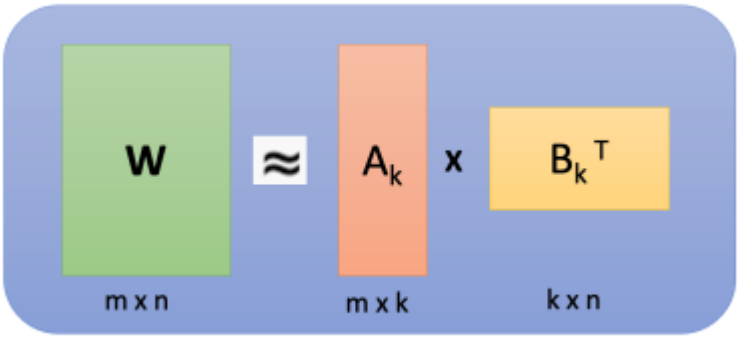
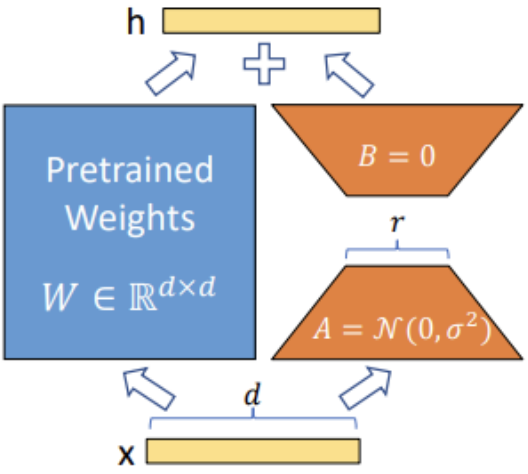
아이펠톤 프로젝트 계획서

개발아이템명	***LoRA-LRD:*** *Low Rank Decomposition with LoRA Approach*		
소속	*Aiffel* 온라인 *5*기 리서치		
신청자 성명	황인준, 맹선재, 박혜원, 양주영	담당퍼실	이영빈

□ 프로젝트 아이템 개요(요약)

아이템 소개	*LoRA 를 활용한 모델의 레이어 단위의 차원 분석 Model Compression 을 위한 Low Rank Decomposition (*LRD) 와 더불어, Adapter 의 개념을 융합한 LoRA 를 역으로 활용하여, pretrained 모델의 weight 를 LRD 할 수 있다는 관점 도입 *LoRA : Low-Rank Adaptation 은 Pretrained Model weights 는 고정시킨 상태로, 분해 weights 행렬들 (decomposition weights matrices) 을 Transformer architecture 의 레이어 마다 추가하고 해당 행렬들에 대해서만 가중치를 업데이트하여, 학습 파라미터를 절대적으로 줄인 방법이다. *LRD : Low Rank Decomposition A **Low Rank Decomposition** or Low Rank Factorization of a layer L would give us a new layer $\tilde{L}$ with two weight matrices $A \in \mathbb{R}^{r \times d_2}$ and $B \in \mathbb{R}^{d_1 \times r}$, and a bias $\tilde{b} \in \mathbb{R}^{d_1 \times 1}$, where $r << d_{min}$ such that for a n batch of input vectors $X \in \mathbb{R}^{d_2 \times n}$ the batch of output vectors $Y \in \mathbb{R}^{d_1 \times n}$ is, $$Y = \tilde{L}(X) = BAX + \tilde{b} \approx L(X) = WX + b \quad (1)$$

아이템의 특징 및 차별성	- 기존 LoRA 는 학습에 사용되는 파라미터 수를 효율적으로 줄여서, 파인튜닝을 하는 방식 (Parameter-Efficient Fine-Tuning (PEFT)) 중 하나로만 사용되고 있다. - 그런데, 결국 LoRA 는 $W = W_0 + \Delta W$ (W_0 는 freeze 된 pretrained weights 를 의미하고, ΔW 는 update 되는 weights 를 의미함) 에서, ΔW 에 대한 LRD 를 하는 것과 동일하다. - 우리는 이러한 LoRA 의 LRD적 성질을 사용하여 W 에 대하여 LRD 를 하고자 한다.
이미지	 Figure 1: Low-Rank Decomposition  Figure 2: LoRA

1. 문제인식 (Problem)

1-1 프로젝트의 목표 및 목적**(**필요성**)**

◦ 문제 인식 : 모델 내 데이터 처리 과정 분석의 어려움

- 우리는 모델이 출력하는 결과물은 쉽게 확인 가능하지만, 입력 값들이 모델 내부에서 어떠한 차원으로 변환되어 특징을 추출하는지 상세히 알기는 어렵다는 문제를 인식하였다.
- 이를 해결할 방안을 모색하는 과정에서, 모델의 fine-tuning 에 사용되는 LoRA 가 dense layer 각각에 병렬적으로 붙일 수 있다는 것에 착안하여, LoRA 를 활용한 layer 의 상세 분석이 가능하지 않을까라는 생각에 도달하였다.

◦ LoRA의 다른 활용 방안 제시

- 모델들의 weight 들을 특정 rank 로 Decompose 하여, Model 을 compress 하는 접근 방식은 이미 존재한다. (LRD)
- 우리는 역으로 각 layer 들이 decompose 되어도 성능이 저하하지 되지 않는, rank (이상적인 R) 를 LoRA 를 사용하여 찾아 나감으로써, 기존 model fine-tuning 의 목적 보다는 model 차원 분석의 도구로 LoRA 를 활용하는 방법을 제안하고자 한다.

1-2 아이템의 독창성

◦ Fine tuning 이 아닌 layer-by-layer 차원 분석과 Model Compression 도구로서의 LoRA

- 학습이 끝난 Model 의 특정 layer 에서의 weight 에 0.9를 scaling 하고, low rank adapter 를 병렬로 연결해 준다. Adapter 를 제외한 모든 layer 는 freeze 한후, 학습에 사용한 데이터로 다시 fine-tuning 해준다. Adapter 의 rank 가 적절했다면, 학습된 ΔW 는 $0.1W$ 와 동일할 것이다.

◦ 기존 LRD와의 차별성

- 기존의 LRD 는 weight 를 수학적으로 분리하려는 시도나, 전체 weight 를 작은 차원의 여러 layer로 대체하려는 시도를 했지만, 이번 프로젝트는 원본의 weight 중 일부 만을 fine-tuning (LoRA) 기법으로 변환하려는 시도를 하기때문에 안정성이 더 높을것이라고 기대한다.
- LoRA 를 통하지 않고 rank 를 찾으려는 시도도 유효 할 수 있지만, local minimum 에 고립될 위험이 더 크고 decompose 가능한 적절한 rank 를 찾는데 더 느리고 안정적이지 못한다. 원본 weight 의 10% 만을 fine-tuning (LoRA) 으로 복구하는 과정은 90%의 가이드라인과 함께하는 학습이기 때문에 난이도가 더 쉽고, 효율적으로 rank 를 찾는데 도움을 줄 수 있다.

2. 개발 및 연구 내용

2-1. 구현 내용 상세(구현 가능성)

[Key Concept of LoRA-LRD]

$$W = W_0 + \Delta W\downarrow$$
$$W = 0.9W + 0.1\,W\downarrow$$
$$0.9W + BA \cong 0.9W + 0.1W\downarrow$$
$$10 \times BA = 10 \times 0.1W\hookleftarrow$$

$W = W_0 + \Delta W$ (LoRA)

$W = W_0(= 0.9W) + \Delta W(= 0.1W)$ (Goal)

$0.9W + BA \simeq 0.9W + 0.1W$ (After fine-tuning, recovery of weight)

$10 \times BA \simeq 10 \times 0.1W$ (Equivalent as model layer compression)

$Rank\ Loss = \Sigma_{ij}(W_{ij} - 10B_{ik}A_{kj})^2$ (Demo rank selection loss)

[구현 방식**]**

- 모델을 학습 시킨 뒤, original weights (W) 을 얻는다.
- original weights (W) 에 0.9 를 곱하여, $W_0(= 0.9W)$ 를 얻는다.
- $W_0(= 0.9W)$ 를 freeze 된 weight 값으로 두고, LoRA 방식에 따라 BA 에 해당되는 Weights 들을 동일한 task 및 동일한 dataset 에 대하여 학습시킨다.
- 만약 rank (r) 가, 해당 Layer 를 optimal 하게 decompose 하는 r 이라면, BA 의 Weights 들은 0.1 W 와 동일할 것이다.
- 따라서, $10 \times BA$, W 간의 차이를, optimal 한 r 를 찾기 위한 loss 로 둘 수 있다.
- 이 loss 를 최소화 하는 r 를 찾은 것이 실험의 결과가 될 것이다.

2-2. 개발 아이템 기대효과

- 모델의 각 레이어들이, 몇 개의 rank 를 통해 특징을 추출하는지 경향성을 분석 할 수 있다. 이러한 분석 결과들은 새로운 모델을 설계할 때, 기본적인 가이드라인으로 사용할 수 있다.
- fine-tuned 모델의 layer 단위의 compression 이 가능해진다. (모델의 경량화)
- 큰 모델에 대해서 데이터 뭉치에 중요하게 작용하는 특성의 갯수를 알게되어 데이터 분석에 도움을 줄 수 있다.

3. 실행 계획

3-1. 기간내 프로젝트 구현 완성을 위한 전략

◦ 실험 설계

[실험 진행 순서**]**

- 앞서, 2.1 에서 제시한 **[**구현 방식**]** 을 간단한 모델부터 LLM까지 점진적으로 적용할 것이다. 모델에 대한 범위를 넓혀간다.

  Step 1. Dense Layer N개로 연결된 단순한 classification 모델에 대해서 LoRA-LRD 적용

  Step 2. Transformer 모델 대해서 LoRA-LRD 적용

  Step 3. LLM 의 특정 레이어들에 대해서 LoRA-LRD 적용

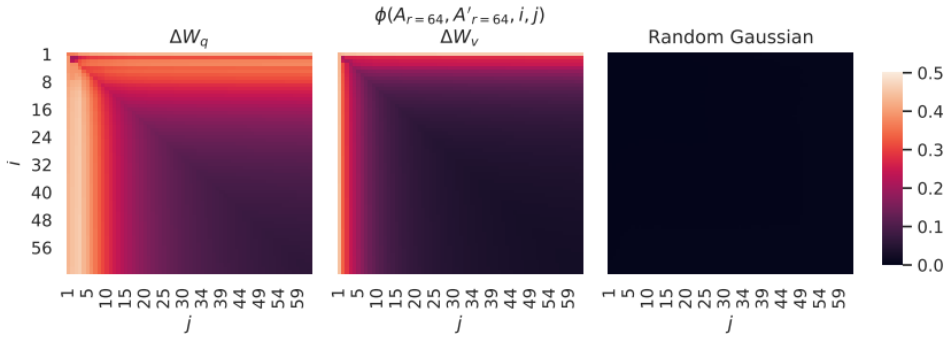
[실험 전략**]**

- Optimal 한 r 을 찾기 위한 전략

  (더 효과적인 방법은, 추후 실험을 진행하면서 더 조사하고, 설계해볼 예정이다)

  1. Random Search
  2. Grassmann Distance[1] 값으로 r 에 대한 range 찾기

     Ex. m×n 인 layer 에 대해서, r = n/2 값을 주고 학습 시킨 두 개의 weight matrices 에 대하여, 1~ n/2 까지의 Grassmann distance 값을 계산하여, similarity 를 계산한다. 이 결과 값을 통해 추정해나갈 r 에 대한 range 를 설정한다.



[1] 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. In International Conference on Learning Representations, 2022. 논문의 'Subspace similarity between different r ' 단락에서 사용한 방법을 채택함.

3-2. 아이펠톤 기간 내 마일스톤

Task	목표기간	세부내용
가설 계획	*10.24 ~ 10.27*	설정한 가설에 대해 멘토와 토의
LoRA 코드 분석	*10.30 ~ 11.1*	*microsoft*와 *huggingface*에서 제공된 *LoRA* 코드를 분석 후 가설에 따라 적용할 방식에 대해 논의
실험 설계	*11.2 ~ 11.6*	*r* 값을 찾기 위한 효율적인 방법 조사 및 설계 데이터 및 모델 설계
간단한 모델 학습 및 *fine-tuning*	*11.7 ~ 11.14*	간단한 모델의 모든 *Layer* 에 대해서 *optimal* 한 *r* 을 찾고 이 과정 중에서 *r* 에 대한 인사이트를 도출
복잡한 모델 학습 및 *fine-tuning*	*11.14 ~ 11.20*	간단한 모델에 대한 실험 과정 중에 도출해낸 결과들을 바탕으로, 복잡한 모델에 대해서도 *optimal* 한 *r* 을 탐색 *(*특정 *Layer* 들에 대해서만 진행*)*
결과 분석 및 논문 작성	*11.21 ~ 12.05*	진행된 실험들에 대해서 보다 더 깊은 분석 및 이를 기반으로 논문 작성

3-3. 팀장 및 팀원의 역할 분배

순번	주요 담당업무	역할 상세	인원
1	실험 설계 및 조사	실험에 대한 설계에 필요한 자료를 조사하고 오류가 없는지 점검	*4*
2	데이터 확보	실험에 필요한 데이터를 확보	*1 ~ 2*
3	모델 구현 및 학습	간단한 모델은 직접 구현하고 실험에 적합한 큰 모델에 대해 조사 및 선별	*3*
4	실험 결과 정리	통제된 조건 하에서, 각자 실험을 진행하고 결과를 공유해서 그 의미를 파악하고 진행 방향에 대해 점검 및 논의	*4*

4. Reference

1. LoRA 코드

https://github.com/microsoft/LoRA/tree/main

https://huggingface.co/docs/peft/conceptual_guides/lora

2. Attention is All You Need 논문

https://arxiv.org/abs/1706.03762

3. LoRA 논문

https://arxiv.org/abs/2106.09685

4. Strategies for Applying Low Rank Decomposition to Transformer-Based Models 논문

https://neurips2022-enlsp.github.io/accepted_papers.html

5. LRD image

https://smashinggradient.com/2023/05/23/30-compression-of-llms-with-low-rank-decomposition-of-attention-weight-matrices/

6. Pretraining a Transformer from scratch with KerasNLP

https://keras.io/guides/keras_nlp/transformer_pretraining/

7. grassmann distance 관련 논문

https://dl.acm.org/doi/abs/10.1145/1390156.1390204

8. Vera 논문

https://arxiv.org/abs/2310.11454