



# LLM 파인튜닝 가이드

## 1. 기초 개념 이해

### 사전학습 vs 파인튜닝

#### 사전학습(Pretraining)

- **정의:** 언어의 기본 능력을 익히는 단계
- **데이터:** 대규모 일반 텍스트 (책, 뉴스, 위키 등)
- **목표:** 문법, 어휘, 상식 등 언어 전반의 기초 능력 학습
- **방식:** 자가지도학습 (Self-Supervised Learning)
  - 예: 문장에서 단어를 가리고 맞추게 하는 마스크 방식
- **비유:** 모든 분야의 책을 읽으며 언어 감각을 익히는 과정

#### 파인튜닝(Fine-tuning)

- **정의:** 특정 목적에 맞게 모델을 조정하는 단계
- **데이터:** 작은 규모의 레이블 데이터 (질문-답변 쌍 등)
- **목표:** 특정 작업(번역, 대화, 감정분류 등)에 특화
- **방식:** 지도학습 (Supervised Learning)
- **비유:** 특정 분야(의학, 법률 등)에 대해 집중적으로 공부하는 과정

### 파인튜닝 공식

$$W_{ft} = W_{pt} + \Delta W$$

(파인튜닝된 가중치 = 사전학습된 가중치 + 가중치 업데이트)



## 2. 효율적 파인튜닝 기법 (PEFT)

### PEFT란 무엇인가?

#### 핵심 아이디어

- 전체 파라미터를 다시 학습하지 않고 일부만 업데이트
- 메모리와 시간을 대폭 절약하면서도 높은 성능 유지

**비유:** 모델의 전체 건물(전체 파라미터)을 다시 짓는 대신, 창문과 조명(일부 파라미터)만 교체하여 전체 분위기(출력 성능)를 효과적으로 바꾸는 전략

## LoRA (Low-Rank Adaptation)

### 🔑 핵심 개념

- **Low-Rank:** 낮은 계수를 가진 간단한 구조
- **Adaptation:** 새로운 상황에 맞게 조정하는 과정

### 작동 원리

1. 기존 모델의 가중치(W)는 고정
2. 각 가중치 행렬에 저랭크 행렬 A, B 추가
3. 학습은 오직 A, B 행렬만 수행 → 전체 파라미터의 0.1~1%만 업데이트

### 수학적 표현

$$W_{ft} = W_{pt} + \underset{\text{Approximation}}{\Delta W} = W_{pt} + AB$$

where  $W_{ft}, W_{pt}, \Delta W, AB \in \mathbb{R}^{d \times d}$   
and  $A \in \mathbb{R}^{d \times r}, B \in \mathbb{R}^{r \times d}$

### 실제 예시

- 4096차원 가중치를 8차원 행렬 두 개로 근사하면 →  $r = 8$
- 기존 가중치에 소규모로 학습된 조정값(A×B)을 더해서 성능 향상

**비유:** 하나의 큰 가구를 통째로 들여놓는 대신, 두 개의 작고 가벼운 부품으로 나눠 조립하면 공간도 절약되고 이동·설치도 훨씬 간편

## QLoRA (Quantized LoRA)

### 배경

- LoRA도 메모리 효율적이지만, 기본 모델은 여전히 고정밀도 형식
- 사전학습된 모델 자체를 4비트로 양자화하여 메모리 사용량을 극적으로 줄임

### 작동 방식

- 기존 가중치 W를 4비트로 양자화 (예: NF4 형식)

- LoRA처럼  $\Delta W = A \times B$  구조만 학습
- **메모리 절약**: 최대 75% 이상 감소

## 양자화(Quantization) 이해



### 개념

- 모델의 가중치와 연산을 더 낮은 비트 수로 표현하는 기법
- 32비트 → 16비트, 8비트, 심지어 4비트로 축소

### 비트별 표현 가능한 값

비트 수	표현 가능한 값의 개수
1비트	2개 (0~1)
4비트	16개 (0~15)
8비트	256개 (0~255)
16비트	65,536개
32비트	약 43억 개

### 장점과 단점

-  메모리 최대 75% 이상 절감
-  수치 표현 정밀도 하락으로 약간의 성능 저하 가능

## 3. 실전 도구와 프레임워크

### PyTorch

#### 특징

- 동적 계산 그래프: 실행 시점에 그래프 구성, 직관적 코드 작성
- 자동 미분: Autograd 엔진으로 역전파 자동 처리
- 다재다능함: 다양한 모델 구조와 실험에 유연하게 활용

### Hugging Face

개념: LLM을 위한 종합 플랫폼 (AI의 앱스토어)

#### 핵심 기능

1. **모델**: 한 줄로 불러오고, 한 줄로 공유

```
from transformers import AutoModelForCausalLM, AutoTokenizer
model = AutoModelForCausalLM.from_pretrained("meta-llama/Llama-2-7b-hf")
```

#### 1. 데이터셋: 실습용부터 연구용까지 다양하게 제공

```
from datasets import load_dataset
dataset = load_dataset("squad")
```

#### 1. **Hugging Face Hub**: 모델, 데이터셋, 데모 페이지가 프로젝트 카드처럼 정리

## Unsloth

**개념**: 대규모 언어 모델을 더 빠르고 가볍게 파인튜닝하기 위한 최신 도구

### 주요 장점

- 파인튜닝 시 학습 속도 2배 이상
- 메모리 절약 최대 60%
- 기존 PyTorch 기반 코드에서 메모리 사용량 50~60% 이상 절감
- 4bit 양자화, Flash Attention, Gradient Checkpointing 등 자동 적용

## 4. 핵심 하이퍼파라미터

### Learning Rate (학습률)

**개념**: 모델이 얼마나 빨리 배울지 정하는 숫자

### 영향

- **너무 크면**: 엉뚱한 방향으로 튀고, 학습이 불안정
- **너무 작으면**: 학습이 너무 느려져서 거의 배우지 못함

**권장사항**: Warmup and Scheduler 함께 사용

- **Warmup**: 처음엔 천천히 시작해서 점점 속도를 높임
- **Scheduler**: 시간이 지나면 다시 천천히 줄여서 안정적으로 마무리

### Warmup Steps






**개념**: 학습 초기에 학습률을 서서히 증가시키는 단계

- **비유:** "준비 운동 없이 달리면 다친다"는 개념과 유사
- **설정:** 일반적으로 전체 step의 1~5% 수준

## Batch Size

**개념:** 한 번에 사용하는 데이터 샘플 수

**영향**

- **작으면:**
  -  GPU 메모리 부담 적고, 빠르게 자주 업데이트
  -  다양한 데이터 패턴을 더 잘 반영
  -  결과가 흔들릴 수 있어 학습이 조금 불안정
- **크면:**
  -  안정적인 평균값으로 학습, 손실이 부드럽게 감소
  -  메모리를 더 많이 소모, 가끔 오버피팅

**권장:** 보통 4~64 사이 값, GPU 메모리 상황에 맞춰 조정

## Iteration vs Epoch

개념	의미
Iteration	배치 하나를 학습하는 반복 (한 번)
Epoch	전체 데이터를 한 번 학습하는 것

**예시:** 데이터 1,000개 / 배치 크기 100 → 1 Epoch = 10 Iteration

## LoRA 전용 설정

**r (rank)**

- LoRA에서 삽입되는 작은 행렬의 내부 차원 수
- r이 작으면: 학습 파라미터 수와 연산량 줄어듦
- r이 크면: 표현력 올라가지만 자원 소비도 증가

**lora\_dropout**

- 학습 중 일부 뉴런을 무작위로 끄는 비율
- 과적합 방지하고 모델의 일반화 성능 향상
- 예: lora\_dropout=0.05 → 5% 확률로 뉴런을 꺼서 학습

## 5. 실무 필수 지식

### 토큰나이저(Tokenizer)

개념: 문장을 토큰으로 나누는 도구

과정: 텍스트 → 토큰 → 숫자 ID

"나는 학교에 간다" → ['\_\_나는', '\_\_학교', '에', '\_\_간다'] → [123, 456, 78, 910]

주의사항: 모델마다 토큰나이저 방식이 달라서 반드시 모델과 같은 토큰나이저 사용 필수

### GPU 선택 가이드

구분	가속기 종류	Colab 지원	용도/특징	파인튜닝 적합성
T4 GPU	Colab 무료/Pro	✅ 무료/Pro	저렴한 보급형, 약 16GB VRAM	❌ 대형 모델에는 부족
L4 GPU	Colab Pro+	❌ 무료/✅ Pro+	T4보다 빠르고 효율적	💎 추론 위주, 학습에는 제한
A100 GPU	Colab Pro+	❌ 무료/✅ Pro+	고성능 (40~80GB), 대형 모델 학습 최적	✅ LLM 파인튜닝에 적합
TPU v2-8	Google Cloud	❌ Colab 기본 미지원	병렬 처리 특화	💎 복잡한 설정 필요

권장:

- 실습/소형 모델: T4 GPU
- 본격적인 파인튜닝: A100 GPU 이상

## 6. 파인튜닝 실행 로드맵

### 7단계 실행 체크리스트

단계	주요 내용	핵심 포인트
1. 사전학습 모델 선택	LLaMA, Mistral 등	오픈소스 여부, 모델 크기(B 단위), 토큰나이저 호환
2. 데이터셋 준비	질문-응답 형식 (Alpaca 등)	포맷 변환 (JSON, ChatML), 전처리 필요
3. 토큰나이저 설정	텍스트 → 토큰 변환	모델과 동일한 토큰나이저 사용

단계	주요 내용	핵심 포인트
4. 파인튜닝 방식 선택	Full, LoRA, QLoRA	자원 상황에 따라 선택 (LoRA는 경량)
5. 하이퍼파라미터 설정	learning_rate, batch_size 등	실험적으로 조정 필요
6. 학습 수행	Trainer, SFTTrainer 등	GPU/TPU 활용, 체크포인트 저장
7. 모델 저장 및 배포	save_pretrained(), Hub 업로드	추론 API 개발, 로컬 또는 Hugging Face 배포

## 각 단계별 주의사항

### 1단계: 모델 선택

- 사용 목적에 맞는 크기 선택 (7B, 13B, 70B 등)
- 라이선스 확인 (상업적 이용 가능 여부)
- 언어 지원 확인 (한국어 성능)

### 2단계: 데이터 준비

- 고품질 데이터가 소량이 대량의 저품질 데이터보다 효과적
- 일관된 포맷 유지
- 적절한 데이터 전처리 수행

### 3-4단계: 방식 선택

- 리소스가 제한적이면 LoRA/QLoRA 선택
- 최고 성능이 필요하면 Full Fine-tuning 고려

### 5단계: 하이퍼파라미터

- 작은 학습률부터 시작 ( $1e-5 \sim 5e-4$ )
- 과적합 방지를 위한 조기 종료 설정
- 정기적인 체크포인트 저장

### 6-7단계: 학습 및 배포

- 학습 과정 모니터링 (loss, accuracy)
- 검증 데이터로 성능 평가
- 배포 전 충분한 테스트

## 7. 참고자료 및 추가학습

### 유용한 링크 모음

- **Hugging Face Transformers:** <https://huggingface.co/docs/transformers>
- **LoRA 논문:** "LoRA: Low-Rank Adaptation of Large Language Models"
- **QLoRA 논문:** "QLoRA: Efficient Finetuning of Quantized LLMs"
- **Unsloth:** <https://github.com/unslothai/unsloth>

### 실습 예제 추천

1. 초급: Hugging Face의 sentiment analysis fine-tuning
2. 중급: LoRA를 사용한 chatbot fine-tuning
3. 고급: QLoRA를 활용한 대형 모델 domain adaptation

### 심화 학습 방향

- **RLHF (Reinforcement Learning from Human Feedback)**
- **DPO (Direct Preference Optimization)**
- **Multi-modal Fine-tuning**
- **Distributed Training 기법**

---

## 마무리

파인튜닝은 사전학습된 LLM을 특정 목적에 맞게 조정하는 핵심 기술입니다. PEFT 기법들, 특히 LoRA와 QLoRA를 활용하면 제한된 자원으로도 효과적인 파인튜닝이 가능합니다.

### 성공적인 파인튜닝의 핵심:

1. 명확한 목적과 고품질 데이터
2. 적절한 모델과 기법 선택
3. 체계적인 하이퍼파라미터 튜닝
4. 충분한 실험과 검증

**기억할 점:** 파인튜닝은 예술과 과학의 조합입니다. 이론적 이해를 바탕으로 많은 실험을 통해 최적의 결과를 얻을 수 있습니다.