Cross-Modal Fine-Tuning

Align then Refine

Tran Trong Khiem

AI lab tranning

2024/05/29

Idea

Problem setup:

- Tranformers pre-train body model g_s in domain D^s.
- Dataset $\{x_i^t, y_i^t\}_{i=1}^N$ in domain D^t.
- Dataset $\{x_i^s, y_i^s\}_{i=1}^N$ in domain D^s

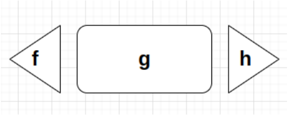


Figure 1: proposed model

embedder: f extract feature from input $f(.)$

- random init for f_t.
- using pre-train and freezing for f_s

predictor: h generate output $h(.)$.

Idea(cnt.)

Proposed workflow:

1. Stage 1 : Embedder training: mitigate modality gap.
2. Stage 2 : Modality gap and rank estimation.
3. Stage 3 : Full fine-tuning using LoRA.

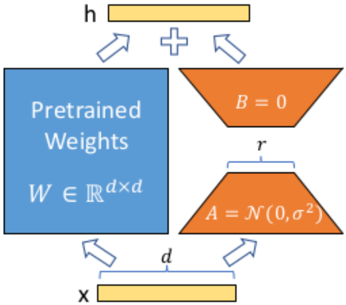


Figure 2: LORA.

Stage 3: LORA training

LORA:

- A pre-trained weight matrix $W_0 \in R^{d \times k}$.
- Low-rank decomposition : $W_0 + \Delta W = W_0 + BA$.
  - $B \in \mathbb{R}^{d \times r}, \quad A \in \mathbb{R}^{r \times k}, \quad$ the rank $r << \min(d, k)$
- During training, W_0 is **frozen**, while A and B contain **trainable parameters**.

Expectation:

- Adapt the rank r depending on the **current modality gap**.
- $r_i = r(D_i)$ in layer i. The larger the modality gap, the larger the rank r.

Modality gap and rank estimation

We have :

- D_i is modaily gap at layer i.
- r_i is rank LORA at layer i.
- $r \sim P(r|D)$.

Problem:

- What hypothesis determines the relationship between the optimal rank r in LoRA and the modality gap D ?
- How to derive $P(r|D)$?

Experiment

Goal: Hypothesis for testing

- The LoRA rank adapted in cross-modality transfer is directly proportional to the modality gap.

Experiment setup two experiments for hypothesis testing:

1. Adapt LORA in ORCA model with different rank r (1,2,4,8,16,64, 128, 256, 512, . . .) in large modality task.
    - Target dataset: Darcy - flow.
    - Source dataset: ImageNet-21k.
2. Adapt LORA in ORCA model with different rank r (1,2,4,8,16,64, 128, 256, 512, . . .) in small modality task.
    - Target dataset : CIFAR100.
    - Source dataset: ImageNet-21k.

Experiment 2

Experiment 2 Setup:

- Source dataset: ImageNet-21k.
- Target dataset: CIFAR100.
- Embedder training: disable.
- Full fine-tune training: 50 epochs.
- LORA rank: 1.

Experiment 2 result:

Models	Trainable params	Prediction errors (↓)
ORCA(100 epochs pp)	90M	0.0653
ORCA(50 epochs)	90M	0.0664
LORA r= 1(50 epochs)	0.18 M	0.0849

Table 1: Prediction errors (↓)

Experiment1

Experiment1 setup:

- Source dataset : ImageNet-21k.
- Target dataset : Darcy-flow.
- Pre-train model : Swin Transformers(90 M params).
- Train embedder : Disable.
- Full fine-tuning : 100 epochs
- Using LoRa with different rank : 1,2,4,8,16,64.

Experiment result1

Experiment result1:

Models	Trainable params	Prediction errors (↓)
ORCA	90M	0.0076
Fine-tune	90M	0.0078
LORA r= 1	0.08 M	0.0968
LORA r= 2	0.15 M	0.0885
LORA r= 4	0.3 M	0.0823
LORA r= 8	0.6 M	0.0797
LORA r= 16	1.2 M	0.0782
LORA r= 64	4.6 M	0.0771
LORA r= 128	9.2 M	0.0771
LORA r= 256	18.4 M	0.0771
LORA r= 512	36.9 M	0.0770
LORA r= 1024	73.9 M	0.0771

Table 2: Prediction errors (↓)

Experiment result3

Experiment result3:

Models	Trainable params	Prediction errors (↓)
ORCA	90M	0.0076
Fine-tune	90M	0.0078
LORA r= 1	3.4 M	0.0104
LORA r= 2	3.6 M	0.0094
LORA r= 4	4 M	0.0086
LORA r= 5	4.23 M	0.0095
LORA r= 6	4.45 M	0.0088
LORA r= 7	4.67 M	0.0087
LORA r= 8	4.8 M	0.0096
LORA r= 16	6.6 M	0.0095
LORA r= 64	17 M	0.0092
LORA r= 128	30.8 M	0.0087

Table 3: Prediction errors (↓)

Experiment1.1

Experiment setup1.1:

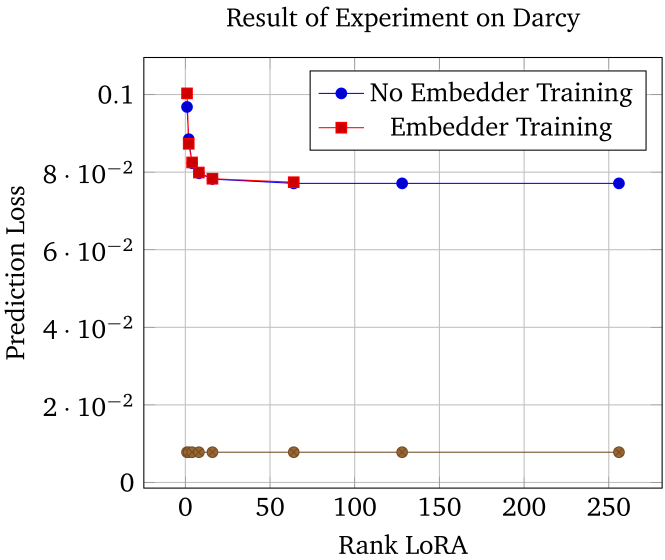
- Source dataset : ImageNet-21k.
- Embedder dataset : CIFAR10.
- Target dataset : Darcy-flow.
- Pre-train model : Swin Transformers(90 M params).
- Train embedder : Using OTDD - 60 epochs(same OTDD loss).
- Full fine-tuning : 100 epochs.
- Using LoRa with different rank : 1,2,4,8,16,64.

Experiment result 1.1

Models	Trainable params	Prediction errors (↓)
ORCA	90M	0.0076
ORCA-LORA r= 1	0.08 M	0.1003
ORCA-LORA r= 2	0.15 M	0.0873
ORCA-LORA r= 4	0.3 M	0.0825
ORCA-LORA r= 8	0.6 M	0.0799
ORCA-LORA r= 16	1.2 M	0.0783
ORCA-LORA r= 64	4.6 M	0.0774

Table 4: Prediction errors (↓)

Conclusion


Result of Experiment on Darcy
No Embedder Training
Embedder Training
Prediction Loss
0.1
$8 \cdot 10^{-2}$
$6 \cdot 10^{-2}$
$4 \cdot 10^{-2}$
$2 \cdot 10^{-2}$
0
0
50
100
150
200
250
Rank LoRA

Conclusion

With large modality:

- With $(r < 64)$, A larger LoRA rank results in better model performance.
- Performance increases slowly from $r = 16$
- Embedder training does not affect model performance.
- With $r > 64$, A larger LoRA rank does not affect model performance.

With small modality:

- with $r = 1$, LORA achieves good performance.

Relatework

Problem: There is a substantial performance gap when comparing LoRA to full fine-tuning.

- Updating only a **fraction of the model's parameters**.
- It inadequate to fit the intricacies presented in the training data.

Delta-LoRA:

- Base on the mathematical property : $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial AB}$

Algorithm 1: Delta-LoRA

Input: Learning rate η; weight decay β; total training iterations T; low rank r; scale factor α; start steps K; update ratio λ.

$\boldsymbol{A}$ is initialized by Kaiming Initialization, $\boldsymbol{B} = \boldsymbol{0}$ and $\boldsymbol{W}$ is initialized with pre-trained weights.

for $t = 0, ..., T-1$ **do**

Sample a mini-batch and compute gradients for $\{\boldsymbol{A},\boldsymbol{B}\}$ in each Delta-LoRA module.

Update the first and second moments maintained by the optimizer with the computed gradients, and get the normalized gradients $\widehat{g}_{\boldsymbol{A}}$ and $\widehat{g}_{\boldsymbol{B}}$.

$\boldsymbol{A}^{(t+1)} \leftarrow \boldsymbol{A}^{(t)} - \eta\widehat{g}_{\boldsymbol{A}} - \eta\beta\boldsymbol{A}^{(t)}$

$\boldsymbol{B}^{(t+1)} \leftarrow \boldsymbol{B}^{(t)} - \eta\widehat{g}_{\boldsymbol{B}} - \eta\beta\boldsymbol{B}^{(t)}$

if $t > K$ **do**

$\boldsymbol{W}^{(t+1)} \leftarrow \boldsymbol{W}^{(t)} + \lambda \cdot \frac{\alpha}{r} \cdot (\boldsymbol{A}^{(t+1)}\boldsymbol{B}^{(t+1)} - \boldsymbol{A}^{(t)}\boldsymbol{B}^{(t)})$

end if

end for

Output: the fine-tuned parameters $\{\boldsymbol{W}^{(T)}, \boldsymbol{A}^{(T)}, \boldsymbol{B}^{(T)}\}$

India buffet process

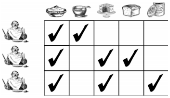


Figure 4: India buffet process

- A stochastics process define a distribution over **infinite binary matrices**.
- Indian restaurants offer buffets with an infinite number of dishes.
- N customers enter a restaurant one after another.
- **First customer** takes first $d_0 \sim Poisson(\alpha)$ dishes.
- $i-$ **th customer** moves along the buffet.
    - Sampling dishes with $p_k = \frac{m_k}{i}$.
    - $m_k$is the number of previous customers who have chosen dish k.
    - tries a $d_k \sim Poisson(\frac{\alpha}{i})$ number of new dishes .

India buffet process

- Indicate which customers chose which dishes using a **binary matrix** Z.
  - N rows and infinitely many columns.
  - $z_{ik} = 1$ if the i-th customer sampled the k-th dish.
- The **probability** of any particular matrix being produced by this process is

$$P(Z) = \frac{\alpha_+^K}{\prod_{i=1}^N K_1^{(i)}!} \exp\left(-\alpha H_N\right) \prod_{k=1}^{K_+} \frac{(N - m_k)!(m_k - 1)!}{N!}$$

  - $H_N = \sum_i^N \frac{1}{i}$
  - m_k is the number of previous customers who have chosen dish k.
  - $K_1^{(i)}$ is the **number of new dishes** sampled by the i-th customer.
  - K_+ is the number of disshes for which $m_k > 0$.

India buffet process

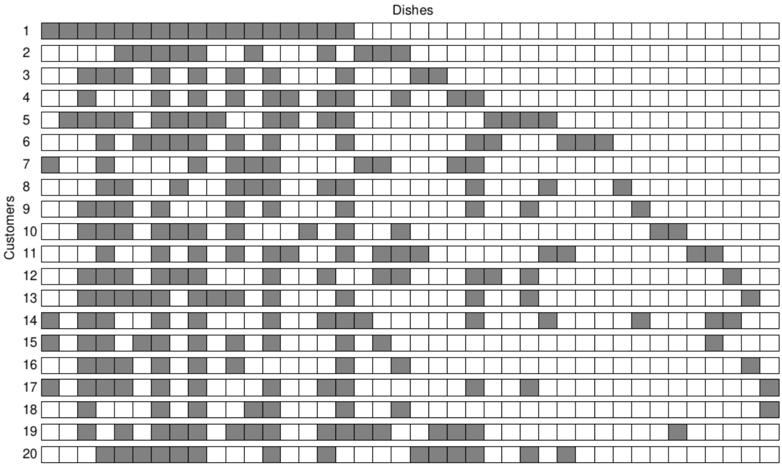


Figure 5: A binary matrix generated by the Indian buffet process with $\alpha = 10$.

India buffet process

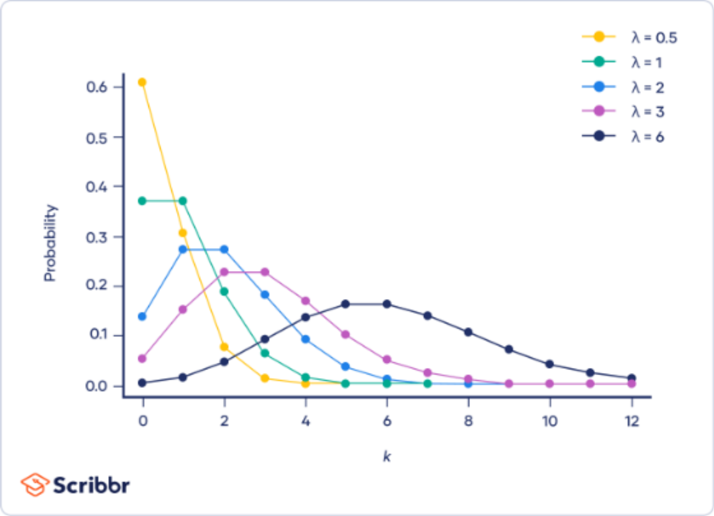


Figure 6: Poisson distribution

INTRODUCTION

Goal: Fine tuning by **adapting only some parameters for new tasks**.

- Store and load a **small number** of task-specific parameters.
- Boost the operational efficiency.

Hypothesis:

- The change in weights during model adaptation has a low "intrinsic rank".
- proposed **Low-Rank Adaptation** (LoRA) approach.

LORA:

- Train some dense layers in a neural network indirectly by optimizing rank decomposition matrices of the dense layers' change during adaptation.
- Keeping the **pre-trained weights frozen**.

LORA

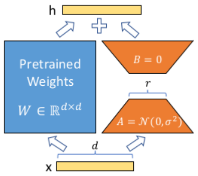


Figure 7: LORA.

- A **pre-trained weight** matrix $W_0 \in R^{d \times k}$.
- Represent a **low-rank decomposition** : $W_0 + \Delta W = W_0 + BA$
  - $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$. The rank $r << min(d, k)$.
- W_0 is frozen, while A and B contain trainable parameters.
- For $h = W_0 x$, modified forward pass yields :

$$h = W_0 x + \Delta W x = W_0 x + BAx$$

UNDERSTANDING THE LOW-RANK UPDATES

Research questions:

1. Which weight matrices in a Transformer model should we apply **LORA** to ?
2. What is the optimal rank r for **LoRA**?

Which weight matrices in a Transformer model should we apply LORA to ?

	# of Trainable Parameters = 18M						
Weight Type Rank r	W_q 8	W_k 8	W_v 8	W_o 8	W_q, W_k 4	W_q, W_v 4	W_q, W_k, W_v, W_o 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**

Figure 8: Validation accuracy on WikiSQL and MultiNLI after applying LoRA in GPT-3

- W_q, W_k, W_v, and W_o to refer to the query/key/value/output projection matrices in the self-attention module.
- Preferable to adapt more weight matrices than adapting a single type of weights with a larger rank.

What is the optimal rank r for LoRA ?

	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

Figure 9: Validation accuracy on WikiSQL and MultiNLI with different rank r in GPT-3

Subspace similarity between different r

- The adaptation matrix can indeed have a very low rank.

Using Modality gap for rank LORA estimation ?

We have :

- D_i is modaily gap at layer i.
- r_i is rank LORA at layer i.
- $r \sim P(r|D)$.

Problem:

- What hypothesis determines the relationship between the optimal rank r in LoRA and the modality gap D ?
- How to derive $P(r|D)$?

Introduction

LoRA blocks are parameter efficient, they suffer from two problems:

1. The size of these blocks is fixed and cannot be modified after training.
2. **Optimizing their rank** requires an exhaustive search and effort.

DyLoRA: technique to address these two problems together.

- trains LoRA blocks for a range of ranks instead of a **single rank**.
- **Outperform LoRA** in a much wider range of ranks **without adding to the training time**.

DyLoRA

In each **LoRA module**:

- The **up-projection** matrix $W_{\text{up}} \in \mathbb{R}^{m \times r}$
- The **down-projection** matrix $W_{\text{dw}} \in \mathbb{R}^{r \times d}$.
- Train the LoRA in the range $r \in \text{Range}[r_{\text{min}}, r_{\text{max}}]$.
  - r_{min} and r_{max} can be treated as new hyper-parameters.

At **each training step**,

- Sample $b \sim p_B(\cdot)$, where $b \in \{r_{\text{min}}, r_{\text{min}} + 1, \ldots, r_{\text{max}}\}$.
  - p_B is a pre-defined categorical distribution.
- **Forward**:
  - $W_{\text{dw}\downarrow b} = W_{\text{dw}}[: b, :]$ and $W^b_{\text{dw}} = W_{\text{dw}}[b, :]$
  - $W_{\text{up}\downarrow b} = W_{\text{up}}[: b, :]$ and $W^b_{\text{up}} = W_{\text{up}}[b, :]$
  - $h = W_0 x + \frac{\alpha}{b} W_{\text{up}\downarrow b} W_{\text{dw}\downarrow b} x$

DyLoRA

Backward:

- Given input and output $(x, y) = (x_i, y_i)_{i=1}^{N}$, define **dynamic loss** function as:

$$\mathcal{L}_{\downarrow b}^{\mathcal{DY}} = \sum_{i=1}^{N} l(f(x_i; W_{\mathrm{dw}\downarrow b}, W_{\mathrm{up}\downarrow b}), y_i)$$

- **If FROZEN then:**

$$W_{\mathrm{up}}^{b} = W_{\mathrm{up}}^{b} - \eta \nabla_{W_{\mathrm{up}}^{b}} \mathcal{L}_{\downarrow b}^{\mathcal{DY}}$$

$$W_{\mathrm{dw}}^{b} = W_{\mathrm{dw}}^{b} - \eta \nabla_{W_{\mathrm{dw}}^{b}} \mathcal{L}_{\downarrow b}^{\mathcal{DY}}$$

Else:

$$W_{\mathrm{dw}\downarrow b} = W_{\mathrm{dw}\downarrow b} - \eta \nabla_{W_{\mathrm{dw}\downarrow b}} \mathcal{L}_{\downarrow b}^{\mathcal{DY}}$$

$$W_{\mathrm{up}\downarrow b} = W_{\mathrm{up}\downarrow b} - \eta \nabla_{W_{\mathrm{up}\downarrow b}} \mathcal{L}_{\downarrow b}^{\mathcal{DY}}$$

Experiment

Model	Accuracy MNLI	Accuracy SST-2	F1 MRPC	Mathews CoLA	Accuracy QNLI	Accuracy QQP	Accuracy RTE	Pearson 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

Introduction

Problem:

- Adapter matrices A and B in LoRA are updated with the same learning rate.
- The same learning rate for A and B does not allow efficient feature learning.

LoRA+:

- Different learning rates for the LoRA adapter matrices A and B.
- **Improves performance** 1% – 2%
- Finetuning speed (up to $\sim$ 2X SpeedUp)

LORA+

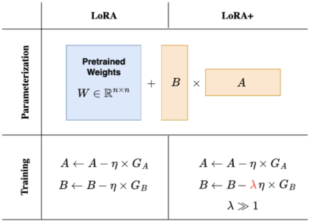


Figure 10: LORA+

LORA:

- Weight matrix $W^* \in \mathbb{R}^{n_1 \times n_2}$ in the **pretrained model**.
- Fine-tuning process with a low-rank decomposition:

$$W = W^* + \Delta W = W^* + \frac{\alpha}{r} BA$$

- $B \in \mathbb{R}^{n_1 \times r}$, $A \in \mathbb{R}^{r \times n_2}$ are trainable, $r \ll \min(n_1, n_2)$ and $\alpha \in \mathbb{R}$ are tunable constants.

An Intuitive Analysis of LoRA

LoRA with a Toy Model

- Linear model : $f(x) = (W^* + ba^\top)x$
  - $x \in R^n, n_1 = 1, n_2 = n, r = 1$
  - Loss function : $L(\theta) = \frac{1}{2}(f(x) - y)^2$ with $\theta = (a, b)$.
    - (x, y) is an input-output datapoint.

Initialization:

- Gaussian initialization of the weights as follows: $a_i \sim \mathcal{N}(0, \sigma_a^2)$, $b \sim \mathcal{N}(0, \sigma_b^2)$
- Two possible schemes:
  - Init[1]: $\sigma_b^2 = 0$, $\sigma_a^2 = \Theta(n^{-1})$
  - Init[2]: $\sigma_b^2 = \Theta(1)$, $\sigma_a^2 = 0$

An Intuitive Analysis of LoRA

Learning rate:

- The gradients are given by:

$$\frac{\partial \mathcal{L}}{\partial b} = a^T x(f(x) - y)$$

$$\frac{\partial \mathcal{L}}{\partial a} = b(f(x) - y)x$$

- Let $U_t = (f_t(x) - y)$. At step t with learning rate $\eta > 0$, we have :

$$b_t = b_{t-1} - \eta a_{t-1}^T x U_{t-1}$$

$$a_t = a_{t-1} - \eta b_{t-1} U_{t-1} x$$

- Then, we have : $\Delta f_t = f_t(x) - f_{t-1}(x) = \delta_t^1 + \delta_t^2 + \delta_t^3$
    - $\delta_t^1 = -\eta b_{t-1}^2 U_{t-1} \|x\|^2$
    - $\delta_t^2 = -\eta (a_{t-1}^\top x)^2 U_{t-1}$ and $\delta_t^3 = \eta^2 U_{t-1}^2 b_{t-1} (a_{t-1}^\top x) ||x||^2$

An Intuitive Analysis of LoRA

Learning rate:

- **Goal** : As n grows, a desirable property is that $\Delta f_t = \Theta(1)$.
- **Proposition 1**: Assume that LoRA weights are initialized with Init[1] or Init[2] with learning rate $\eta = \Theta(n^c)$ for some $c \in \mathbb{R}$.
    - It is impossible to have $\delta_t^i = \Theta(1)$ for $i \in \{1, 2\}$ for any $t > 0$.
    - Fine-tuning with LoRA in this setup is inefficient.
- **Proposition 2**: With learning rate $\eta_a = \Theta(n^{-1})$ and $\eta_b = \Theta(1)$
    - we have for all $t > 1$, $i \in \{1, 2, 3\}$, $\delta_{ti} = \Theta(1)$.
    - Fine-tuning with LoRA in this setup is efficient.

Stability and Feature Learning with LoRA in the Infinite Width Limit

Notation:

- Z denotes the input to LORA layer and $\bar{Z}$ the output.

$$\bar{Z} = W^* Z + \frac{\alpha}{r} BAZ$$

- Define LoRA features (Z_A, Z_B) as $Z_A = AZ$ and $Z_B = BZ_A$.

Definition 3(Stability): LoRA finetuning is stable if for all LoRA layers.

- For all training steps t, we have Z, Z_A, $Z_B = \mathcal{O}(1)$ as $n \to \infty$.

Definition 4 (Stable Feature Learning with LoRA): LoRA finetuning induces stable feature learning.

- It is stable.
- For all LoRA layers and finetuning step t, we have :

$$\Delta Z_B^t = Z_B^t - Z_B^{t-1} = \Theta(1)$$

Stability and Feature Learning with LoRA in the Infinite Width Limit

After step t,Z_B is updated as follows :

$$\Delta Z_B^t = B_{t-1}\Delta Z_A^t + \Delta B_t Z_A^{t-1} + \Delta B_t \Delta Z_A^t$$
$$= \delta_t^1 + \delta_t^2 + \delta_t^3$$

Definition 5 (Efficient Learning): LoRA fine- tuning is efficient.

- it is stable.
- for all LoRA layers in the model, all steps $t > 1$, and $i \in \{1, 2\}$, we have $\delta_t^i = \Theta(1)$.

Theorem 1 (Efficient LoRA (Informal)):

- it is impossible to achieve efficiency with $\eta_A = \eta_B$.
- LoRA finetuning is efficient with $\eta_A = \Theta(n^{-1})$ and $\eta_B = \Theta(1)$.

Experiments with Language Models

Roberta-base:

- Finetuning with $\alpha = r = 8$
- $\eta_B \gg \eta_A$, outperforming the standard practice where $\eta_A = \eta_B$.

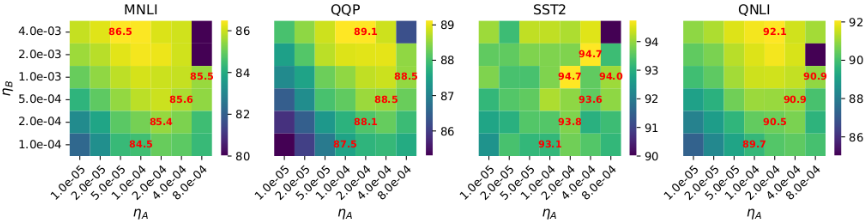


Figure 11: LORA result.

Introduction

QLORA:

- Reduces the average memory requirements of finetuning without degrading performance.
- Introduces multiple innovations designed:
  1. **4-bit NormalFloat**: optimal quantization data type for normally distributed data.
  2. **Double Quantization**: quantizes the quantization constants, saving an average of about 0.37 bits per parameter.
  3. **Paged Optimizers**: Avoid the gradient checkpointing memory spikes.

Background

Block-wise k-bit Quantization:

- Quantization is the process of discretizing an input from a representation that holds more information to a representation with less information.
- Quantizing a 32-bit Floating Point (FP32) tensor into a Int8 tensor :

$$X^{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(X^{\text{FP32}})}X^{\text{FP32}}\cdot\right) = \text{round}\left(X^{\text{FP32}} \cdot c_{\text{FP32}}\right)$$

- **Problem**: **large magnitude values** are not utilized well.
- **Solution**: chunk the **input tensor** into blocks that are independently quantized, each with their own quantization constant c.

QLORA Finetuning

4-bit NormalFloat Quantization:

- ensures each quantization bin has an **equal number of values assigned** from the input tensor.

Double Quantization:

- the process of quantizing the quantization constants for **additional memory savings**.

Paged Optimizers:

- automatic page-to-page transfers between the CPU and GPU for error-free GPU processing in the scenario where the GPU occasionally runs out-of-memory.

QLORA:

- define QLORA for a single linear layer as follows:

$$Y^{\text{BF16}} = X^{\text{BF16}} \cdot \text{doubleDequant}(c^{\text{FP32}}, c^{k\text{-bit}}, W^{\text{NF4}}) + X^{\text{BF16}} \cdot L^{\text{BF16}} \cdot L^{\text{BF16}}$$

Introduction

Problem:

- What is the minimum rank of the LoRA adapters required to adapt a pre-trained model f to match the target model $\bar{f}$.
- How does the model architecture affect the minimal rank ?

Contributions:

- Characterize the LoRA rank for Fully Connected Neural Networks **(FNN)** and Transformer Networks **(TFN)**.
- Identify the necessary LoRA-rank for adapting a frozen model to exactly match a target model.

Theorem 1: Let f be a target FNN(or TFN) and f_0 be a frozen FNN(or TFN).

- Under mild conditions on ranks and network architectures. There exist low-rank adapters such that a f_0 is exactly equal to f.

Expressive Power of Linear Models with LoRA

Simplest scenario: both the target model $\bar{f}$ and the frozen model f_0 are **linear**.

- Target Model : $\bar{f}(x) = \bar{W}x$
- Frozen Model : $f_0(x) = (\prod_{l=1}^{L} W_l)x$
- For a given LoRA-rank $R \in [D]$, Adapted Model :

$$f(x) = (W_L + \Delta W_L) \cdots (W_1 + \Delta W_1)x$$

  - $\text{rank}(\Delta W_l) \leq R$ for all $l \in [L]$.

Expressive Power of Linear Models with LoRA

Lemma 1: Define error matrix $E := \bar{W} - \prod_{l=1}^{L} W_l$ and $R_E = \text{rank}(E)$. For a given LoRA-rank $R \in [D]$, assume that all $(W_l)_{l=1}^{L}$ and $\prod_{l=1}^{L} W_l + LR_r(E)$ is non-singular for all $r \leq R(L-1)$.

$$\min || \prod_{l=1}^{L} (W_l + \Delta W_l) - \bar{W} || = \sigma_{RL+1}(E)$$

Thus, when $R \geq [\frac{R_E}{L}]$, the optimal solution satisfies $\sum_{l=1}^{L} (W_l + \Delta W_l) = W$, implying $f = f_0$.

Expressive Power of FNNs with LoRA

MMD

Define: MMD is a distance (difference) between feature means.
Denote:

- X and $\phi(X) \in \mathcal{F}$ is the a feature map.
- Assuming $\mathcal{F}$ satisfies the necessary conditions:
  - X, Y such that $k(X, Y) = \langle \phi(X), \phi(Y) \rangle_{\mathcal{F}}$

Feature Mean:

- Given $\mathcal{X} \sim P$ we have feature means :

$$\mu_P = \mathbb{E}_{X \sim P}[\phi(X)]$$

Maximum mean discrepancy:

$$\mathrm{MMD}(P, Q) = \|\mathbb{E}_{X \sim P}[\phi(X)] - \mathbb{E}_{Y \sim Q}[\phi(Y)]\|_{\mathcal{F}} = \|\mu_P - \mu_Q\|$$

Optimal transport(OP): Comparing by 'transporting'

Figure 12: Optimal transport

Optimal transport

- a method to find least-cost schemes to transport dirt and rubble from one place to another.
- $\mathrm{OT}_c(\alpha, \beta) := \min_{\pi \in \Pi(\alpha,\beta)} \int_{X \times X} c(x,y)\, d\pi(x,y)$.
  - $\Pi(\alpha, \beta)$ be the set of joint probability distributions on $X \times X$.
- $W_p(\alpha, \beta) \hat{=} OT(\alpha, \beta)^{1/p}$ is called the p-Wasserstein distance.

References

1. Junhong Shen, Liam Li, Lucio M. Dery , Corey Staten,Mikhail Khodak,Graham Neubig,Ameet Talwalkar;Cross-Modal Fine-Tuning: Align then Refine
2. MMD.
3. OTDD
4. Paloma García-de-Herreros, Vagrant Gautam, Philipp Slusallek, Dietrich Klakow, Marius Mosbach,What explains the success of cross-modal fine-tuning with ORCA?
5. Lincan Cai, Shuang Li, Wenxuan Ma, Jingxuan Kang , Binhui Xie, Zixun Sun, Chengwei Zhu, Enhancing Cross-Modal Fine-Tuning with Gradually Intermediate Modality Generation.