LoRA: Low-Rank Adaptation of Large Language Models

1. Introduction

배경 :

NLP는 대부분 LLM을 사전학습해서 사용하고 있다. 모델이 커지고 더 많은 데이터를 학습할수록 다운스트림 작업에서 성능이 계속 좋아진다.

문제점 :

새로운 작업마다 모델을 full fine-tuning 하면 모든 파라미터를 다시 학습해야 한다. GPT-3 같은 초거대 모델은 수천억 개 파라미터가 있기에 태스크마다 따로 저장하면 메모리, 저장공간, 배포 비용이 엄청나다.

기존 해결책과 한계 :

adapter tuning - 모델 중간에 작은 모듈을 추가해 학습 (빠르고 효율적) → 하지만 추론이 느려짐.

prompt/prefix tuning - 입력 앞에 특별한 토큰(학습된 벡터)을 붙여서 모델을 적응 → 입력 길이를 차지하고 최적화가 어려움

즉 효율은 있지만 성능/추론 속도 트레이드오프가 생김.

LoRA 아이디어 :

모델이 학습하면서 바뀌는 가중치 변화는 생각보다 단순하다는 가정에서 출발. 원래의 거대한 가중치는 그대로 두고, 변화량만 두 개의 작은 행렬 곱(저랭크 분해)으로 표현해서 학습. 그래서 학습할 파라미터 수가 극히 작아짐.

장점 :

태스크별로 필요한 건 작은 행렬 2개(A,B) 뿐 → 저장, 배포가 쉽다. 학습이 끝나면 원래 가중치와 합쳐서 사용 가능 → 추론 속도 저하 없음. 다른 기법과도 같이 쓸 수 있다.

결과 :

RoBERTa, DeBERTa, GPT-2, GPT-3 같은 모델에서 테스트 → 풀 파인튜닝만큼 좋거나 오히려 더 좋음.

특히 GPT-3 175B 같은 초거대 모델에서도 효과적임 → 실무적으로 매우 강력한 방법.

---

2. Problem Statement

full fine tuning 방식의 문제 :

거대 언어모델은 이미 수천억 개의 파라미터를 가지고 있다. 새로운 작업을 할 때 풀 파인튜닝을 하면, 모델 전체 파라미터를 다시 학습해야 하고 저장해야 한다.

LoRA의 발상 :

모델이 변하는 부분을 작고 간단한 구조로 근사할 수 있지 않을가? 큰 파라미터 업데이트를 모두 학습하는 대신 훨씬 작은 파라미터 집합만 학습해서 업데이트를 대신 표현하자는 아이디어.

---

3. Aren’t Existing Solutions Good Enough?

Adapter Tuning

- 방법: 모델 블록 사이에 작은 모듈(어댑터 레이어)을 넣어서, 그 부분만 학습시키는 방식.
- 장점: 파라미터 수를 크게 줄일 수 있음 (원래 모델의 1% 이하).
- 단점: 추론할 때 시간이 늘어남.

- 왜냐하면 어댑터는 모델의 계산 흐름에 “추가 단계”를 넣는 거라, 아무리 작아도 순차적으로 실행해야 함.
- 특히 온라인 서비스처럼 한 번에 한 문장(배치=1)을 처리하는 경우, 지연이 확 늘어남.
- GPT-2 같은 모델에서 실험했더니, 20~30% 이상 속도가 느려짐.

Prefix Tuning

- 방법: 입력 문장의 앞부분에 “학습된 프리픽스 벡터”를 붙여서 모델을 조정.
- 장점: 모델 파라미터는 거의 안 건드림.
- 단점:
  - 학습이 어렵고 불안정함 → 파라미터 수를 늘려도 성능이 꼭 좋아지지 않음.
  - 프리픽스 때문에 입력 시퀀스 길이의 일부를 뺏김 → 실제 태스크에 쓸 수 있는 문장 길이가 줄어듦 → 성능 손해 발생.

---

4. OUR METHOD

4.1 Low-Rank-Parametrized Update Matrices

핵심 아이디어 :

- 신경망의 가중치 행렬은 보통 full rank인데, 실제로 작업별로 모델을 조금 조정할 때 필요한 변화(업데이트)는 단순한 저차원(저랭크) 구조만으로 충분할 수 있다는 점에 착안.
- 그래서 LoRA는 원래 가중치 W_0는 그대로 두고, 업데이트 ΔW만 작은 두 행렬의 곱(BA)으로 표현.

구현 방식 :

- W_0는 고정(freeze)하고, 새로운 두 행렬만 학습.
- 즉, 순전파 계산은 $h = W0x + BAx$ 이런 식으로 원래 결과에 “조정 값”을 더하는 구조.
- 초기엔 B=0으로 두어서 시작 시 모델은 원래와 똑같이 작동 → 안정적인 학습이 가능.
- 스케일링 팩터를 두어, 랭크 크기를 바꿔도 하이퍼파라미터를 크게 다시 조정할 필요가 없다.

풀 파인튜닝과의 관계 :

- 랭크 r을 충분히 크게 잡으면 풀 파인튜닝과 같은 표현력을 가질 수 있다.
- 즉, LoRA는 풀 파인튜닝을 “압축한 일반화 버전”이라고 볼 수 있다.
- 차이점은, 어댑터 방법은 결국 작은 MLP에 가까워지고, 프리픽스 방법은 입력 길이에 제약을 받는데, LoRA는 원래 모델의 잠재력을 그대로 쓸 수 있다는 점이다.

장점 – 추론 지연 없음 :

- 학습이 끝난 뒤에는 BA를 W_0에 합쳐서 저장할 수 있다.
- 그러면 추론할 때는 그냥 원래 모델처럼 실행되므로 추가 지연(latency)이 전혀 없음.
- 태스크를 바꿀 때도 단순히 BABABA만 교체하면 되니까, 메모리와 속도 모두 효율적.

요약 :

LoRA는 “원래 가중치는 그대로 두고, 변화량만 작은 두 행렬로 압축해서 학습”하는 방법이다. 이렇게 하면 학습은 훨씬 가볍고, 추론은 원래 모델과 똑같이 빠르며, 필요한 경우엔 풀 파인튜닝 수준까지 확장할 수도 있다.

4.2 Applying LoRA to Transformer

어디에 LoRA를 적용할까?

- Transformer에는 Self-Attention 모듈에 4개의 가중치 행렬과, MLP 모듈에 2개의 가중치 행렬이 있다.

- LoRA는 원칙적으로 모든 가중치에 적용할 수 있지만, 논문에서는 단순성과 효율성을 위해 어텐션 부분(Wq, Wv 등)만 조정하고 MLP는 동결했다.
- LayerNorm, bias 같은 다른 부분은 아직 실험하지 않았고, 추후 연구로 남겨둠.

실질적 장점 :

- VRAM 절약:
  - GPT-3 175B 훈련 시 필요 VRAM을 1.2TB → 350GB로 줄임.
  - 옵티마이저 상태를 거의 안 저장해도 되기 때문.
- 체크포인트 저장 크기 감소:
  - LoRA rank r=4r=4r=4, query/value 행렬만 적응했을 때
  - 모델 저장 크기 350GB → 35MB (10,000배 축소!).
- GPU 자원 절약: 더 적은 GPU로도 큰 모델 학습 가능.
- 태스크 전환 용이: 전체 모델을 교체하지 않고 LoRA 가중치(BA)만 교체하면 되므로, 맞춤형 모델을 쉽게 스위칭 가능.
- 학습 속도 향상: GPT-3 175B 실험에서 25% 더 빠름 (대부분 파라미터는 gradient 계산 안 해도 되기 때문).

한계점 :

- LoRA 가중치(A, B)를 W에 합쳐버리면, 서로 다른 태스크를 한 번에 배치(batch) 처리하기 어렵다.
- 물론 W에 합치지 않고 태스크별 LoRA 모듈을 유지하면 가능하지만, 이 경우엔 추론 속도가 중요하지 않은 상황에서만 쓸 수 있다.

요약 :

LoRA는 Transformer에서 어텐션 부분만 살짝 바꿔서 학습해도 효율이 엄청 좋다.

- 메모리도 확 줄고, 저장공간도 10,000배 줄고, 학습 속도도 빨라진다.
- 게다가 태스크마다 작은 LoRA 가중치만 바꿔 끼우면 되니까, 한 모델로 여러 작업을 쉽게 전환할 수 있다.
- 다만, 여러 태스크를 동시에 한 배치에서 처리하는 데는 조금 제약이 있다는 한계가 있다.

---

5. EMPIRICAL EXPERIMENTS

실험 목표 :

- LoRA가 정말 효율적인지 확인하기 위해 RoBERTa, DeBERTa, GPT-2, GPT-3 같은 다양한 언어모델에서 실험.
- 태스크 범위도 넓게 자연어 이해(NLU) (GLUE 벤치마크)부터 자연어 생성(NLG) (WikiSQL, SAMSum)까지 확인.

5.1 BASELINES

비교 대상 :

LoRA를 기존 방법들과 비교하기 위해 여러 가지 방식을 실험

- 풀 파인튜닝(FT): 모델 전체 파라미터를 다 학습하는 가장 기본적인 방법.
- FT-Top2: GPT-2에서 마지막 두 층만 파인튜닝하는 변형.
- BitFit: 파라미터 중 바이어스(bias)만 학습하는 초간단 방법.
- Prefix Tuning
  - PreEmbed: 입력에 새로운 토큰(학습 가능한 벡터)을 앞이나 중간에 붙여서 조정.
  - PreLayer: 각 Transformer 층마다 학습 가능한 벡터를 추가하는 더 강력한 버전.
- Adapter Tuning

    - AdapterH: 원래 제안된 방식, Transformer 블록 사이에 작은 MLP 모듈 삽입.
    - AdapterL: 더 가벼운 버전, MLP 뒤와 LayerNorm 뒤에만 추가.
    - AdapterP: Pfeiffer 변형.
    - AdapterDrop: 일부 어댑터를 아예 빼서 더 가볍게 한 버전.
- LoRA: 기존 가중치는 그대로 두고, 작은 두 행렬(BA)만 학습해서 업데이트하는 방식. 대부분 W_q, W_v에만 적용.

주요 결과 (GLUE 벤치마크) :

- LoRA 성능: 풀 파인튜닝(FT)과 거의 똑같거나 더 잘 나옴.
- 파라미터 수:
    - 풀 파인튜닝: 수억~수십억 개 학습.
    - LoRA: 몇 백만 개(원래 모델의 0.1~0.3% 수준)만 학습.
- 효율성: 같은 성능을 훨씬 적은 파라미터로 달성 → 저장·학습 비용이 크게 줄어듦.

요약 :

LoRA는 RoBERTa, DeBERTa, GPT 계열에서 풀 파인튜닝만큼 성능을 내면서도 학습해야 할 파라미터는 극도로 적음 을 보여줬다.

→ 기존 방법들(BitFit, Prefix, Adapter 등)보다 안정적이고 효율적이라는 게 입증.

5.2 ROBERTA BASE/LARGE

배경 :

- RoBERTa는 BERT의 학습 방법을 개선한 모델로, 비슷한 크기의 모델 대비 성능이 강력.
- 최근에는 더 큰 모델들이 GLUE 같은 벤치마크에서 RoBERTa를 넘어섰지만, 여전히 “중간 크기의 실용적인 모델”로 연구자와 실무자들이 많이 쓰고 있다.

사용한 모델 :

- RoBERTa-base: 약 1.25억 파라미터
- RoBERTa-large: 약 3.55억 파라미터
- 두 모델 모두 HuggingFace Transformers 라이브러리에서 가져와서 실험에 사용.

비교 방법 :

LoRA가 기존 방법(특히 어댑터)과 공정하게 비교되도록 세팅을 맞춤

1. 배치 크기와 시퀀스 길이를 동일하게 맞춤
    - 모든 태스크에서 batch size 동일, sequence length는 128로 고정.
    - 이렇게 해야 adapter 방식과 동일한 조건에서 비교 가능.
2. 초기화 방식 통일
    - MRPC, RTE, STS-B 같은 작은 데이터셋은 보통 MNLI로 먼저 적응된 모델을 가져다 쓰는 경우가 있음.
    - 하지만 LoRA 비교에서는 항상 사전학습 모델 그대로를 초기화로 사용.
    - 즉, 불필요하게 유리한 사전 적응을 배제하고, 진짜 “제로에서 태스크 적응” 성능을 보도록 했음.

결과 :

- LoRA는 RoBERTa-base, RoBERTa-large 모두에서 풀 파인튜닝과 거의 동일하거나 더 나은 성능을 기록
- 특히 필요한 학습 파라미터 수는 훨씬 적음 (수억 개 → 수십만 개 수준).
- 어댑터 기반 방법들과 비교했을 때도 성능에서 손해가 없거나 오히려 더 좋았음.

요약 :

RoBERTa는 중간 크기 모델인데, 여기서도 LoRA가 효율적이라는 걸 실험으로 보여줬다. 공정한 조건(배치·길이·초기화 동일)에서 비교했는데도, LoRA는 적은 파라미터만 학습해도 성능은 풀 파인튜닝급이었다. 이는 LoRA가 "대형 모델뿐 아니라 중형 모델에도 충분히 실용적이다"라는 중요한 증거.

5.3 DeBERTaXXL (NLU – 초대형 모델)

DeBERTa-XXL은 BERT 계열의 최신 변형으로, 15억(1.5B) 파라미터를 가진 초대형 모델로, GLUE, SuperGLUE 같은 대표적인 자연어 이해(NLU) 벤치마크에서 이미 강력한 성능을 내는 모델이다.

연구진이 궁금했던 건 "이렇게 큰 모델에서도 LoRA가 풀 파인튜닝만큼 성능을 낼 수 있을까?" 였다. 결론적으로 LoRA는 훨씬 적은 파라미터만 학습했음에도, 풀 파인튜닝 DeBERTa-XXL과 거의 동일한 성능을 보여줬어다. 즉, 초대형 모델에도 LoRA가 적용 가능하다는 걸 입증함.

5.4 GPT-2 Medium/Large (NLG – 생성 모델)

지금까지는 이해(NLU) 중심의 태스크였는데, 이제는 생성(NLG) 모델인 GPT-2 Medium & Large를 실험했다. LoRA는 GPT-2 Medium/Large에서도 다른 방식들(어댑터, 프리픽스 튜닝 등)보다 성능이 좋거나 비슷하면서, 학습 파라미터 수는 훨씬 적음. 즉, LoRA는 텍스트 생성 같은 NLG 작업에도 효과적이라는 걸 보여줌.

5.5 Scaling UP to GPT-3 175B

이 부분은 LoRA의 최종 테스트라고 볼 수 있다. 1750억 파라미터짜리 GPT-3에 LoRA를 적용했을 때도 성능이 잘 나오는지 확인.

실험 결과 :

LoRA는 WikiSQL, MNLI, SAMSum 세 가지 태스크에서 풀 파인튜닝과 같거나 오히려 더 좋은 성능을 냈으며 학습해야 하는 파라미터 수는 훨씬 적은데도 불구하고 성능은 그대로거나 상승했다.

- 파라미터 수를 많이 늘린다고 성능이 무조건 좋아지진 않는다.
- 특히 Prefix 방식들은 특수 토큰을 너무 많이 쓰면 성능이 급격히 떨어졌다.
    - Prefix-embedding → 256개 이상 토큰
    - Prefix-layer → 32개 이상 토큰
- 원인: 특수 토큰이 너무 많으면 입력 데이터 분포가 사전학습 때와 달라져서 모델이 혼란스러워진다고 추측.

LoRA의 강점 :

초거대 모델(GPT-3 175B)에서도 LoRA는 안정적이고 강력하다. 다른 방법들보다 스케일업 효율이 좋고 성능도 우수하다는 게 실험으로 확인됐다. 즉, LoRA는 작은 모델뿐 아니라 초대형 모델까지도 문제없이 확장 가능한 방법이다.

---

6. RELATEDWORKS

"LoRA가 왜 필요한지, 어떤 연구 전통 속에서 나온 건지"를 정리한 부분.

1. **Transformer 모델들 :**
   - Transformer는 요즘 NLP의 기본 모델 구조.
   - BERT, GPT-2 같은 모델이 나오면서 “사전학습 + 파인튜닝” 패러다임이 대세가 됨.
   - GPT-3는 1750억 파라미터짜리 초거대 모델.
2. **프롬프트 엔지니어링 & 파인튜닝 :**
   - GPT-3는 프롬프트만 잘 만들어도 적응할 수 있지만, 성능이 프롬프트 품질에 크게 좌우됨 → 프롬프트 엔지니어링 필요.
   - 파인튜닝은 모든 파라미터를 다 업데이트하는 방식인데, GPT-3 같은 초대형 모델은 메모리·비용이 너무 커서 비현실적임.
3. **파라미터 효율적 적응 방법들 :**
   - Adapter: 기존 레이어 사이에 작은 모듈 추가 → 효율적이지만 추론 속도 느려짐.
   - COMPACTER: 어댑터를 더 효율적으로 매개변수화.
   - Prefix/Embedding 튜닝: 입력 토큰 일부를 학습 → 확장은 가능하지만 입력 길이를 잡아먹음.
   - LoRA: 저랭크 행렬로 업데이트를 제한 → 학습은 효율적이고, 추론 시엔 원래 가중치에 합쳐져서 추가 지연 없음.
4. **저랭크 구조와 딥러닝 :**
   - 머신러닝 문제는 본질적으로 저랭크 구조를 자주 가짐.
   - 과매개변수화된 신경망도 학습 끝나면 저랭크 성질을 보임.
   - 기존 연구들은 네트워크 자체를 저랭크로 학습했지만, 동결된 모델을 저랭크 업데이트로만 조정하는 방법은 없었음.
   - 이론적으로도 저랭크 구조는 학습 효율성과 성능에 중요한 역할을 한다고 알려져 있음.

---

7. CONCLUSION AND FUTURE WORK

LoRA의 핵심 장점 :

- 풀 파인튜닝은 너무 비쌈: 하드웨어 자원도 많이 들고, 태스크마다 모델을 따로 저장/로드해야 해서 비효율적임.
- LoRA는 해결책:
  - 추론 속도 늦추지 않음
  - 입력 길이 줄이지 않음
  - 모델 품질 유지
  - 태스크 간 빠른 전환 가능 (공통 파라미터를 공유하기 때문)
- Transformer에 집중했지만, 사실상 dense layer가 있는 모든 신경망에 적용 가능.

앞으로의 연구 과제 :

1. 다른 방법과 결합 → LoRA + Adapter 같은 식으로 시너지 낼 수 있음.
2. 메커니즘 이해 → 사전학습된 특징이 LoRA로 어떻게 태스크에 맞춰지는지 아직 불명확. LoRA는 이 과정을 풀 파인튜닝보다 더 연구하기 쉽게 해줌.
3. 적용 대상 선정 → 지금은 “어느 가중치에 LoRA를 적용할까?”를 경험적으로 고르는데, 이를 더 과학적으로 정할 방법이 필요.
4. 랭크 결핍 가설 → 업데이트가 저랭크라면, 원래 가중치 W도 저랭크일 수 있음 → 이 부분은 새로운 연구 아이디어로 이어질 수 있음.

요약 :

LoRA는 “효율적이고 빠른 적응”을 가능하게 만드는 방법이다. 풀 파인튜닝이 가진 비용 문제와 비효율성을 극복하면서도 성능은 유지하거나 오히려 높일 수 있다. 앞으로는 LoRA를 다른 기법과 결합하거나, 더 깊이 있는 원리 연구에 활용할 수 있는 여지가 많다.