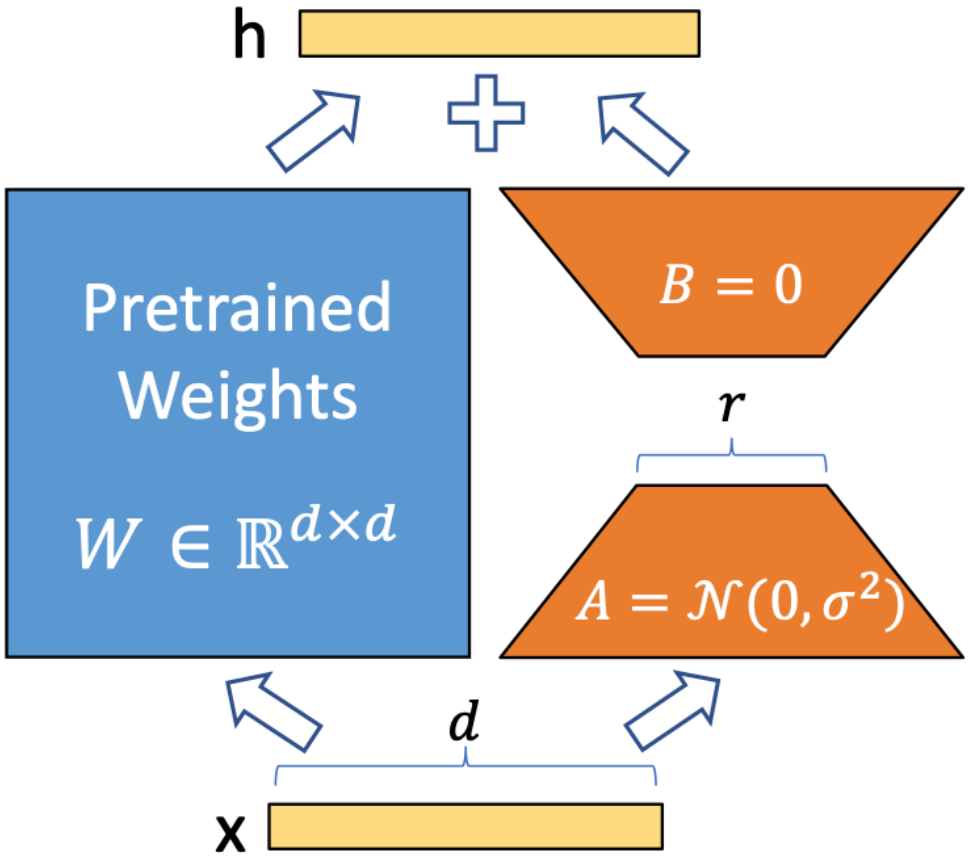
LoRA: Low-Rank Adaptation of Large Language Models

Introduction



자연어처리에서는 대규모 사전학습된 모델을 여러 태스크에 맞게 적용시키는 방법을 주로 사용한다. 하지만 모델의 크기가 점점 커지면서 모든 parameter를 다시 task에 맞도록 학습시키는 fine-tuning은 비용적으로 불가능에 가까워졌다. 예시로 GPT-3의 경우 175B(1750억개)의 parameter를 가진다.

일부 parameter만 학습시키거나, 외부 모듈을 사용하는 방식도 있지만, 여전히 성능 손실, 추론 속도 저하 등의 문제가 존재한다. 그래서 위 연구 팀에서는 기존 연구에서 parameter가 과도하게 많은 모델이 사실은 낮은 차원의 공간에서 학습된다는 아이디어에서 시작해, parameter 업데이트도 low-rank 구조일 수 있다는 가설에 기반해 LoRA(Low-Rank Adaptation)을 제안한다.

LoRA는 4가지 장점을 가진다.

- W는 그대로 두고, 차원이 낮춰진 행렬 A, B만 교체하면 된다. 따라서 빠르고 비용이 적게 든다.
- LoRA는 대부분의 parameter가 고정되어 있기 때문에 계산량이 줄고 GPU 메모리가 절감된다.
- A, B는 deploy 단계에서 W에 합쳐질 수 있기 때문에, 추론 시에 latency가 존재하지 않는다.
- LoRA는 Prefix-tuning과 같은 다양한 다른 방법들과 함께 사용될 수 있다.

Problem Statement

$$\max_{\Phi} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(P_{\Phi}(y_t|x,y_{<t})\right)$$

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log\left(p_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x,y_{<t})\right)$$

대부분의 자연어 생성 문제는 조건부 확률을 최대화 하는 문제로 표현할 수 있다. 모델 전체의 parameter를 업데이트하는 방식에서는 각 downstream task마다 모델 파라미터 수만큼에 해당하는 학습을 진행해야 한다. 본 논문에서는 이에 대한 대안으로, 사전학습된 parameter는 freeze하고, 전체 모델 parameter를 학습하는 대신, 낮은 차원의 적은 수의 parameter Θ를 통해 학습을 진행하게 된다. 이 Θ는 pre-trained GPT-3(175B) 모델의 경우 0.01% 수준의 parameter만으로 학습이 가능하다.

Aren't Existing Solutions Good Enough?

Batch Size Sequence Length $\|\Theta\|$	32 512 0.5M	16 256 11M	1 128 11M
Fine-Tune/LoRA	1449.4±0.8	338.0±0.6	19.8±2.7
$\text{Adapter}^{\text{L}}$	1482.0±1.0 (+2.2%)	354.8±0.5 (+5.0%)	23.9±2.1 (+20.7%)
$\text{Adapter}^{\text{H}}$	1492.2±1.0 (+3.0%)	366.3±0.5 (+8.4%)	25.8±2.2 (+30.3%)

Adapter Layers Introduce Inference latency

다양한 종류의 adapter가 존재하지만, Houlsby et al. (2019)를 예를 들어서 설명하자면, 이 모델은 기존 transformer 블록 사이에 작은 MLP 모듈을 삽입해서 해당 모듈만 학습한다. 학습 parameter 수는 줄어들지만, 추론 속도가 증가하고, 따라서 실시간 추론 환경에서 문제가 될 수 있다. 또한, 병목 현상이 심화될 수 있고, adapter가 작은 모듈임에도, 대형 모델에서는 여전히 크다.

Directly Optimizing the Prompt is Hard

Prefix tuning 방식(Li & Liang, 2021)은 학습 가능한 토큰(prefix)를 붙여 모델의 출력을 유도한다. 모델 parameter는 건드리지 않고, 입력만 바꾸기 때문에 추론시에 지연이 발생하지는 않지만, 제약이 많고, 학습이 불안정하고, 입력 길이가 제한되는 문제가 존재한다.

Our Method

Low-Rank Parametrized Update Matrices

$$h = W_0x + \Delta x = W_0x + BAx$$

핵심 개념은 기존 Dense layer의 parameter W를 완전히 학습하는 대신, 업데이트된 parameter를 low-rank 행렬인 BA로 근사한다. LoRA는 $\Delta W = BA$ 형태로 간주하고 학습을 진행하기 때문에, 모델 전체 parameter는 고정되며, A와 B만 학습된다.

- 랭크 r을 충분히 키우면 LoRA는 일반적인 fine-tuning처럼 전체 W를 학습할 수 있다. 즉, parameter 수를 늘리면 fine-tuning에 수렴한다
- 추론 시점에서는 $W = W_0 + BA$를 미리 계산해서 한 번만 덮어쓰기 하면 parameter가 업데이트되기 때문에, 추론 시의 latency가 full fine-tuning 수준에 수렴한다.

Applying LoRA to Transformer

Transformer에는 4가지 self attention module(W_q, W_k, W_v, W_o)과 두개의 MLP module이 존재한다. 위 논문에서는 MLP 모듈, bias, LayerNorm은 freeze해놓고, attention weights만 학습하는 전략을 사용한다. 이를 통해 파라미터 효율성을 유지하고, 구조를 간단화하고, 충분한 성능을 확보할 수 있다.

Empirical Experiments

본 논문에서는 LoRA의 성능을 평가하기 위해 RoBERTa(Liu et al., 2019), DeBERTa(He et al., 2021), GPT-2(Radford et al., b), GPT-3 175B(Brown et al., 2020) 모델들에 대해 실험을 진행한다. 실험 task는 자연어 이해(NLU, Natural Language

Understanding), 자연어 생성(NLG, Natural Language Generation)까지의 넓은 범위를 다룬다.

Model & Method	# Trainable Parameters	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B	Avg.
RoB_{base} (FT)*	125.0M	**87.6**	94.8	90.2	**63.6**	92.8	**91.9**	78.7	91.2	86.4
RoB_{base} (BitFit)*	0.1M	84.7	93.7	**92.7**	62.0	91.8	84.0	81.5	90.8	85.2
RoB_{base} ($Adpt^D$)*	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
RoB_{base} ($Adpt^D$)*	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
RoB_{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**
RoB_{large} (FT)*	355.0M	90.2	**96.4**	**90.9**	68.0	94.7	**92.2**	86.6	92.4	88.9
RoB_{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**
RoB_{large} ($Adpt^P$)†	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
RoB_{large} ($Adpt^P$)†	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
RoB_{large} ($Adpt^H$)†	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
RoB_{large} ($Adpt^H$)†	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
RoB_{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**
DeB_{XXL} (FT)*	1500.0M	91.8	**97.2**	92.0	72.0	**96.0**	92.7	93.9	92.9	91.1
DeB_{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**

Baseline

LoRA의 비교대상은 다음과 같다.

- Fine Tuning(FT) : 모든 parameter 업데이트하면서 학습
- Bias-only or BitFit : bias parameter만을 학습
- Prefix-embedding tuning(PreEmbed) : 입력에 학습이 가능한 토큰을 삽입
- Prefix-layer tuning(PreLayer) : PreEmbed의 연장, transformer layer에서 활성화를 학습, 기존 layer의 계산 결과가 학습 가능한 것들로 대체됨
- Adapter tuning : 각 층 사이에 adapter 모듈을 삽입하여 학습,
  - H : fully connected layer에 삽입
  - L : MLP 모듈과 LayerNorm 뒤에만 삽입
  - D : 몇개의 adapter layer를 drop함

RoBERTa base/large

사전학습된 RoBERTa base(125M)와 large(335M) 모델을 이용해 학습했을 때, LoRA가 1/400 이하의 parameter만으로 학습해도 성능은 유사하거나 향상되는 것을 확인할 수 있다.

DeBERTa XXL(1.5B)

약 0.3%에 달하는 parameter만으로도 성능이 FineTuning보다 높은 것을 확인할 수 있다.

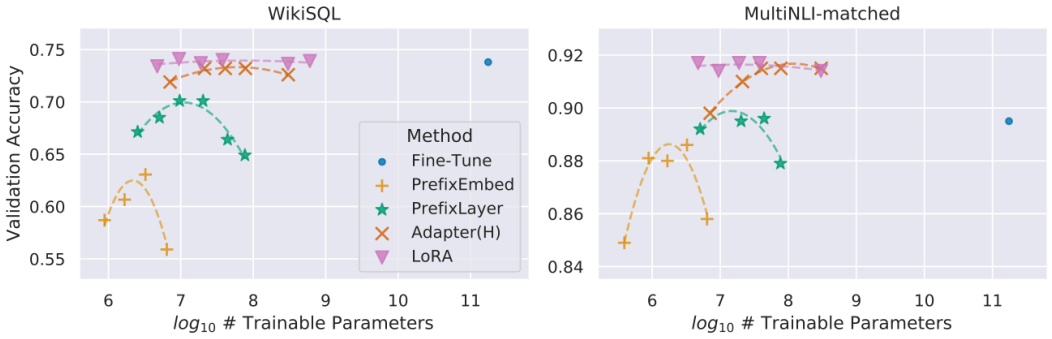
GPT-2 medium/large

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 ($\text{Adapter}^{\text{L}}$)*	0.37M	66.3	8.41	45.0	69.8	2.40
GPT-2 M ($\text{Adapter}^{\text{L}}$)*	11.09M	68.9	8.71	46.1	71.3	2.47
GPT-2 M ($\text{Adapter}^{\text{H}}$)	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 ($\text{Adapter}^{\text{L}}$)	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 ($\text{Adapter}^{\text{L}}$)	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	$\mathbf{70.4}_{\pm.1}$	$\mathbf{8.89}_{\pm.02}$	$\mathbf{46.8}_{\pm.2}$	$\mathbf{72.0}_{\pm.2}$	$2.47_{\pm.02}$

자연어 생성 task에서도 LoRA가 더 높은 품질의 task를 생성했음을 확인할 수 있다.

GPT-3 175B

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 ($\text{Adapter}^{\text{H}}$)	7.1M	71.9	89.8	53.0/28.9/44.8
GPT-3 ($\text{Adapter}^{\text{H}}$)	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



훨씬 많은 parameter를 가지는 GPT-3에서도 LoRA가 더 적은 parameter만을 학습하고도 full fine tuning보다 높은 score를 기록하였다.

다른 방식에 비해 LoRA가 적은 parameter 수로도 빠르게 성능이 향상된다. 다른 방법은 parameter 수가 늘어남에 따라 성능이 떨어지기도 하지만, LoRA는 일관적으로 높은 성능을 보여준다.

Related Works

Transformer Language Models

Transformer(Vaswani et al., 2017)는 NLP를 완전히 바꿔놓은 모델이다. BERT, GPT-2, GPT-3는 대규모 pretraining 후 fine-tuning 구조로 성능을 향상시켰다. 모델이 클수록 일반적인 fine-tuning은 성능이 좋지만, 비용 문제가 커진다. GPT-3 175B처럼 모델이 매우 클 경우, fine-tuning 자체가 어렵다.

Prompt Engineering & Fine-tuning

GPT-3는 few-shot 학습이 가능하지만, 성능이 prompt 구성 방식에 매우 민감하다. 즉, 안정성과 일관성이 부족하다. Fine-tuning 접근법은 특정 task에 특화된 모델을 만들어 준다. 대부분의 기존 방식은 전체 parameter를 다시 학습하는 방식이라 cost가 많이 필요하다. 따라서 일부 연구에서는 일부 parameter만을 학습하는 방식을 고안했지만, 여전히 효율성이 부족하다.

Parameter-Efficient Adaptation

많은 연구에서는 adapter layer를 기존 layer 사이에 넣어 fine-tuning 시에 parameter 수를 줄이는 방식을 고안했지만, 추론 시 속도 저하 문제와, 추가 parameter가 존재한다는 문제가 있다.

Low-Rank Structures in Deep Learning

많은 DNN은 실제로 저차원 구조(low rank structure) 안에 존재한다. LoRA는 이 가설을 기반으로, parameter의 변화량이 low-rank 구조로 근사가 가능하다는 점을 실험적으로 선보였다.

Understanding the Low-Rank Updates

Which Weight Matrices in Transformer Should We Apply LoRA To?

	# 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**

Transformer 내부의 다양한 weight matrix 중 어디에 LoRA를 적용해야 성능 대비 효율이 가장 좋을까?

Transformer는 W_q, W_k, W_v, W_o와 같은 다양한 weight matrix를 가진다. WikiSQL과 MultiNLI에 LoRA를 적용했을 때, $W_q + W_v$ 조합(query, value matrix에만 LoRA를 적용)에서 성능 향상이 극대화되면서 parameter는 최소화된다.

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

rank r이 모델의 성능에 어떻게 영향을 미칠까?

Weight type과 rank를 변화시키며 LoRA의 성능을 확인해 봤을 때, 놀랍게도, rank가 1일 때에도 W_q, W_v를 학습시키는 데에는 충분했다. 반면, W_q만을 학습시키는 데에는 rank가 높은 것이 좋았다.

rank가 이미 충분히 낮은 상황에서도 LoRA는 경쟁력 있는 성능을 보여준다. 증가하는 rank r이 의미있는 subspace까지 학습시키지는 않음을 의미한다.

Conclusion and Future Work

거대한 language model을 fine-tuning하는 것은 비용 측면에서 비효율적이고, 높은 하드웨어 사양을 요구한다. 본 연구에서는 LoRA를 통해, 추론 latency를 초래하거나, 높은 성능을 위해 input sequence의 길이를 줄이지 않아도 되는 효과적인 방식을 제안한다.

다양한 후속 연구의 방향이 존재한다.

- LoRA가 다른 adaptation method와 함께 사용될 수 있는지
- LoRA의 fine-tuning 작동 방식이 무엇인지
- LoRA를 적용시킬 matrix를 경험적으로 선택했는데, 더 이론화된 방식이 있는지

- ΔW의 rank 불충분성은 W 또한 rank가 불충분함을 의미할 수 있음