



(9주차) LLM Fine-tuning



● Pre-training Fine tuning 패러다임

Fine tuning

- **스케일링 법칙(Scaling Laws)**: 모델 성능이 모델 크기, 데이터셋 크기, 컴퓨팅 파워에 비례하여 증가
- **사전학습 후 파인튜닝 (Pre-train then Fine-tune) 패러다임**: 대규모 데이터로 범용 모델을 사전학습시킨 후, 특정 작업/도메인 데이터로 추가 학습(파인튜닝) 하는 것이 효율적인 접근법이라는 인식 확산



● LLM Fine tuning의 목적

Fine tuning 목적

- 작업 특화: 특정 작업에 맞춘 성능 최적화
- 도메인 적응: 의료, 법률, 금융 등 특정 도메인의 전문 용어와 맥락 반영
- 언어 특화: 한국어의 띄어쓰기, 어미 변화, 고맥락적 표현 반영
- 효율성: 전체 모델 재학습 없이 최소한의 조정으로 성능 향상



● LLM Fine tuning의 필요성

Fine tuning 필요성

- 언어적 한계 극복: 영어 중심의 사전 학습 데이터로 인해 한국어 특화 작업에서 성능 저하 발생
- 작업별 요구사항: 감정 분석은 이진 분류, 텍스트 생성은 문맥 연속성 유지 등 작업별 최적화 필요
- 문화적 맥락: 한국어 특유의 정서, 유머, 관용구 등을 모델에 반영
- 비용 효율성: 전체 모델을 재학습시키는 대신 파인 튜닝으로 계산 자원 절약
- 실시간 응용: 한국어 챗봇, 고객 지원 시스템, 콘텐츠 생성 등에서 빠르고 정확한
- 응답 제공작업 특화: 특정 작업에 맞춘 성능 최적화



● LLM Fine tuning의 개요

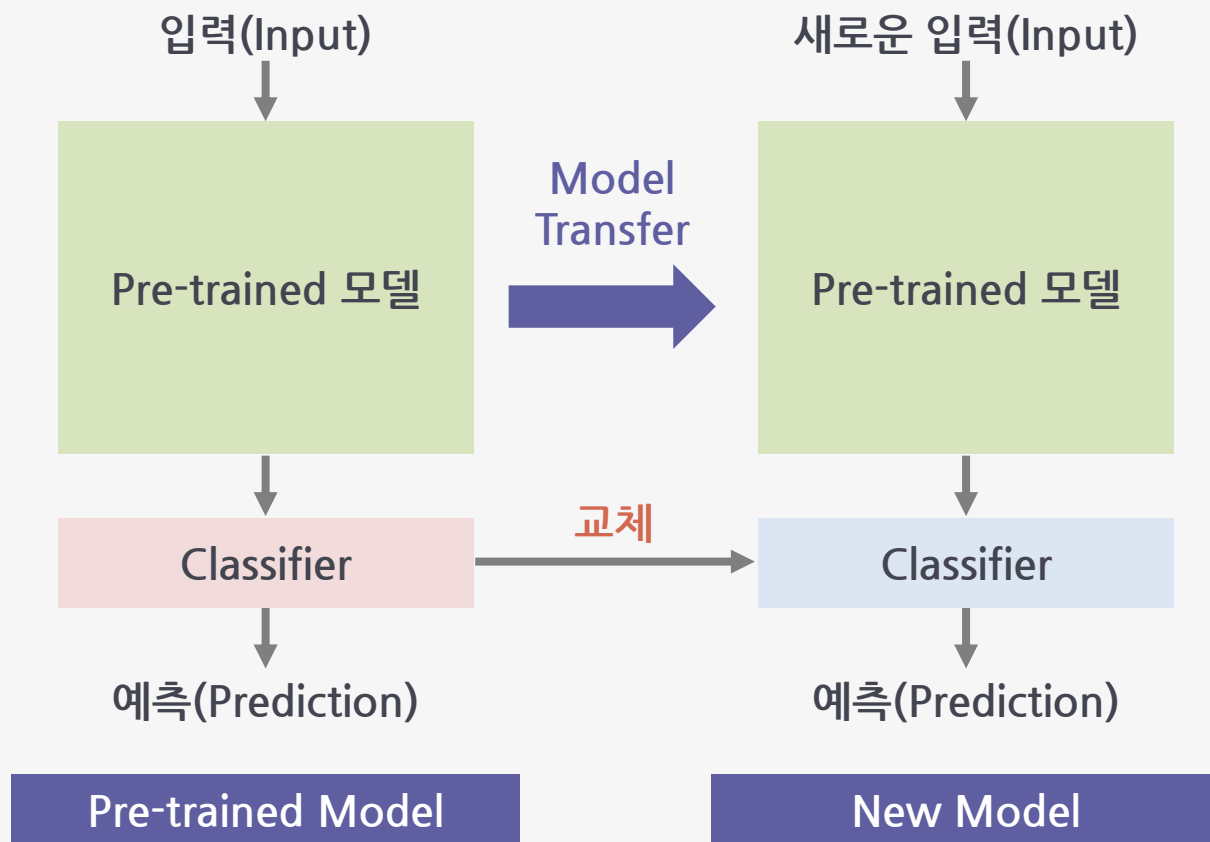
Fine tuning이란?

대규모 데이터 셋으로 사전에 학습이 완료된
언어모델(Pre-trained Language Model)을 사용



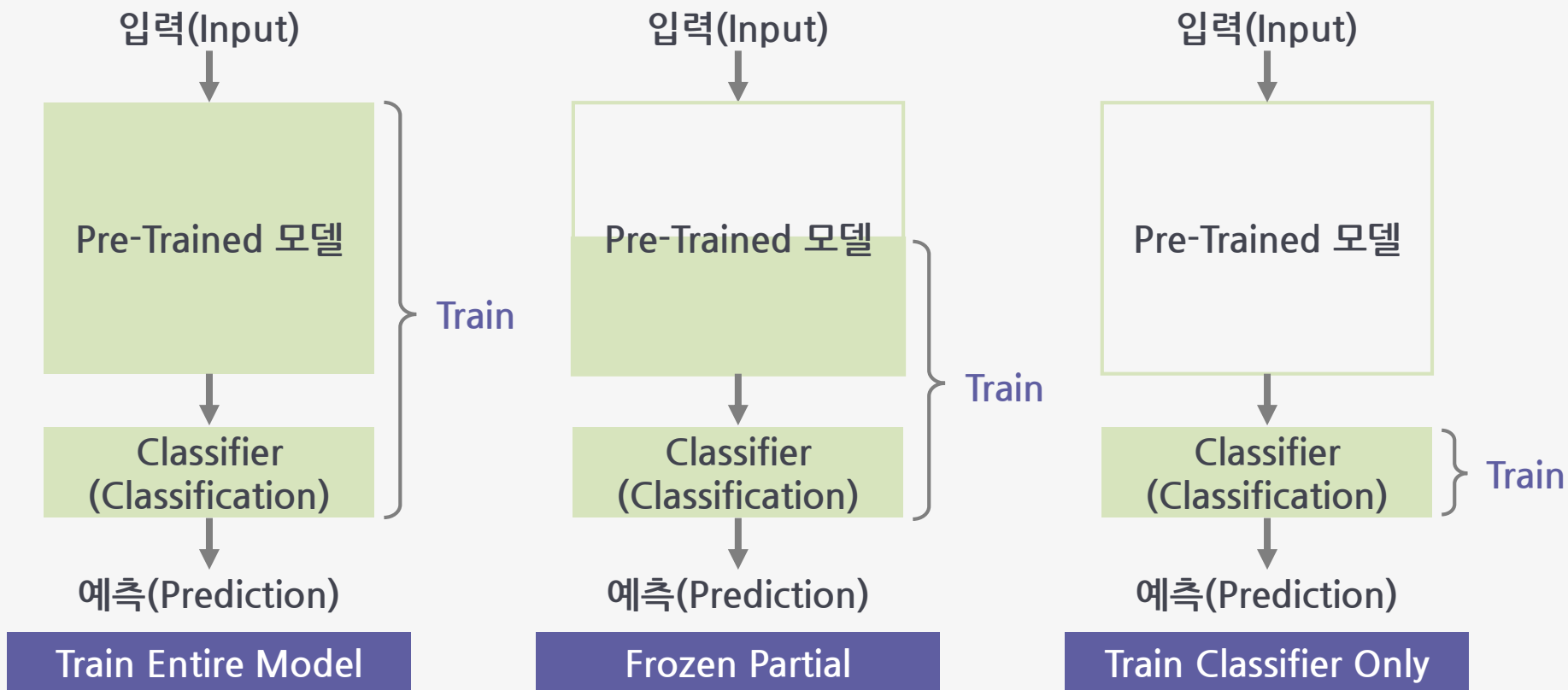
도메인 적응 (Domain Adaptation), 작업 특화 (Task Specialization),
및 언어 특화 등을 위한 추가 학습 프로세스

● Fine tuning의 개요



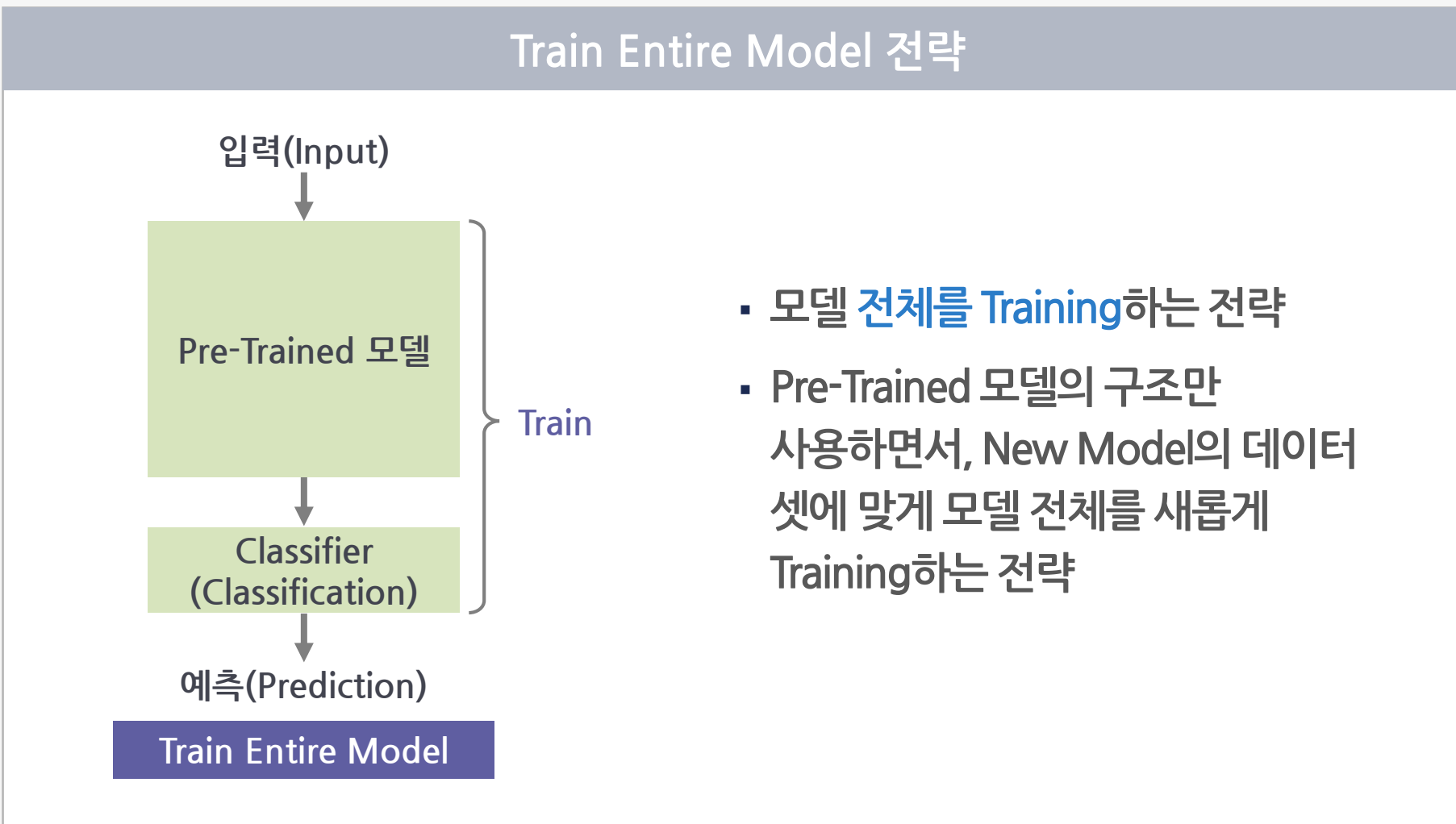
● Pre-Trained 모델의 Fine-Tuning 전략

- 전이 학습을 통해 새롭게 생성된 New Model에 대한 최적화된 Train 전략
- Train Entire Model, Frozen Partial, Train Classifier Only 전략이 존재



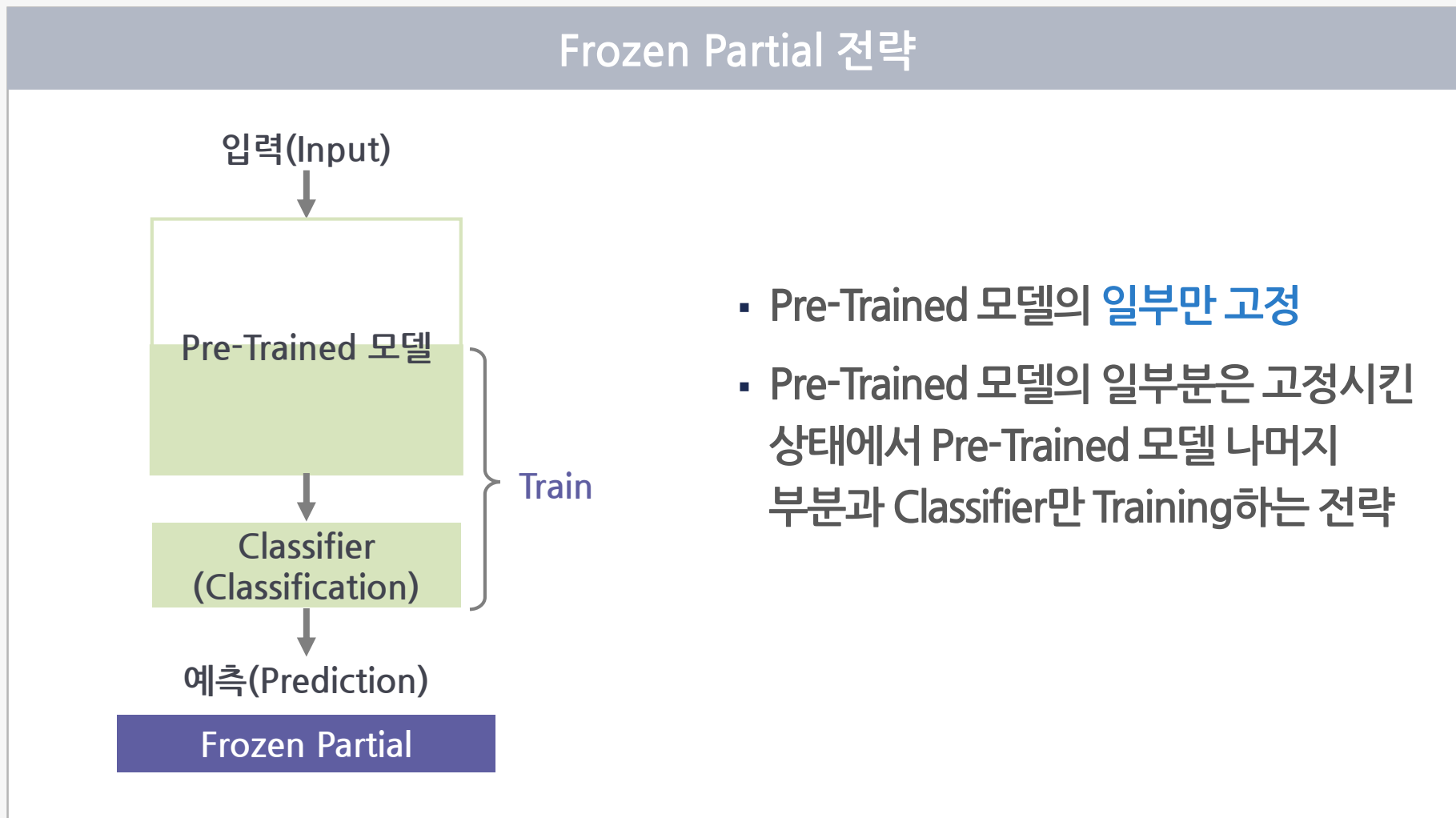


● Pre-Trained 모델의 Fine-Tuning 전략



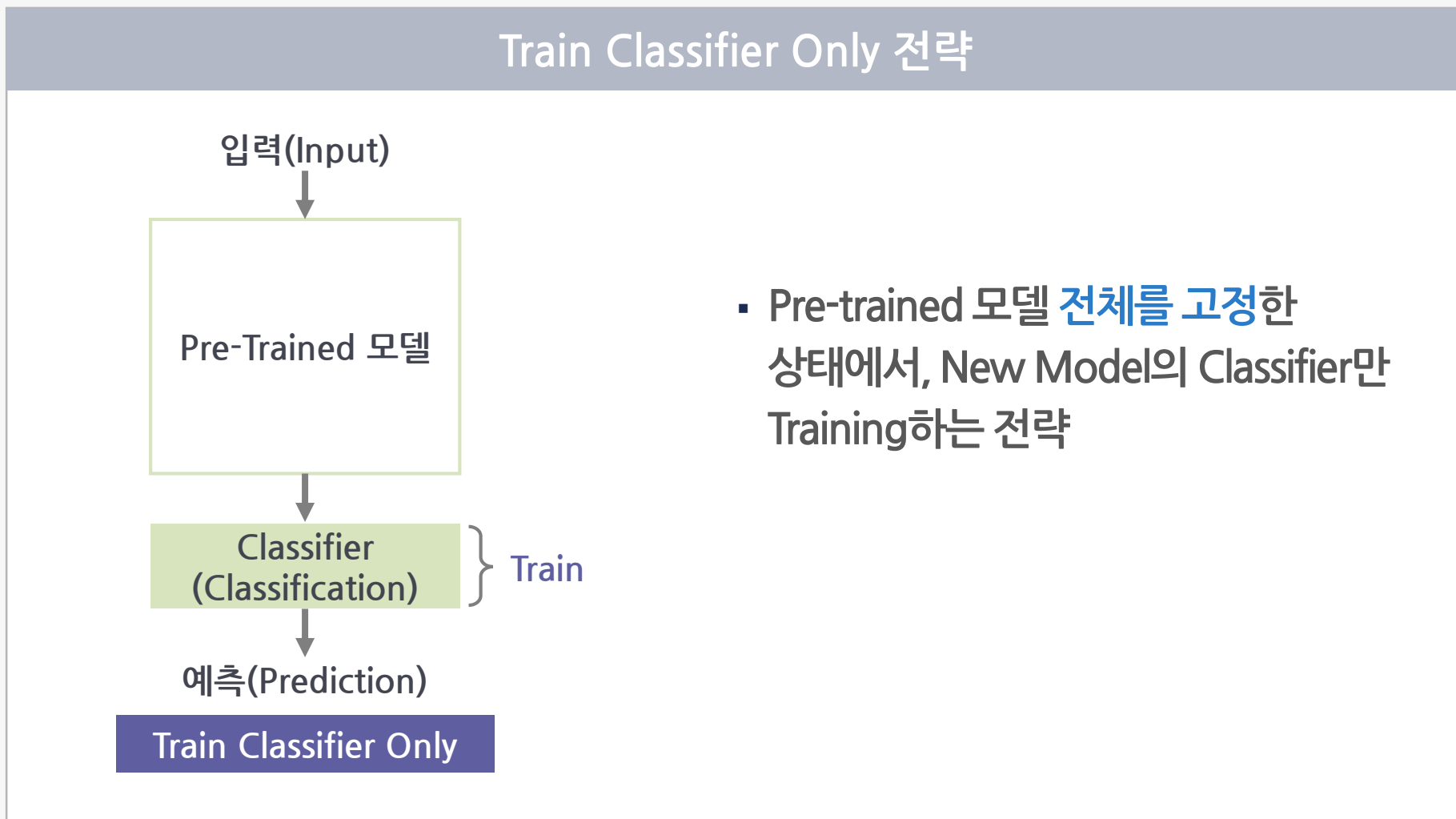


● Pre-Trained 모델의 Fine-Tuning 전략



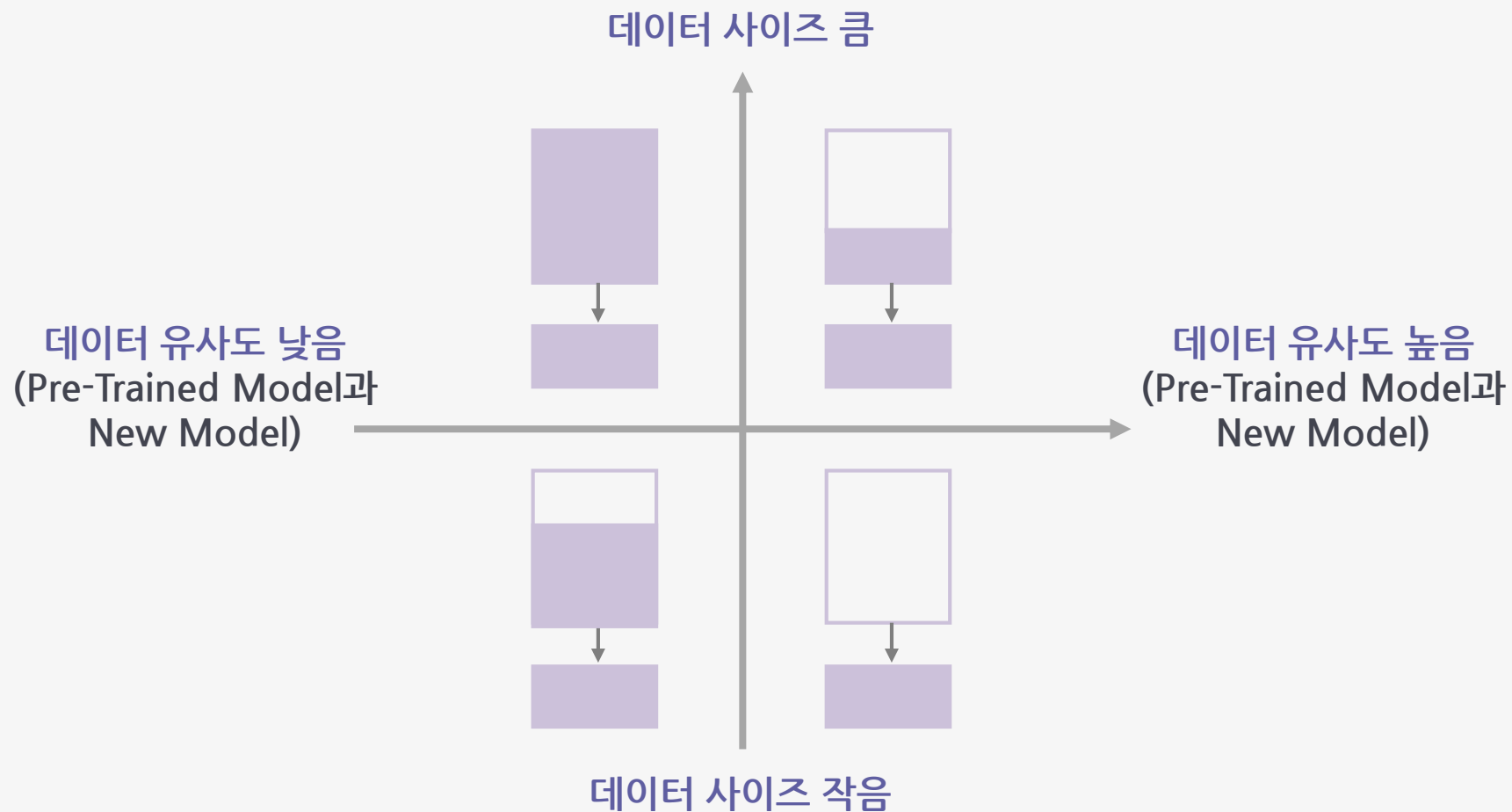


● Pre-Trained 모델의 Fine-Tuning 전략



● 전이 학습 전략의 선택

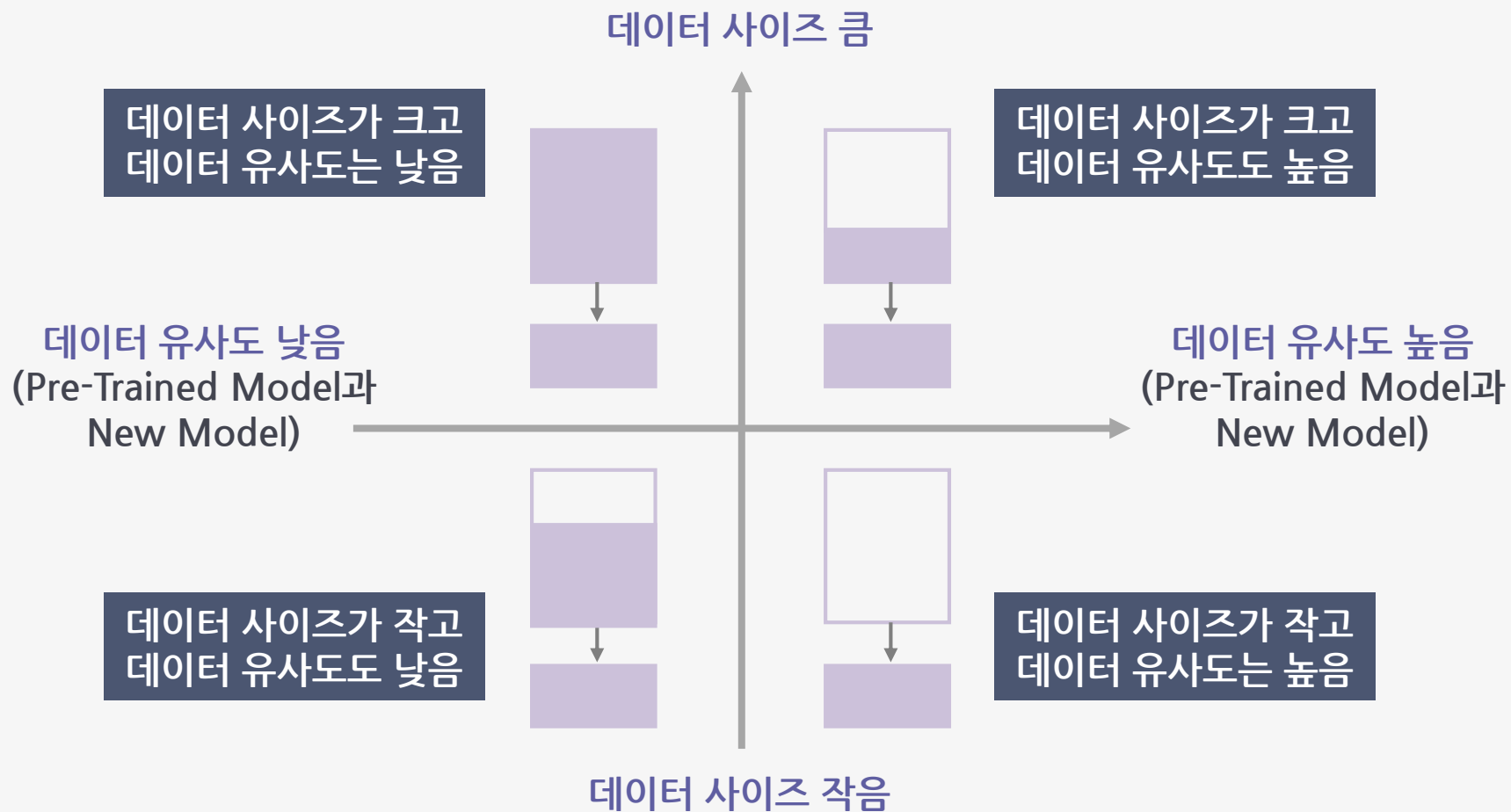
데이터 크기와 데이터 유사도에 따라 선택 필요





● 전이 학습 전략의 선택

데이터 크기와 데이터 유사도에 따라 선택 필요





● LLM Fine tuning의 절차



데이터셋 준비

작업에 맞는 고품질
데이터 수집 및 전처리



모델 선택

LLaMA, Gemma 등
사전 학습된 모델 선택



학습 환경 설정

하이퍼파라미터(학습
률, 배치 크기 등) 설정



모델 학습

준비된 데이터로 파인
튜닝 수행



평가 및 검증

테스트 데이터로 모델
성능 평가 및 최적화



배포 및 유지보수

학습된 모델 배포 및
성능 모니터링



● LLM Full Fine tuning의 문제점

상세 설명

- 모델 파라미터와 VRAM:

모델 가중치 로드 뿐만 아니라, 학습 중 발생하는 그래디언트(Gradient), 옵티마이저 상태(Optimizer States, AdamW는 파라미터당 약 8바이트 추가 필요), 활성화(Activation) 값 등을 저장하기 위해 모델 크기의 몇 배에 달하는 VRAM이 필요.

- 예시 (대략적 추정): Llama 3 8B 모델 풀 파인튜닝 시 (FP16 기준) 최소 40GB 이상의 VRAM이 필요할 수 있으며, 배치 크기나 시퀀스 길이에 따라 더 증가.

Llama 3 70B는 훨씬 더 많은 자원 요구 (A100 80GB 여러 대 필요).

- 학습 시간: 데이터셋 크기, 모델 크기, GPU 성능 및 개수에 따라

수 시간에서 수 일, 수 주까지 소요

- 저장 비용: 70B 모델 하나가 FP16으로 약 140GB. 작업별로 파인튜닝 모델을 저장 시 스토리지 부담 가중



● PEFT(Parameter Efficient Fine Tuning) 개요

PEFT(Parameter Efficient Fine Tuning) 이란?

PEFT는 대규모 모델의 모든 파라미터를 학습시키는 대신, 소수의 파라미터만 조정하여 효율적으로 파인 튜닝하는 기법



메모리와 계산 자원이 제한적인 컴퓨팅 환경에서 유용



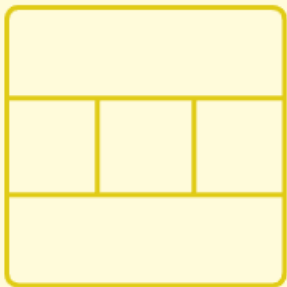
● PEFT(Parameter Efficient Fine Tuning) 장점

PEFT 장점

- 메모리 효율성: 전체 가중치 대신 소수 파라미터만 업데이트
- 학습속도 개선: 학습 시간과 계산 자원 절약
- 모듈화: 작업별로 독립적인 PEFT 모듈 생성 가능
- 전이 학습: 동일한 사전 학습 모델을 여러 작업에 활용
- 저장 효율성: PEFT로 학습된 파라미터만 저장하여 디스크 공간 절약
- 확장성: 대규모 모델에서도 적용 가능

PEFT(Parameter Efficient Fine Tuning)

● PEFT(Parameter Efficient Fine Tuning) 종류



LoRA/QLoRA

저차원 행렬 곱으로
가중치 행렬
업데이트를 근사



Adapter

레이어 사이에 작
은 신경망 모듈을
삽입하고 해당
모듈만 학습



Prompt Tuning

입력 임베딩에 학습
가능한 가상 토큰을
추가



Prefix Tuning

각 레이어의 어텐션
메커니즘에 학습
가능한 프리픽스를
추가

PEFT(Parameter Efficient Fine Tuning)



● PEFT(Parameter Efficient Fine Tuning) 종류

기법	장점	단점/고려사항	추천 상황
LoRA / QLoRA	성능 저하 적음, 높은 파라미터 효율성, 적용 용이성, QLoRA는 메모리 극대화	최적 r 값, target_modules 등 하이퍼파라미터 튜닝 필요	대부분의 파인튜닝 작업, 특히 자원 제약 시 강력 추천
Adapter	파라미터 효율성 높음, 모듈식 구조 다중 작업 적응 용이	추론 시 약간의 지연 시간 증가 가능성, 최적 bottleneck 크기 찾기	여러 작업을 하나의 모델로 처리하거나, 모듈 교체가 중요할 때
Prompt Tuning	극도의 파라미터 효율성 (가장 적음) 원본 모델 불변	복잡한 작업에서 성능 한계 가능성 프롬프트 길이 제약	매우 적은 파라미터 수정이 필요하거나, 간단한 작업 적응 시
Prefix Tuning	Prompt Tuning보다 표현력 높음 원본 모델 불변	Prompt Tuning보다 파라미터 다소 많음, 구현 복잡성 약간 증가	Prompt Tuning으로 성능 부족 시 대안으로 고려



● LoRA(Low-Rank Adaptation) 개요

LoRA(Low-Rank Adaptation) 란?

모델의 가중치 행렬에 저차원 행렬을 추가하는 파인 튜닝하는 기법



학습 파라미터 수를 대폭 감소가 가능한 PEFT 기법

LoRA(Low-Rank Adaptation) 개요

LoRA

LoRA는 다음과 같은 방식으로 가중치 업데이트를 수행

$$W = W_0 + \Delta W, \Delta W = BA$$

$$BA \in \mathbb{R}^{d \times r}$$

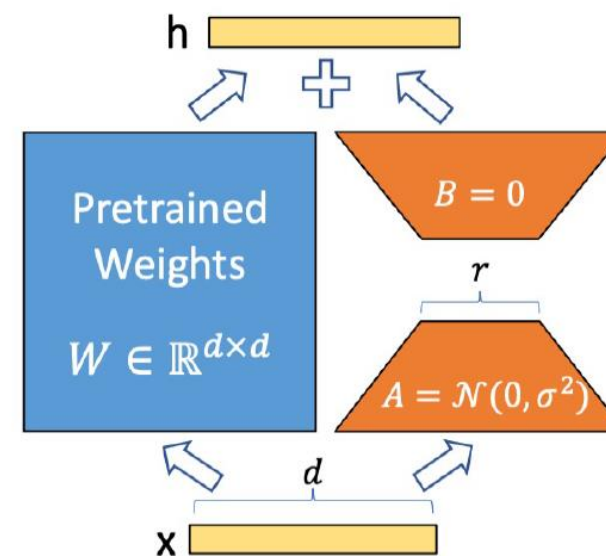
- W_0 : 원래의 사전 학습된 가중치 행렬 (고정)
- ΔW : 추가 학습되는 가중치 변화
- $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times d}$: 저차원 행렬, r 은 랭크

LoRA는 B 와 A 만 학습하므로 파라미터 수가 크게 감소

예) $d = 4096$, $k = 4096$, $r = 8$ 일 경우,

원래 파라미터 수 $d \times d = 16.8\text{M}$ 에 비해

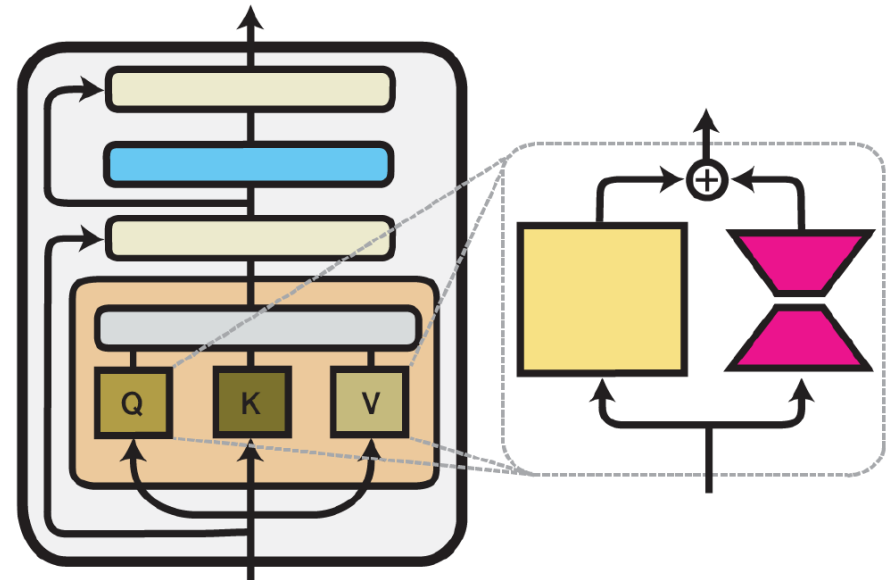
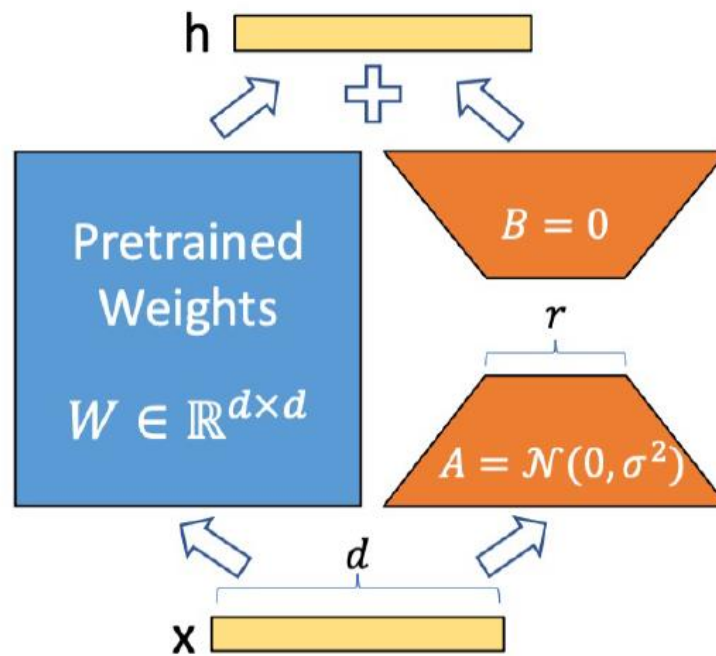
LoRA는 $d \times r + r \times d \approx 65.5\text{K}$ 만 학습



PEFT(Parameter Efficient Fine Tuning)

LoRA(Low-Rank Adaptation) 개요

LoRA





◉ Unsloth 개요

Unsloth 개념

- 대규모 언어 모델(LLM) 파인튜닝(미세조정)을 위한 오픈소스 라이브러리
- 속도 및 메모리 효율성 극대화에 초점
- <https://github.com/unslothai/unsloth>

Unsloth 목표

- 고성능 GPU 없이도 사용자 맞춤형 LLM 제작 지원
- LLM 파인튜닝 진입 장벽 완화













● Unsloth 특징 및 장점

Unsloth 특징 및 장점

- 속도 향상: 기존 방식(Hugging Face) 대비 2배~최대 30배 빠른 학습
- 메모리 사용량 감소: LoRA/QLoRA 효율적 구현으로 VRAM 사용량 최대 80% 절감
저사양 GPU 및 무료 클라우드 환경(예: Colab)에서도 대형 모델 파인튜닝 가능
(예: 7B 모델, 5GB VRAM)
- 정확도: 속도 및 메모리 효율성 개선에도 불구하고, 모델 정확도 손실 최소화 또는 향상
- 사용 편의성: 간결한 API 제공, 소량의 코드로 파인튜닝 시작 가능
- 지원 모델 다양성: Llama, Mistral, Phi, Gemma, Qwen 등 주요 오픈소스 LLM 지원



● Unsloth 지원 LLM 모델

Unsloth supports	Free Notebooks	Performance	Memory use
Qwen3 (14B)	 Start for free	2x faster	70% less
GRPO (R1 reasoning)	 Start for free	2x faster	80% less
Gemma 3 (4B)	 Start for free	1.6x faster	60% less
Llama 3.2 (3B)	 Start for free	2x faster	70% less
Phi-4 (14B)	 Start for free	2x faster	70% less
Llama 3.2 Vision (11B)	 Start for free	2x faster	50% less
Llama 3.1 (8B)	 Start for free	2x faster	70% less
Mistral v0.3 (7B)	 Start for free	2.2x faster	75% less
Ollama	 Start for free	1.9x faster	60% less
DPO Zephyr	 Start for free	1.9x faster	50% less



● Unsloth 지원 LLM 모델

구분	FP32 (기준)	FP16	BF16
총 비트 수	32	16	16
부호 비트	1	1	1
지수 비트	8	5 (좁은 범위)	8 (넓은 범위, fp32와 유사)
가수 비트	23	10 (fp32보다 낮음)	7 (fp16보다도 낮음)
표현 범위	넓음	매우 좁음	넓음 (fp32와 유사)
정밀도	높음	중간	낮음
주요 장점	-	메모리↓, 속도↑, 높은 정밀도(bf16 대비)	메모리↓, 속도↑, 넓은 범위로 학습 안정성↑
주요 단점	메모리↑, 속도↓	표현 범위 매우 좁음, 오버/언더플로우	정밀도 매우 낮음
딥러닝 선호도	기본	추론, 정밀도 중요시	학습 (특히 대규모 모델)