MODULE 03

품질을 조금 양보하고 속도와 메모리를 얻는 법

모델의 크기를 줄이는 기술인 양자화를 다룹니다. 얼마나 줄일 수 있고, 그 대가로 무엇을 내주는지, 그리고 내 장비에 실제로 들어가는지를 직접 계산해보는 것까지가 이번 모듈의 목표입니다.

01 모델의 무게는 어디서 오나

모델 파일의 실체는 **수십억 개의 숫자(파라미터) 덩어리**입니다. 이 숫자 하나하나를 몇 비트(bit, 0 또는 1 하나를 담는 가장 작은 저장 단위)로 저장하느냐가 파일 크기, 즉 우리가 흔히 말하는 "모델의 무게"를 결정합니다. (단위 감각이 더 필요하면 워밍업을 참고하세요.)

지금까지 다뤄온 기본 정밀도는 **FP16(FP=부동소수점, 소수점 위치를 유동적으로 표현하는 방식 중 16비트짜리)**입니다. 8비트를 묶은 것이 1바이트이므로 16비트는 2바이트이고, 대략 파라미터 수 × 2바이트로 계산하면 되는데, 이 기준으로는 **7B(=70억 개) 모델이 약 14GB, 30B(=300억 개) 모델이 약 60GB**에 이릅니다.

문제는 여기서 시작됩니다. 게이밍 노트북의 VRAM은 보통 8~12GB 수준이라, **가장 작은 축인 7B 모델조차 그대로는 빠듯합니다.** 그래서 파일 크기를 줄이는 방법이 필요해지는데, 그 답이 이번 모듈의 주제인 양자화입니다.

02 양자화 = 정밀한 반올림

양자화(quantization)는 각 숫자를 표현하는 데 쓰는 비트 수를 줄이는 작업입니다. 16비트를 8비트(Q8)로, 다시 4비트(Q4)로 낮추면 파일 크기는 각각 절반, 1/4로 줄어듭니다.

> 💡 **비유**

> 원주율 3.14159265를 3.14로 적는 것과 비슷합니다. 대부분의 계산에서는 둘의 차이가 드러나지 않지만, 아주 미세한 구분이 필요한 계산에서는 그 오차가 보이기 시작합니다. 양자화도 마찬가지로 대부분의 상황에서는 티가 안 나지만, 정밀함이 필요한 순간에는 품질 차이가 나타날 수 있습니다.

표기 읽는 법: `Q4_K_M`
앞의 `Q4`는 "파라미터를 4비트로 양자화했다"는 뜻이고, 뒤의 `K_M`은 세부 양자화 방식과 크기 변형을 나타내는 표기입니다 — `K`·`K_M`·`_0` 등은 양자화 세부 알고리즘의 차이이며, 같은 비트 수라면 `K_M` 계열이 보통 더 안전한 기본값입니다. 실무에서 가장 널리 쓰는 기본값이 바로 `Q4_K_M`이고, 품질을 조금 더 지키고 싶다면 `Q6_K`나 `Q8_0`을 고릅니다.

양자화의 효과는 두 가지입니다. 첫째, 파일이 작아지니 **더 작은 VRAM에도 모델이 들어갑니다.** 둘째, 파일이 작아지면 매 토큰을 생성할 때마다 읽어야 하는 메모리 양도 줄어들어 **속도까지 빨라집니다**(모듈 1에서 본 "대역폭 ÷ 모델 크기" 공식을 떠올려 보세요).

03 GGUF — 로컬 추론용 표준 포맷

GGUF는 llama.cpp 계열 엔진(Ollama, LM Studio 포함)이 읽는 단일 파일 포맷입니다. 양자화를 마친 모델은 대부분 이 포맷으로 배포됩니다.

HuggingFace(모델 파일들이 모여 있는 대표 웹사이트)에서 로컬 실행용 모델을 찾을 때 저장소 이름에 **"GGUF"**가 붙은 것을 받아야 하는 이유가 여기 있습니다. 같은 모델이라도 한 저장소 안에 Q2부터 Q8까지 여러 양자화 버전의 파일이 함께 올라가 있고, **파일 크기가 곧 그 모델을 돌리는 데 필요한 VRAM의 대략적인 기준**이 됩니다.

04 필요 VRAM 산식 — 이 계산이 서빙 계획의 절반이다

필요 VRAM ≈ ① 모델 파일 크기(GGUF 그대로) + ② KV 캐시(컨텍스트 길이·동시 사용자에 비례) + ③ 작업 버퍼(대략 1~2GB)

예를 들어 **8B 모델을 Q4_K_M으로 양자화하면 파일 크기가 약 4.9GB**(계산기는 메타데이터 여유분 5%를 더해 5.0GB로 표시됩니다)입니다. 12GB VRAM 노트북이라면 KV 캐시와 버퍼까지 포함해도 여유 있게 구

동되고, 컨텍스트도 넉넉히 잡을 수 있습니다.

반대의 경우도 있습니다. **24GB 서버에 파일 크기 22GB짜리 모델을 올리면**, 파일 자체는 들어가더라도 KV 캐시가 앉을 자리가 거의 남지 않아 컨텍스트를 제대로 잡을 수 없습니다. **"파일이 들어간다"와 "쓸 수 있다"는 서로 다른 이야기**라는 뜻입니다.

05 직접 계산해보기 (인터랙티브 계산기)

위 산식을 그대로 계산기로 옮겼습니다. 값을 바꾸면 바로 다시 계산됩니다. 감을 잡는 기준으로, 컨텍스트 길이는 대략 **A4 1장 ≈ 500~700토큰**으로 환산해 볼 수 있습니다. 그리고 **bpw**(bits per weight, 파라미터 하나당 평균 비트 수)는 `K_M` 같은 세부 방식에 따라 정수가 아닌 값(예: 4.8)이 붙고, 계산기 결과에는 메타데이터 등 여유분으로 약 5%를 더 얹고, 작업 버퍼는 산식의 1~2GB 범위 중 중간값인 1.5GB로 고정해 계산합니다.

파라미터 수 (B, 10억 단위)	8
양자화	Q4_K_M (4.8bpw)
1인당 컨텍스트 길이	8k 토큰
동시 사용자 수	1명 (혼자)
KV 캐시 부담 (크고 구형 구조일수록 무거운 쪽 — 모르면 '보통')	보통 (0.1GB / 1k tok)
내 GPU 메모리	12GB

모델 무게 5.0GB + KV 캐시 0.8GB + 버퍼 1.5GB = 총 **7.3GB** → **여유 있음** (GPU 12GB 기준)

※ 이 계산기는 산식을 몸에 붙이기 위한 학습용 근사치입니다. 실전 산정은 정밀도(Precision)·동시 처리(Batch)·컨텍스트 길이(Seq Len)까지 반영하는 APX ML VRAM Calculator를 권장하고, 실제 배치 전에는 반드시 실측으로 확정하세요.

↗ 심화 학습

더 정밀한 외부 VRAM 계산기 — 정밀도·배치·시퀀스 길이까지 반영: **APX ML VRAM Calculator** · 위 계산기가 감 잡기용이라면 이쪽은 실전 산정용입니다. "Precision, Batch, Seq Len을 무시하면 성능이 무너진다"는 것이 실무자들의 공통 경험담입니다.

체계적으로 더 배우려면: **Hugging Face LLM Course** · 영어(일부 한국어 번역), 무료 실습형 — 양자화·모델 포맷 챕터 포함.

양자화·압축의 최전선(연구 단계): **Google TurboQuant** · **arXiv**

06 품질은 얼마나 떨어지나

일반적인 경험칙은 이렇습니다. **요약·분류·추출처럼 범위가 좁은 태스크에서는 Q4_K_M이 원본(FP16)과 체감 차이가 거의 없습니다.** 반면 미세한 논리 전개, 숫자 계산, 긴 추론이 필요한 태스크는 Q6_K나 Q8_0에서 더 안정적인 결과를 보입니다.

하지만 결론은 하나로 모입니다. **소문이 아니라 내 태스크의 골든셋(우리 업무와 비슷한 실제 예시 묶음)으로 원본과 양자화본을 직접 비교하는 것**입니다. 이 비교 방법은 모듈 6(평가)에서 자세히 다룹니다.

🔗 우리 연구회에선

우리 연구회의 8월 목표가 정확히 이것입니다. 이번 모듈에서 배운 산식으로 각자 '우리 서버(24GB)에 어떤 모델·양자화·컨텍스트 조합이 최적인지'를 먼저 계산해보고, 그 계산을 실측 결과와 대조해 보는 것입니다.

CHECK

이해했는지 확인해 봅시다

틀려도 괜찮습니다 — 오답을 고르면 왜 아닌지 설명이 나오고, 정답을 찾을 때까지 다시 고를 수 있습니다. 점수는 첫 시도 기준입니다.

문제 1 / 5

Q1. 7B(70억 파라미터) 모델을 FP16으로 저장하면 대략 몇 GB인가?

① 약 3.5GB

② 약 7GB

③ 약 14GB

④ 약 28GB

← 이전
모듈 2 · LLM 동작 원리

다음 →
모듈 4 · 서빙 엔진