Speculative Decoding & Multi-Token Prediction

Fast Inference from Transformers via Speculative Decoding

Yaniv Leviathan [*1] Matan Kalman [*1] Yossi Matias [1]

EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test

Yuhui Li[1,3], Fangyun Wei[2], Chao Zhang[1], Hongyang Zhang[3,4]
[1]Peking University [2]Microsoft Research
[3]University of Waterloo [4]Vector Institute
yuhui.li@stu.pku.edu.cn, fawe@microsoft.com
c.zhang@pku.edu.cn, hongyang.zhang@uwaterloo.ca

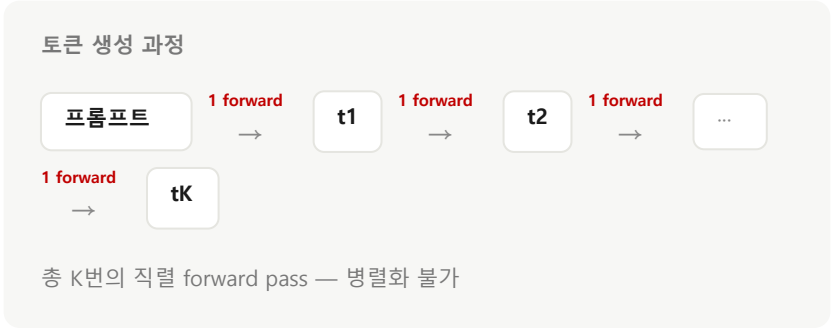
2026. 06. 04

Contents

LLM 추론은 왜 느린가?

Autoregressive decoding의 본질적 한계

- K개 토큰 생성 = K번의 순차(serial) forward pass
- 각 step마다 전체 파라미터를 HBM→캐시로 로드 → **memory-bound**
- GPU 연산력(FLOPs)은 미활용, 메모리 대역폭이 병목
- 모델이 클수록 토큰당 지연(latency) 증가



디코딩 속도의 병목: 연산량이 아니라 메모리 접근 횟수

Speculative Decoding vs Multi-Token Prediction

Speculative Decoding (SD)

- 별도의 **작은 draft model**이 여러 토큰을 먼저 제안
- 큰 target model이 **한 번의 병렬 forward로 검증**
- Rejection sampling으로 **출력 분포 무손실 보장**
- 대표: Leviathan et al. (ICML '23), EAGLE 계열

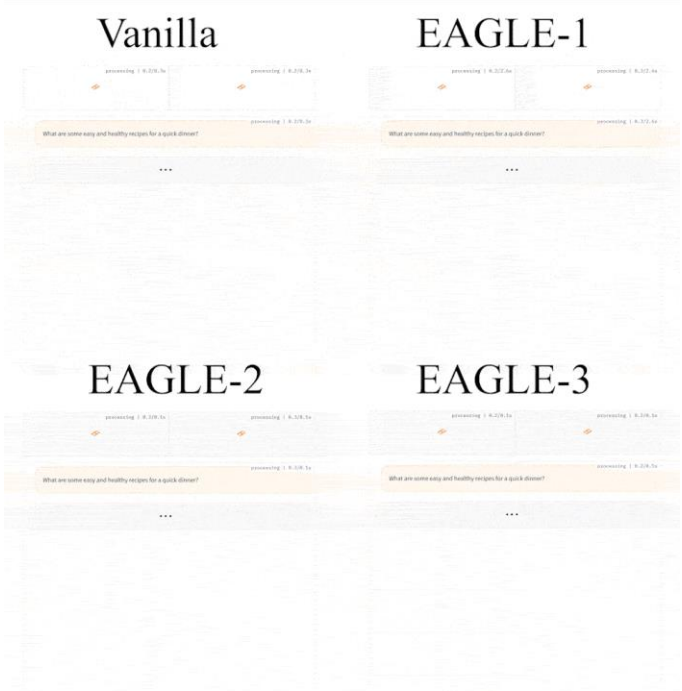
Multi-Token Prediction (MTP)

- **모델 스스로** 미래 여러 토큰을 동시에 예측 (추가 head/모듈)
- 별도 draft 모델 불필요 — self-drafting
- 학습 시그널로도 활용 가능 (DeepSeek-V3 등)
- 대표: Medusa, DeepSeek-V3, Gemma 4

현대 SOTA (EAGLE-3)는 SD의 self-drafting 계열

target 내부 정보를 재사용해 초안 생성하고, **SD식 rejection sampling**으로 출력 무손실을 보장

Speculative Decoding 적용 효과 (시연)



Vanilla autoregressive vs EAGLE-3 — 동일 모델·동일 출력 조건

- 동일 모델, 동일 출력 — 디코딩 방식만 상이
- 위: 일반 autoregressive decoding (토큰 1개씩 순차 생성)
- 아래: speculative decoding (EAGLE-3) — 복수 토큰을 한 사이클에 확정

동일 모델·동일 출력 조건에서 디코딩 방식 차이에 따른 생성 속도 비교

Demo: SafeAILab/EAGLE (github.com/SafeAILab/EAGLE)

오늘 다룰 두 논문

논문 1

Fast Inference from Transformers via Speculative Decoding

Leviathan, Kalman, Matias · Google Research

ICML 2023 · arXiv:2211.17192

Draft model + target model 구조로 출력 분포 동일성을 수학적으로 보장하며 2–3× 속도 향상

논문 2

EAGLE-3: Scaling up Inference Acceleration of LLMs via Training-Time Test

Li et al. · PKU · MSR · Waterloo · Vector

2025 · arXiv:2503.01840

다층 feature fusion + training-time test로 self-drafting 품질을 향상, 최대 6.5× 속도 달성

> 두 논문은 Speculative Decoding의 원리적 기초(논문 1)와 현재 SOTA (논문 2)을 각각 대표

Speculative Decoding 방법론 분류

① 별도 Draft 모델

작은 LM이 초안 생성

- 기성 소형 모델 사용
- — Leviathan et al. (2023)
- KD로 정렬 강화
- — DistillSpec
- **장점**: 모델 수정 불필요

② Self-Drafting

본체가 스스로 초안 생성

- 추가 head 방식
- — Medusa
- feature 재사용 1-layer
- — EAGLE 계열
- 내장 MTP 모듈
- — DeepSeek-V3
- **장점**: 높은 수용률 α

③ Draft-Free

모델 없이 초안 생성

- n-gram trie 활용
- — Lookahead
- datastore 검색
- — REST
- **장점**: 학습·추가 모델 전혀 불필요

모든 방식이 동일한 검증 framework(병렬 forward + 수용/보정)를 공유 — 차이는 **"초안을 누가 만드는가"**

Part 1.

Speculative Decoding (Leviathan et al., ICML 2023)

Fast Inference from Transformers via Speculative Decoding

Yaniv Leviathan, Matan Kalman, Yossi Matias · Google Research · ICML 2023 · arXiv:2211.17192

Motivation

왜 LLM 추론은 느린가?

- 큰 autoregressive 모델 추론: **K 토큰 생성 = K번의 직렬 forward pass**
- 핵심 관찰 ①: 어려운 LM 태스크 안에도 **쉬운 구간**이 존재 → 작은 모델로 충분히 근사 가능
- 핵심 관찰 ②: 남는 연산 자원을 **병렬성(speculative execution)**으로 활용 가능

SD 방식의 세 가지 특성

- 추가 재학습 불필요 — target model 및 draft model 모두 기존 체크포인트 그대로 사용
- 모델 구조 변경 없음 — target model 아키텍처를 수정하지 않음
- 출력 분포 완전 동일 — rejection sampling에 의해 수학적으로 보장

아이디어 — 작은 모델이 제안하고, 큰 모델이 한 번에 검증

① Draft model Mq

작고 빠른 보조 모델

γ개 토큰을 autoregressive로 순차 제안

→

② Target model Mp

크고 정확한 메인 모델

γ+1 위치 → 병렬 forward 1회로 일괄 평가

→

③ Speculative Sampling

각 토큰을 확률 비교로 수용/거부

보정 토큰 추가 → **n+1개 토큰 확정**

- **γ** = lookahead — 한 사이클에 제안하는 토큰 수 (하이퍼파라미터)
- 거부되어도 **최소 1토큰**은 항상 전진 (보정 토큰으로 보장)

Leviathan et al., "Fast Inference from Transformers via Speculative Decoding", ICML 2023 (arXiv:2211.17192)

Speculative Sampling 절차

1. Draft Mq가 현재 prefix에서 γ개 토큰 x1…xγ 를 순차 샘플
2. Target M가 prefix + x…x 에 대해 **p…p 을 단 1회 병렬 계산**
3. 각 토큰을 좌→우로 검사: **min(1, p(x) / q(x))** 로 수용 여부 결정
4. 거부 시: 보정 분포 **norm(max(0, p−q))**에서 재샘플
5. 결과: x…x + 보정 토큰 1개 = **한 사이클에 n+1 토큰 확정**

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.
▷ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.
for $i = 1$ **to** γ **do**
  $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$
  $x_i \sim q_i(x)$
end for
▷ Run M_p in parallel.
$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$
  $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$
▷ Determine the number of accepted guesses n.
$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$
$n \leftarrow \min(\{i - 1 \mid 1 \le i \le \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$
▷ Adjust the distribution from M_p if needed.
$p'(x) \leftarrow p_{n+1}(x)$
if $n < \gamma$ **then**
  $p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$
end if
▷ Return one token from M_p, and n tokens from M_q.
$t \sim p'(x)$
return $prefix + [x_1, \ldots, x_n, t]$

Theorem (무손실 보장): 이 절차의 출력 분포는 M 단독 샘플링 분포와 동일

Leviathan et al., "Fast Inference from Transformers via Speculative Decoding", ICML 2023 (arXiv:2211.17192)

Acceptance Rule과 분포 보존

수용 판단 규칙

조건	수용 확률	처리
q(x) ≤ p(x)	**1 (무조건 수용)**	토큰 확정
q(x) > p(x)	p(x) / q(x)	확률적 수용
거부 발생 시	—	norm(max(0, p−q)) 재샘플

- draft가 **과대평가한 만큼**만 확률적으로 깎아냄
- 거부 시 남은 확률질량 **(p−q)**에서 재샘플 → p 분포 정확히 복원
- greedy(temp=0): draft 토큰 = target argmax일 때만 수용

α = 1 − DTV(p, q)

수용률 = 두 분포의 유사도 — draft가 target을 잘 근사할수록 α↑

실제 디코딩 Trace (논문 Figure 1)

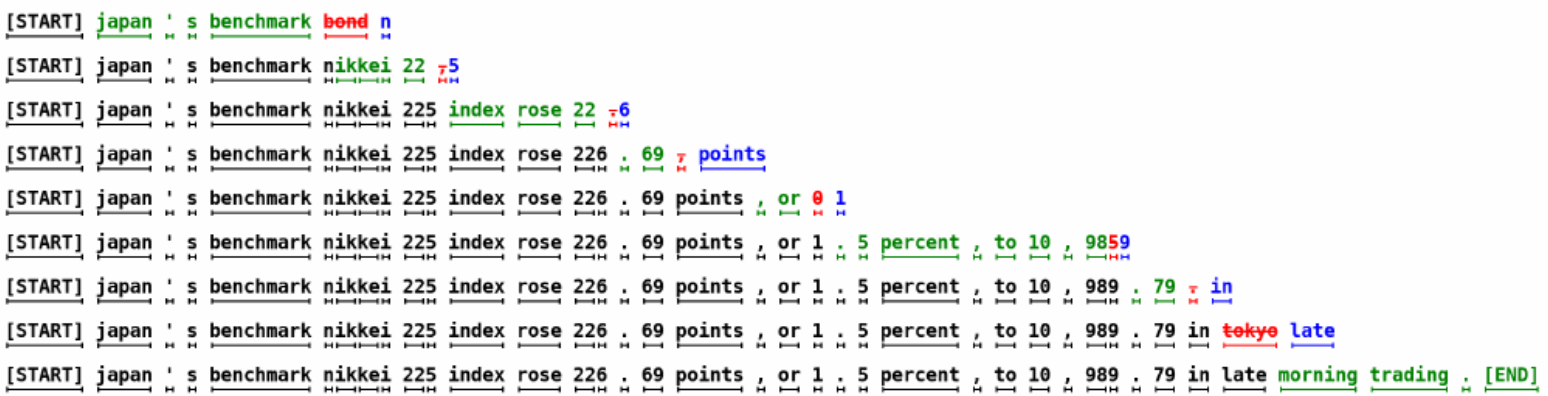
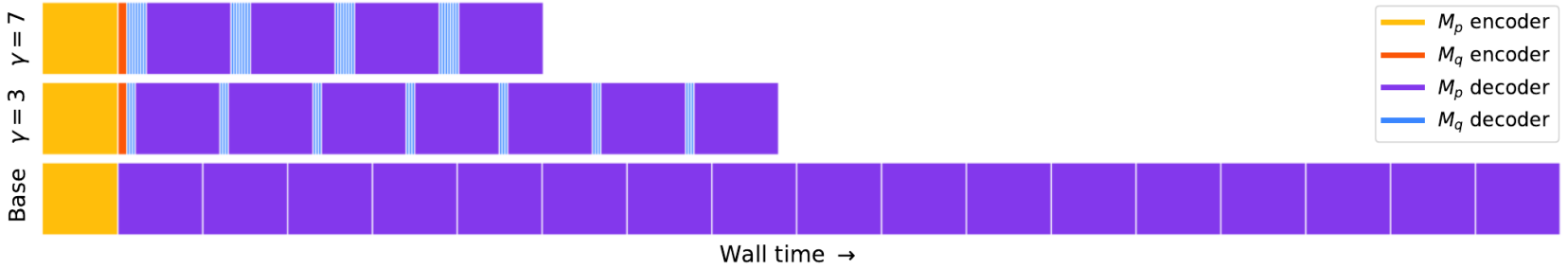


Figure 1. Our technique illustrated in the case of unconditional language modeling. Each line represents one iteration of the algorithm. The **green** tokens are the suggestions made by the approximation model (here, a GPT-like Transformer decoder with 6M parameters trained on lm1b with 8k tokens) that the target model (here, a GPT-like Transformer decoder with 97M parameters in the same setting) accepted, while the **red** and **blue** tokens are the rejected suggestions and their corrections, respectively. For example, in the first line the target model was run only once, and 5 tokens were generated.

- **38토큰** 문장을 target 모델 단 **9회** 실행으로 생성 (일반 autoregressive: 38회)
- 한 번의 target forward로 여러 draft 토큰을 병렬 검증 — 수용된 토큰은 모두 확정됨

Figure 1: Leviathan et al., "Fast Inference from Transformers via Speculative Decoding", ICML 2023 (arXiv:2211.17192)

실행 타이밍으로 보는 Speculative Decoding



파란 블록 = draft Mq

작은 모델의 순차 실행. 한 칸이 짧다 = 빠르다

보라 블록 = target Mp

γ개 토큰을 단 1회의 병렬 forward로 일괄 검증

위 두 줄 vs 맨 아래 baseline

γ=7, γ=3 모두 동일 출력을 훨씬 짧은 시간에 완료 — 직렬 실행을 병렬 검증으로 단축

Figure: Leviathan et al., "Fast Inference from Transformers via Speculative Decoding", ICML 2023 (arXiv:2211.17192)

수용률 α와 기대 생성 토큰 수

α 정의

α = 한 토큰이 수용될 기대 확률

draft model이 target model을 얼마나 잘 근사하는가

사이클당 기대 생성 토큰 수

$$E[\text{tokens}] = (1 - \alpha^{\gamma+1}) / (1 - \alpha)$$

- α = 0.8, γ = 5 → 사이클당 약 **6.07 토큰**
- α가 높을수록 γ를 키우는 이득이 증가
- γ → ∞ 일 때 E[tokens] → 1/(1 − α)

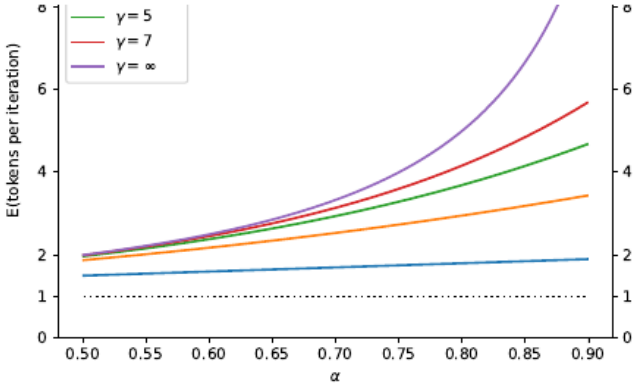


Figure 2. The expected number of tokens generated by Algorithm 1 as a function of α for various values of γ.

Figure 2: Leviathan et al., "Fast Inference from Transformers via Speculative Decoding", ICML 2023 (arXiv:2211.17192)

Walltime Speedup과 최적 γ

Walltime Speedup 공식

Speedup = (1 − αγ+1) / ((1 − α)(γc + 1))

- **c** = draft / target 실행시간 비 (실험: **~0.02** 수준)
- α > c 이면 항상 speedup 보장
- α · c에 따라 최적 γ 존재
- 예: α = 0.75, c = 0.02 → **γ ≈ 6**

- 총 FLOPs는 다소 증가 — 단 walltime은 단축
- 메모리 대역폭 병목을 병렬성으로 우회하는 것이 핵심

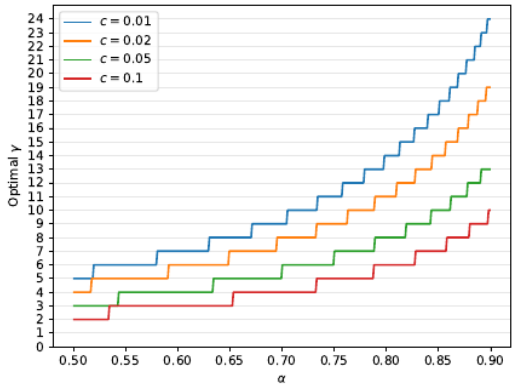


Figure 3. The optimal γ as a function of α for various values of c. F

Table 1. The total number of arithmetic operations and the inference speed vs the baseline, for various values of γ and α, assuming $c = \hat{c} = 0$.

α	γ	OPERATIONS	SPEED
0.6	2	1.53X	1.96X
0.7	3	1.58X	2.53X
0.8	2	1.23X	2.44X
0.8	5	1.63X	3.69X
0.9	2	1.11X	2.71X
0.9	10	1.60X	6.86X

Figures 3 & Table 1: Leviathan et al., "Fast Inference from Transformers via Speculative Decoding", ICML 2023 (arXiv:2211.17192)

T5-XXL 실험 결과 (논문 Table 2)

Table 2. Empirical results for speeding up inference from a T5-XXL 11B model.

TASK	M_q	TEMP	γ	α	SPEED
ENDE	T5-SMALL ★	0	7	0.75	**3.4X**
ENDE	T5-BASE	0	7	0.8	2.8X
ENDE	T5-LARGE	0	7	0.82	1.7X
ENDE	T5-SMALL ★	1	7	0.62	**2.6X**
ENDE	T5-BASE	1	5	0.68	2.4X
ENDE	T5-LARGE	1	3	0.71	1.4X
CNNDM	T5-SMALL ★	0	5	0.65	**3.1X**
CNNDM	T5-BASE	0	5	0.73	3.0X
CNNDM	T5-LARGE	0	3	0.74	2.2X
CNNDM	T5-SMALL ★	1	5	0.53	**2.3X**
CNNDM	T5-BASE	1	3	0.55	2.2X
CNNDM	T5-LARGE	1	3	0.56	1.7X

- T5-small draft 조합이 최적: EnDe **3.4×**, CNN/DM **3.1×** (greedy)
- rejection sampling 방식으로 출력 분포 완전 동일 보장 — 품질 손실 없음

Table 2: Leviathan et al., "Fast Inference from Transformers via Speculative Decoding", ICML 2023 (arXiv:2211.17192)

Part 1 핵심 정리

- 작은 draft 모델이 **γ개 토큰을 순차 제안** → target 모델이 1회 forward로 병렬 검증
- rejection sampling 기반 수용으로 **출력 분포 완전 보존** — 재학습 불필요
- T5-XXL 기준 재학습 없이 **2 ~ 3.4×** walltime 단축
- 이후 Medusa, EAGLE 등 self-drafting 계열 연구의 **공통 이론적 토대**

- bigram 같은 초간단 draft ($\alpha \approx 0.2$)로도 ~1.25× 가능 — 프레임워크의 범용성 입증
- LaMDA 137B에서도 동작 확인 (모델 크기 무관)

Part 2.

EAGLE-3 (Li et al., 2025)

Scaling up Inference Acceleration of Large Language Models via Training-Time Test

Yuhui Li, Fangyun Wei, Chao Zhang, Hongyang Zhang — PKU · MSR · Waterloo · Vector Institute, arXiv:2503.01840 (2025)

EAGLE 계열의 흐름: 1 → 2

EAGLE-1

- **Feature-level autoregression** — 토큰 대신 target의 top-layer feature를 예측
- target의 내부 정보를 재사용 → vanilla SD보다 강력한 draft 품질
- 1-layer 경량 draft head — **self-drafting**, MTP 계열
- 별도 draft 모델 불필요 — target 가중치 동결 유지

EAGLE-2

- Draft를 **트리(tree)**로 확장 → 더 많은 후보 경로 탐색
- Confidence 기반 **동적 트리 조정** — 유망한 경로에 자원 집중
- EAGLE-1 대비 **20~40% 추가 가속**
- Vicuna-13B 기준: 3.07× → **4.26×**

Leviathan(별도 draft 모델) → **EAGLE** (target 내부 feature 재사용 self-draft) — MTP 아이디어와의 융합

EAGLE-1 — Feature 수준 Autoregression

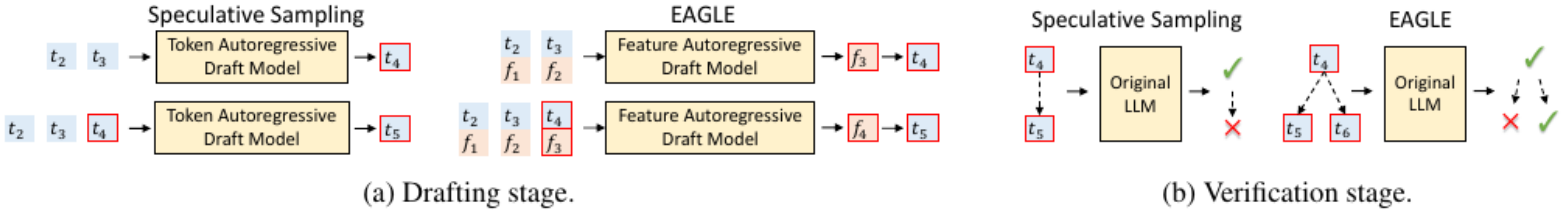


Figure 3: Comparison of standard speculative sampling and EAGLE. For simplicity, EAGLE's tree-structured draft is shown only in the verification stage, while the illustration of the drafting stage uses a chain-structured draft. Here, t_i denotes the i-th token embedding, and f_i denotes the i-th feature vector in the second-to-top-layer of LLM before LM head.

- 토큰 대신 target의 **top-layer feature**를 autoregress
  ➔ feature 수준이 토큰보다 규칙적이라 예측 용이
- sampling의 무작위성은 직전 토큰 embedding을 함께 입력해 보정
- draft는 **1-layer decoder**만 학습 (embedding / LM head는 target 재사용)
- 성능: Vicuna-13B **3.07×** — vanilla SD / Medusa(2.07×) 대비 우위

Figure: Li et al., "EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees", arXiv:2406.16858 (2024)

Draft Tree — 정적 트리에서 동적 트리로

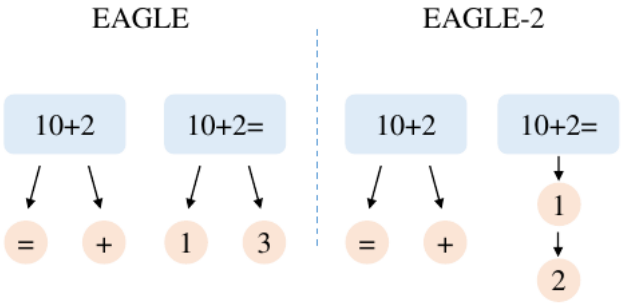


Figure 4: Differences between EAGLE and EAGLE-2. EAGLE always uses a fixed draft shape. When the query is "10+2=", the next token is very likely to be correctly predicted as "1". However, with a static draft tree, EAGLE would still add two candidates, even though the probability of the other candidate "3" being correct is very low. EAGLE-2, on the other hand, adjusts the shape of draft tree based on the context. When the query is "10+2", the next token is difficult to predict, so EAGLE-2 adds two candidates. For the simpler query "10+2=", EAGLE-2 adds only one candidate "1".

- 초안을 체인이 아닌 **트리**로 — 여러 후보 경로를 한 번의 forward로 동시 검증 (tree attention)
- EAGLE-1 / Medusa: **고정 트리** — 모든 문맥에 같은 모양
- 관찰: 수용률은 문맥 의존 + draft confidence가 수용률의 좋은 근사 (calibrated)
- EAGLE-2: confidence 누적값 기준 **expansion → rerank** 2단계로 트리를 문맥별 동적 구성
- 효과: EAGLE-1 대비 **+20~40%** (Vicuna-13B 3.07× → **4.26×**), 추가 학습 불필요

Figure: Li et al., "EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees", arXiv:2406.16858 (2024)

EAGLE의 feature 제약과 스케일링 한계

- EAGLE 학습 loss = feature 회귀(**l_fea**) + token 예측(**l_token**)
- feature 예측이 **추가 제약**으로 작용 → 표현력 제한
- 학습 데이터를 8배 늘려도 speedup **정체(plateau)**
- multi-step drafting 시: 자신이 예측한 feature를 다시 입력 → **train-test 분포 불일치**
- step이 깊어질수록 수용률 급락: 2-step째 α: EAGLE **35%** vs EAGLE-3 **85%**

feature 제약 + 분포 불일치가 함께 스케일링의 천장을 형성 → EAGLE-3의 출발점

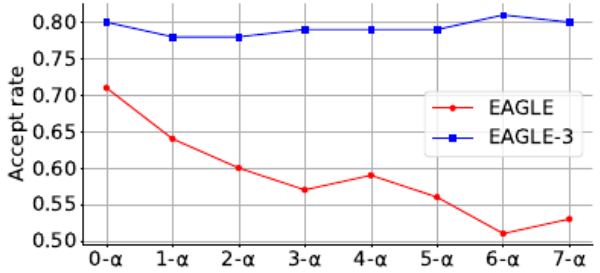


Figure 7: Acceptance rate of EAGLE and EAGLE-3 on MT-bench, with the target model being LLaMA-Instruct 3.1 8B. Hereby, n-α refers to the acceptance rate when the input contains n estimated features, under the condition that the previous estimated tokens are all accepted by the target model.

Figure 7: Li et al., "EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test", arXiv:2503.01840 (2025)

방법 ①: Feature 예측 제약 제거

EAGLE-1 / 2 (기존)

draft 출력 → target feature에 맞도록 회귀 제약 → LM head

feature regression loss (l_fea) 포함 → 표현 자유도 제한

데이터 증가 시 speedup 정체

EAGLE-3 (제안)

draft 출력 → **제약 없는 자유 벡터** → LM head로 직접 token 예측

feature regression loss 완전 제거 → **표현 자유도 확보**

스케일링 병목의 첫 번째 원인 해소

- Ablation: 제약 제거만으로 **3.16× → 3.82×** (MT-bench, LLaMA-3.1-8B-Instruct)
- Feature 정합 없이도 token 예측 정확도 향상 — 제약이 오히려 표현을 제한하였음

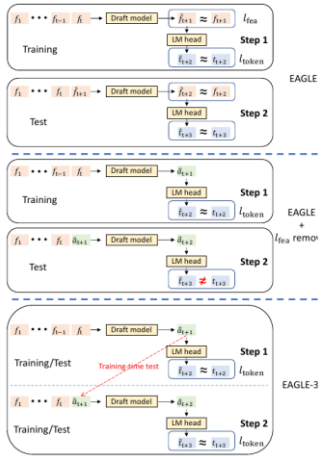


Figure 3: Illustration of **training-time test** (the bottom part) and its comparison with other draft methods (the upper and middle parts). f denotes the feature, t denotes the token, and a represents the unconstrained vectors. We use the hat to denote the predictions from models. All the methods shown in the figure use the token sequence from the previous time step, but for simplicity, this is not depicted in the figure. The input to EAGLE-3 is not actually f, but it is not shown in this figure. We will provide a detailed explanation in the following section.

To summarize, this paper introduces EAGLE-3, an enhanced version of EAGLE that achieves a significant speedup. EAGLE-3 is parallelized

방법 ②: Multi-layer Feature Fusion

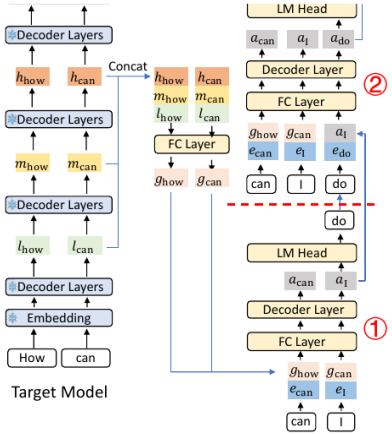


Figure 5: Diagram of the EAGLE-3 inference pipeline, illustrating the three steps of the draft model. l, m, and h represent the low, middle, and high-level features of the target model, respectively. e denotes the embedding.

3 EAGLE-3

In this section, we provide a detailed description of the implementation of EAGLE-3.

3.1 Inference Pipeline

Consistent with other speculative sampling methods, EAGLE-3 alternates between the drafting and verification stages. The difference between

- Top-layer feature만으로는 "다음-다음 토큰" 예측에 정보 **부족**
- Target의 **low / mid / high** 3개 층 feature를 추출하여 융합
- **concat[f_low ; f_mid ; f_high] → FC → g** → draft 입력으로 사용
- 융합 feature g + 직전 token embedding → 1-layer decoder가 초안 생성

> **g** = FC(concat[f_low ; f_mid ; f_high])
> draft input = **g** ⊕ token_emb(x_prev)

Ablation: multi-layer fusion 추가로 **3.82× → 4.40×**

EAGLE-3 파이프라인 단계별 구성

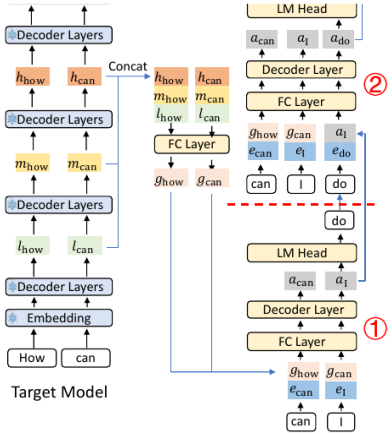


Figure 5: Diagram of the EAGLE-3 inference pipeline, illustrating the three steps of the draft model. l, m, and h represent the low, middle, and high-level features of the target model, respectively. e denotes the embedding.

3 EAGLE-3

In this section, we provide a detailed description of the implementation of EAGLE-3.

3.1 Inference Pipeline

Consistent with other speculative sampling methods, EAGLE-3 alternates between the drafting and verification stages. The difference between

1 **Step ①**

target 직전 forward에서 **low / mid / high feature** 추출 → g 융합; 직전 token embedding과 함께 1-layer draft가 첫 초안 token 생성

2 **Step ②**

draft 출력 **(a)**를 다시 입력으로 연결 → 다음 초안 token 생성 (자기 예측을 입력으로 받는 구조)

3 **Step ③**

반복하며 **draft 트리**를 확장 → target이 한 번의 병렬 forward로 일괄 검증

Training-Time Test = 학습 시 ①②③을 그대로 재현 → 추론과 학습의 분포 불일치 제거

방법 ③: Training-Time Test

- 문제: 학습은 ground-truth 입력, 추론은 **자기 예측 입력** → 분포 불일치 발생
- 해법: **학습 중에 multi-step drafting을 그대로 시뮬레이션**
- 자신의 예측 출력을 다시 입력받는 상황을 학습이 직접 경험 (①②③ 단계 재현)
- Attention mask를 step 구조에 맞게 조정 → 단계 간 정보 흐름 제어

2-step째 수용률 α 비교:

EAGLE: 98% → 72% → **35%**

EAGLE-3: 98% → 92% → **85%**

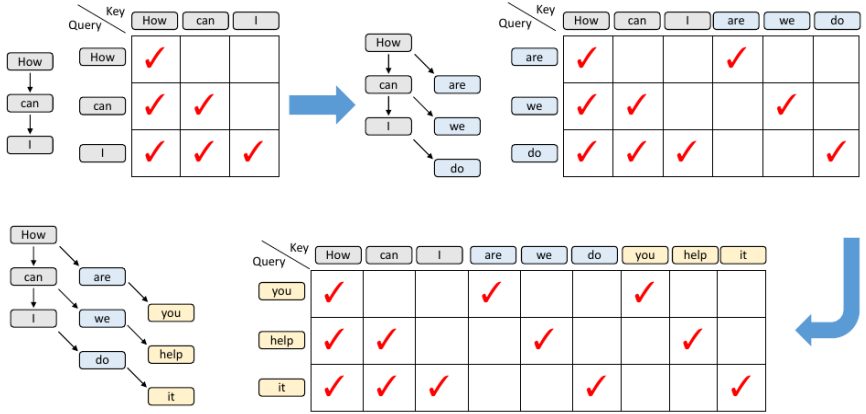


Figure 6: Diagram of the attention causal masks during training-time test. It sequentially shows a native training step (the first step) and two simulated training steps (the second and third steps). The arrows between tokens represent contextual relationships. The gray tokens represent the training data while the blue and yellow tokens represent the first- and second-round predictions by the draft model, respectively.

EAGLE-3 메인 결과 (Table 1)

Table 1: Speedup ratios and average acceptance lengths τ of different methods. V represents Vicuna, L31 represents LLaMA-Instruct 3.1, L33 represents LLaMA-Instruct 3.3, and DSL represents DeepSeek-R1-Distill-LLaMA. SpS denotes standard speculative sampling, with its draft model being Vicuna-68M. Methods like Medusa relax acceptance conditions under non-greedy settings, which do not guarantee lossless acceleration. Therefore, we do not compare EAGLE-3 with these methods when temperature=1.

		MT-bench		HumanEval		GSM8K		Alpaca		CNN/DM		Mean	
Model	Method	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Temperature=0													
V 13B	SpS	1.93x	2.27	2.23x	2.57	1.77x	2.01	1.76x	2.03	1.93x	2.33	1.92x	2.24
	PLD	1.58x	1.63	1.85x	1.93	1.68x	1.73	1.16x	1.19	2.42x	2.50	1.74x	1.80
	Medusa	2.07x	2.59	2.50x	2.78	2.23x	2.64	2.08x	2.45	1.71x	2.09	2.12x	2.51
	Lookahead	1.65x	1.69	1.71x	1.75	1.81x	1.90	1.46x	1.51	1.46x	1.50	1.62x	1.67
	Hydra	2.88x	3.65	3.28x	3.87	2.93x	3.66	2.86x	3.53	2.05x	2.81	2.80x	3.50
	EAGLE	3.07x	3.98	3.58x	4.39	3.08x	3.97	3.03x	3.95	2.49x	3.52	3.05x	3.96
	EAGLE-2	4.26x	4.83	4.96x	5.41	4.22x	4.79	4.25x	4.89	3.40x	4.21	4.22x	4.83
	EAGLE-3	**5.58x**	**6.65**	**6.47x**	**7.54**	**5.32x**	**6.29**	**5.16x**	**6.17**	**5.01x**	**6.47**	**5.51x**	**6.62**
L31 8B	EAGLE-2	3.16x	4.05	3.66x	4.71	3.39x	4.24	3.28x	4.12	2.65x	3.45	3.23x	4.11
	EAGLE-3	**4.40x**	**6.13**	**4.85x**	**6.74**	**4.48x**	**6.23**	**4.82x**	**6.70**	**3.65x**	**5.34**	**4.44x**	**6.23**
L33 70B	EAGLE-2	2.83x	3.67	3.12x	4.09	2.83x	3.69	3.03x	3.92	2.44x	3.55	2.85x	3.78
	EAGLE-3	**4.11x**	**5.63**	**4.79x**	**6.52**	**4.34x**	**6.15**	**4.30x**	**6.09**	**3.27x**	**5.02**	**4.12x**	**5.88**
DSL 8B	EAGLE-2	2.92x	3.80	3.42x	4.29	3.40x	4.40	3.01x	3.80	3.53x	3.33	3.26x	3.92
	EAGLE-3	**4.05x**	**5.58**	**4.59x**	**6.38**	**5.01x**	**6.93**	**3.65x**	**5.37**	**3.52x**	**4.92**	**4.16x**	**5.84**
Temperature=1													
V 13B	SpS	1.62x	1.84	1.72x	1.97	1.46x	1.73	1.52x	1.78	1.66x	1.89	1.60x	1.84
	EAGLE	2.32x	3.20	2.65x	3.63	2.57x	3.60	2.45x	3.57	2.23x	3.26	2.44x	3.45
	EAGLE-2	3.80x	4.40	4.22x	4.89	3.77x	4.41	3.78x	4.37	3.25x	3.97	3.76x	4.41
	EAGLE-3	**4.57x**	**5.42**	**5.15x**	**6.22**	**4.71x**	**5.58**	**4.49x**	**5.39**	**4.33x**	**5.72**	**4.65x**	**5.67**
L31 8B	EAGLE-2	2.44x	3.16	3.39x	4.39	2.86x	3.74	2.83x	3.65	2.44x	3.14	2.80x	3.62
	EAGLE-3	**3.07x**	**4.24**	**4.13x**	**5.82**	**3.32x**	**4.59**	**3.90x**	**5.56**	**2.99x**	**4.39**	**3.45x**	**4.92**
L33 70B	EAGLE-2	2.73x	3.51	2.89x	3.81	2.52x	3.36	2.77x	3.73	2.32x	3.27	2.65x	3.54
	EAGLE-3	**3.96x**	**5.45**	**4.36x**	**6.16**	**4.17x**	**5.95**	**4.14x**	**5.87**	**3.11x**	**4.88**	**3.95x**	**5.66**
DSL 8B	EAGLE-2	2.69x	3.41	3.01x	3.82	3.16x	4.05	2.64x	3.29	2.35x	3.13	2.77x	3.54
	EAGLE-3	**3.20x**	**4.49**	**3.77x**	**5.28**	**4.38x**	**6.10**	**3.16x**	**4.30**	**3.08x**	**4.27**	**3.52x**	**4.89**

최대 **6.47×** (HumanEval, Vicuna-13B) · 5-task 평균 **5.51×** · EAGLE-2 대비 **약 1.3–1.4×** 일관 개선 (temp=0)

데이터 스케일링과 Speedup

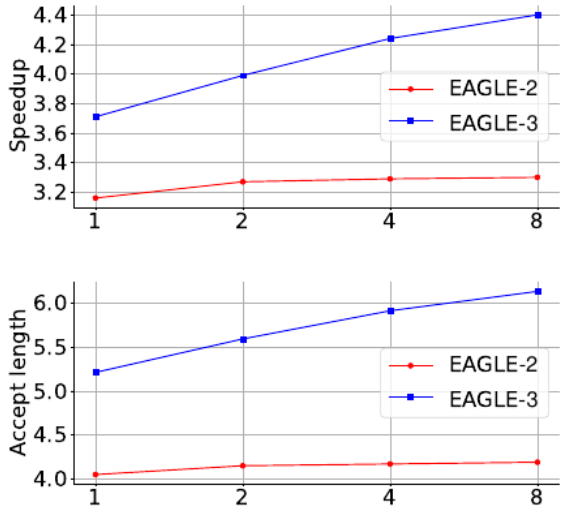


Figure 1: Scaling law evaluated on the MT-bench using LLaMA-Instruct 3.1 8B as the target model, with the x-axis representing the data scale relative to ShareGPT. The new architectural designs in EAGLE-3 enable an increasing scaling curve, which was never observed in the previous works.

> Inference 가속의 **Scaling Law** — 최초 확인
> 학습 데이터 1× → 8× 증가 시 EAGLE-3 speedup 지속 상승

- EAGLE-2: 데이터 8배 증가에도 speedup **정체 (plateau)**
- EAGLE-3: 데이터 증가 → speedup **지속 상승**
- feature 제약 제거 + training-time test가 스케일링을 가능하게 함
- LLM 사전학습과 유사하게, **추론 가속에도 scaling law 성립**

Li et al., "EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test", arXiv:2503.01840 (2025)

Production 검증: SGLang / vLLM

Table 4 — SGLang (LLaMA-3.1-8B-Instruct, H100, MT-bench)

Table 4: Throughput at batch size = 1 on H100 when the target model is LLaMA-Instruct 3.1 8B and the testing dataset is MT-bench. The experiments were conducted by the SGLang team.

Method	Throughput (bs=1)
SGLang (w/o speculative, 1x H100)	158.34 tokens/s
SGLang + EAGLE-2 (1x H100)	244.10 tokens/s
SGLang + EAGLE-3 (1x H100)	373.25 tokens/s

Table 5 — vLLM (LLaMA-3.1-8B-Instruct, A100, MT-bench)

Table 5: Throughput improvement under different batch sizes on A100 and LLaMA-Instruct 3.1 8B for the MT-Bench dataset, with vLLM without speculative sampling as the baseline (1.00x).

Batch size	2	4	8	16	24	32	48	56
EAGLE	1.30x	1.25x	1.21x	1.10x	1.03x	0.93x	0.82x	0.71x
EAGLE-3	1.75x	1.68x	1.58x	1.49x	1.42x	1.36x	1.21x	1.01x

- 기존 SD(EAGLE)은 **큰 배치에서 역효과** — BS ≥ 24에서 baseline 미만 (0.99×)
- EAGLE-3는 BS=64에서도 **+38% 처리량** (1.38×) 유지 — 배치 크기에 강건
- vLLM에서도 동일 경향: BS=2에서 **1.75×** 처리량 향상
- SGLang / vLLM 프로덕션 프레임워크에 기본 탑재 — 실서비스 적용 검증

Part 3.

그 밖의 주요 방법론과 프로덕션 생태계

Medusa · DeepSeek-V3 MTP · Lookahead · SGLang · vLLM · Gemma 4

Medusa — Multiple Decoding Heads (ICML 2024)

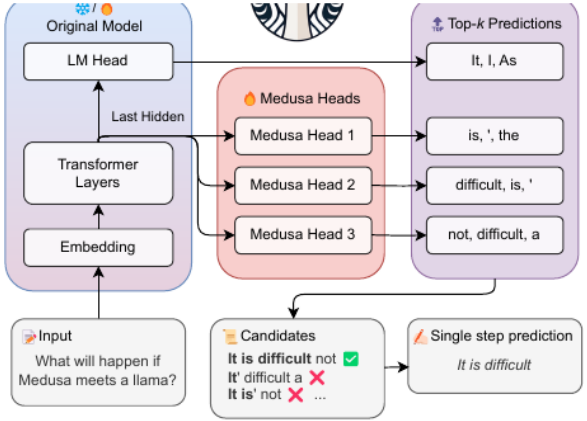


Figure 1. MEDUSA introduces *multiple heads* on top of the last hidden states of the LLM, enabling the prediction of several subsequent tokens in parallel (Section 2.1.1). During inference, each head generates multiple top predictions for its designated position. These predictions are assembled into candidates, which are processed in parallel using a *tree-based attention* mechanism (Section 2.1.2). The final step is to verify the candidates and accept a continuation. Besides the standard rejection sampling scheme, a *typical acceptance* scheme (Section 2.3.1) can also be used here to select reasonable continuations, and the *longest accepted candidate prefix* will be used for the next decoding phase.

- backbone 위에 여러 decoding head를 추가해 미래 토큰을 **병렬 예측**
  ➔ 별도 draft 모델 불필요
- **tree attention**으로 다수 후보 경로를 한 번의 forward pass로 동시 검증
- **Medusa-1**: backbone 동결, head만 학습 (무손실 가속) / **Medusa-2**: 공동 학습으로 가속 극대화
- 성능: Medusa-1 약 **2.2×**, Medusa-2 **2.3–2.8×** (Vicuna-13B MT-Bench 최대 **3.16×**)

Cai et al., "Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads", ICML 2024 (arXiv:2401.10774)

DeepSeek-V3 — 학습·추론 통합 MTP (2024)

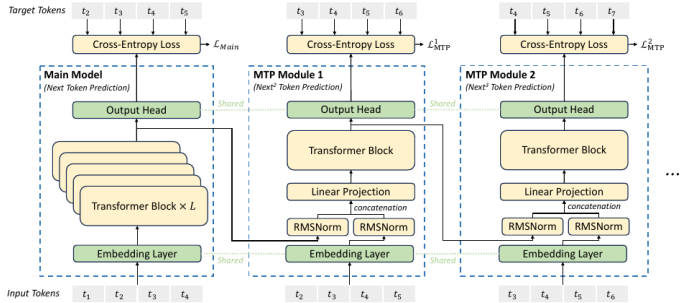


Figure 3 | Illustration of our Multi-Token Prediction (MTP) implementation. We keep the complete causal chain for the prediction of each token at each depth.

학습

- **1-layer MTP module**이 다음-다음 토큰까지 예측 — 학습 시 그널 밀도 강화 (embedding·output head는 본체와 공유)

추론

- 학습된 MTP module을 speculative decoding의 **draft**로 재사용
- 두 번째 토큰 acceptance rate **85~90%** (주제 전반에서 일관)
- 디코딩 속도 약 **1.8× TPS** 향상
- frontier 모델에서 학습+추론 MTP 통합을 처음 검증

DeepSeek-AI, "DeepSeek-V3 Technical Report", arXiv:2412.19437 (2024)

Lookahead — Draft 모델 없는 n-gram 기반 가속

Lookahead: An Inference Acceleration Framework for Large Language Model with Lossless Generation Accuracy

Table 2: Inference Speed(token/s) of different methods on various models, datasets and devices. As LookaheadDecoding only offers implementation for Llama, we do not conduct experience on AntGLM-10B Model.

model	dataset	device	transformers	vLLM	LLMA	LaDe	lookahead(ours)
AntGLM-10B	AntRAG	A100-80G	52.4	52.06(x0.99)	165.4(x3.16)	-	280.9(x5.36)
AntGLM-10B	AntRAG	A10	20.3	20.29(x1.00)	59.5(x2.93)	-	105.1(x5.18)
AntGLM-10B	AntRAG	V100-32G	27.3	27.28(x1.00)	64.3(x2.36)	-	118.9(x4.36)
Llama-7B	Dolly	A100-80G	50.4	91.04(x1.81)	60.7(x1.20)	66.7(x1.32)	106.8(x2.12)
Llama-7B	Dolly	A10	31.4	32.58(x1.04)	35.8(x1.14)	38.2(x1.22)	55.7(x1.77)
Llama-7B	GSM8k	A100-80G	41.4	92.09(x2.22)	69.0(x1.67)	94.3(x2.28)	111.3(x2.69)
Llama-7B	GSM8k	A10	31.4	32.73(x1.04)	41.1(x1.31)	48.9(x1.56)	68.1(x2.17)
Llama-7B	HumanEval-x	A100-80G	51.1	90.47(x1.77)	66.5(x1.30)	88.3(x1.73)	161.5(x3.16)
Llama-7B	HumanEval-x	A10	30.9	32.46(x1.05)	42.1(x1.36)	46.0(x1.49)	89.6(x2.90)
Llama-13B	Dolly	A100-80G	39.9	51.67(x1.29)	59.0(x1.48)	45.2(x1.13)	84.6(x2.12)
Llama-13B	Dolly	V100-32G	20.5	22.07(x1.08)	23.9(x1.17)	23.2(x1.13)	35.2(x1.72)
Llama-13B	GSM8k	A100-80G	42.9	52.06(x1.21)	51.8(x1.21)	64.8(x1.51)	103.4(x2.41)
Llama-13B	GSM8k	V100-32G	22.0	22.43(x1.02)	25.8(x1.17)	27.8(x1.26)	45.6(x2.07)
Llama-13B	HumanEval-x	A100-80G	35.0	51.49(x1.47)	57.7(x1.65)	66.1(x1.89)	137.3(x3.92)
Llama-13B	HumanEval-x	V100-32G	21.5	22.33(x1.04)	28.8(x1.34)	27.5(x1.28)	57.0(x2.65)

- 프롬프트·생성 이력의 **n-gram을 trie에 저장**해 draft 후보로 사용 — 학습·추가 모델 불필요
- 공통 prefix를 공유하는 다중 분기 후보를 한 번에 검증 (**무손실**)
- LLaMA-7B 기준 Dolly **2.1×** · GSM8K **2.7×** · 코드(HumanEval-X) **3.2~3.9×**
- 검색·RAG처럼 입력과 출력이 겹치는 실서비스에서 **5~6× 사례** (Alipay 운영)

Zhao et al., "Lookahead: An Inference Acceleration Framework...", arXiv:2312.12728 (KDD 2024)

REST — Retrieval 기반 Speculative Decoding (NAACL 2024)

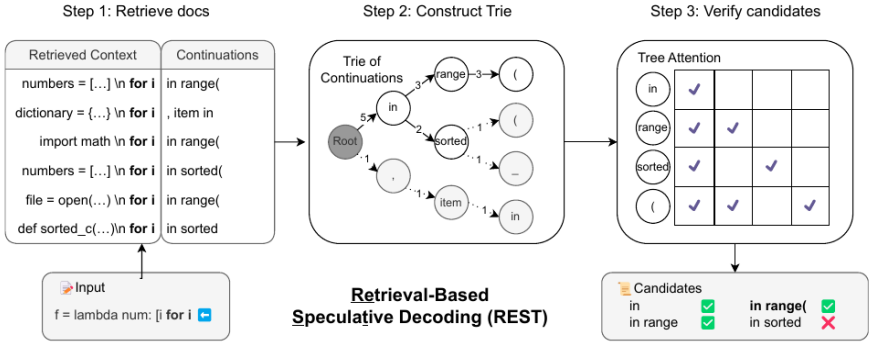


Figure 1: Overview of REST. During inference, the input context is utilized as the query to retrieve docs from the datastore that match the longest suffix of the input. A Trie is constructed using the continuations from the retrieved docs. We prune the low-frequency (weight) branches and the remaining subtree is further used as candidates. Candidates will be fed into the LLM with a tree attention mask for verification. All correct tokens from the start will be accepted, and the draft tokens after the first mistake will be rejected.

- datastore에서 현재 context의 suffix와 일치하는 후속 토큰을 검색해 **draft**로 사용
- 검색 결과를 **Trie**로 구성, 고빈도 prefix를 후보로 — 검색+구성 **1ms 미만**
- draft 모델·학습 전혀 불필요 — 모든 LM에 **plug-and-play**
- 성능: HumanEval(코드) **2.1~2.4×** · MT-Bench **1.6~1.8×** (CodeLlama/Vicuna 7·13B)

He et al., "REST: Retrieval-Based Speculative Decoding", NAACL 2024 (arXiv:2311.08252)

DistillSpec — KD로 수용률 α 높이기 (ICLR 2024)

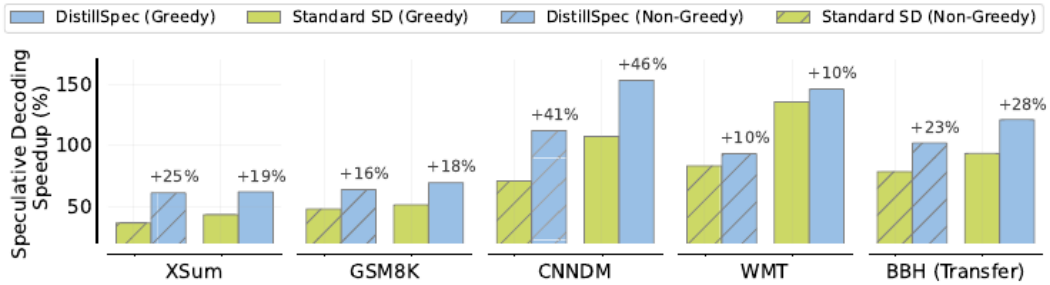


Figure 1: Performance comparison of standard speculative decoding (SD) vs. our proposed DistillSpec, with small- and XL-sized models from the T5 v1.1 family (Raffel et al., 2020) being utilized as the draft and the target models, respectively. DistillSpec enhances SD speed by better aligning the draft with the target via white-box knowledge distillation, resulting in a consistent 10−45% speedup improvement over standard SD across various datasets. The distilled draft model from GSM8K transfers well to 23 unseen BigBenchHard tasks (Suzgun et al., 2022), resulting in an average speedup of 26%. See §5.1 for additional details.

- knowledge distillation으로 draft 분포를 target에 정렬 → token **수용률 α** 직접 향상
- 핵심: draft 자신의 출력(**on-policy data**)으로 학습 + task·디코딩 방식에 맞는 **divergence** 선택
- 성능: 표준 SD 대비 **+10~46%** 추가 가속, 미학습 태스크 전이에서도 **+13~15%**

Zhou et al., "DistillSpec: Improving Speculative Decoding via Knowledge Distillation", ICLR 2024 (arXiv:2310.08461)

프로덕션 적용 ① — SGLang의 MTP 서빙

SGLang + DeepSeek-V3 MTP (LMSYS)

- DeepSeek-V3의 MTP(NextN)를 EAGLE 방식 draft로 통합
- **16×H200** 소규모 배포: throughput **+60.8%** (4-token, rank당 82.0 tok/s)
- **128×H200** 대규모: **+14.2%**
- 생성 결과 동일성(exact equivalence) 보장

AMD MI300X에서의 검증 (ROCm 블로그)

- MI300X **8-GPU**에서 DeepSeek-V3 NextN + SGLang 운영
- Random 데이터셋: **1.25~2.11×** (낮은 concurrency에서 최대)
- ShareGPT: **1.36~1.80×**
- NVIDIA 외 하드웨어에서도 동작 검증

MTP 기반 speculative decoding은 프레임워크·하드웨어 전반에서 production 검증 단계에 도달

LMSYS, "Accelerating SGLang with Multiple Token Prediction" (lmsys.org)
AMD, "Efficient LLM Serving with MTP: DeepSeek V3 + SGLang" (rocm.blogs.amd.com)

프로덕션 적용 ② — vLLM 표준화와 Gemma 4

vLLM — SD 방식의 표준 인터페이스

- **-speculative-config** 하나로 방식 선택
- 모델 기반: EAGLE · MTP · Draft Model · PARD · MLP
- 경량: N-gram · Suffix Decoding
- 문서 가이드: EAGLE·MTP·Draft는 low-QPS에서 **"High gain"** 분류

Gemma 4 — MTP drafter 내장 (Google)

- 경량 MTP drafter가 target과 activation·KV cache 공유
- 품질 저하 없이 최대 **3×** 가속
- Apple Silicon 26B MoE: batch 4~8 기준 약 **2.2×**
- 엣지(E2B·E4B)부터 클라우드까지 동일 적용

SD/MTP는 연구를 넘어 주요 서빙 스택의 기본 기능으로 정착

vLLM Documentation, "Speculative Decoding" (docs.vllm.ai)
Google, "Multi-token prediction in Gemma 4" (blog.google)

시스템 관점 — Batch 크기와 SD의 이득

작은 batch (낮은 QPS) — memory-bound

- GPU 연산 유닛이 유휴 — 검증 연산이 사실상 **무료**
- SD 이득 최대 (**3~6×**)

큰 batch (높은 QPS) — compute-bound

- 연산이 포화 — draft·검증이 **실비용**으로 전환
- 이득 감소, 역효과 가능 (EAGLE: BS=24에서 **0.93×**)

SGLang · H100 · LLaMA-3.1-8B, EAGLE-3 논문 Table 4

vLLM 가이드 — 낮은 QPS에서 EAGLE·MTP **"High gain"**, 높은 QPS에선 방식·수용률에 따라 신중히 선택

Li et al., arXiv:2503.01840
vLLM Documentation (docs.vllm.ai)

두 논문 비교: SD의 처음과 지금

항목	Leviathan et al. (2023)	EAGLE-3 (2025)
Draft 방식	별도 소형 모델 (off-the-shelf)	target feature 재사용 self-draft/ self-drafting + feature fusion
추가 학습	불필요	draft head 학습 필요 (target 동결)
구조	선형 γ-체인	동적 트리 (EAGLE-2 계승)
무손실 보장	O (rejection sampling)	O (동일 검증 framework)
속도	2 – 3.4×	**4 – 6.5×**
배포	개념 검증 (TPU)	vLLM / SGLang 프로덕션 탑재

검증 framework는 그대로, **"초안을 누가 어떻게 만드는가"**가 3년간의 핵심 진화

separate draft model → MTP 기반 self-drafting + feature fusion + training-time test

방법론 종합 비교

방법	초안 방식	추가 학습	무손실	대표 성능	비고
Leviathan SD (2023)	별도 소형 모델	불필요	O	2 – 3.4×	SD framework 정립
Medusa (2024)	추가 decoding heads	head 학습	O	2.2 – 2.8×	tree attention
EAGLE-1/2 (2024)	feature 재사용 1-layer	draft 학습	O	**3.1×** / **4.3×**	동적 트리(EAGLE-2)
EAGLE-3 (2025)	multi-layer fusion	draft 학습	O	**4 – 6.5×**	scaling law, vLLM/SGLang
DeepSeek-V3 MTP (2024)	내장 MTP 모듈	본체 공동학습	O	~1.8× TPS	학습 시그널 겸용
Lookahead (2024)	n-gram trie	불필요	O	2 – 3.9×	실서비스 5~6× 사례
REST (2024)	datastore 검색	불필요	O	1.6 – 2.4×	코드에서 강함
DistillSpec (2024)	KD로 draft 정렬	KD 학습	O	SD +10~46%	α 향상 기법

출처: 본 발표에서 다룬 각 논문·기술블로그

한계와 적용 가이드

알려진 한계

- 대배치·고QPS에서 이득 감소
-
— compute-bound 전환 시 draft 연산이 실비용
- draft-target 분포 불일치 도메인에서 α 하락
-
— 전문 분야·저자원 언어에서 비효율 발생
- 트리 검증의 KV cache·메모리 오버헤드
- 학습형 drafter는 target 모델마다 재학습 필요

상황별 선택 가이드

- 코드·템플릿 등 반복 패턴 多
-
→ **Lookahead · REST** (학습 불필요)
- 단일 모델 최고 가속 목표
-
→ **EAGLE-3**
- 학습 파이프라인 통제 가능
-
→ **MTP 내장** (DeepSeek-V3 방식)
- 기성 모델 조합만 가능
-
→ **Leviathan SD + DistillSpec**

Takeaways & Discussion

1. SD = **무손실** LLM 추론 가속의 표준 framework (draft → 병렬 verify → 수용/보정)

2. 성능의 열쇠 = 수용률 **α**와 draft 비용 **c**의 균형 — draft 품질과 속도 간 트레이드오프

3. self-drafting(EAGLE 계열)이 별도 draft 모델을 대체 — target 내부 정보 재사용이 α를 끌어올림

4. EAGLE-3: feature 제약 제거 + training-time test → **데이터 스케일링**까지 확보 (최대 6.5×)

남은 질문 (Discussion)

더 큰 배치 / 장문 컨텍스트에서의 이득 · reasoning 모델 특화 drafting · 학습-추론 통합 (MTP pretraining과의 결합)

Thank you

Q & A

이정섭 · 2026. 06. 04 · 주간세미나

Leviathan et al., "Fast Inference from Transformers via Speculative Decoding", ICML 2023 (arXiv:2211.17192)
Li et al., "EAGLE-3: Scaling up Inference Acceleration of LLMs via Training-Time Test", arXiv:2503.01840 (2025)