE	FT[1]	774.03M	68.5	8.78	46.0	69.9	2.45
	Adpt[L1]	0.88M	69.1	8.68	46.3	71.4	2.49
	Adpt[L1]	23.00M	68.9	8.70	46.1	71.3	2.45
	LoRA	0.77M	70.1	8.80	46.7	**71.9**	2.52
	VeRA	**0.17M**	**70.3**	**8.85**	**46.9**	71.6	**2.54**

DoRA: Weight-Decomposed Low-Rank Adaptation

$$W' = \underline{m}\frac{V + \Delta V}{||V + \Delta V||_c} = \underline{m}\frac{W_0 + \underline{BA}}{||W_0 + \underline{BA}||_c}$$

$$m = ||W_0||_c \text{ and } V = W_0$$



DoRA: Weight-Decomposed Low-Rank Adaptation

Table 1. Accuracy comparison of LLaMA 7B/13B with various PEFT methods on eight commonsense reasoning datasets. Results of all the baseline methods are taken from (Hu et al., 2023). DoRA†: the adjusted version of DoRA with the rank halved.

Model	PEFT Method	# Params (%)	BoolQ	PIQA	SIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
ChatGPT	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
LLaMA-7B	Prefix	0.11	64.3	76.8	73.9	42.1	72.1	72.9	54.0	60.6	64.6
	Series	0.99	63.0	79.2	76.3	67.9	75.7	74.5	57.1	72.4	70.8
	Parallel	3.54	67.9	76.4	78.8	69.8	78.9	73.7	57.3	75.2	72.2
	LoRA	0.83	68.9	80.7	77.4	78.1	78.8	77.8	61.3	74.8	74.7
	DoRA† (Ours)	0.43	70.0	82.6	79.7	83.2	80.6	80.6	65.4	77.6	**77.5**
	DoRA (Ours)	0.84	68.5	82.9	79.6	84.8	80.8	81.4	65.8	81.0	**78.1**
LLaMA-13B	Prefix	0.03	65.3	75.4	72.1	55.2	68.6	79.5	62.9	68.0	68.4
	Series	0.80	71.8	83	79.2	88.1	82.4	82.5	67.3	81.8	79.5
	Parallel	2.89	72.5	84.9	79.8	92.1	84.7	84.2	71.2	82.4	81.4
	LoRA	0.67	72.1	83.5	80.5	90.5	83.7	82.8	68.3	82.4	80.5
	DoRA† (Ours)	0.35	72.5	85.3	79.9	90.1	82.9	82.7	69.7	83.6	**80.8**
	DoRA (Ours)	0.68	72.4	84.9	81.5	92.4	84.2	84.2	69.6	82.8	**81.5**

DoRA: Weight-Decomposed Low-Rank Adaptation

Table 5. Average scores on MT-Bench assigned by GPT-4 to the answers generated by fine-tuned LLaMA-7B/LLaMA2-7B.

Model	PEFT Method	# Params (%)	Score
LLaMA-7B	LoRA	2.31	5.1
	DoRA (Ours)	2.33	**5.5**
	VeRA	0.02	4.3
	DVoRA (Ours)	0.04	**5.0**
LLaMA2-7B	LoRA	2.31	5.7
	DoRA (Ours)	2.33	**6.0**
	VeRA	0.02	5.5
	DVoRA (Ours)	0.04	**6.0**

Hydra: Multi-head Low-rank Adaptation for Parameter Efficient Fine-tuning

$$
\begin{aligned}
h &= f(x; W_0, b_0) + g(x; A) + g(f(x; W_0, b_0); B) \\
&= W_0 x + b_0 + A_{\text{up}} A_{\text{down}} x \\
&\quad + B_{\text{up}} B_{\text{down}} W_0 x + B_{\text{up}} B_{\text{down}} b_0,
\end{aligned}
$$

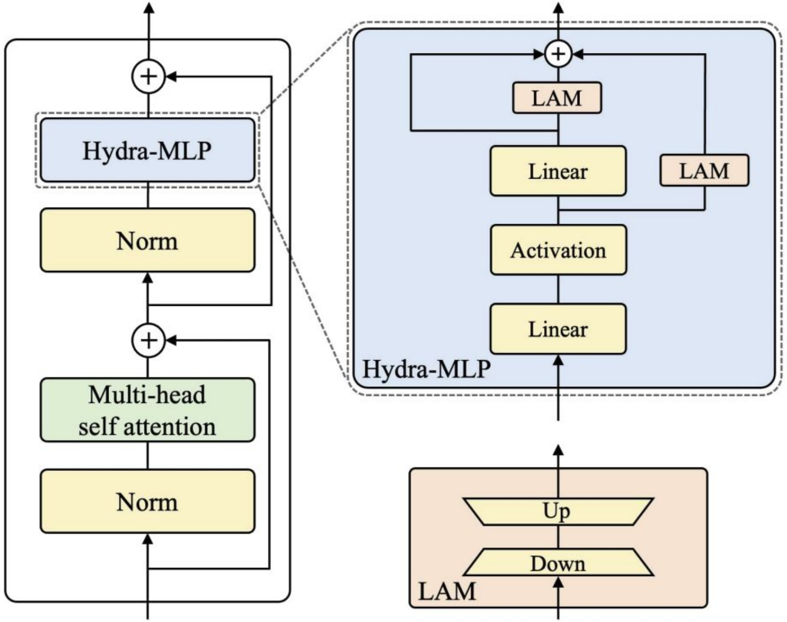


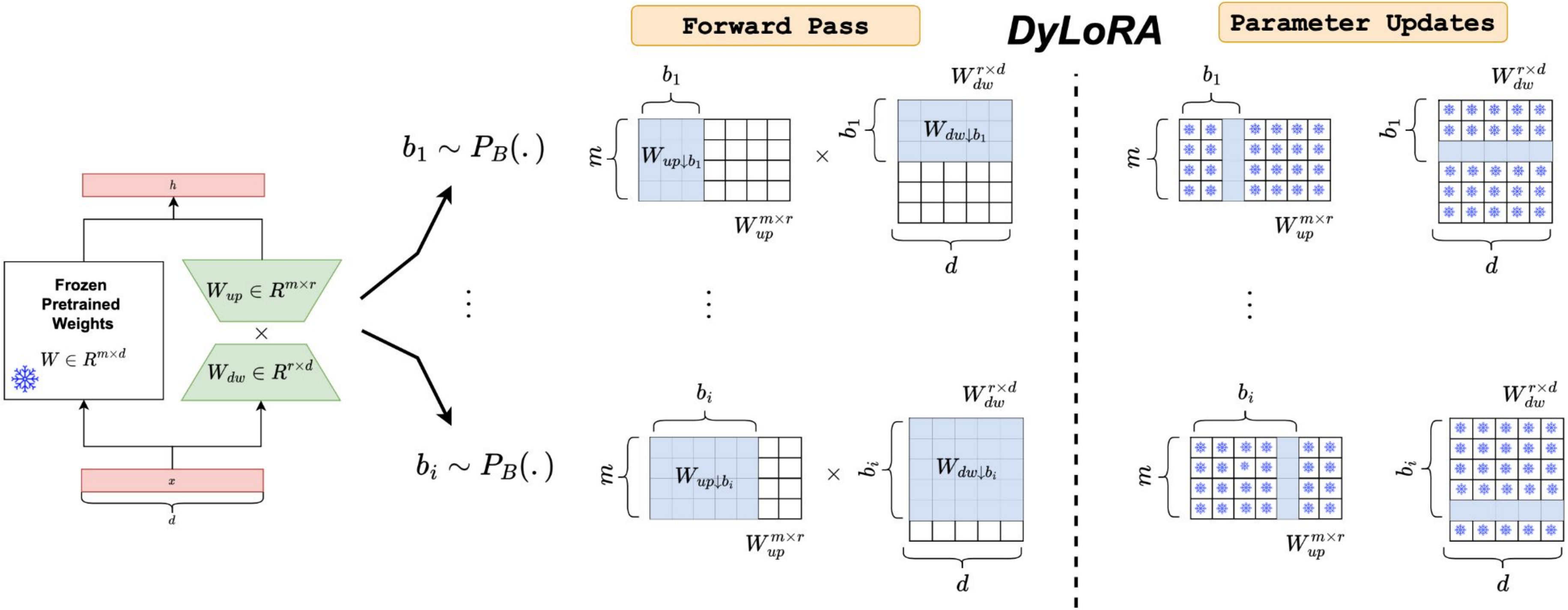
Figure 2. *Hydra*-MLP in a transformer architecture in the training phase. Linear Adapter Module (LAM) implements down projection and up projection on its input in order.

Hydra: Multi-head Low-rank Adaptation for Parameter Efficient Fine-tuning

Method	#Params (M)	Avg.	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
Full tuning	125	86.4	**87.6**	94.8	90.2	63.6	92.8	**91.9**	78.7	91.2
BitFit [80]	0.1	85.2	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8
AdapterDrop [64]	0.3	84.4	87.1	94.2	88.5	60.8	93.1	90.2	71.5	89.7
AdapterDrop [64]	0.9	85.4	87.3	94.7	88.4	62.6	93.0	90.6	75.9	90.3
LoRA [33]	0.3	87.2	87.5	**95.1**	89.7	63.4	**93.3**	90.8	86.6	91.5
Hydra	0.3	**87.9**	87.5	95.0	92.2	**65.4**	92.8	90.8	**87.4**	**91.7**

Table 3. Natural language understanding results. We report the Matthew's correlation for CoLA, Pearson correlation for STS-B, and accuracy for the others.

DyLoRA: Parameter-Efficient Tuning of Pretrained Models using Dynamic Search-Free Low Rank Adaptation



DyLoRA: Parameter-Efficient Tuning of Pretrained Models using Dynamic Search-Free Low Rank Adaptation

Algorithm 1 DyLoRA - Training

Require:
$r \in$Range$[r_{min}, r_{max}]$; i: the number of training iterations; α: a scaling factor; p_B: probability distribution function for rank selection; $X \in \mathbb{R}^{d \times n}$: all input features to LORA; $W_0 \in \mathbb{R}^{m \times d}$ the original frozen pretrained weight matrix

Require: $W_{dw} \in \mathbb{R}^{r \times d}$; $W_{up} \in \mathbb{R}^{m \times r}$, FROZEN: whether to keep the lower ranks frozen when updating the higher ranks

while t < i **do**:

Forward:

// sample a specific rank, during test is given

$b \sim p_B(.)$

// truncate down-projection matrix

$W_{dw\downarrow b} = W_{dw}[:b,:]$

$W^b_{dw} = W_{dw}[b,:]$

// truncate up-projection matrix

$W_{up\downarrow b} = W_{up}[:,:b]$

$W^b_{up} = W_{up}[:,b]$

// calculate the LoRA output

$h = W_0 X + \frac{\alpha}{b} W_{up\downarrow b} W_{dw\downarrow b} X$

Backward:

if FROZEN **then**

// only update the unique parameters of the selected rank

$W^b_{dw} \leftarrow W^b_{dw} - \eta \nabla_{W^b_{dw}} \mathcal{L}^{\mathcal{DY}}_{\downarrow b}$

$W^b_{up} \leftarrow W^b_{up} - \eta \nabla_{W^b_{up}} \mathcal{L}^{\mathcal{DY}}_{\downarrow b}$

else

$W_{dw\downarrow b} \leftarrow W_{dw\downarrow b} - \eta \nabla_{W^b_{dw\downarrow b}} \mathcal{L}^{\mathcal{DY}}_{\downarrow b}$

$W_{up\downarrow b} \leftarrow W_{up\downarrow b} - \eta \nabla_{W^b_{up\downarrow b}} \mathcal{L}^{\mathcal{DY}}_{\downarrow b}$

end if

end while

	Accuracy	Accuracy	F1	Mathews	Accuracy	Accuracy	Accuracy	Pearson	
Model	**MNLI**	**SST-2**	**MRPC**	**CoLA**	**QNLI**	**QQP**	**RTE**	**STS-B**	**Avg**
Rank = 1									
LoRA	$34.60_{\pm 3.69}$	$69.61_{\pm 7.99}$	$83.47_{\pm 3.90}$	$25.57_{\pm 9.71}$	$53.00_{\pm 2.95}$	$44.30_{\pm 7.50}$	$57.55_{\pm 5.51}$	$76.07_{\pm 6.06}$	54.90
DyLoRA (Frozen)	$85.36_{\pm 0.26}$	$93.51_{\pm 0.49}$	$90.75_{\pm 0.70}$	$56.95_{\pm 1.54}$	$91.70_{\pm 0.28}$	$87.87_{\pm 0.17}$	$66.79_{\pm 8.54}$	$89.95_{\pm 0.24}$	82.86
DyLoRA	$85.59_{\pm 0.07}$	$93.23_{\pm 0.63}$	$91.58_{\pm 0.69}$	$57.93_{\pm 2.12}$	$91.95_{\pm 0.14}$	$88.37_{\pm 0.15}$	$74.80_{\pm 1.48}$	$90.30_{\pm 0.13}$	**84.22**
Rank = 2									
LoRA	$40.53_{\pm 6.17}$	$82.75_{\pm 5.08}$	$88.00_{\pm 1.81}$	$43.30_{\pm 4.67}$	$63.42_{\pm 2.99}$	$59.21_{\pm 6.13}$	$68.88_{\pm 1.26}$	$85.51_{\pm 1.94}$	66.45
DyLoRA (Frozen)	$85.74_{\pm 0.28}$	$93.76_{\pm 0.52}$	$91.09_{\pm 0.45}$	$56.88_{\pm 2.09}$	$92.03_{\pm 0.22}$	$88.21_{\pm 0.07}$	$63.90_{\pm 12.85}$	$90.25_{\pm 0.15}$	82.73
DyLoRA	$86.02_{\pm 0.06}$	$93.81_{\pm 0.30}$	$91.66_{\pm 0.46}$	$59.91_{\pm 1.88}$	$92.39_{\pm 0.25}$	$89.33_{\pm 0.05}$	$76.03_{\pm 1.61}$	$90.60_{\pm 0.09}$	**84.97**
Rank = 3									
LoRA	$58.95_{\pm 6.02}$	$90.00_{\pm 1.27}$	$89.66_{\pm 1.25}$	$56.78_{\pm 1.88}$	$79.26_{\pm 4.80}$	$72.58_{\pm 4.09}$	$72.49_{\pm 2.30}$	$88.80_{\pm 0.29}$	76.07
DyLoRA (Frozen)	$85.78_{\pm 0.25}$	$93.76_{\pm 0.26}$	$91.78_{\pm 0.89}$	$58.86_{\pm 0.32}$	$92.17_{\pm 0.18}$	$88.40_{\pm 0.0}$	$70.90_{\pm 6.14}$	$90.50_{\pm 0.29}$	84.02
DyLoRA	$86.70_{\pm 0.09}$	$94.11_{\pm 0.33}$	$91.56_{\pm 0.86}$	$60.97_{\pm 2.01}$	$92.77_{\pm 0.21}$	$89.76_{\pm 0.07}$	$77.11_{\pm 2.97}$	$90.69_{\pm 0.14}$	**85.46**
Rank = 4									
LoRA	$72.10_{\pm 5.25}$	$91.56_{\pm 0.34}$	$89.62_{\pm 0.92}$	$58.53_{\pm 3.93}$	$85.09_{\pm 1.20}$	$80.78_{\pm 3.73}$	$73.07_{\pm 2.29}$	$89.28_{\pm 0.72}$	80.00
DyLoRA (Frozen)	$85.93_{\pm 0.19}$	$93.85_{\pm 0.33}$	$91.28_{\pm 0.71}$	$59.25_{\pm 1.05}$	$92.27_{\pm 0.16}$	$88.52_{\pm 0.08}$	$71.12_{\pm 2.46}$	$90.53_{\pm 0.18}$	84.10
DyLoRA	$86.82_{\pm 0.04}$	$94.40_{\pm 0.13}$	$92.06_{\pm 0.46}$	$59.81_{\pm 1.71}$	$92.91_{\pm 0.31}$	$89.80_{\pm 0.10}$	$77.40_{\pm 2.72}$	$90.86_{\pm 0.06}$	**85.53**
Rank = 5									
LoRA	$78.61_{\pm 3.97}$	$92.82_{\pm 0.46}$	$90.75_{\pm 0.96}$	$60.37_{\pm 3.10}$	$88.97_{\pm 0.90}$	$85.26_{\pm 1.56}$	$73.21_{\pm 2.17}$	$89.90_{\pm 0.30}$	82.49
DyLoRA (Frozen)	$85.95_{\pm 0.17}$	$93.78_{\pm 0.26}$	$91.28_{\pm 0.64}$	$59.41_{\pm 1.30}$	$92.30_{\pm 0.17}$	$88.56_{\pm 0.09}$	$71.48_{\pm 2.92}$	$90.60_{\pm 0.20}$	84.17
DyLoRA	$87.00_{\pm 0.10}$	$94.29_{\pm 0.41}$	$91.73_{\pm 0.60}$	$60.52_{\pm 1.07}$	$93.01_{\pm 0.28}$	$90.04_{\pm 0.10}$	$76.90_{\pm 2.11}$	$90.97_{\pm 0.20}$	**85.56**
Rank = 6									
LoRA	$83.02_{\pm 1.59}$	$93.49_{\pm 0.88}$	$91.28_{\pm 0.63}$	$61.94_{\pm 2.27}$	$90.32_{\pm 0.76}$	$87.54_{\pm 1.51}$	$76.68_{\pm 1.16}$	$90.12_{\pm 0.12}$	84.30
DyLoRA (Frozen)	$85.98_{\pm 0.16}$	$93.76_{\pm 0.46}$	$91.12_{\pm 0.43}$	$58.95_{\pm 1.10}$	$92.46_{\pm 0.14}$	$88.68_{\pm 0.13}$	$72.64_{\pm 2.44}$	$90.64_{\pm 0.23}$	84.28
DyLoRA	$86.97_{\pm 0.20}$	$94.27_{\pm 0.37}$	$91.44_{\pm 0.64}$	$60.16_{\pm 1.70}$	$93.01_{\pm 0.21}$	$90.07_{\pm 0.14}$	$77.33_{\pm 1.66}$	$91.03_{\pm 0.20}$	**85.53**
Rank = 7									
LoRA	$85.44_{\pm 0.78}$	$93.62_{\pm 0.35}$	$91.27_{\pm 0.73}$	$62.19_{\pm 2.66}$	$91.88_{\pm 0.23}$	$89.51_{\pm 0.30}$	$75.52_{\pm 1.41}$	$90.35_{\pm 0.24}$	84.97
DyLoRA (Frozen)	$86.08_{\pm 0.14}$	$93.97_{\pm 0.17}$	$91.02_{\pm 0.70}$	$58.76_{\pm 0.94}$	$92.30_{\pm 0.10}$	$88.77_{\pm 0.06}$	$73.50_{\pm 1.67}$	$90.68_{\pm 0.15}$	84.38
DyLoRA	$86.82_{\pm 0.10}$	$94.27_{\pm 0.33}$	$91.38_{\pm 0.59}$	$59.51_{\pm 1.75}$	$92.99_{\pm 0.26}$	$90.04_{\pm 0.06}$	$77.91_{\pm 1.58}$	$91.07_{\pm 0.19}$	**85.50**
Rank = 8									
LoRA	$86.82_{\pm 0.18}$	$94.01_{\pm 0.30}$	$91.48_{\pm 0.73}$	$62.08_{\pm 1.37}$	$92.39_{\pm 0.39}$	$90.42_{\pm 0.02}$	$74.51_{\pm 0.41}$	$90.48_{\pm 0.24}$	85.27
DyLoRA (Frozen)	$86.10_{\pm 0.04}$	$93.69_{\pm 0.41}$	$91.19_{\pm 0.79}$	$58.52_{\pm 0.95}$	$92.47_{\pm 0.18}$	$88.82_{\pm 0.06}$	$73.29_{\pm 2.49}$	$90.68_{\pm 0.14}$	84.35
DyLoRA	$86.76_{\pm 0.13}$	$94.36_{\pm 0.38}$	$91.38_{\pm 0.83}$	$59.51_{\pm 1.84}$	$93.00_{\pm 0.32}$	$89.91_{\pm 0.08}$	$77.55_{\pm 0.59}$	$91.05_{\pm 0.19}$	**85.44**
Best (Rank)									
LoRA	87.03(8)	94.50(6)	92.25(7)	**66.05**(7)	92.81(8)	**90.45**(8)	77.98(6)	90.87(8)	86.49
DyLoRA (Frozen)	86.18(7)	94.50(**2**)	**92.93(3)**	61.57(5)	92.70(**6**)	88.88(8)	75.81(7)	90.89(**6**)	85.43
DyLoRA	**87.17(6)**	**94.72**(7)	92.79(8)	63.32(**3**)	**93.56**(8)	90.17(**6**)	**80.14(4)**	**91.36**(7)	**86.66**
Full Rank									
Fine Tune*	**87.6**	**94.8**	90.2	63.6	92.8	**91.9**	78.7	91.2	86.4

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning

$$W = W^{(0)} + \Delta = W^{(0)} + P\Lambda Q$$

$$R(P, Q) = \|P^\top P - I\|_{\mathsf{F}}^2 + \|QQ^\top - I\|_{\mathsf{F}}^2$$

$$\mathcal{G}_i = \{P_{*i}, \lambda_i, Q_{i*}\}$$

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning

$$\Lambda_k^{(t+1)} = \mathcal{T}(\tilde{\Lambda}_k^{(t)}, S_k^{(t)}), \text{ with } \mathcal{T}(\tilde{\Lambda}_k^{(t)}, S_k^{(t)})_{ii} = \begin{cases} \tilde{\Lambda}_{k,ii}^{(t)} & S_{k,i}^{(t)} \text{ is in the top-}b^{(t)} \text{ of } S^{(t)}, \\ 0 & \text{otherwise,} \end{cases}$$

Singular value magnitude: $S_{k,i} = |\lambda_{k,i}|$

Sensitivity-based importance: $S_{k,i} = s(\lambda_{k,i}) + \frac{1}{d_1}\sum_{j=1}^{d_1} s(P_{k,ji}) + \frac{1}{d_2}\sum_{j=1}^{d_2} s(Q_{k,ij})$

$$I(w_{ij}) = |w_{ij}\nabla_{w_{ij}}\mathcal{L}|$$

$$s^{(t)}(w_{ij}) = \overline{I}^{(t)}(w_{ij}) \cdot \overline{U}^{(t)}(w_{ij})$$

$$\overline{I}^{(t)}(w_{ij}) = \beta_1 \overline{I}^{(t-1)}(w_{ij}) + (1-\beta_1) I^{(t)}(w_{ij})$$

$$\overline{U}^{(t)}(w_{ij}) = \beta_2 \overline{U}^{(t-1)}(w_{ij}) + (1-\beta_2)\left|I^{(t)}(w_{ij}) - \overline{I}^{(t)}(w_{ij})\right|$$

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning

Algorithm 1 AdaLoRA

Input: Dataset $\mathcal{D}$; total iterations T; budget schedule $\{b^{(t)}\}_{t=0}^{T}$; hyperparameters $\eta, \gamma, \beta_1, \beta_2$.
for $t = 1, \ldots, T$ **do**
  Sample a mini-batch from $\mathcal{D}$ and compute the gradient $\nabla \mathcal{L}(\mathcal{P}, \mathcal{E}, \mathcal{Q})$;
  Compute the sensitivity $I^{(t)}$ in (8) for every parameter in $\{\mathcal{P}, \mathcal{E}, \mathcal{Q}\}$;
  Update $\overline{I}^{(t)}$ as (9) and $\overline{U}^{(t)}$ as (10) for every parameter in $\{\mathcal{P}, \mathcal{E}, \mathcal{Q}\}$;
  Compute $S_{k,i}^{(t)}$ by (7), for $k = 1, \ldots, n$ and $i = 1, \ldots, r$;
  Update $P_k^{(t+1)} = P_k^{(t)} - \eta \nabla_{P_k} \mathcal{L}(\mathcal{P}, \mathcal{E}, \mathcal{Q})$ and $Q_k^{(t+1)} = Q_k^{(t)} - \eta \nabla_{Q_k} \mathcal{L}(\mathcal{P}, \mathcal{E}, \mathcal{Q})$;
  Update $\Lambda_k^{(t+1)} = \mathcal{T}(\Lambda_k^{(t)} - \eta \nabla_{\Lambda_k} \mathcal{L}(\mathcal{P}, \mathcal{E}, \mathcal{Q}), S_k^{(t)})$ given the budget $b^{(t)}$.
end for
Output: The fine-tuned parameters $\{\mathcal{P}^{(T)}, \mathcal{E}^{(T)}, \mathcal{Q}^{(T)}\}$.

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning

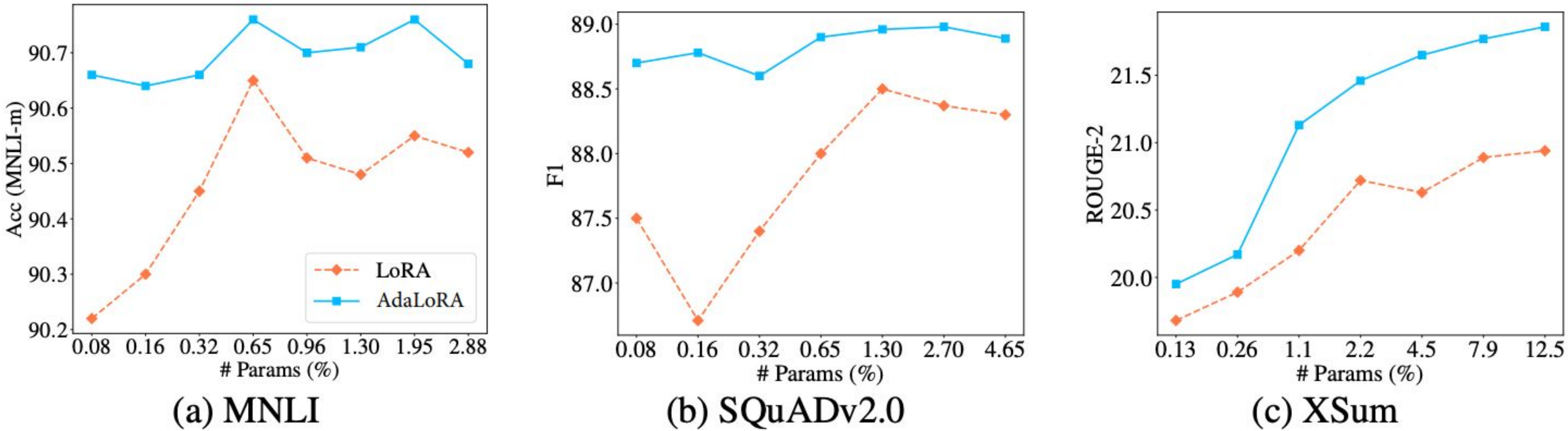
Method	# Params	MNLI m/mm	SST-2 Acc	CoLA Mcc	QQP Acc/F1	QNLI Acc	RTE Acc	MRPC Acc	STS-B Corr	All Ave.
Full FT	184M	89.90/90.12	95.63	69.19	**92.40/89.80**	94.03	83.75	89.46	91.60	88.09
BitFit	0.1M	89.37/89.91	94.84	66.96	88.41/84.95	92.24	78.70	87.75	91.35	86.02
HAdapter	1.22M	90.13/90.17	95.53	68.64	91.91/89.27	94.11	84.48	89.95	91.48	88.12
PAdapter	1.18M	90.33/90.39	95.61	68.77	92.04/89.40	94.29	85.20	89.46	91.54	88.24
$\text{LoRA}_{r=8}$	1.33M	90.65/90.69	94.95	69.82	91.99/89.38	93.87	85.20	89.95	91.60	88.34
AdaLoRA	1.27M	**90.76/90.79**	**96.10**	**71.45**	**92.23/89.74**	**94.55**	**88.09**	**90.69**	**91.84**	**89.31**
HAdapter	0.61M	90.12/90.23	95.30	67.87	91.65/88.95	93.76	85.56	89.22	91.30	87.93
PAdapter	0.60M	90.15/90.28	95.53	69.48	91.62/88.86	93.98	84.12	89.22	91.52	88.04
HAdapter	0.31M	90.10/90.02	95.41	67.65	91.54/88.81	93.52	83.39	89.25	91.31	87.60
PAdapter	0.30M	89.89/90.06	94.72	69.06	91.40/88.62	93.87	84.48	89.71	91.38	87.90
$\text{LoRA}_{r=2}$	0.33M	90.30/90.38	94.95	68.71	91.61/88.91	94.03	85.56	89.71	**91.68**	88.15
AdaLoRA	0.32M	**90.66/90.70**	**95.80**	**70.04**	**91.78/89.16**	**94.49**	**87.36**	**90.44**	91.63	**88.86**

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning

Squad – datasets for question answering and reading comprehension from a set of Wikipedia articles

	SQuADv1.1				SQuADv2.0			
Full FT	86.0 / 92.7				85.4 / 88.4			
# Params	0.08%	0.16%	0.32%	0.65%	0.08%	0.16%	0.32%	0.65%
HAdapter	84.4/91.5	85.3/92.1	86.1/92.7	86.7/92.9	83.4/86.6	84.3/87.3	84.9/87.9	85.4/88.3
PAdapter	84.4/91.7	85.9/92.5	86.2/92.8	86.6/93.0	84.2/87.2	84.5/87.6	84.9/87.8	84.5/87.5
LoRA	86.4/92.8	86.6/92.9	86.7/93.1	86.7/93.1	84.7/87.5	83.6/86.7	84.5/87.4	85.0/88.0
AdaLoRA	**87.2/93.4**	**87.5/93.6**	**87.5/93.7**	**87.6/93.7**	**85.6/88.7**	**85.7/88.8**	**85.5/88.6**	**86.0/88.9**

AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning



GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection

$$G_t = USV^\top \approx \sum_{i=1}^{r} s_i u_i v_i^\top$$

$$P_t = [u_1, u_2, ..., u_r], \quad Q_t = [v_1, v_2, ..., v_r]$$

Definition 3.4 (Gradient Low-rank Projection (**GaLore**)). Gradient low-rank projection (**GaLore**) denotes the following gradient update rules (η is the learning rate):

$$W_T = W_0 + \eta \sum_{t=0}^{T-1} \tilde{G}_t, \qquad \tilde{G}_t = P_t \rho_t (P_t^\top G_t Q_t) Q_t^\top, \tag{11}$$

where $P_t \in \mathbb{R}^{m \times r}$ and $Q_t \in \mathbb{R}^{n \times r}$ are projection matrices.

Algorithm 2: Adam with GaLore

Input: A layer weight matrix $W \in \mathbb{R}^{m \times n}$ with $m \leq n$. Step size η, scale factor α, decay rates β_1, β_2, rank r, subspace change frequency T.

Initialize first-order moment $M_0 \in \mathbb{R}^{n \times r} \leftarrow 0$

Initialize second-order moment $V_0 \in \mathbb{R}^{n \times r} \leftarrow 0$

Initialize step $t \leftarrow 0$

repeat

  $G_t \in \mathbb{R}^{m \times n} \leftarrow -\nabla_W \varphi_t(W_t)$

  if $t \bmod T = 0$ **then**

    $U, S, V \leftarrow \text{SVD}(G_t)$

    $P_t \leftarrow U[:, :r]$ {Initialize left projector as $m \leq n$}

  else

    $P_t \leftarrow P_{t-1}$ {Reuse the previous projector}

  end if

  $R_t \leftarrow P_t^\top G_t$ {Project gradient into compact space}

  UPDATE(R_t) **by Adam**

    $M_t \leftarrow \beta_1 \cdot M_{t-1} + (1 - \beta_1) \cdot R_t$

    $V_t \leftarrow \beta_2 \cdot V_{t-1} + (1 - \beta_2) \cdot R_t^2$

    $M_t \leftarrow M_t / (1 - \beta_1^t)$

    $V_t \leftarrow V_t / (1 - \beta_2^t)$

    $N_t \leftarrow M_t / (\sqrt{V_t} + \epsilon)$

  $\tilde{G}_t \leftarrow \alpha \cdot P N_t$ {Project back to original space}

  $W_t \leftarrow W_{t-1} + \eta \cdot \tilde{G}_t$

  $t \leftarrow t + 1$

until convergence criteria met

return W_t

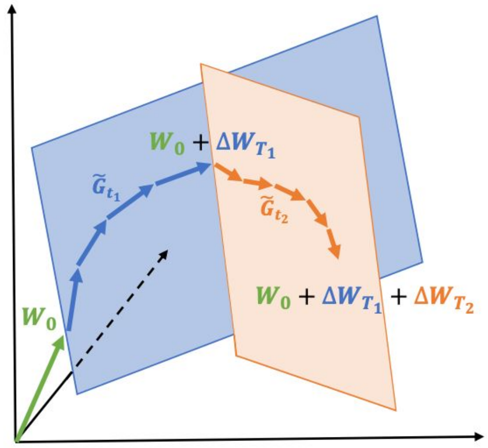


Figure 2: Learning through low-rank subspaces ΔW_{T_1} and ΔW_{T_2} using GaLore. For $t_1 \in [0, T_1 - 1]$, W are updated by projected gradients $\tilde{G}_{t_1}$ in a subspace determined by fixed P_{t_1} and Q_{t_1}. After T_1 steps, the subspace is changed by re-computing P_{t_2} and Q_{t_2} for $t_2 \in [T_1, T_2 - 1]$, and the process repeats until convergence.

GaLore: Memory-Efficient LLM Training by Gradient Low-Rank Projection

	60M	**130M**	**350M**	**1B**
Full-Rank	34.06 (0.36G)	25.08 (0.76G)	18.80 (2.06G)	15.56 (7.80G)
GaLore	**34.88** (0.24G)	**25.36** (0.52G)	**18.95** (1.22G)	**15.64** (4.38G)
Low-Rank	78.18 (0.26G)	45.51 (0.54G)	37.41 (1.08G)	142.53 (3.57G)
LoRA	34.99 (0.36G)	33.92 (0.80G)	25.58 (1.76G)	19.21 (6.17G)
ReLoRA	37.04 (0.36G)	29.37 (0.80G)	29.08 (1.76G)	18.33 (6.17G)
r/d_{model}	128 / 256	256 / 768	256 / 1024	512 / 2048
Training Tokens	1.1B	2.2B	6.4B	13.1B

Вопросы