2025.02.06 (Winter Seminar)

Research on Methodologies Used in LLMs

1. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Model [arXiv 2024]

2. DeepSeek-OCR 2: Visual Causal Flow [arXiv 2026]

Sogang University
Vision & Display Systems Lab, Dept. . of Artificial Intelligence

Presented By
HEEDONG SEO

1) Z. Shao, P.Wang, Q.Zhu, R.Xu, J.Song, X.bi, H.Zhang, M.Zhang, Y.K. Li, Y.Wu, D.Guo, “DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Model,” arXiv preprint arXiv:2402.03300, 2024

2) H.Wei, Y.Sun, Y.Li, “DeepSeek-OCR 2: Visual Causal Flow,” arXiv preprint arXiv:2601.20552, 2026

Outline

- LLM 구조
  - LLM (DeepLearning) 모델 진행 : Training / Inference
  - LLM 진행 구조 : 데이터 수집 및 정제 → Pretraining → SFT → RL

- DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Model [arXiv 2024][1)]
  - 데이터 수집 : Seed-Positive (수학 관련), Common Crawl-Negative (수학과 관련 없는)
  - SFT : CoT, PoT, TIR
  - RL : GRPO

- DeepSeek-OCR 2: Visual Causal Flow [arXiv 2026][2)]
  - DeepEncoder V2 제안

LLM 구조

- LLM (DeepLearning) 모델은 Training / Inference로 동작 구분
  - Training : Transformer 구조 기반으로 진행 텍스트를 Tokenization + Embedding하여 다음 토큰을 예측 (학습), 모델의 지능과 성격 형성
    - Transformer, GRPO, PPO, SFT, QLoRA등의 방법론 활용
  - Inference : 모델이 완성된 후 학습이 완료된 모델을 활용하여 사용자의 질문에 대한 답변을 생성하고 문제를 해결하는 단계
    - Prompt Engineering (CoT, ToT 등), Tokenizer, Embedding, Bit 양자화 등의 방법론 활용

구분	Training (학습/개발)	Inference (추론/활용)
목적	모델의 지능과 성격 형성	생성 및 문제 해결
가중치(W)	업데이트 됨 (변함)	고정됨 (안 변함)
핵심 요소	Transformer, GRPO, SFT, QLoRA	Prompt Engineering, Tokenizer, Embedding
추론 기법	학습 데이터로서의 CoT	CoT, ToT, LoT (질문 기법)
하드웨어 이슈	학습용 VRAM 확보 (H100 등)	양자화를 통한 구동 효율화

LLM 구조

- LLM은 Pretraining, Supervised Fine Tuning (SFT), Reinforcement Learning (RL)와 같이 3단계로 구성
  - Data 수집 및 정제 → 언어의 구조를 이해하는 Pretraining 진행 → 사용자의 지시에 따른 적절한 응답을 배우는 SFT → 모델의 답변을 최적화하는 RL로 진행

Data Preparation

- **Data Collection : Web Data (Common Crawl), GitHub, arXiv 등**
- **Data Refinement : 중복제거, 저품질 문자 필터링 등**
- **Synthetic Data**

Pretraining

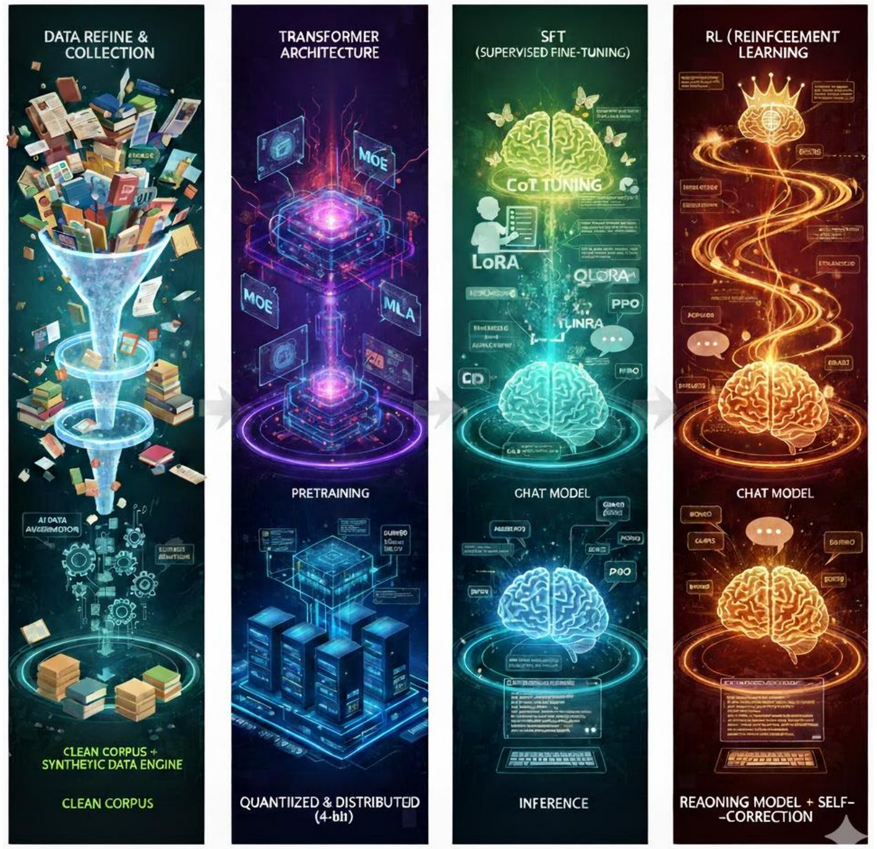
- **Tokenization**
- **Embedding**
- **Positional Encoding**
- **Transformer**

SFT

- **Prompt Engineering**
- **Full Fine-Tuning**
- **Parameter-Efficient Fine-Tuning(PEFT)**

RL

- **Direct Preference Optimization (DPO)**
- **Proximal Policy Optimization (PPO)**
- **Group Relative Policy Optimization (GRPO)**



Pretraining

- Pretraining은 연구 방법들은 목적, 아키텍처, 데이터, 연산 효율성이라는 4가지 관점에서 분류
  - 학습 목적 함수 연구 : 모델이 데이터를 통해 무엇을 어떻게 배울 것인가를 결정하는 수식적 접근
    - Causal Language Modeling (CLM) : 이전 토큰들을 보고 바로 다음에 올 단어 예측 (LLM의 표준 방식)
    - Masked Language Modeling (MLM) : 문장 중간에 Mask를 뚫어놓고 주변 문맥을 통해 빈칸을 채우도록 학습 (BERT 모델)
    - Denoising : 문장의 순서를 섞거나 단어를 임의로 삭제한 뒤 원래 문장으로 복구 (T5와 같은 Encoder-Decoder 모델)
  - 아키텍처 구조 최적화 연구 : 학습 효율을 극대화하고 모델의 용량을 키우기 위한 구조 설계 연구
    - Mixture of Experts (MoE) : 전체 파라미터 중 데이터 처리에 필요한 특정 Expert 뉴런들만 선택적으로 활성화
      - ⁘ 연산 비용 ↓, 지식 총량 비약적 확대
    - Attention 효율화 (MLA, GQA) : Multi-head Latent Attention (MLA)이나 Grouped Query Attention (GQA) 등을 통해 연산량 조절
      - ⁘ KV Cache 효율 ↑
  - 데이터 중심 연구 : 모델의 크기보다 '어떤 데이터를 넣느냐' 에 집중하는 연구
    - Scaling Laws : 데이터의 양, 모델의 크기, 컴퓨팅 자원 사이의 상관관계 분석, 최적의 학습 조합 선택
    - Curriculum Learning : 초기에는 쉬운 일반 문장으로 학습하다가, 점차 수학 등 복잡한 데이터를 주입하는 순서를 최적화
    - Synthetic Data Generation : 고품질 텍스트 부족 문제를 해결하기 위해 신규 생성한 데이터를 학습에 활용

Pretraining

- 연산 및 인프라 효율화 연구 : 하드웨어 지원을 극한으로 활용하기 위한 기술적 연구
  - Flash Attention : GPU의 메모리 접근 방식을 최적화하여 Attention 연산 속도를 하드웨어 수준에서 가속
  - Mixed Precision & BF16 : FP32 대신 FP16, BF16 등의 저정밀도 연산을 섞어 사용하여 속도는 높이고 메모리는 감소
  - Distributed Training : 수만 개의 GPU를 효율적으로 연결하여 학습을 분산 처리하는 병렬화 (Pipeline 등) 기법 연구

Supervised Fine Tuning (SFT)

- SFT는 학습 범위(효율성), 데이터의 성격(목적)에 따라 분류
  - 학습 범위 및 자원 효율성 연구 : 모델의 파라미터를 얼마나 수정 하 느냐에 초점을 맞춤
    - Full Fine-Tuning : 모델의 모든 가중치 (W)를 업데이트, 거대한 GPU(VRAM) 필요
    - Parameter-Efficient Fine-Tuning (PEFT) : 대부분의 가중치는 고정하고 극히 일부의 파라미터만 학습시키는 방식
      - Low-Rank Adaptation (LoRA) : 가중치 행렬 사이에 작은 크기의 '어댑터' 행렬을 삽입하여 이것만 학습, GPU 메모리 사용량 ↓
      - Quantized LoRA : 모델을 4-bit 등으로 양자화한 상태에서 LoRA 적용, 개인용 GPU(RTX 4090 등)에서도 대형 모델 학습 가능
      - Prompt Tuning/Prefix Tuning : 모델의 가중치는 건드리지 않고, 입력 앞단에 학습 가능한 '가상 토큰(Soft Prompt)' 붙여 최적화
  - 데이터 구성 및 목적에 따른 연구 : 데이터를 어떤 형식으로 구성하여 학습시키느냐에 따라 모델의 '능력'이 결정
    - Instruction Tuning : "질문-답변" 쌍을 학습시켜 모델이 사용자의 지시사항을 따르도록 만듦
    - Chain of Thought (CoT) Tuning : 단순 정답이 아니라 문제-<사고 과정>-정답의 흐름이 담긴 데이터를 학습 (Deespeek-R1)
    - Dialogue Tuning : 대화의 맥락(Context)을 유지하며 멀티턴 대화를 주고받는 데이터를 학습시켜 챗봇으로 성능 향상
    - Rejection Sampling : 모델이 생성한 여러 답변 중 보상 모델이 높게 평가한 것들만 골라 다시 SFT 데이터로 사용

Reinforcement Learning (RL)

- RL는 학습 알고리즘, 선호도 반영 방식, 추론 유도 기법에 따라 분류
  - 학습 알고리즘에 따른 연구 : 모델의 정책을 수학적으로 업데이트하는 핵심 방법론
    - Proximal Policy Optimization (PPO) : Actor-Critic 구조를 사용하는 전통적인 알고리즘, Critic 모델을 유지 → 연산 자원이 많이 소모
    - Group Relative Policy Optimization (GRPO) : DeepSeek에서 제안한 방식으로, 별도의 Critic 모델 없이 그룹 내 답변들 상대적 보상을 계산
    - Direct Preference Optimization (DPO) : 복잡한 보상 모델 학습 과정 없이, 사람이 선호하는 답변 쌍 데이터를 직접 손실 함수로 반영
  - 보상 및 선호도 반영 방식에 따른 분류
    - Reinforcement Learning from Human Feedback (RLHF) : 사람의 직접 답변의 우선순위를 매긴 데이터를 바탕으로 보상 모델 학습
    - Reinforcement Learning from AI Feedback (RLAIF) : 사람 대신 더 뛰어난 성능을 가진 AI 모델이 보상을 결정하도록 하여 데이터 확장성↑
    - Rule-based Reward : 수학 문제의 정답 여부 처럼 명확한 정답이 존재하는 경우, 수식이나 실행 결과로 보상
  - 추론 유도 및 사고 전략에 따른 연구 : 학습된 모델이 실제로 답을 낼 때 논리적인 흐름을 타도록 만드는 기법, 프롬프트 기술이지만 RL의 학습 타겟이 되어 모델에 내재화
    - Chain of Thought (CoT) : “단계별로 생각해보자”와 같이 중간 추론 과정을 생성하도록 유도하여 복잡한 문제 해결 능력을 높임
    - Tree of Thoughts (ToT) : 여러 추론 경로를 나무 가지처럼 생성하고, 각 경로의 유망함을 평가하며 최적의 해답을 탐색
    - Logical/Line of Thoughts (LoT) : 논리적 일관성과 흐름을 강조하여 결론에 도달하도록 유도하는 전략

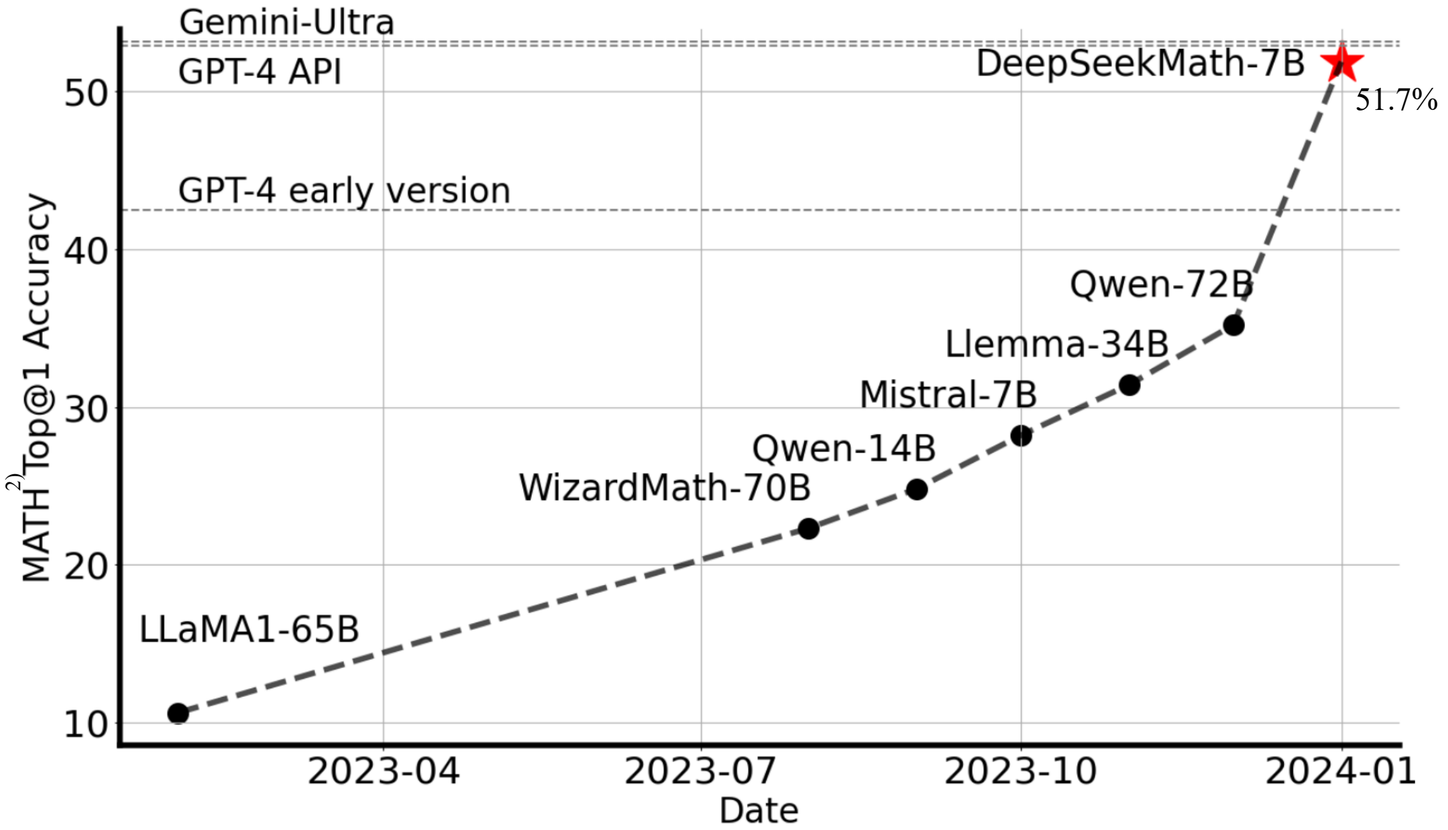
DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Model [arXiv 2024]

1) Z. Shao, P.Wang, Q.Zhu, R.Xu, J.Song, X.bi, H.Zhang, M.Zhang, Y.K. Li, Y.Wu, D.Guo, "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Model," arXiv preprint arXiv:2402.03300, 2024

2) MATH benchmark : 대수, 정수론, 조합론, 해석/미적분, 기하등 전통적 수학 대회 문제 전반을 포함, 고교 상위 ~ 대학 초급, 심화 문제를 포함하여 다단계 논증을 요구

DeepSeekMath 7B[1)]

- DeepSeekMath 7B 모델 구성
  - DeepSeek-Coder-Base-v1.5 7B 모델을 바탕으로, Common Crawl에서 수집한 1,200억(120B) 개의 수학 관련 토큰과 자연어 및 코드 데이터를 활용하여 추가 사전 학습을 진행



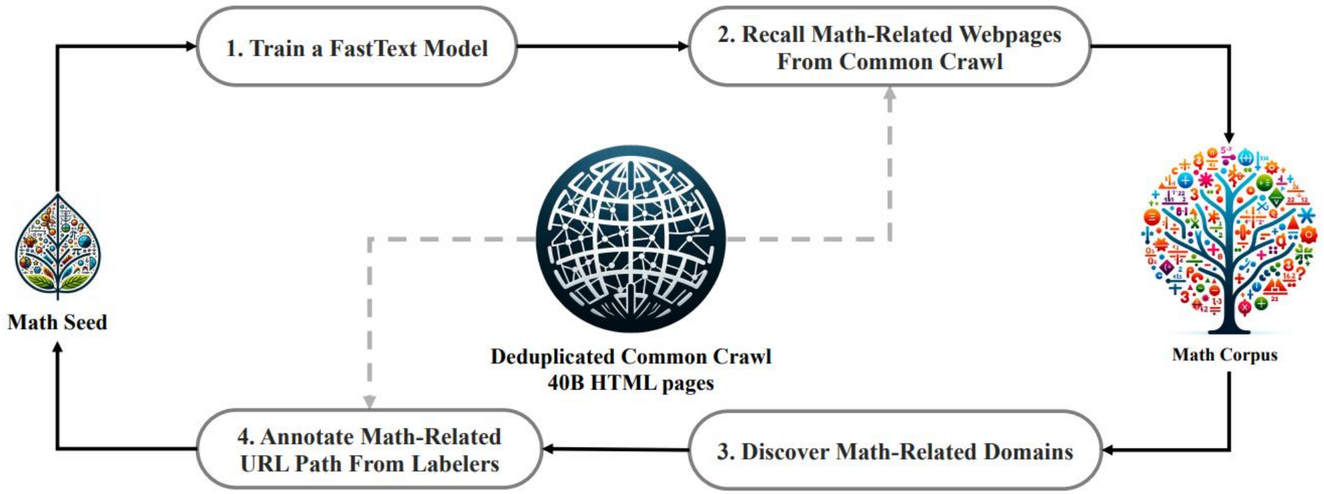
◆ 외부 toolkit이나 투표 기법에 의존하지 않고도 경진 대회 수준의 MATH 벤치마크에서 51.7%라는 인상적인 점수를 기록

1) fastText : 단어·n-gram 임베딩을 평균한 뒤 선형 분류기를 사용하는 매우 빠르고 경량의 supervised 분류기
2) 영어 수학 벤치마크 : GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021)
3) 중국어 벤치마크 : CMATH (Wei et al., 2023), AGIEval (Zhong et al., 2023)

DeepSeekMath 7B

- DeepSeekMath-Base 7B 모델 Data 수집
  - Seed corpus로 OpenWebMath를 사용, Common Crawl에서 수학 관련 웹페이를 탐색 (fastText[1] 분류기로 훈련)
    - Seed corpus에서 random하게 500,000개의 데이터 선택 (Positive = 수학), Common Crawl에서 500,000개의 데이터 선택 (Negative ≠ 수학) → fastText 분류 진행 → 40B HTML 웹페이지 확보 → 수집되지 않은 수학 데이터를 수동으로 주석 처리하여 Positive data 확보 → fastText 분류 → … → … (반복)
      - 4번의 데이터 수집 반복 끝에 120B 토큰에 해당하는 35.5M개의 수학 웹페이지 확보 (OpenWebMath 크기의 9배)
      - 벤치마크 오염을 방지하기 위해 영어[2]/중국어[3] 수학 벤치마크의 질문이나 답변을 포함하는 웹 페이지를 필터링
      - 500B 토큰 : DeepSeekMath Corpus (120B) 56% + AlgebraicStack 4% + arXiv 10% + Github code 20% + natural language data from Common Crawl (English and Chinese) 10%

Data Preparation



- 훈련 Library : fastText
- 벡터 차원 : 256
- 학습률 : 0.1
- n-gram 최대길이 : 3
- 단어 최소 출현 횟수 : 3
- 훈련 에포크 : 3

◆ 수학관련 웹페이지를 수집하는 반복적인 파이프 라인

1) 영어 수학 벤치마크 : GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021)
2) 중국어 벤치마크 : CMATH (Wei et al., 2023), AGIEval (Zhong et al., 2023)

DeepSeekMath 7B

- DeepSeekMath-Base 7B 모델 Data 수집
  - 다른 corpus와 성능 비교 시 DeepSeekMath Corpus가 가장 우수한 성능 확인 (DeepSeek-LLM 1.3B, few-shot CoT prompting 기준)

Math Corpus	Size	English Benchmarks					Chinese Benchmarks		
		GSM8K	MATH	OCW	SAT	MMLU STEM	CMATH	Gaokao MathCloze	Gaokao MathQA
No Math Training	N/A	2.9%	3.0%	2.9%	15.6%	19.5%	12.3%	0.8%	17.9%
MathPile	8.9B	2.7%	3.3%	2.2%	12.5%	15.7%	1.2%	0.0%	2.8%
OpenWebMath	13.6B	11.5%	8.9%	3.7%	31.3%	29.6%	16.8%	0.0%	14.2%
Proof-Pile-2	51.9B	14.3%	11.2%	3.7%	43.8%	29.2%	19.9%	5.1%	11.7%
DeepSeekMath Corpus	**120.2B**	**23.8%**	**13.6%**	**4.8%**	**56.3%**	**33.1%**	**41.5%**	**5.9%**	**23.6%**

◆ 서로 다른 corpus에서 학습된 DeepSeek-LLM 1.3B의 성능 (few-shot chain-of-Thought prompting 사용)

GSM8K [Cobbe et al., 2021]: 초등학생 수준의 수학 문제 해결 능력을 평가
MATH [Hendrycks et al., 2021]: 경쟁 시험 수준의 어려운 수학 문제 해결 능력을 평가
OCW (Open CourseWare) [Lewkowycz et al., 2022a]: 대학 수준의 수학 및 과학 과정을 기반으로 한 문제
SAT [Azerbayev et al., 2023]: 미국 대학수학능력시험(SAT)의 수학 섹션 문제
MMLU-STEM [Hendrycks et al., 2020]: 과학, 기술, 공학, 수학(STEM) 분야에 걸친 광범위한 다지선다형 질문을 포함
Chinese Benchmarks: 중국어 수학 능력을 평가하는 벤치마크
CMATH [Wei et al., 2023]: 중국 초등학생 수준의 수학 시험을 기반 벤치마크
Gaokao MathCloze, Gaokao MathQA [Zhong et al., 2023]: 중국 대학 입학 시험 수학 문제로, 각각 빈칸 채우기 및 질문 답변 형식

DeepSeekMath 7B

- SFT Data Curation
  - CoT, PoT, TIR 형식으로 주석이 달린 776K개의 훈련 예제로 구성 (영어,중국어 수학 문제 포함)
    - Chain-of-Thought (CoT) : 모델이 최종 답변에 도달하기까지의 중간 추론 단계를 텍스트로 자세히 설명하도록 하는 것
      - "이 문제는 먼저 A를 계산하고, 그 다음 B를 사용하여 C를 구한다"와 같이 단계별 사고 과정 표시
    - Program-of-Thoght (PoT) : 모델이 문제를 해결하기 위해 파이썬 코드와 같은 프로그램을 작성하고 실행하는 것을 포함
      - 복잡한 계산이나 논리적 절차가 필요한 문제에 유용
    - Tool-Integrated Reasoning(TIR) : 모델이 문제 해결 과정에서 외부 도구(예: 계산기, 연산 라이브러리)를 호출하여 사용
      - PoT와 유사하지만, 특정 도구 사용에 더 중점

SFT

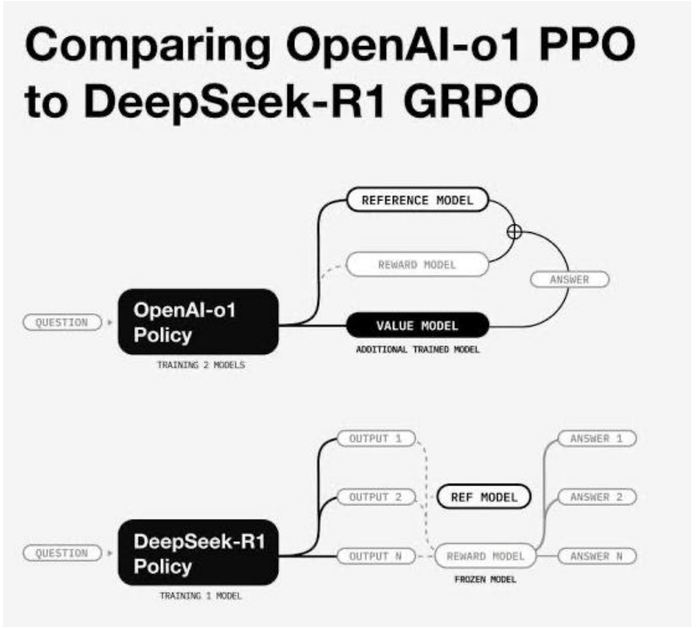
구분	CoT	PoT	TIR
Prompt	lambda q: f"""Let's solve this step by step. Question: {q} Solution: Step 1:"""	lambda q: f"""Solve this problem by writing and executing Python code. Question: {q} Write a Python program to solve this step by step: ```python # Your code here ``` Execute the code and provide the result:"""	lambda q: f"""Solve this problem by integrating logical reasoning with external tools. Question: {q} Plan: 1. Analyze the question and identify which steps require external tools. 2. Execute tool calls and interpret the results. 3. Synthesize the final answer based on the tool outputs and logical deduction. Solution: Step 1: [Identify the first sub-problem] Tool Call: [Tool name and parameters] Result: [Output from the tool] Step 2: [Reasoning based on the result...]"""

DeepSeekMath 7B

- DeepSeekMath-Base 7B : GRPO 기법 적용
  - Group Relative Policy Optimization (GRPO) : PPO의 변형으로, critic model을 사용하지 않고 그룹 점수에서 baseline을 추정하여 훈련 자원을 크게 감소
    - Proximal Policy Optimization (PPO) 는 actor-critic 알고리즘으로, policy model (π_θ)와 value function (V_φ)를 함께 훈련하여 어드밴티지 A_t 를 계산
      - Policy Model 외에 Value Model이라는 추가적인 모델을 학습해야 하므로, 메모리 및 계산 자원 부담이 큼
    - GRPO는 이 문제를 해결하기 위해, 각 q (질문)에 대해 이전 정책 $\pi_{\theta old}$에서 출력 그룹 $\{o_1, o_2, \ldots, o_G\}$를 샘플링하고, 이 그룹의 평균 보상을 기준선으로 사용
      - Value Model 없이도 효과적인 강화 학습을 가능

RL

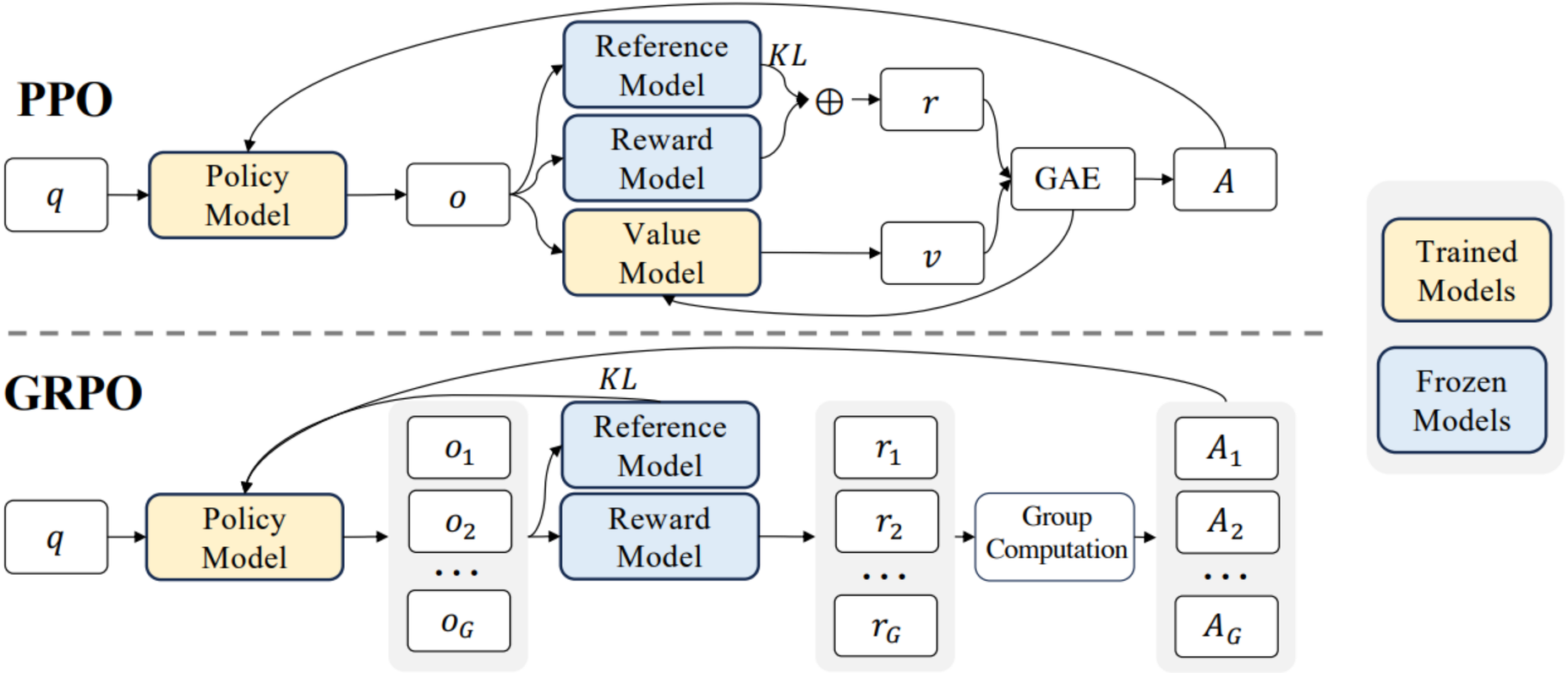
Feature	OpenAI o1 (PPO)	DeepSeek R1 (GRPO)
Models Trained	2 (policy + critic)	1 (policy only)
Training Method	Compares responses one by one	Ranks multiple responses at once
Computational Cost	High (training two models)	Low (training only one model)
Training Speed	Slower	Faster
Self-Verification	Weak	Strong (better ranking method)



◆ Source : https://www.appypieautomate.ai/blog/comparison/openai-o1-ppo-vs-deepseek-r1-grpo

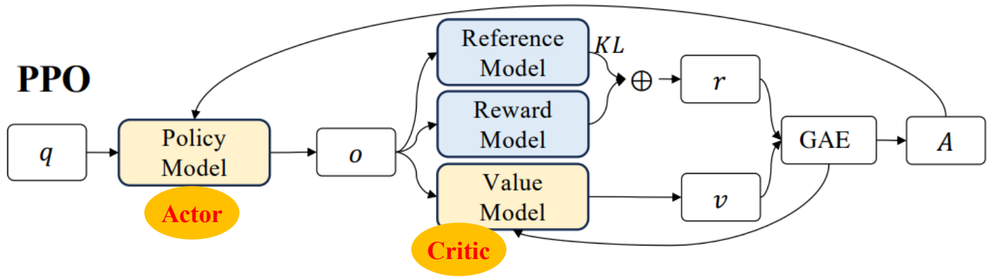
DeepSeekMath 7B

- PPO vs GRPO
  - PPO는 value model(critic)을 학습하고 토큰별 advantage를 GAE로 계산하는 반면, GRPO는 여러 후보 출력(오프라인/온라인으로 샘플된 o1...oG)을 그룹 단위로 평가하여 value 모델을 없애고 그룹 내 상대 보상으로 advantage를 만듦



DeepSeekMath 7B

- PPO[1)]
  - PPO는 정책 업데이트의 안정성을 유지하면서도 효율적인 학습을 가능하게 하는 것



$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|} \sum_{t=1}^{|o|} \min\left[\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})} A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})}, 1-\varepsilon, 1+\varepsilon\right) A_t\right]$$

PPO 목적 함수 (최대화)

clip : 상한/하한 제한 (policy update가 급진적으로 이루어져 학습이 불안정해지는 것을 방지)

- q : 입력 질문
- Policy Model : 현재 학습 중인 모델로, 입력 q에 대한 응답 o를 생성 (학습 모델)
- o : Policy Model이 생성한 응답
- Reference Model: Policy Model의 이전 버전이거나 특정 기준이 되는 모델 (고정)
  Policy Model이 Reference Model에서 너무 멀리 벗어나지 않도록 KL 발산 페널티를 계산하는 데 사용
- Reward Model : o의 품질을 평가하여 보상 r을 생성 (외부적으로 학습되거나 미리 정의된 규칙에 기반할 수 있음)
- Value Model: 현재 상태(또는 응답 o)의 가치를 예측하는 모델 (가치 v를 출력, 학습 모델)
  - KL 발산 (KL): Policy Model과 Reference Model 간의 KL 발산이 계산
    Policy Model의 업데이트가 너무 급격하게 이루어져 불안정해지는 것을 방지)
- r : Reward Model에서 나온 보상 r에 KL 발산 페널티가 더해져 최종 보상 신호가 됨

$$r_t = r_\varphi(q, o_{\leq t}) - \beta \log \frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{ref}(o_t|q,o_{<t})}$$

KL 발산 페널티

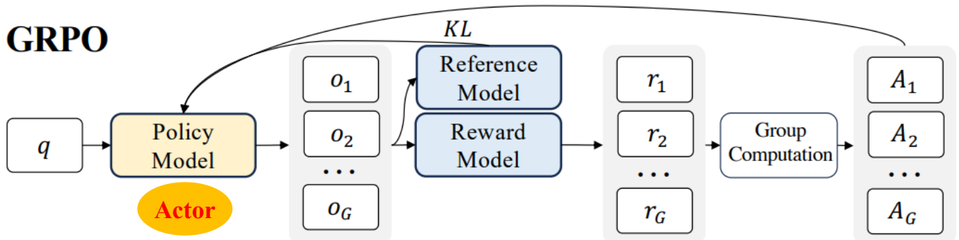
- GAE(Generalized Advantage Estimation) : GAE는 최종 보상 r과 Value Model의 가치 v를 이용하여 어드밴티지 A를 계산
  어드밴티지는 특정 행동이 평균보다 얼마나 더 좋은지를 나타내는 지표
- 어드밴티지 (A): 계산된 어드밴티지 A는 Policy Model을 업데이트하는 데 사용 $A_t = Q(s_t, a_t) - V(s_t)$
  Policy Model은 이 A를 최대화하는 방향으로 학습

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

DeepSeekMath 7B

- GRPO
  - Value Model이 필요 없으므로 PPO 대비 메모리 사용량과 계산 복잡성을 크게 줄일 수 있음



GPPO 목적 함수(최대화)

$$\mathcal{J}_{GRPO}(\theta) = \mathbb{E}[q \sim P(Q), \{o_i\}_{i=1}^{G} \sim \pi_{\theta_{old}}(O|q)]$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}\hat{A}_{i,t}, \text{clip}\left(\frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{old}}(o_{i,t}|q,o_{i,<t})}, 1-\varepsilon, 1+\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right]\right\}$$

clip : 상한/하한 제한 (policy update가 급진적으로 이루어져 학습이 불안정해지는 것을 방지)

KL 발산

특징	PPO	GRPO
Value Model	존재(학습됨), 가치 v를 출력	존재하지 않음
어드밴티지 계산	개별 응답의 보상 r과 가치 v를 GAE로 통합	동일 질문에 대한 여러 응답의 그룹 보상 r_G에서서 추정
KL 페널티 처리	보상 r에 직접 더해져 최종 보상 신호에 포함	손실 함수에 별도의 정규화 항으로 직접 추가됨
메모리/자원 효율성	추가 Value Model로 인해 높음	Value Model이 없어 효율적

- q : 입력 질문
- Policy Model : 현재 학습 중인 모델로, 입력 q에 대한 응답 o를 생성 (학습 모델)
- o : Policy Model이 생성한 응답 (PPO는 한 개의 o(응답) 생성, GRPO는 여러 개의 응답 (o_1, o_2…o_G)을 샘플링)
- Reference Model: Policy Model의 이전 버전이거나 특정 기준이 되는 모델 (고정)
  Policy Model이 Reference Model에서 너무 멀리 벗어나지 않도록 KL 발산 페널티를 계산하는 데 사용
- Reward Model : o의 품질을 평가하여 보상 r을 생성 (외부적으로 학습되거나 미리 정의된 규칙에 기반할 수 있음)
- Value Model: X
- Group Computation : PPO와 달리 Value Model이 없고, 동일한 질문 q에 대해 생성된 여러 응답들의 보상(r_1, ..r_G)을 한데 모아 그룹 연산을 수행 (이 연산을 통해 각 응답에 대한 상대적인 어드밴티지 (A_1,….,A_G)를 추정
  *각 응답의 보상 r_i에서 그룹 전체 보상의 평균을 빼는 방식으로 상대적인 우위 계산 가능
  정규화된 보상을 어드벤티지로 사용 ===> $\hat{A}_{i,t} = \tilde{r}_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$
- KL 발산 (KL): 보상에 페널티로 직접 더해지는 대신, Policy Model을 업데이트하는 손실 함수에 직접 정규화 항으로 추가

$$\mathbb{D}_{KL}\left[\pi_\theta||\pi_{ref}\right] = \frac{\pi_{ref}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - \log\frac{\pi_{ref}(o_{i,t}|q,o_{i,<t})}{\pi_\theta(o_{i,t}|q,o_{i,<t})} - 1 \quad \text{* DKL guaranteed to be positive}$$

- 어드밴티지 (A_1,..,A_G) : 그룹 연산을 통해 계산된 어드밴티지 (A_1,..,A_G) 는 Policy Model을 업데이트 하는 데 사용

DeepSeekMath 7B

- DeepSeekMath-Base 7B 모델 검증 결과
  - DeepSeekMath-RL 7B는 CoT 추론을 활용하여 GSM8K와 MATH에서 각각 88.2%와 51.7%의 정확도 달성

Model	Size	English Benchmarks		Chinese Benchmarks	
		GSM8K	MATH	MGSM-zh	CMATH
Chain-of-Thought Reasoning					
Closed-Source Model					
Gemini Ultra	-	94.4%	53.2%	-	-
GPT-4	-	92.0%	52.9%	-	86.0%
Inflection-2	-	81.4%	34.8%	-	-
GPT-3.5	-	80.8%	34.1%	-	73.8%
Gemini Pro	-	86.5%	32.6%	-	-
Grok-1	-	62.9%	23.9%	-	-
Baichuan-3	-	88.2%	49.2%	-	-
GLM-4	-	87.6%	47.9%	-	-
Open-Source Model					
InternLM2-Math	20B	82.6%	37.7%	-	-
Qwen	72B	78.9%	35.2%	-	-
Math-Shepherd-Mistral	7B	84.1%	33.0%	-	-
WizardMath-v1.1	7B	83.2%	33.0%	-	-
DeepSeek-LLM-Chat	67B	84.1%	32.6%	74.0%	80.3%
MetaMath	70B	82.3%	26.6%	66.4%	70.9%
SeaLLM-v2	7B	78.2%	27.5%	64.8%	-
ChatGLM3	6B	72.3%	25.7%	-	-
WizardMath-v1.0	70B	81.6%	22.7%	64.8%	65.4%
DeepSeekMath-Instruct	7B	82.9%	46.8%	73.2%	84.6%
DeepSeekMath-RL	7B	**88.2%**	**51.7%**	**79.6%**	**88.8%**

Tool-Integrated Reasoning					
Closed-Source Model					
GPT-4 Code Interpreter	-	97.0%	69.7%	-	-
Open-Source Model					
InternLM2-Math	20B	80.7%	54.3%	-	-
DeepSeek-LLM-Chat	67B	86.7%	51.1%	76.4%	85.4%
ToRA	34B	80.7%	50.8%	41.2%	53.4%
MAmmoTH	70B	76.9%	41.8%	-	-
DeepSeekMath-Instruct	7B	83.7%	57.4%	72.0%	84.3%
DeepSeekMath-RL	7B	**86.7%**	**58.8%**	**78.4%**	**87.6%**

❖Closed-Source Mode : 가중치 또는 핵심 코드가 공개되지 않으며, 보통 API 형태로만 접근 가능하거나 사용에 제약이 있는 모델
개발사·운영사가 통제권을 갖고 업데이트·정책·안전장치(콘텐츠 필터 등)를 운영

❖Open-Source Model : 모델의 코드와 학습된 가중치(weights)를 공개하고, 사용·수정·배포를 허용하는 라이선스를 제공하는 모델
투명성(데이터/아키텍처/훈련 절차)을 통해 재현성, 감사(audit), 커스터마이징이 가능

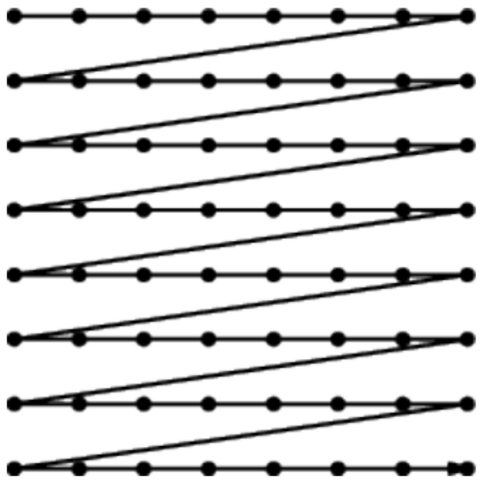
DeepSeek-OCR 2: Visual Causal Flow [arXiv 2026]

1) H.Wei, Y.Sun, Y.Li, “DeepSeek-OCR 2: Visual Causal Flow,” arXiv preprint arXiv:2601.20552, 2026

2) OCR(Optical Character Recognition, 광학 문자 판독) : 이미지(사진, 스캔 문서) 속의 텍스트를 기계가 읽을 수 있는 디지털 데이터로 변환하는 기술

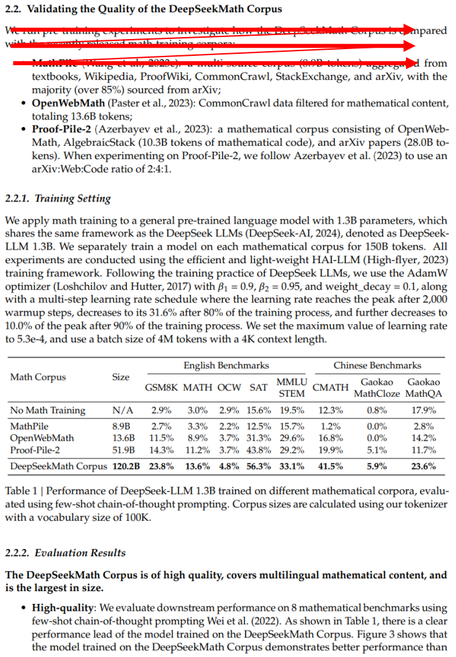
DeepSeek-OCR 2[1)]

- DeepEncoder V2 제안 : 이미지 의미에 따라 시각 토큰을 동적으로 재정렬하는 새로운 인코더
  - 시각 토큰을 의미 기반으로 재정렬하는 DeepEncoder V2를 도입해 문서 OCR[2)] 및 읽기 순서 성능 향상
    - 기존 Vision-Language Model (VLM) 인코더는 이미지 패치를 고정된 raster-scan 순서로 평탄화하여 LLM에 입력
    - 인간의 시각은 의미 기반의 유동적 스캔(인과적 흐름)을 따름 → 복잡한 레이아웃에서 읽기 순서/논리 손실 발생
      - 2D 이미지를 두 단계의 1D 인과적(reasoning) 흐름으로 처리하여 더 인간 유사한 읽기 순서와 고품질 OCR 달성



◆ Raster-Scan Odering

위에서부터
왼쪽 → 오른쪽



2.2. Validating the Quality of the DeepSeekMath Corpus

We run pre-training experiments to investigate how the DeepSeekMath Corpus is compared with the recently released math-training corpora:

- **MathPile** (Wang et al., 2023c): a multi-source corpus (8.9B tokens) aggregated from textbooks, Wikipedia, ProofWiki, CommonCrawl, StackExchange, and arXiv, with the majority (over 85%) sourced from arXiv;
- **OpenWebMath** (Paster et al., 2023): CommonCrawl data filtered for mathematical content, totaling 13.6B tokens;
- **Proof-Pile-2** (Azerbayev et al., 2023): a mathematical corpus consisting of OpenWebMath, AlgebraicStack (10.3B tokens of mathematical code), and arXiv papers (28.0B tokens). When experimenting on Proof-Pile-2, we follow Azerbayev et al. (2023) to use an arXiv:Web:Code ratio of 2:4:1.

2.2.1. Training Setting

We apply math training to a general pre-trained language model with 1.3B parameters, which shares the same framework as the DeepSeek LLMs (DeepSeek-AI, 2024), denoted as DeepSeek-LLM 1.3B. We separately train a model on each mathematical corpus for 150B tokens. All experiments are conducted using the efficient and light-weight HAI-LLM (High-flyer, 2023) training framework. Following the training practice of DeepSeek LLMs, we use the AdamW optimizer (Loshchilov and Hutter, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight_decay = 0.1, along with a multi-step learning rate schedule where the learning rate reaches the peak after 2,000 warmup steps, decreases to its 31.6% after 80% of the training process, and further decreases to 10.0% of the peak after 90% of the training process. We set the maximum value of learning rate to 5.3e-4, and use a batch size of 4M tokens with a 4K context length.

Math Corpus	Size	English Benchmarks					Chinese Benchmarks		
		GSM8K	MATH	OCW	SAT	MMLU STEM	CMATH	Gaokao MathCloze	Gaokao MathQA
No Math Training	N/A	2.9%	3.0%	2.9%	15.6%	19.5%	12.3%	0.8%	17.9%
MathPile	8.9B	2.7%	3.3%	2.2%	12.5%	15.7%	1.2%	0.0%	2.8%
OpenWebMath	13.6B	11.5%	8.9%	3.7%	31.3%	29.6%	16.8%	0.0%	14.2%
Proof-Pile-2	51.9B	14.3%	11.2%	3.7%	43.8%	29.2%	19.9%	5.1%	11.7%
DeepSeekMath Corpus	**120.2B**	**23.8%**	**13.6%**	**4.8%**	**56.3%**	**33.1%**	**41.5%**	**5.9%**	**23.6%**

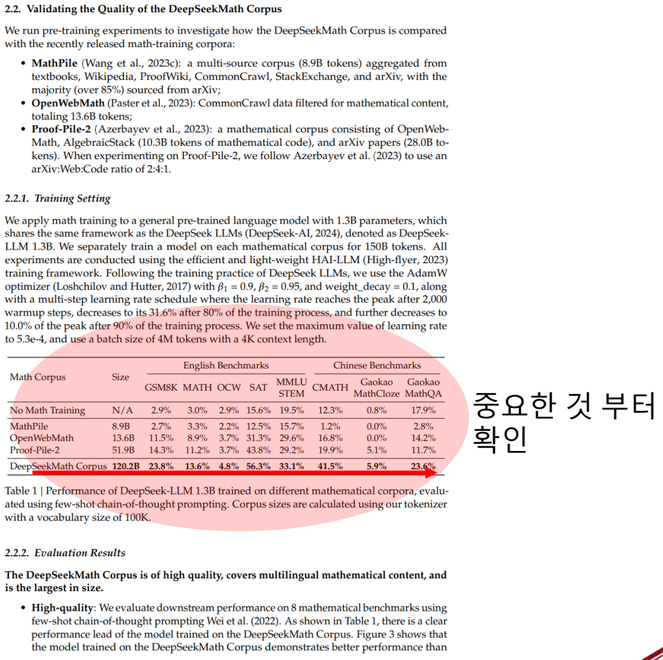
Table 1 | Performance of DeepSeek-LLM 1.3B trained on different mathematical corpora, evaluated using few-shot chain-of-thought prompting. Corpus sizes are calculated using our tokenizer with a vocabulary size of 100K.

2.2.2. Evaluation Results

The DeepSeekMath Corpus is of high quality, covers multilingual mathematical content, and is the largest in size.

- **High-quality**: We evaluate downstream performance on 8 mathematical benchmarks using few-shot chain-of-thought prompting Wei et al. (2022). As shown in Table 1, there is a clear performance lead of the model trained on the DeepSeekMath Corpus. Figure 3 shows that the model trained on the DeepSeekMath Corpus demonstrates better performance than

◆ Raster-Scan



2.2. Validating the Quality of the DeepSeekMath Corpus

We run pre-training experiments to investigate how the DeepSeekMath Corpus is compared with the recently released math-training corpora:

- **MathPile** (Wang et al., 2023c): a multi-source corpus (8.9B tokens) aggregated from textbooks, Wikipedia, ProofWiki, CommonCrawl, StackExchange, and arXiv, with the majority (over 85%) sourced from arXiv;
- **OpenWebMath** (Paster et al., 2023): CommonCrawl data filtered for mathematical content, totaling 13.6B tokens;
- **Proof-Pile-2** (Azerbayev et al., 2023): a mathematical corpus consisting of OpenWebMath, AlgebraicStack (10.3B tokens of mathematical code), and arXiv papers (28.0B tokens). When experimenting on Proof-Pile-2, we follow Azerbayev et al. (2023) to use an arXiv:Web:Code ratio of 2:4:1.

2.2.1. Training Setting

We apply math training to a general pre-trained language model with 1.3B parameters, which shares the same framework as the DeepSeek LLMs (DeepSeek-AI, 2024), denoted as DeepSeek-LLM 1.3B. We separately train a model on each mathematical corpus for 150B tokens. All experiments are conducted using the efficient and light-weight HAI-LLM (High-flyer, 2023) training framework. Following the training practice of DeepSeek LLMs, we use the AdamW optimizer (Loshchilov and Hutter, 2017) with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight_decay = 0.1, along with a multi-step learning rate schedule where the learning rate reaches the peak after 2,000 warmup steps, decreases to its 31.6% after 80% of the training process, and further decreases to 10.0% of the peak after 90% of the training process. We set the maximum value of learning rate to 5.3e-4, and use a batch size of 4M tokens with a 4K context length.

Math Corpus	Size	English Benchmarks					Chinese Benchmarks		
		GSM8K	MATH	OCW	SAT	MMLU STEM	CMATH	Gaokao MathCloze	Gaokao MathQA
No Math Training	N/A	2.9%	3.0%	2.9%	15.6%	19.5%	12.3%	0.8%	17.9%
MathPile	8.9B	2.7%	3.3%	2.2%	12.5%	15.7%	1.2%	0.0%	2.8%
OpenWebMath	13.6B	11.5%	8.9%	3.7%	31.3%	29.6%	16.8%	0.0%	14.2%
Proof-Pile-2	51.9B	14.3%	11.2%	3.7%	43.8%	29.2%	19.9%	5.1%	11.7%
DeepSeekMath Corpus	**120.2B**	**23.8%**	**13.6%**	**4.8%**	**56.3%**	**33.1%**	**41.5%**	**5.9%**	**23.6%**

Table 1 | Performance of DeepSeek-LLM 1.3B trained on different mathematical corpora, evaluated using few-shot chain-of-thought prompting. Corpus sizes are calculated using our tokenizer with a vocabulary size of 100K.

2.2.2. Evaluation Results

The DeepSeekMath Corpus is of high quality, covers multilingual mathematical content, and is the largest in size.

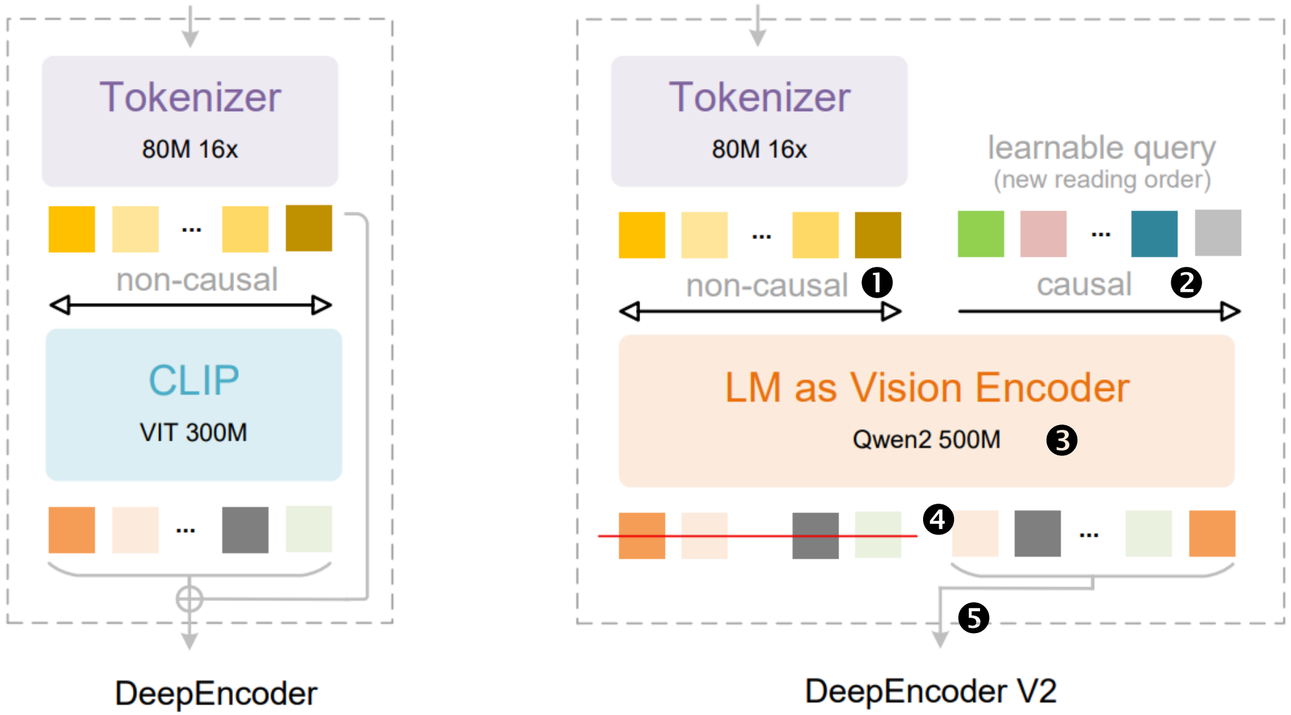
- **High-quality**: We evaluate downstream performance on 8 mathematical benchmarks using few-shot chain-of-thought prompting Wei et al. (2022). As shown in Table 1, there is a clear performance lead of the model trained on the DeepSeekMath Corpus. Figure 3 shows that the model trained on the DeepSeekMath Corpus demonstrates better performance than

중요한 것 부터
확인

◆ Human view

DeepSeek-OCR 2

- DeepEncoder V2 제안
  - DeepEncoder V2는 CLIP을 LLM 스타일 아키텍처로 대체, 시각 토큰 뒤에 Learnable Query인 'Causal Flow Tokens' 추가
  - 시각 토큰은 양방향, 인과 토큰은 인과 주의를 사용하는 맞춤형 Attention Mask를 통해 시각 정보의 인과적 재정렬 수행



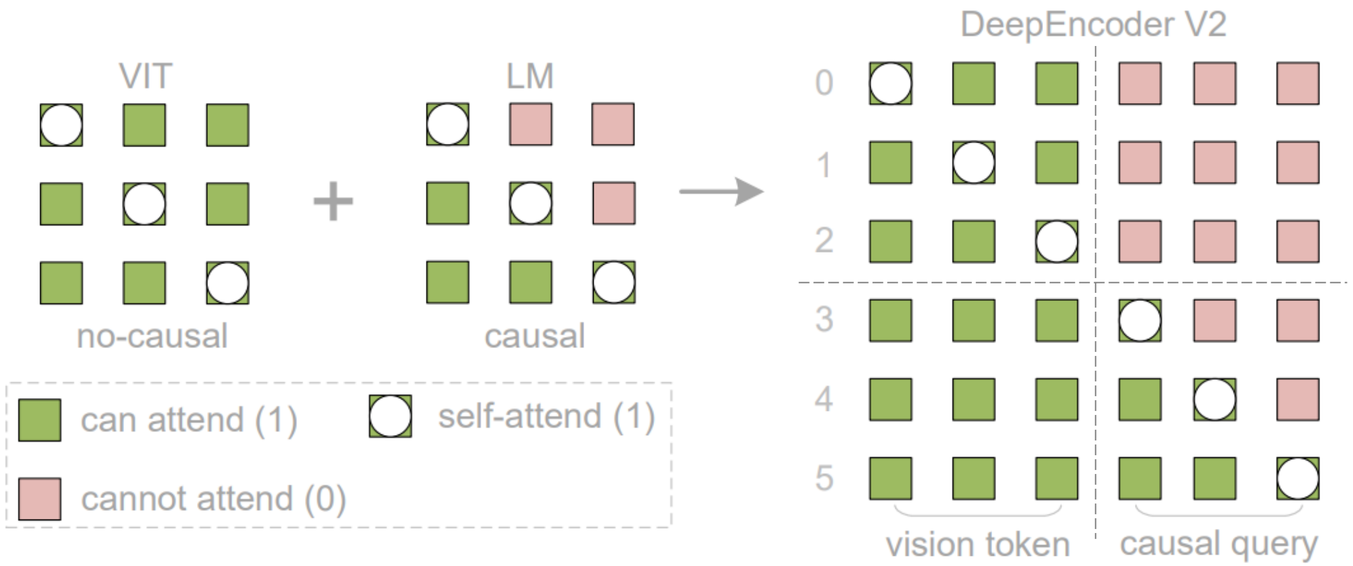
❍ DeepEncoder V2

- 동일한 Tokenizer(80M, 16×)로 시각 토큰을 먼저 생성 ❶
- 그 뒤에 learnable queries를 suffix로 추가 ❷
  - 각 쿼리는 "new reading order" 역할
- LM as Vision Encoder (Qwen2 500M) : LLM 스타일의 Transformer를 시각 토큰 + 쿼리 전체 시퀀스(prefix-concat)로 처리 ❸
  - 시각 토큰 부분에는 bidirectional attention(비인과)
  - 쿼리 부분에는 causal(삼각 마스크, 이전 쿼리만 볼 수 있음)
- 쿼리(후반부) 출력만 추출하여 LLM 디코더로 넘김 ❺
- 효과: 쿼리들이 시각 정보를 보고 "의미적 읽기 순서"로 토큰을 재배열 ❹

→ LLM 디코더의 autoregressive 추론과 자연스럽게 결합

DeepSeek-OCR 2

- DeepEncoder V2 제안
  - VIT 쪽(녹색)은 no-causal으로 시각 토큰들끼리 양방향(attend 가능)
  - LM 쪽(분홍)은 causal인 causal query들이 삼각형(autoregressive) 마스크를 가짐
  - DeepEncoder V2의 전체 마스크 시각화 : 행(쿼리 토큰)에서 열(키/값 토큰)로 향하는 "누가 누구를 볼 수 있는지"를 표시
    - 왼쪽 블록(vision token)은 서로 모두 볼 수 있고, 오른쪽 블록(causal query)은 모든 시각 토큰과 이전의(또는 자기 자신) causal query만 볼 수 있음



* Transformer Attention (기존)

$$\mathrm{Attn}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^{\top}}{\sqrt{d}}\right)V$$

* DeepEncoder V2 Attention(기존)

$$\mathrm{Attn}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^{\top}}{\sqrt{d}} + A\right)V, \quad A_{ij} = \begin{cases} 0 & \text{if } M_{ij} = 1, \\ -\infty & \text{if } M_{ij} = 0. \end{cases}$$

$$M = \begin{bmatrix} \mathbf{1}_{m\times m} & \mathbf{0}_{m\times n} \\ \mathbf{1}_{n\times m} & \mathrm{LowerTri}(n) \end{bmatrix}, \quad \text{where } n = m$$

m: 시각(visual) 토큰 수
n: causal query 토큰 수

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

왼쪽 상단 $\mathbf{1}_{m\times m}$: 시각 토큰들이 서로 모두 볼 수 있음.
왼쪽 하단 $\mathbf{1}_{n\times m}$: causal query들이 모든 시각 토큰을 볼 수 있음.
오른쪽 상단 $\mathbf{0}_{m\times n}$: 시각 토큰은 causal query들을 볼 수 없음(시각 토큰은 prefix로 고정된 상태).
오른쪽 하단 $\mathrm{LowerTri}(n)$: causal query들 사이에는 autoregressive(선행 토큰만 관찰) 구조.

[v1 v2 v3 q1 q2 q3], 마스크 M (1 허용, 0 차단)

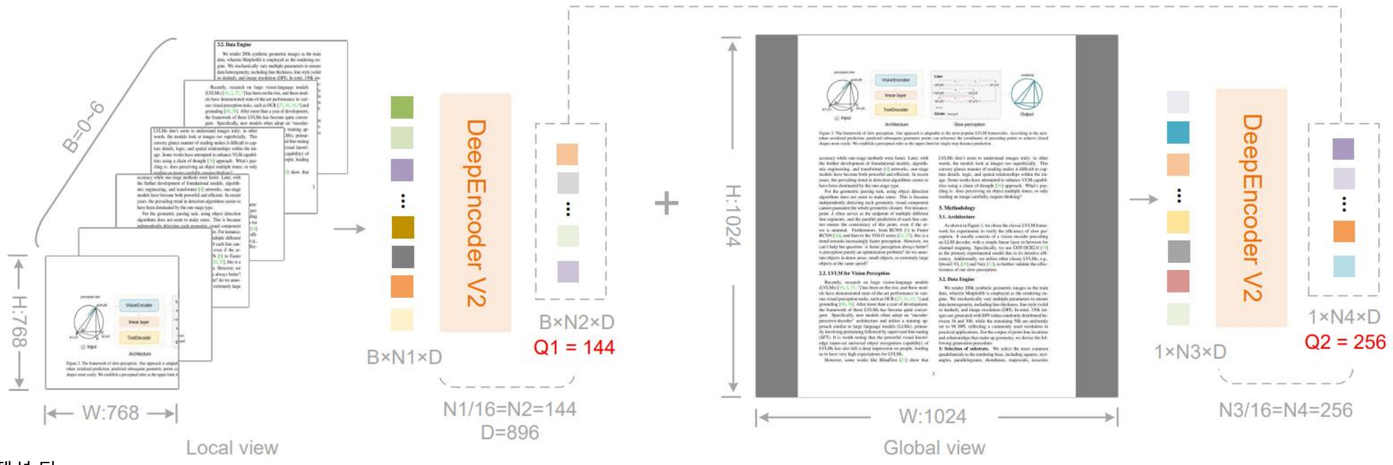
$$A = \begin{bmatrix} 0 & 0 & 0 & -1e9 & -1e9 & -1e9 \\ 0 & 0 & 0 & -1e9 & -1e9 & -1e9 \\ 0 & 0 & 0 & -1e9 & -1e9 & -1e9 \\ 0 & 0 & 0 & 0 & -1e9 & -1e9 \\ 0 & 0 & 0 & 0 & 0 & -1e9 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

DeepSeek-OCR 2

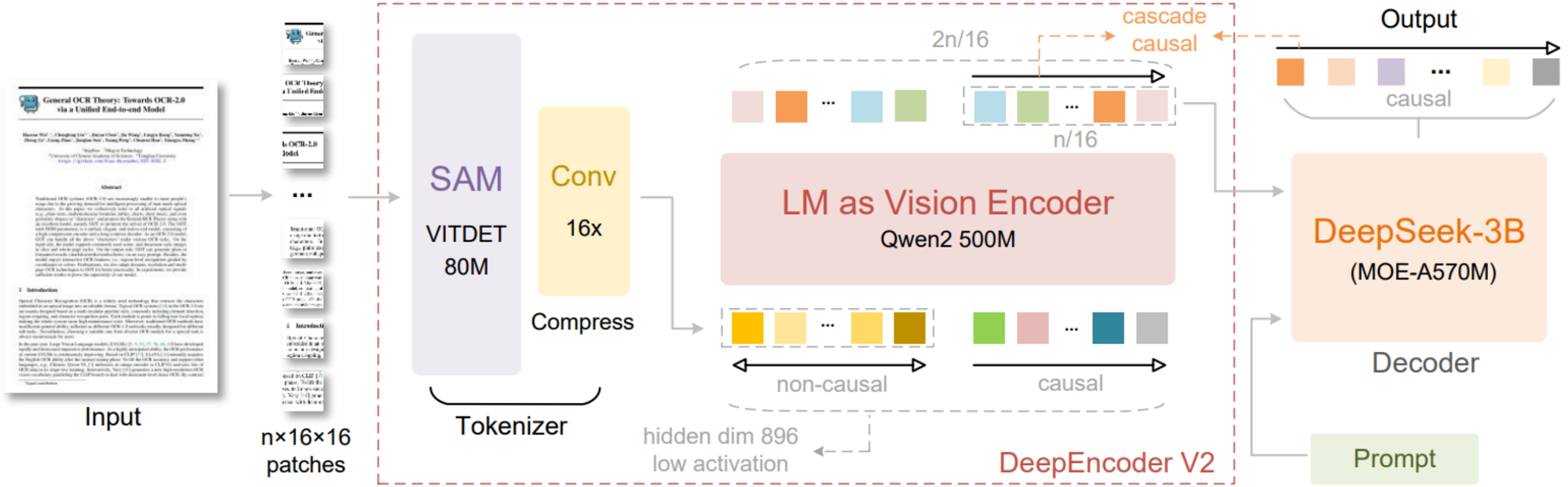
- DeepEncoder V2 제안
  - 시간 토큰의 순서를 동적으로 재정렬

$$\mathbf{O} = \mathcal{D}\left(\pi_Q\left(\mathcal{T}^L\left(\mathcal{E}(\mathbf{I}) \oplus \mathbf{Q}_0; \mathbf{M}\right)\right)\right)$$

- E(I): vision tokenizer가 만든 visual tokens (matrix)
- Q_0: 학습 가능한 causal query embeddings (행렬)
- ⊕: 시퀀스 연결 (visual tokens 앞에 붙이거나 뒤에 붙이는 구성에 따라 다름)
- $T^L(...; M)$: L-layer transformer with attention mask M
- π_Q: transformer 출력에서 query-portion(마지막 n 토큰)을 추출하는 연산
- D: decoder(LLM) — 여기로 쿼리 출력(벡터들)이 전달되어 최종 텍스트 logits O 생성
- 요점: Q_0와 그 출력은 벡터이며 LLM 입력으로 들어가기 전에 (필요 시) LLM embedding 차원으로 프로젝션 됨



$T = 256 + k \times 144, \quad k = 0\sim6$ (논문 제시 : $256 \sim 1120$)



1) OmniDocBench v1.5 : 중국어와 영어로 된 9가지 주요 범주(잡지, 학술 논문, 연구 보고서 등 포함)에 걸쳐 1,355개의 문서 페이지로 구성

DeepSeek-OCR 2

- DeepEncoder V2 제안
  - OmniDocBench v1.5[1]에서의 종합적인 문서 읽기 평가 결과

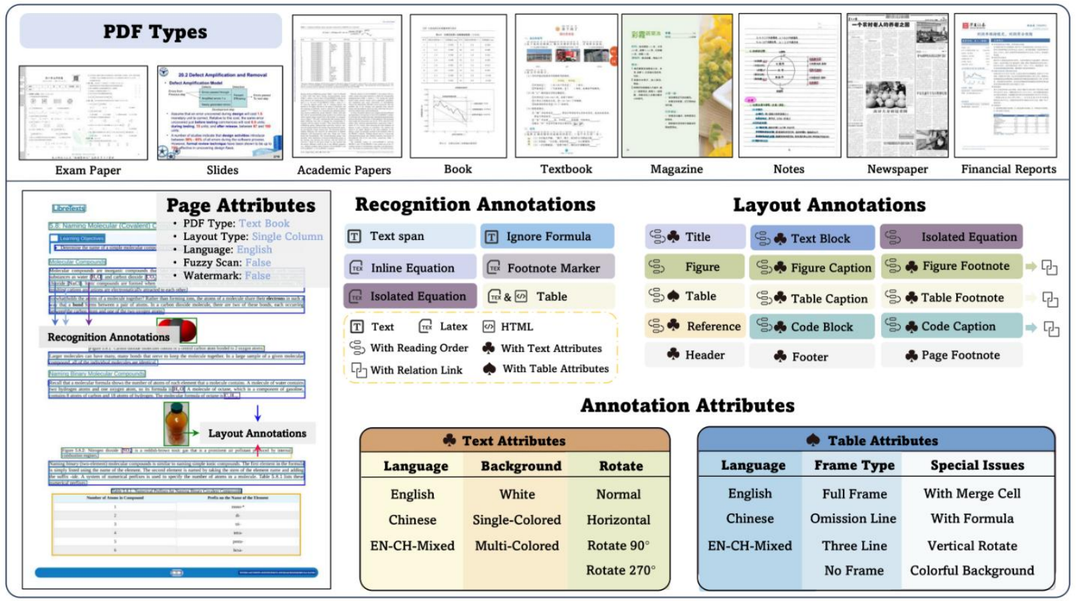
    -DeepSeek-OCR 2는 시각 토큰 예산은 오히려 줄이면서(1120 < 1156) 모든 주요 항목에서 실질적 성능 향상

Model	V-token^{max} ↓	Overall ↑	Text^{Edit} ↓	Formula^{CDM} ↑	Table^{TEDs} ↑	Table^{TEDS_s} ↑	R-order^{Edit} ↑
Pipline							
Marker-1.8.2 [1]	-	71.30	0.206	76.66	57.88	71.17	0.250
MinerU2-pp [45]	-	71.51	0.209	76.55	70.90	79.11	0.225
Dolphin [17]	-	74.67	0.125	67.85	68.70	77.77	0.124
Dolphin-1.5 [17]	-	83.21	0.092	80.78	78.06	84.10	0.080
PP-StructureV3 [13]	-	86.73	0.073	85.79	81.68	89.48	0.073
MonkeyOCR-pro-1.2B [23]	-	86.96	0.084	85.02	84.24	89.02	0.130
MonkeyOCR-3B [23]	-	87.13	0.075	87.45	81.39	85.92	0.129
MonkeyOCR-pro-3B [23]	-	88.85	0.075	87.25	86.78	90.63	0.128
MinerU2.5 [45]	-	90.67	0.047	88.46	88.22	92.38	0.044
PaddleOCR-VL [12]	-	92.86	0.035	91.22	90.89	94.76	0.043
End-to-end Model							
OCRFlux [4]	>6000	74.82	0.193	68.03	75.75	80.23	0.202
GPT-4o [33]	-	75.02	0.217	79.70	67.07	76.09	0.148
InternVL3 [55]	>7000	80.33	0.131	83.42	70.64	77.74	0.113
POINTS-Reader [31]	>6000	80.98	0.134	79.20	77.13	81.66	0.145
olmOCR [36]	>6000	81.79	0.096	86.04	68.92	74.77	0.121
InternVL3.5-241B [49]	>7000	82.67	0.142	87.23	75.00	81.28	0.125
MinerU2-VLM [45]	>7000	85.56	0.078	80.95	83.54	87.66	0.086
Nanonets-OCR-s [2]	>7000	85.59	0.093	85.90	80.14	85.57	0.108
Qwen2.5-VL-72B [9]	>6000	87.02	0.094	88.27	82.15	86.22	0.102
Gemini-2.5 Pro[6]	-	88.03	0.075	85.82	85.71	90.29	0.097
dots.ocr [39]	>6000	88.41	0.048	83.22	86.78	90.62	0.053
OCRVerse [3]	>6000	88.56	0.058	86.91	84.55	88.45	0.071
Qwen3-VL-235B [8]	>6000	89.15	0.069	88.14	86.21	90.55	0.068
DeepSeek-OCR (9-crops)	1156	87.36	0.073	84.14	85.25	89.01	0.085
DeepSeek-OCR 2	1120	91.09	0.048	90.31	87.75	92.06	0.057
	↓ 36	↑ 3.73	↓ 0.025	↑ 6.17	↑ 2.5	↑ 3.05	↓ 0.028

- V-token_max (↓) : 한 페이지당 허용되는 최대 visual token(시각 토큰) 수
- Overall (↑) : 전체 성능(%)
- Text^{Edit} (↓) : 텍스트 인식의 Edit Distance 지표로, 값이 작을수록(↓) 인식 오류가 적음
  - Edit Distance는 원문과 모델 출력 간의 문자 수준 삽입/삭제/치환 횟수의 정규화된 값
    - ED = 0이면 완벽 일치, ED가 0.1이면 평균적으로 10% 정도 편집(불일치)이 있음 (정규화 방법에 따라 절대값 해석은 달라질 수 있음).
- Formula^{CDM} (↑) : 수식 인식 관련 지표(문서의 표기대로 "CDM"로 표기됨)
- Table^{TEDs} / Table^{TEDSs} (↑) : 테이블 구조·내용 추출 관련 지표(두 종류의 테이블 평가 지표) ↑가 클수록 테이블 인식 성능이 좋음을 의미합니다.
- R-order^{Edit} (↓): 문서의 읽기 순서(reading order) 정렬 정확도를 편집거리로 측정한 항목 값이 작을수록(↓) 원래 문서의 논리적 읽기 순서(사람이 읽는 순서)에 가깝게 모델이 요소를 나열했음을 의미
  - R-order^{Edit}가 0.085 → 0.057로 감소했다면, 요소들의 순서가 더 정확해져 downstream(문서 변환 / 요약)에서 더 적은 순서 오류가 발생

감사합니다

Appendix. OmniDocBench v1.5



OmniDocBench v1.5 : 중국어와 영어로 된 9가지 주요 범주(잡지, 학술 논문, 연구 보고서 등 포함)에 걸쳐 1,355개의 문서 페이지로 구성
https://huggingface.co/datasets/opendatalab/OmniDocBench

Appendix. Edit Distance

- Edit Distance : 원문과 모델 출력 간의 문자 수준 삽입/삭제/치환 횟수의 정규화된 값 (정규화된 Levenshtein distance)

$$\mathrm{ED}(r,p) = \frac{\mathrm{Levenshtein}(r,p)}{|r|}$$

$$\mathrm{ED}_{\mathrm{dataset}} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{ED}(r_i, p_i)$$

r: reference(정답) 문자열 (또는 정답 시퀀스) 길이 |r|
p: predicted(모델 출력) 문자열
Levenshtein(r,p): r을 p로 바꾸는 데 필요한 최소 편집 연산(삽입·삭제·치환) 개수

*ED 값은 0에서 이상으로 나오며, 0에 가까울수록(↓) 정답과 유사. 논문에서 R-order ED가 0.085 → 0.057로 감소한 것은 읽기 순서 시퀀스의 평균 편집거리가 줄었다는 의미(순서 오류가 줄음).

Levenshtein

$$\mathrm{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \mathrm{lev}\left(\mathrm{tail}(a), \mathrm{tail}(b)\right) & \text{if } \mathrm{head}(a) = \mathrm{head}(b), \\ 1 + \min \begin{cases} \mathrm{lev}\left(\mathrm{tail}(a), b\right) \\ \mathrm{lev}\left(a, \mathrm{tail}(b)\right) \\ \mathrm{lev}\left(\mathrm{tail}(a), \mathrm{tail}(b)\right) \end{cases} & \text{otherwise} \end{cases}$$

예를 들어, "kitten"과 "sitting" 사이의 레벤슈타인 거리는 3이다. 다음의 3가지 편집을 통해 한 단어를 다른 단어로 바꿀 수 있으며, 3번보다 적은 편집으로는 불가능하기 때문이다.

1. **k**itten → **s**itten ("k"를 "s"로 교체)
2. sitt**e**n → sitt**i**n ("e"를 "i"로 교체)
3. sittin → sittin**g** (끝에 "g" 삽입)

삭제의 간단한 예는 "uninformed"와 "uniformed"에서 볼 수 있으며, 이들의 거리는 1이다.

1. uni**n**formed → uniformed ("n" 삭제)