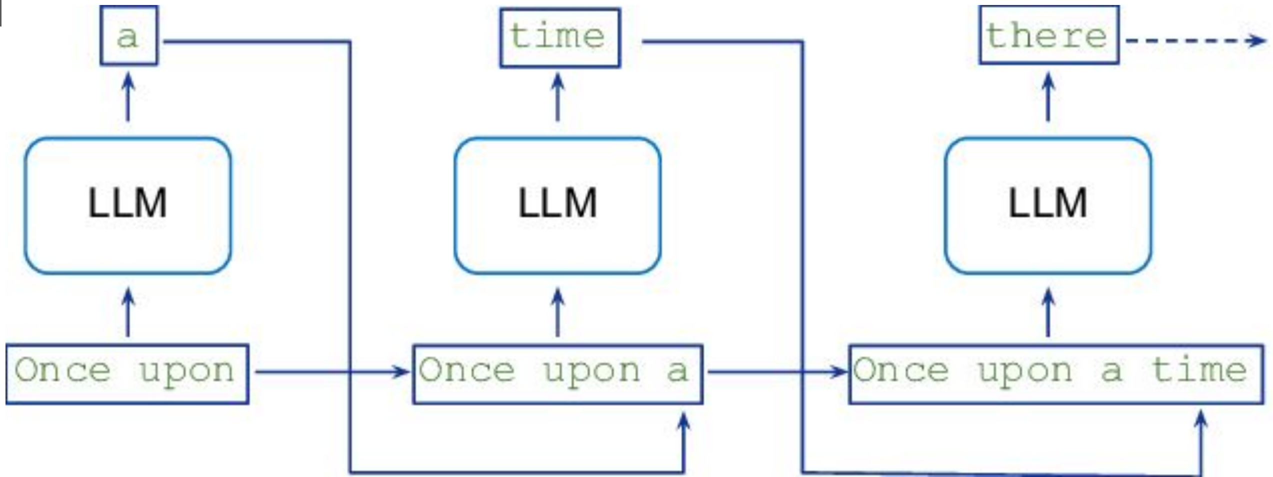
Fast Inference from Transformers via Speculative Decoding

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

Presenter: Ethan Morton

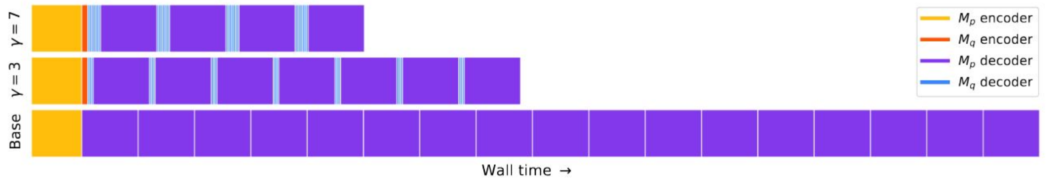
Motivation

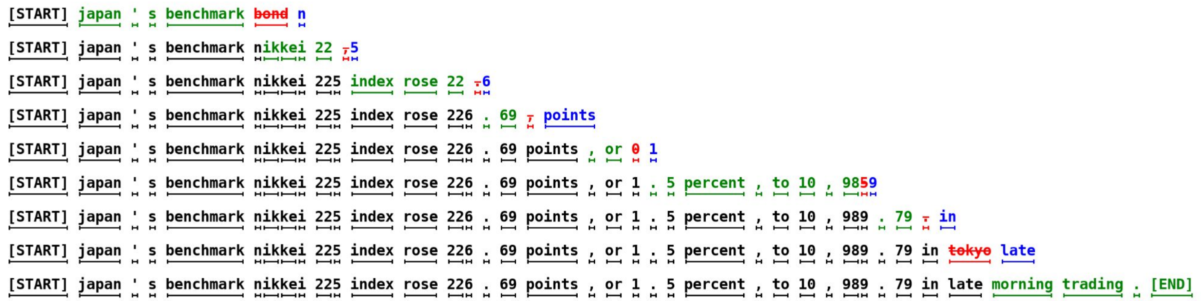
- Autoregressive models are inherently sequential
- Each token requires a full forward pass
- For large model sizes, this is a HUGE bottleneck



Speculative Decoding

- Uses a smaller "draft model" (M_q) to generate a sequence of draft tokens
- A larger "target model" (M_p) to verify and change incorrect tokens





[START] japan ' s benchmark ~~bond~~ n

[START] japan ' s benchmark nikkei 22 ~~,~~5

[START] japan ' s benchmark nikkei 225 index rose 22 ~~,~~6

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 ~~,~~ points

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 0 1

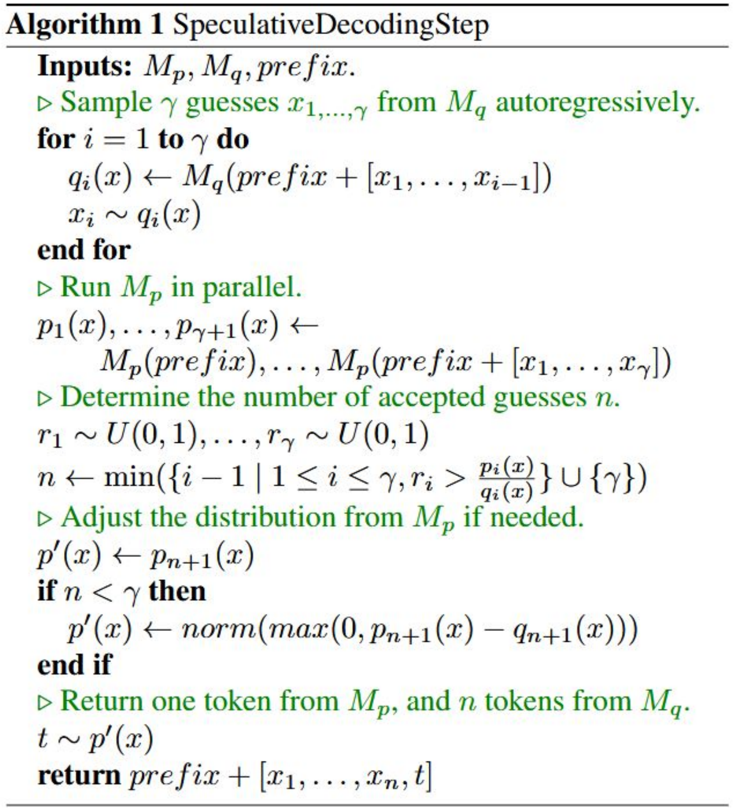
[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 9859

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 ~~,~~ in

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in ~~tokyo~~ late

[START] japan ' s benchmark nikkei 225 index rose 226 . 69 points , or 1 . 5 percent , to 10 , 989 . 79 in late morning trading . [END]

Decoding Loop



Algorithm 1 SpeculativeDecodingStep

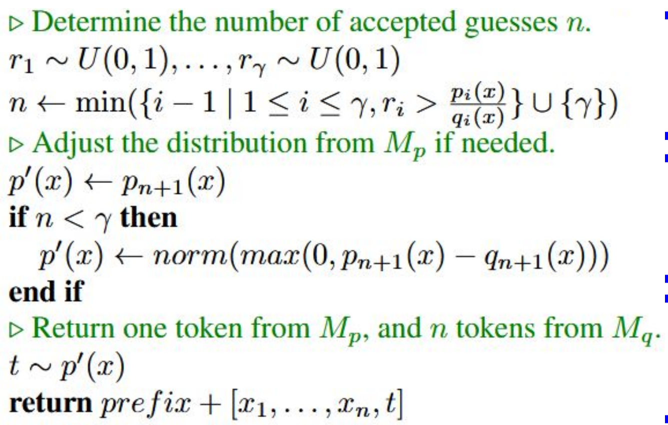
```
Inputs: $M_p, M_q, prefix$.
▷ Sample $\gamma$ guesses $x_{1,\ldots,\gamma}$ from $M_q$ autoregressively.
for $i = 1$ to $\gamma$ do
    $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$
    $x_i \sim q_i(x)$
end for
▷ Run $M_p$ in parallel.
$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$
        $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$
▷ 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 $prefix + [x_1, \ldots, x_n, t]$
```

Speculative generation

Target Verification Pass

Token Comparison/Verification

Speculative Sampling



```
▷ Determine the number of accepted guesses $n$.
$r_1 \sim U(0,1), \dots, r_\gamma \sim U(0,1)$
$n \leftarrow \min(\{i-1 \mid 1 \le i \le \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$
▷ 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 $prefix + [x_1, \dots, x_n, t]$
```

- Token is accepted if p(x)/q(x) is less than a uniformly sampled threshold
- Generated sequence is adjusted. If there’s a rejection, “sync” the models by renormalizing
- New sequence is returned with n accepted draft tokens and one target generated token

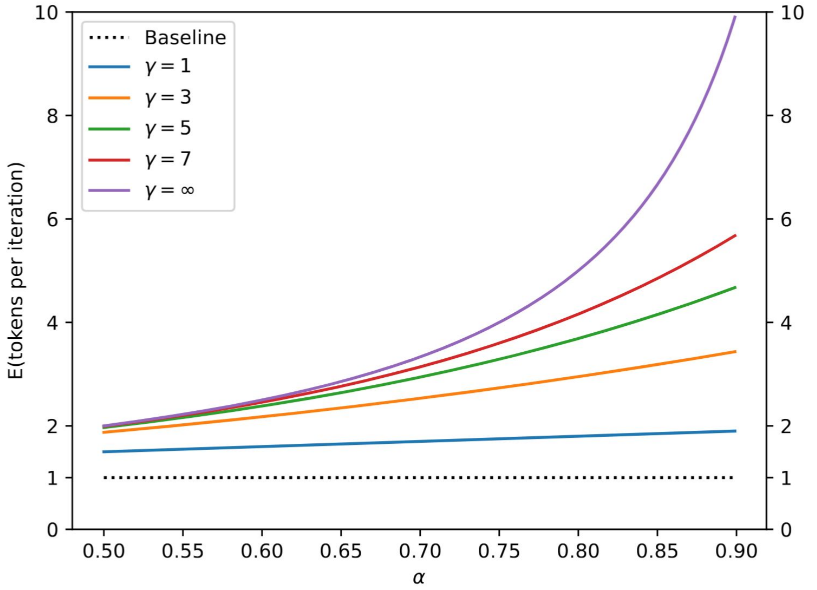
Metrics - Acceptance Length

$$E(\#\ generated\ tokens) = \frac{1-\alpha^{\gamma+1}}{1-\alpha}$$

$$\alpha = E(\beta)$$

β = probability of a draft token to be accepted

γ = draft sequence length



Metrics - Speedup and Ops

Theorem 3.8. *The expected improvement factor in total walltime by Algorithm* $\boxed{1}$ *is* $\frac{1-\alpha^{\gamma+1}}{(1-\alpha)(\gamma c+1)}$.

Theorem 3.11. *The expected factor of increase in the number of total operations of Algorithm* $\boxed{1}$ *is* $\frac{(1-\alpha)(\gamma\hat{c}+\gamma+1)}{1-\alpha^{\gamma+1}}$.

c = cost of running M_q

α = average rate of draft token acceptance

γ = draft sequence length

$\hat{c}$ = ratio of arithmetic operations for M_q to M_p

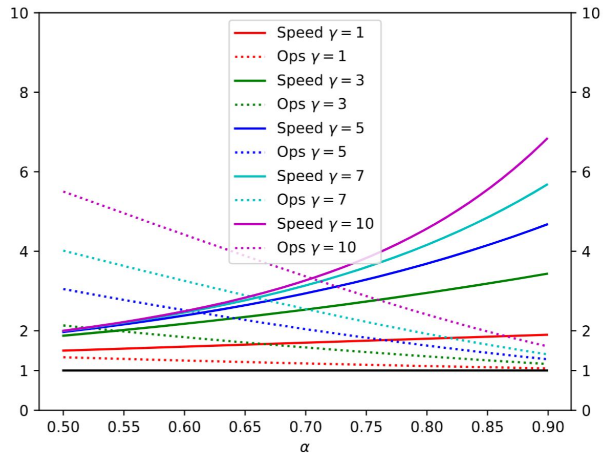


Figure 4. The speedup factor and the increase in number of arithmetic operations as a function of α for various values of γ.

Results - Numerical Predictions

Speedup $= \frac{1-\alpha^{\gamma+1}}{(1-\alpha)(\gamma c+1)}$

Operations $= \frac{(1-\alpha)(\gamma\hat{c}+\gamma+1)}{1-\alpha^{\gamma+1}}$

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

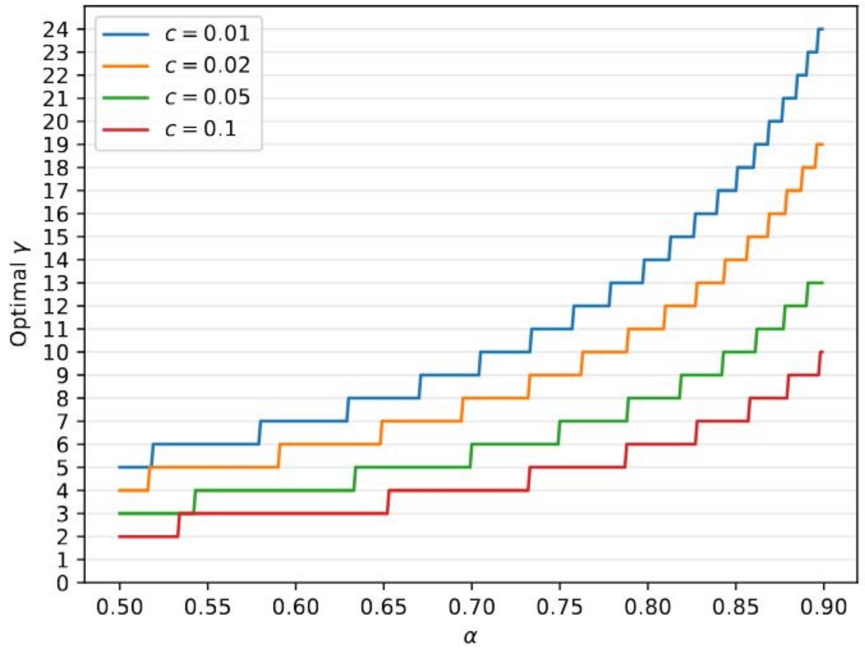


Figure 3. The optimal γ as a function of α for various values of c.

Results

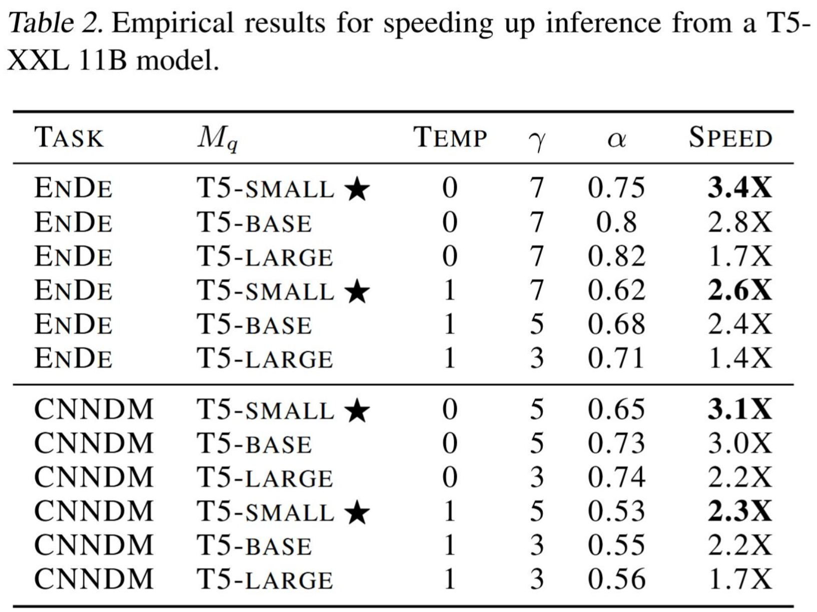


Table 2. Empirical results for speeding up inference from a T5-XXL 11B model.

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

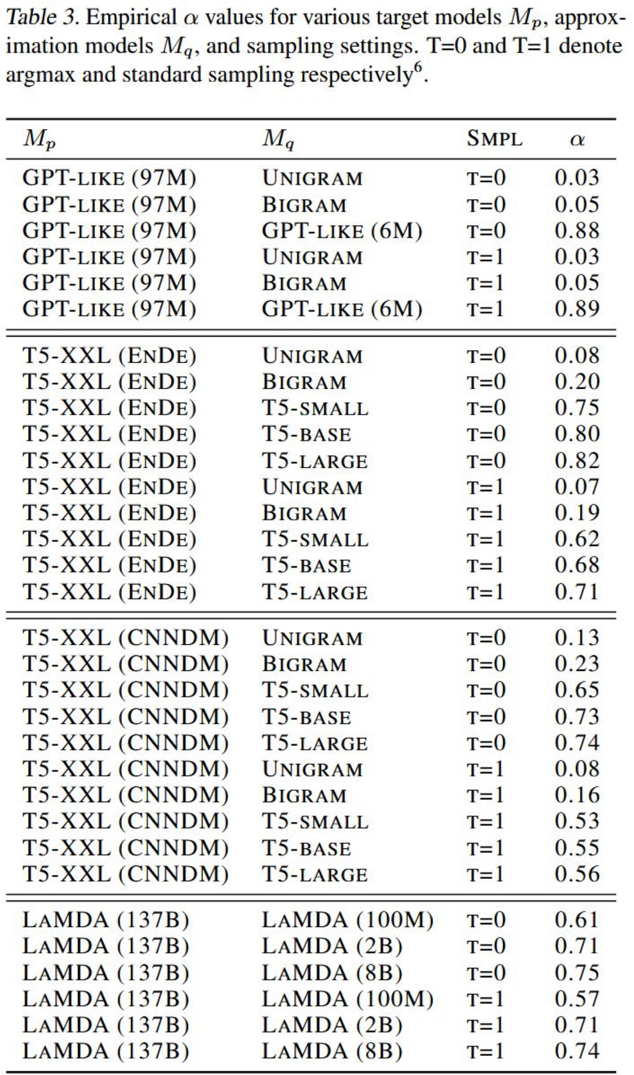


Table 3. Empirical α values for various target models M_p, approximation models M_q, and sampling settings. T=0 and T=1 denote argmax and standard sampling respectively[6].

M_p	M_q	Smpl	α
GPT-like (97M)	Unigram	T=0	0.03
GPT-like (97M)	Bigram	T=0	0.05
GPT-like (97M)	GPT-like (6M)	T=0	0.88
GPT-like (97M)	Unigram	T=1	0.03
GPT-like (97M)	Bigram	T=1	0.05
GPT-like (97M)	GPT-like (6M)	T=1	0.89
T5-XXL (EnDe)	Unigram	T=0	0.08
T5-XXL (EnDe)	Bigram	T=0	0.20
T5-XXL (EnDe)	T5-small	T=0	0.75
T5-XXL (EnDe)	T5-base	T=0	0.80
T5-XXL (EnDe)	T5-large	T=0	0.82
T5-XXL (EnDe)	Unigram	T=1	0.07
T5-XXL (EnDe)	Bigram	T=1	0.19
T5-XXL (EnDe)	T5-small	T=1	0.62
T5-XXL (EnDe)	T5-base	T=1	0.68
T5-XXL (EnDe)	T5-large	T=1	0.71
T5-XXL (CNNDM)	Unigram	T=0	0.13
T5-XXL (CNNDM)	Bigram	T=0	0.23
T5-XXL (CNNDM)	T5-small	T=0	0.65
T5-XXL (CNNDM)	T5-base	T=0	0.73
T5-XXL (CNNDM)	T5-large	T=0	0.74
T5-XXL (CNNDM)	Unigram	T=1	0.08
T5-XXL (CNNDM)	Bigram	T=1	0.16
T5-XXL (CNNDM)	T5-small	T=1	0.53
T5-XXL (CNNDM)	T5-base	T=1	0.55
T5-XXL (CNNDM)	T5-large	T=1	0.56
LaMDA (137B)	LaMDA (100M)	T=0	0.61
LaMDA (137B)	LaMDA (2B)	T=0	0.71
LaMDA (137B)	LaMDA (8B)	T=0	0.75
LaMDA (137B)	LaMDA (100M)	T=1	0.57
LaMDA (137B)	LaMDA (2B)	T=1	0.71
LaMDA (137B)	LaMDA (8B)	T=1	0.74

- Alpha increases with model size
- Speedup is greater for small models
- Higher acceptance/speedup for argmax

Q/A

MEDUSA: Simple LLM Inference Acceleration Framework w/ Multiple Decoding Heads

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, Tri Dao

ICML 2024

Presenter: Qiyang Li

Executive summary

- Problem: autoregressive LLM inference (batch size 1) is memory-bandwidth bound because each output token requires streaming the full model parameters repeatedly.
- Key idea: attach K lightweight decoding heads to the last hidden state to predict multiple future tokens in parallel, form multiple candidate continuations, and verify them jointly via a tree-structured attention mask (no separate draft model).
- Two training modes: MEDUSA-1 (freeze backbone, train only heads) for lossless integration; MEDUSA-2 (joint fine-tuning with a training recipe) for higher head accuracy and larger speedups.
- Results: on Vicuna family, MEDUSA-1 yields ~2.2x speedups; MEDUSA-2 yields up to ~2.8x while maintaining generation quality on MT-Bench.

Background and motivation

- Bottleneck: repeated HBM $\longrightarrow$ accelerator cache transfers dominate latency in single-stream decoding, under-utilizing arithmetic units.
- Speculative decoding uses a separate draft model to generate continuations; practical issues: obtaining/serving a compatible draft, distribution mismatch, extra serving complexity.
- Goal: increase arithmetic intensity per HBM transfer and reduce number of sequential decoding steps with minimal serving complexity and small parameter overhead.

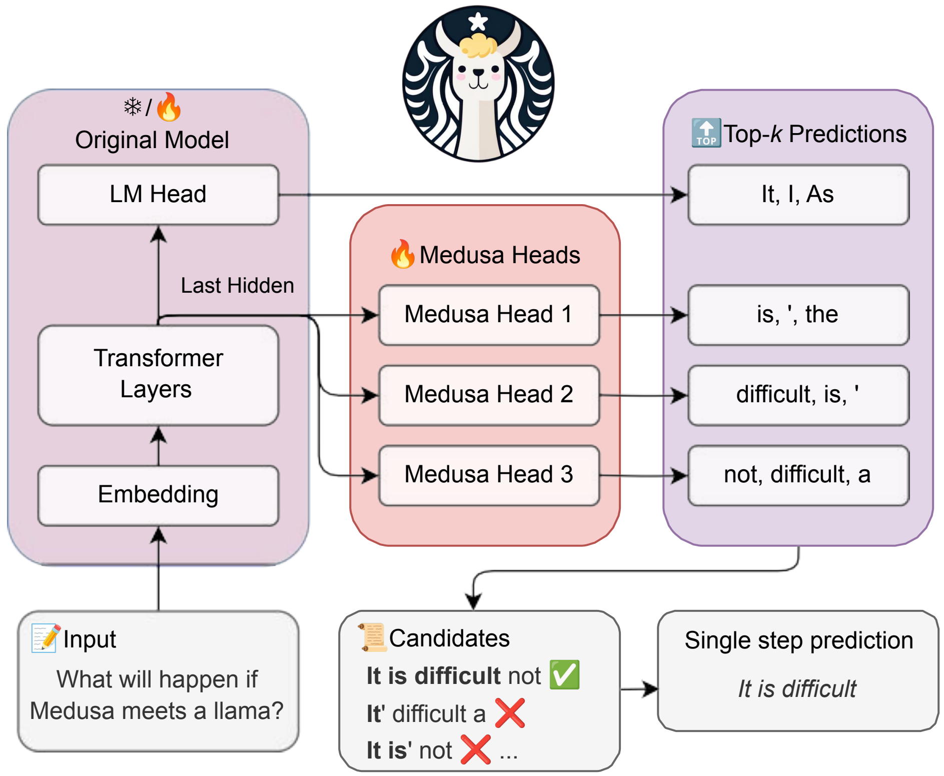


Figure: MEDUSA pipeline: attach extra decoding heads to last hidden states; sample top candidates per head; build candidate continuations; process them in parallel via tree attention; accept the longest verified prefix (rejection sampling or typical acceptance).

MEDUSA heads: architecture and initialization

$$h_t \in \mathbb{R}^d \quad \text{(last hidden state at position } t\text{)}$$

$$\tilde{h}_t^{(k)} = \mathrm{SiLU}\big(W_1^{(k)} h_t\big) + h_t, \quad W_1^{(k)} \in \mathbb{R}^{d\times d}$$

$$p_t^{(k)} = \mathrm{softmax}\big(W_2^{(k)} \tilde{h}_t^{(k)}\big), \quad W_2^{(k)} \in \mathbb{R}^{V\times d}$$

Notes:

- Each head is a single feed-forward layer with residual (lightweight).
- Initialize $W_2^{(k)}$ to match the backbone LM head and $W_1^{(k)} = 0$ so initial outputs align with the backbone logits.
- Heads can be trained with the backbone frozen (MEDUSA-1) or jointly with the backbone (MEDUSA-2).

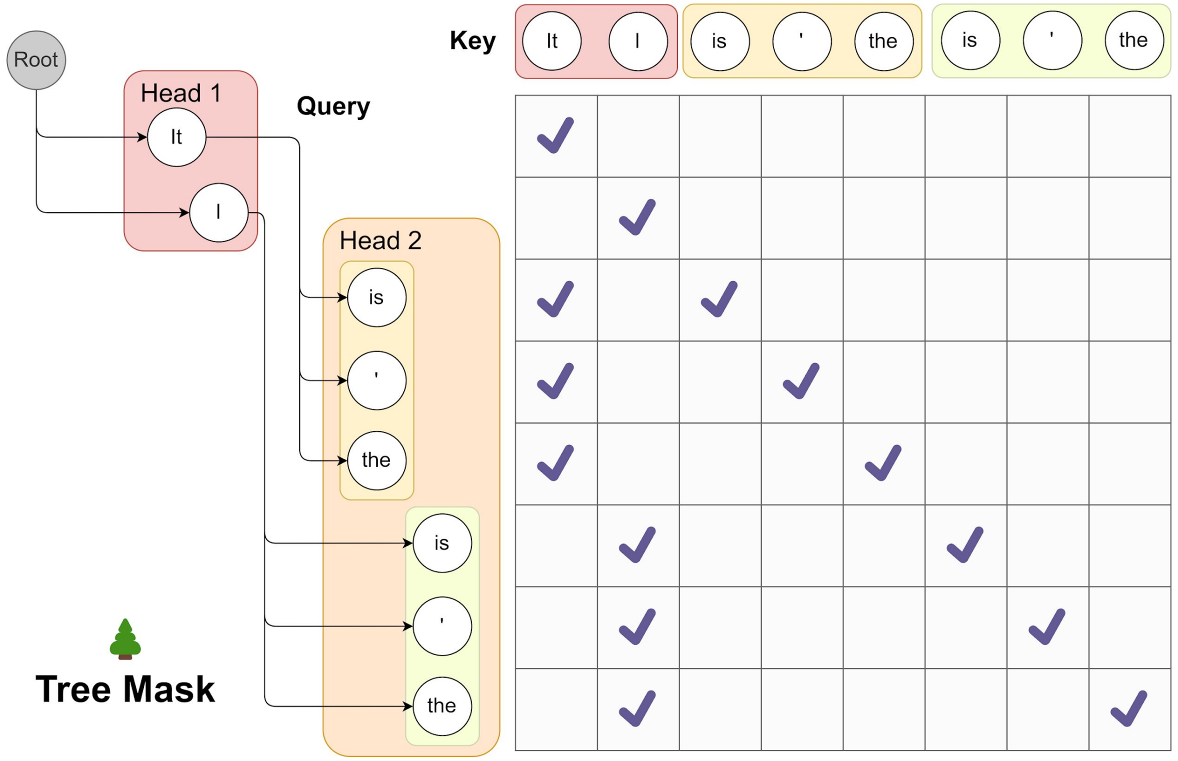


Figure: Tree attention: build candidates from top-s_k tokens per head (Cartesian product). The attention mask enforces that tokens only attend to their predecessors within the same candidate branch; positional indices are adjusted accordingly.

Tree attention: construction and complexity

- Candidate construction: for head k take top s_k tokens; leaf candidates $= \prod_{k=1}^{K} s_k$ (Cartesian product).
- Process candidates in a single forward by arranging tokens as tree nodes with a causal mask that permits attention only along each branch's prefix.
- New tokens computed in one forward: $\sum_{k=1}^{K} \prod_{i=1}^{k} s_i$, increasing arithmetic intensity per HBM transfer.
- Trade-off: larger trees increase expected accepted length but also add compute/attention overhead; choose tree sparsity judiciously.

Training strategies overview

- MEDUSA-1 (frozen backbone): train only the new heads with cross-entropy to ground-truth tokens at offsets. Low memory; compatible with quantized backbone.
- MEDUSA-2 (joint training): jointly train backbone and heads with a combined loss and training recipe to preserve original LM quality and improve head accuracy.
- Self-distillation: if original SFT data is unavailable (or for RLHF models), generate a dataset by sampling the target model and distill using LoRA adapters to avoid hosting two full models.

MEDUSA-1 loss (frozen backbone)

$$\mathcal{L}_{\text{MEDUSA-1}} = \sum_{k=1}^{K} -\lambda_k \log p_t^{(k)}\left(y_{t+k+1}\right)$$

Practical tips:

- Use decay weights λ_k (for example $\lambda_k = c^k$ with $c \approx 0.8$) to downweight farther predictions.
- With a quantized backbone you can train heads on a single consumer GPU (few hours for typical datasets).

Medusa-2 joint training recipe

$$\mathcal{L}_{\text{Medusa-2}} = \mathcal{L}_{\text{LM}} + \lambda_0 \mathcal{L}_{\text{Medusa-1}}, \qquad \mathcal{L}_{\text{LM}} = -\log p_t^{(0)}(y_{t+1})$$

Key techniques:

- Combined loss keeps backbone next-token capability.
- Differential learning rates: smaller LR for backbone, larger LR for heads.
- Heads warmup: two-stage schedule (train heads only, then joint fine-tune with small backbone LR or gradually increasing weight).

Typical acceptance: quick verification (entropy adaptive)

$$\text{Accept } x_{n+k} \text{ if } p_{\text{orig}}(x_{n+k} \mid x_{\leq n+k-1}) > \min\left(\epsilon,\ \delta e^{-H\left(p_{\text{orig}}(\cdot|x_{\leq n+k-1})\right)}\right)$$

- Idea: accept tokens that are *not extremely improbable* under the original model instead of full distribution matching (rejection sampling).
- Greedily accept first token to guarantee progress; among candidates accept the longest prefix satisfying the threshold.
- Empirically yields higher throughput than rejection sampling while keeping generation quality similar.

Self-distillation workflow (when SFT data unavailable)

- Generate a dataset: use public seed prompts (e.g., ShareGPT, UltraChat) and sample the target model (chosen temperature) to produce multi-turn examples.
- Distill backbone behavior using KL between teacher logits (original model) and student logits; to avoid storing two full models, fine-tune with LoRA adapters so teacher = backbone with adapters off.
- Use the generated dataset to train heads (and optionally joint-distill backbone) with minimal extra memory.

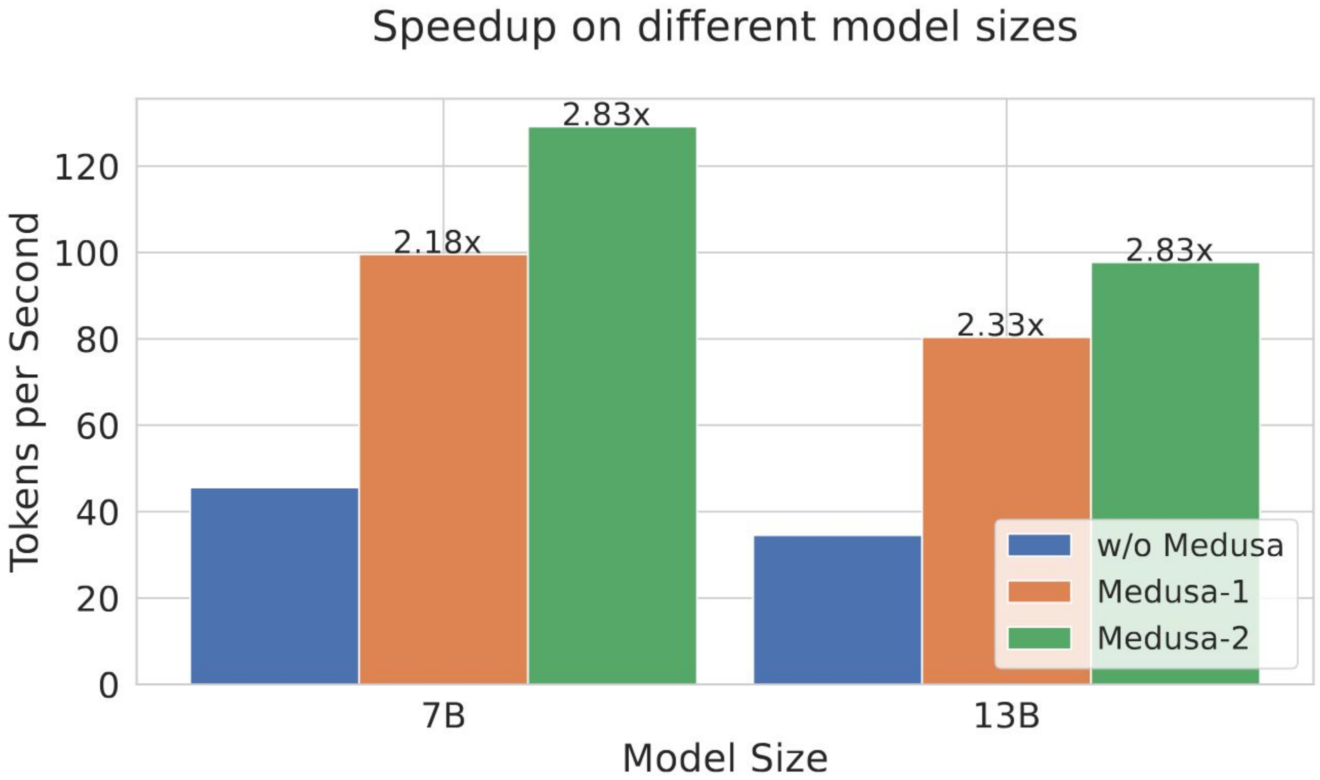


Figure: Tokens/sec comparison: baseline vs. MEDUSA-1 vs. MEDUSA-2 on Vicuna-7B and Vicuna-13B. MEDUSA-1 achieves ~2.2x; MEDUSA-2 up to ~2.8x.

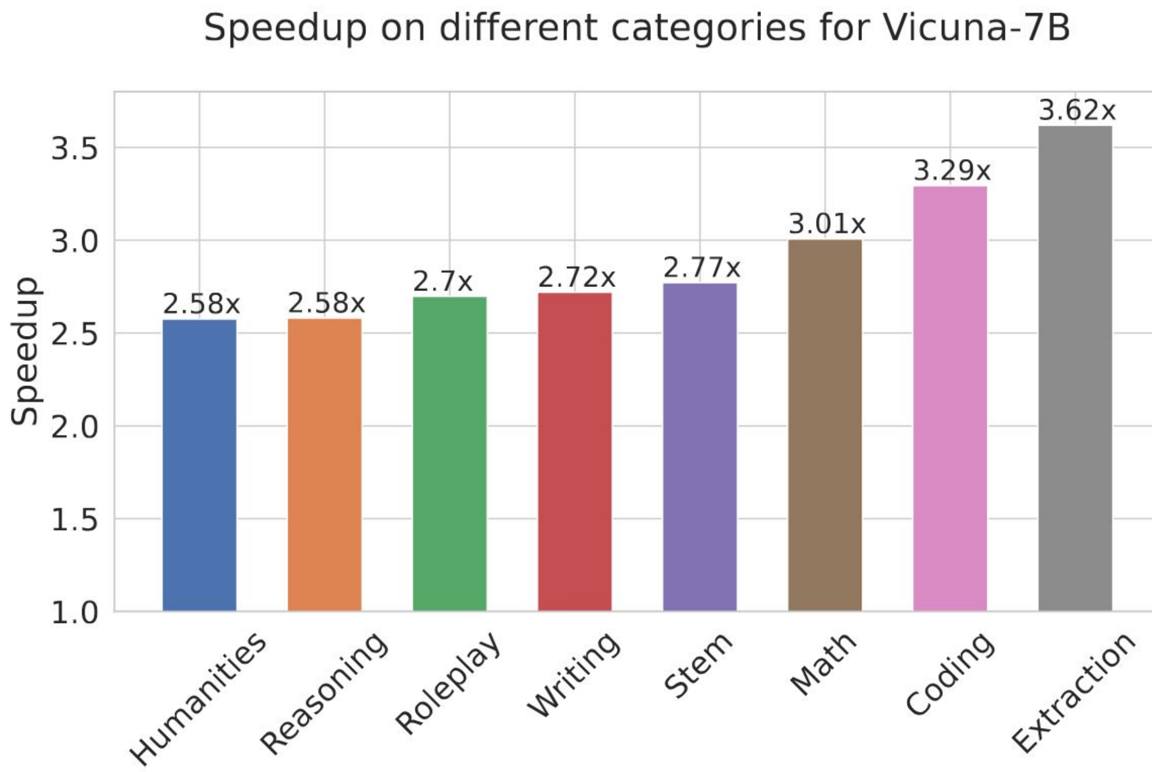


Figure: Detailed speedups per category (Vicuna-7B, MEDUSA-2): some categories like coding and extraction benefit especially (3x+).

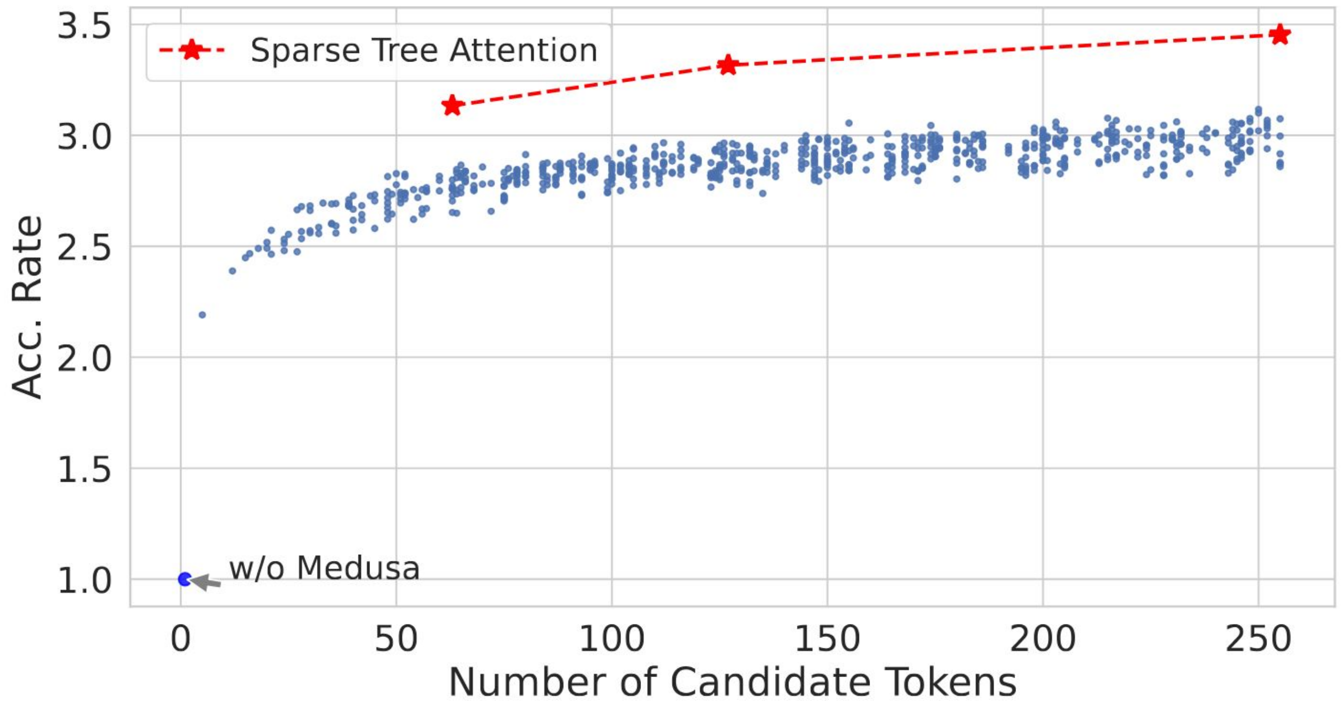


Figure: Acceleration rate for random dense trees (blue dots) vs optimized sparse trees (red stars). Optimized sparse trees (with same node budget) can give higher acceptance length.

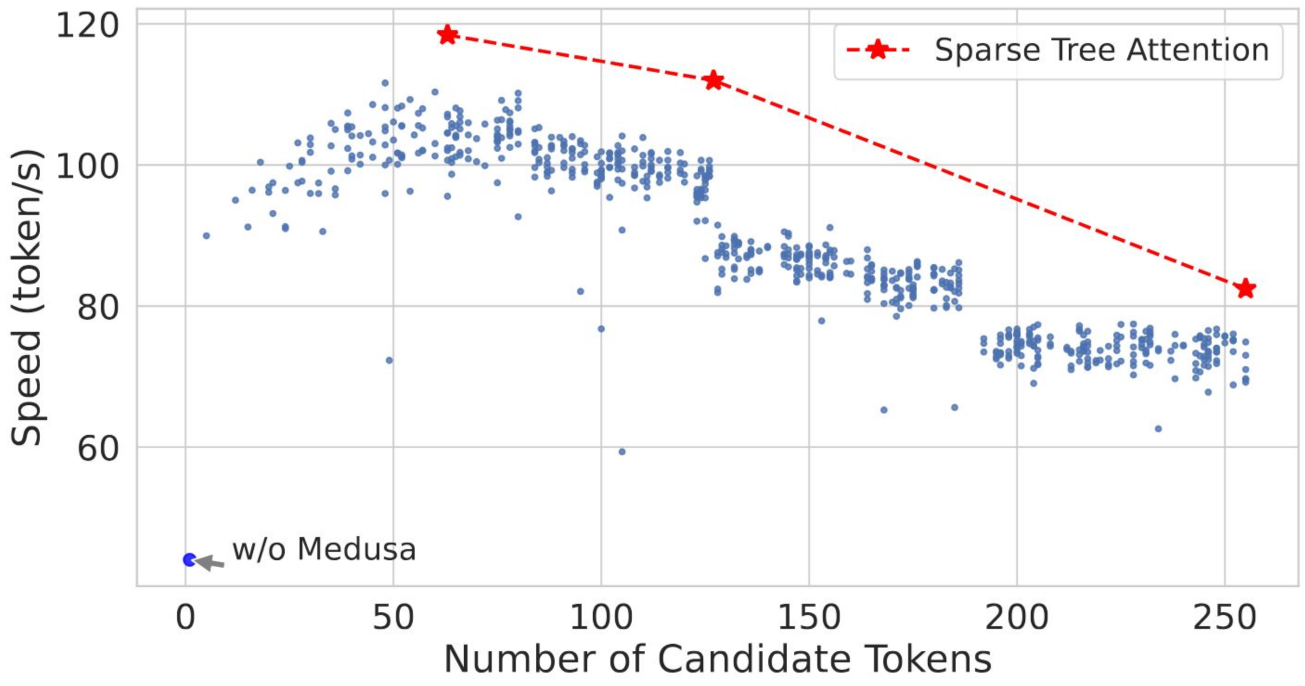


Figure: Raw tokens/sec as tree size increases. Larger trees increase compute and may reduce raw speed; choose tree config to balance expected accepted length and compute overhead.

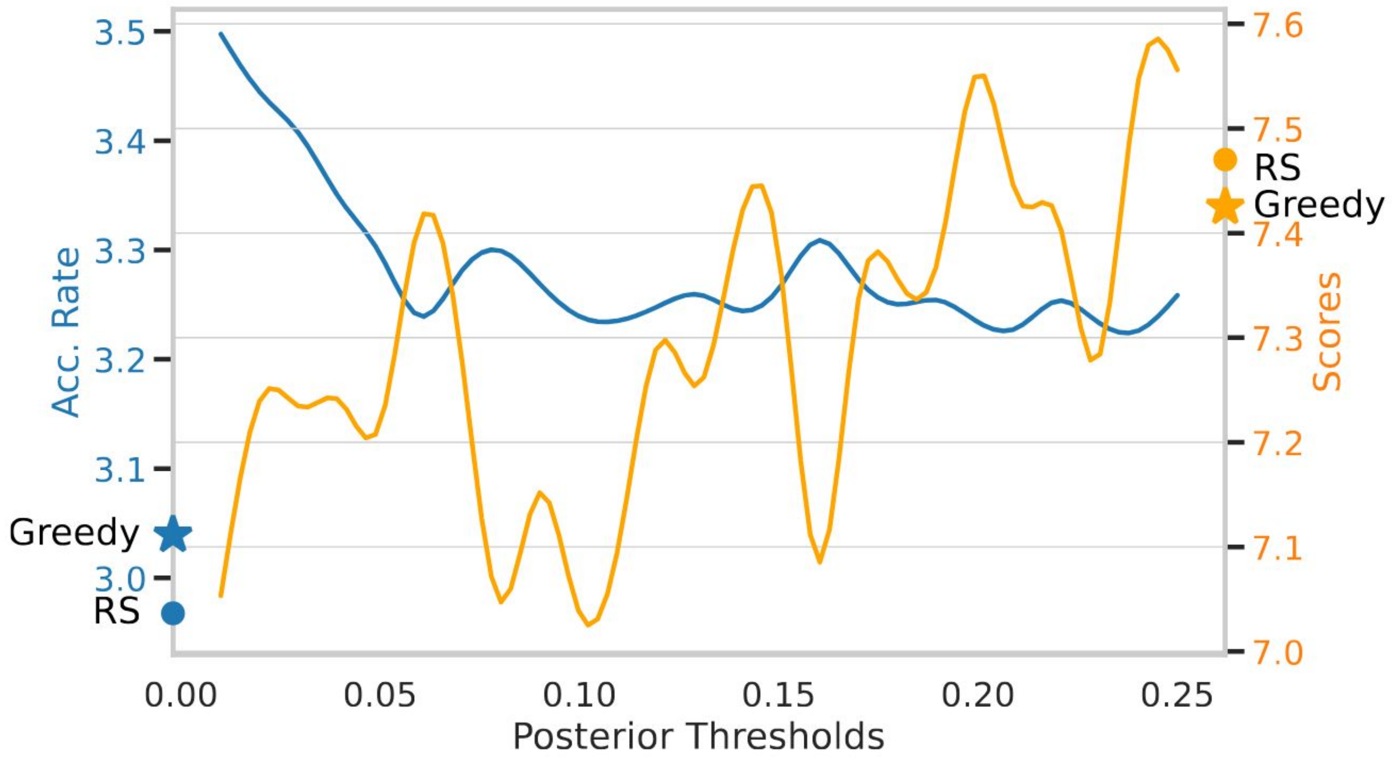


Figure: Typical acceptance ablation: trade-off between threshold and acceleration/quality (writing + roleplay tasks). Higher threshold tends to improve quality at some cost to acceleration.

Ablations and practical results (summary)

- Practical pipeline impact (rough numbers): heads-only (no tree) ~1.5x; add tree attention ~1.9x; optimized sparse tree ~2.2x; full MEDUSA-2 joint training ~2.8x.
- MEDUSA-1 is easy to deploy and preserves original model quality; MEDUSA-2 requires care (warmup, joint loss) but yields larger speedups.
- Typical acceptance accelerates decoding in sampling regimes (temperature > 0) more effectively than rejection sampling in practice.

Limitations and future work

- Limitations: tree attention increases compute per step and can become compute-bound; hyperparameters (K, s_k, thresholds) need tuning per model/task.
- Future work: hardware-aware scheduling, automatic tree construction using per-prediction calibration, extension to larger batch sizes and distributed multi-GPU serving.

Concluding remarks

- MEDUSA offers a simple, parameter-efficient way to accelerate single-stream LLM inference by predicting multiple future tokens and verifying them jointly.
- Two training modes enable both lossless integration (MEDUSA-1) and maximal speedup with preserved quality (MEDUSA-2).
- Empirical results show consistent ~2.2–2.8x speedups on Vicuna and related models with preserved MT-Bench scores.

Q/A

EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty

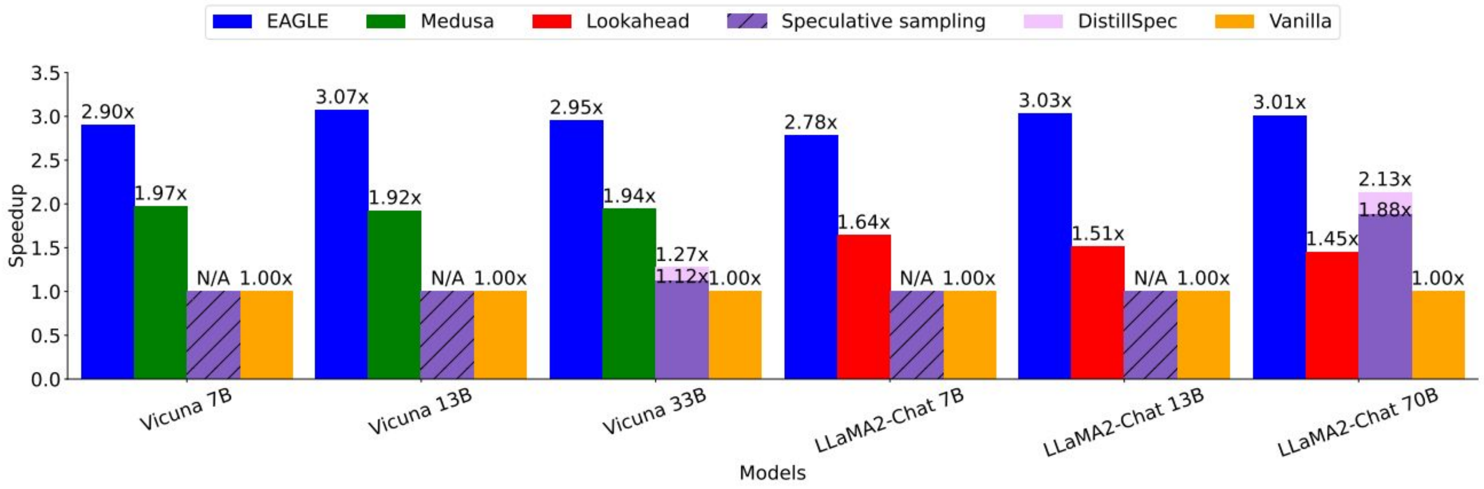
Yuhui Li[♠] Fangyun Wei[‡] Chao Zhang[♠] Hongyang Zhang[♣][†]
[♠]Peking University [‡]Microsoft Research [♣]University of Waterloo [†]Vector Institute
hongyang.zhang@uwaterloo.ca
https://github.com/SafeAILab/EAGLE

Presenter: Lian Lian

Motivation

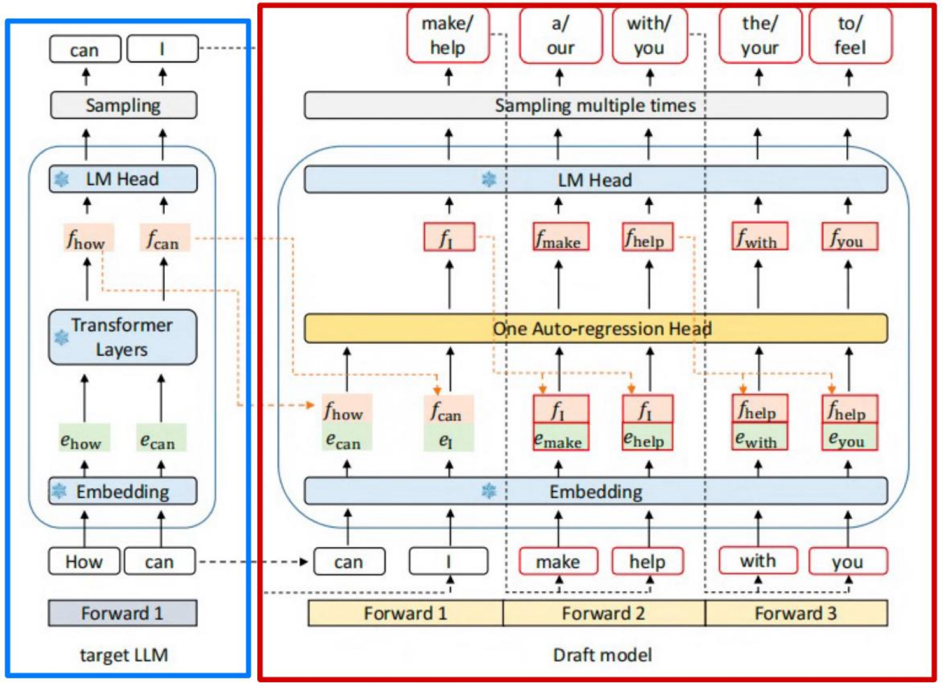
Points:

- Large Language Models (LLMs) → slow inference
- Autoregressive decoding = one token per step
- Latency is critical in dialogue, code, math reasoning



Speculative Sampling Recap

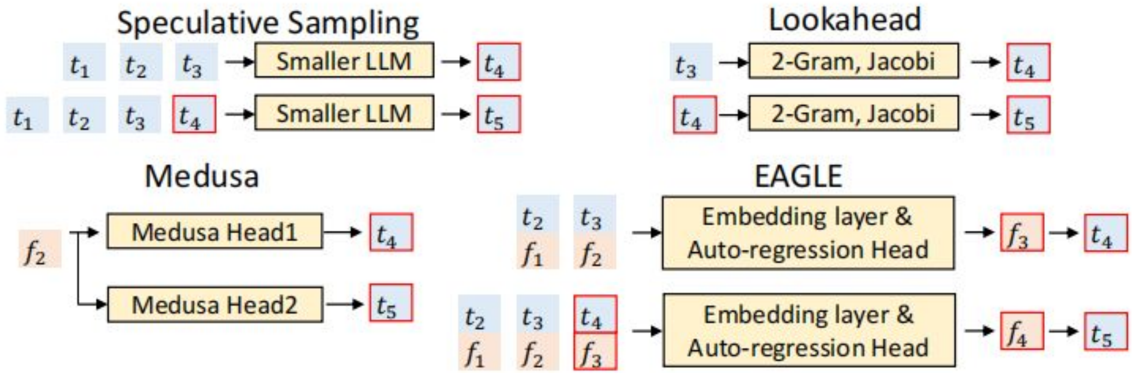
- The **draft model** quickly proposes several candidate tokens in advance, rather than generating them one by one.
- The **target LLM** then evaluates all these drafted tokens in a single forward pass, accepting valid ones and rejecting the rest.
- Overall efficiency depends critically on the **acceptance rate** — the fraction of drafted tokens that survive validation, which determines how many tokens can be generated per step.



Computational process

Challenge of Existing Methods

- Tokens level: discrete, irregular
- Draft often **misaligned** with target
- Prior art: Medusa / Lookahead draft at token level→ limited speedup



Speculative Sampling
t_1 t_2 t_3 → Smaller LLM → t_4
t_1 t_2 t_3 t_4 → Smaller LLM → t_5
Lookahead
t_3 → 2-Gram, Jacobi → t_4
t_4 → 2-Gram, Jacobi → t_5
Medusa
f_2
Medusa Head1 → t_4
Medusa Head2 → t_5
EAGLE
t_2 t_3 f_1 f_2 → Embedding layer & Auto-regression Head → f_3 → t_4
t_2 t_3 t_4 f_1 f_2 f_3 → Embedding layer & Auto-regression Head → f_4 → t_5

Challenge of Existing Methods

Feature trajectories depend on sampled tokens

For example, after “I”, the next hidden state could correspond to “am” or “always”, leading to very different branches.

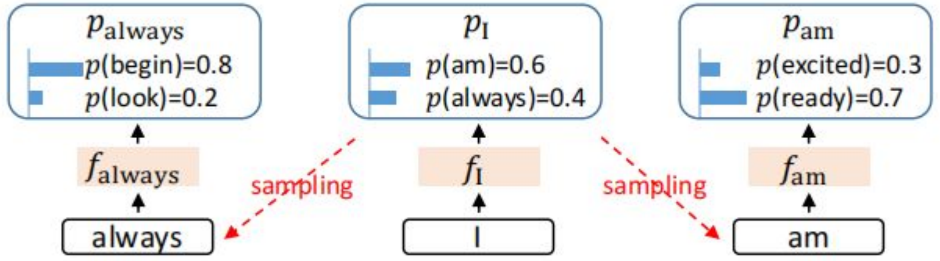


Figure 3: Uncertainty in feature sequences. The next feature following f_I is contingent on the sampling outcome and cannot be determined solely based on f_I, where both “always” and “am” are possible to follow the token “I” and lead to two branches.

Key Observations in EAGLE

1. **Feature-level autoregression = smoother, easier**
2. **But features are still uncertain (depend on sampled token)**

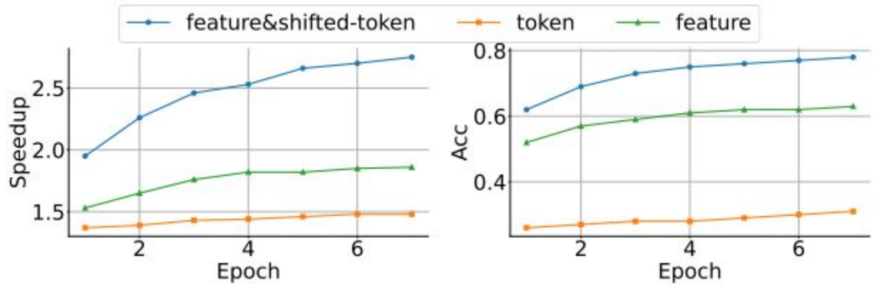
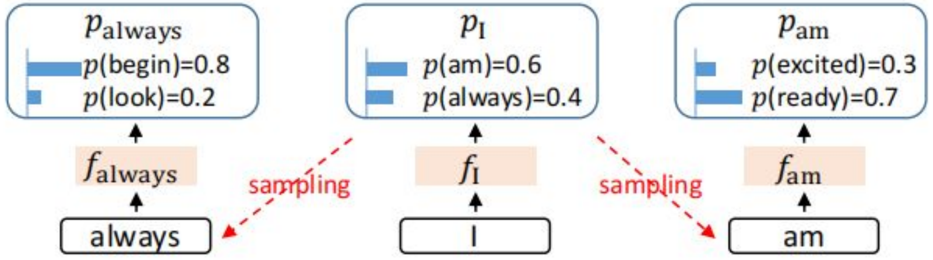


Figure 4: Accuracy and speedup ratio of draft models based on tokens, features and feature&shifted-token at temperature=0, tested on MT-bench with Vicuna 7B as the original LLM. Feature&shifted-token refers to using a feature sequence and a token sequence advanced by one time step as inputs.



Key Observations in EAGLE

Example: “I” → “am” vs “always”

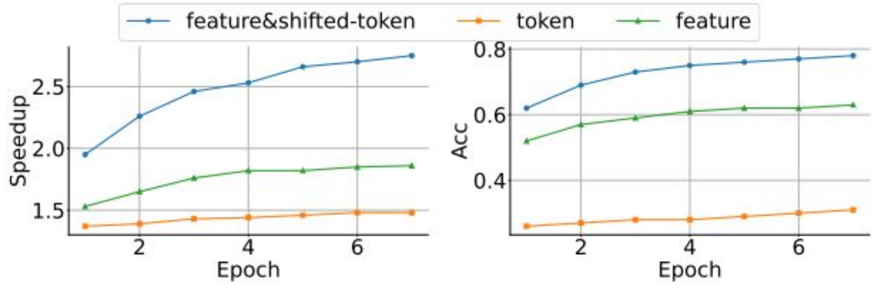
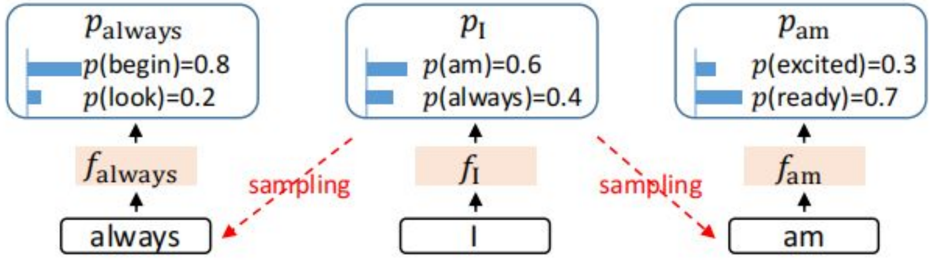
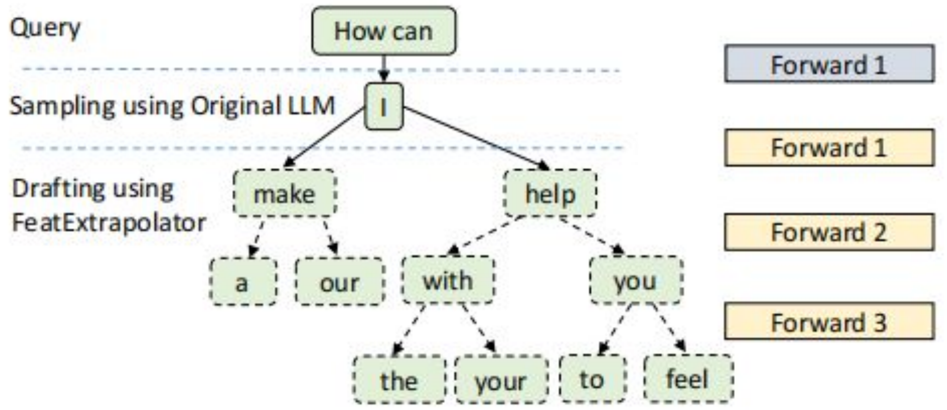


Figure 4: Accuracy and speedup ratio of draft models based on tokens, features and feature&shifted-token at temperature=0, tested on MT-bench with Vicuna 7B as the original LLM. Feature&shifted-token refers to using a feature sequence and a token sequence advanced by one time step as inputs.



EGALE Solution (shifted tokens：token to feature level)

1. **Add one-step-ahead token into feature prediction**
2. **Removes uncertainty → higher accuracy**
3. **Draft features align better with target LLM**
4.
5.



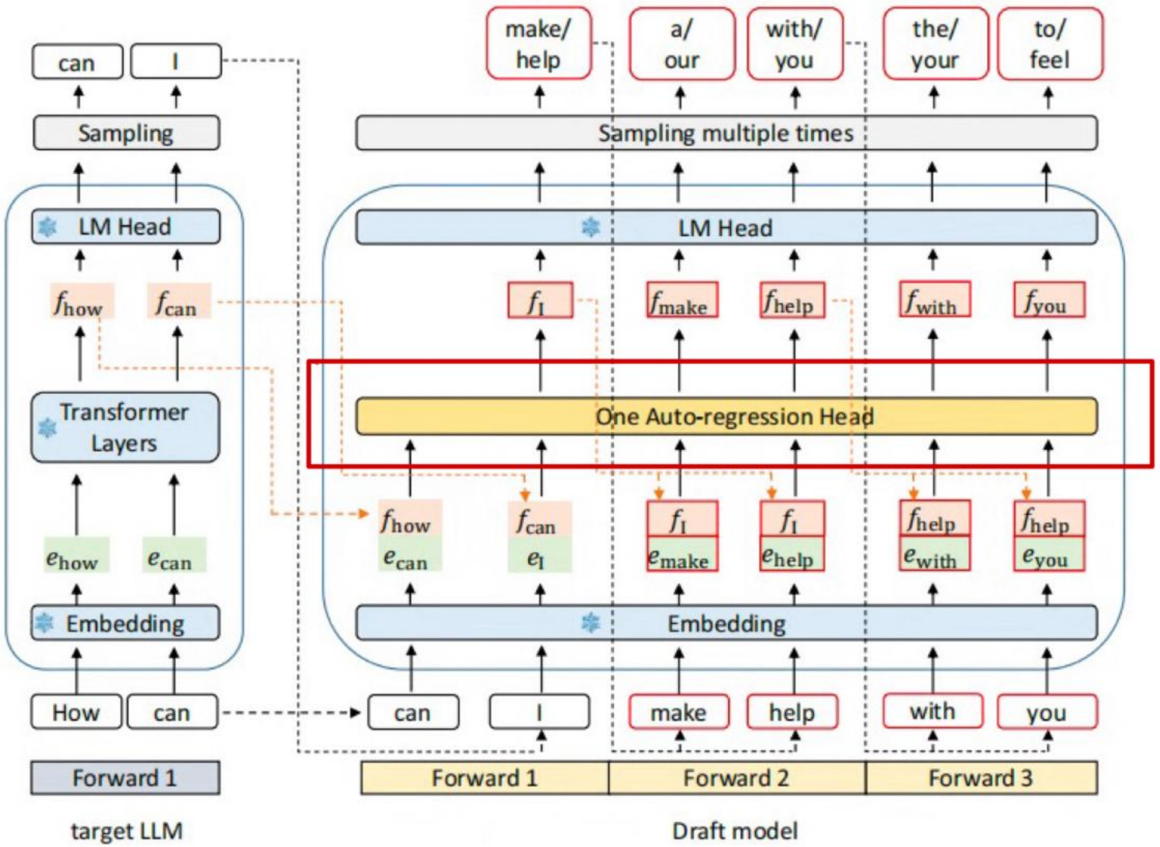
Corresponding generation results for each step

Draft Model Architecture

green blocks represent token embeddings,

orange blocks represent features, red boxes indicate the predicions of the draft model

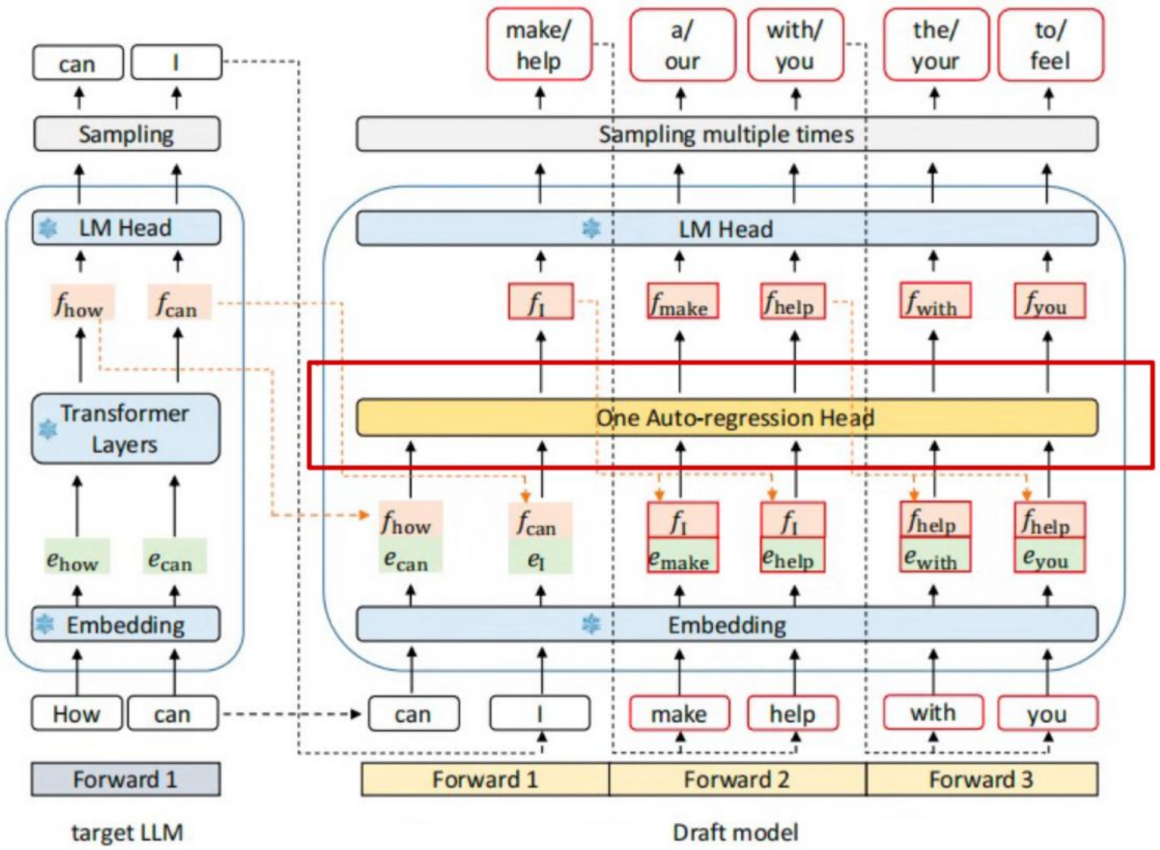
blue modules with snowflake icons represent the use of target LLM parameters, which are not subject to training



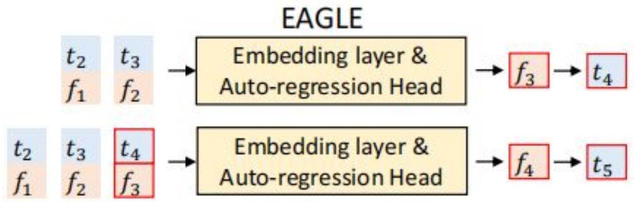
Draft Model Architecture

1. Embedding + LM head (shared)
2. Lightweight autoregression head (trainable)
3. Output: hidden states → mapped to tokens

$$\hat{f}_{t+1} = g(f_t, \text{Embed}(x_{t+1}))$$



Training Objectives



1. SmoothL1 (Feature-level regression)

Regression loss for predicting next hidden feature.

$$L_{reg} = \text{Smooth L1}(f_{i+1}, \text{Draft\_Model}(T_{2:i+1}, F_{1:i})).$$

$$L_{\text{reg}}(d) = \begin{cases} \frac{1}{2\beta} d^2, & |d| < \beta, \\ |d| - \frac{\beta}{2}, & \text{otherwise,} \end{cases} \qquad d = \hat{h}_{t+1} - h_{t+1}$$

2. Cross-Entropy- (token-level classification)

Ensures predicted token distribution matches the ground truth.

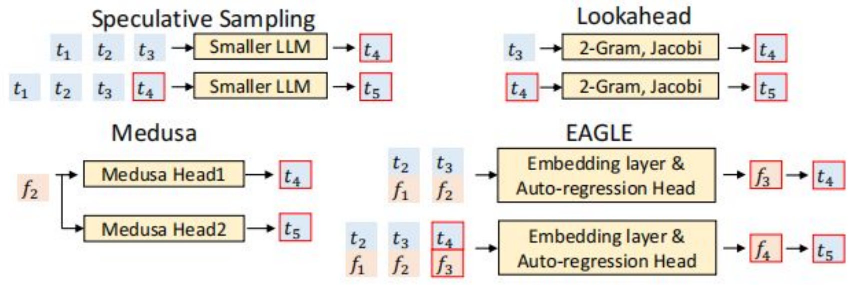
$$p_{i+2} = \text{Softmax}(\text{LM\_Head}(f_{i+1})),$$
$$\hat{p}_{i+2} = \text{Softmax}(\text{LM\_Head}(\hat{f}_{i+1})),$$
$$L_{cls} = \text{Cross\_Entropy}(p_{i+2}, \hat{p}_{i+2}).$$

Training Objectives

Combined Objective

$$L = L_{\text{reg}} + w_{\text{cls}} L_{\text{cls}}$$

- Lreg: Smooth L1 for feature-level Autoregression.
- Lcls: Cross-Entropy for token-level accuracy.



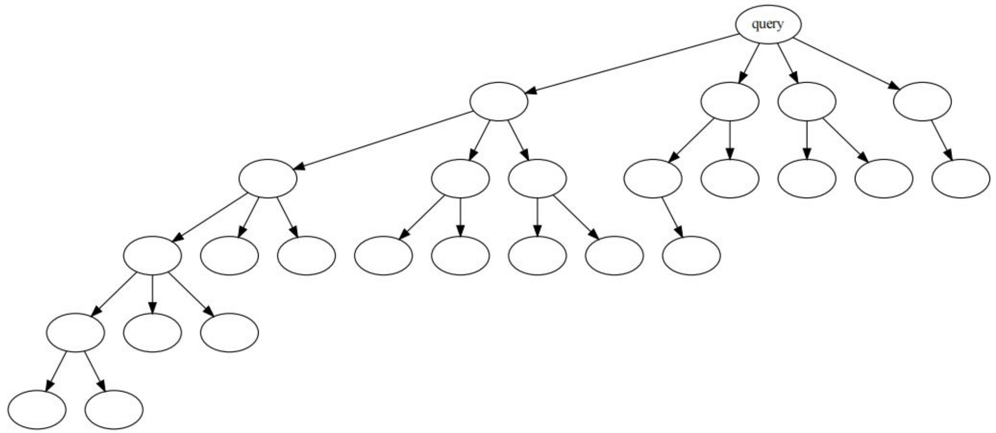
Speculative Sampling
t_1 t_2 t_3 → Smaller LLM → t_4
t_1 t_2 t_3 t_4 → Smaller LLM → t_5
Lookahead
t_3 → 2-Gram, Jacobi → t_4
t_4 → 2-Gram, Jacobi → t_5
Medusa
f_2
Medusa Head1 → t_4
Medusa Head2 → t_5
EAGLE
t_2 t_3 f_1 f_2 → Embedding layer & Auto-regression Head → f_3 → t_4
t_2 t_3 t_4 f_1 f_2 f_3 → Embedding layer & Auto-regression Head → f_4 → t_5

Figure 5: A comparison of the methods for drafting the fourth and fifth tokens, t_4 and t_5. t (represented by blue blocks) denotes tokens, and f (orange blocks) signifies the features, with subscripts indicating their positions in the sequence. The red border indicates the predictions of the draft model. For simplicity, the n in the n-gram for Lookahead, as shown in the figure, has been set to 2.

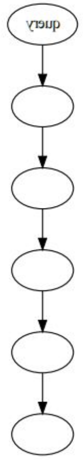
w_{cls} = 0.1 is chosen to balance the two objective

Drafting Process (Tree Attention)

- Vanilla speculative decoding = linear chain
- EAGLE uses **tree attention**, drafting multiple branches per forward pass.
- Increases acceptance length without extra target passes



Draft structure when tree attention is employed



Draft structure without the use of tree attention

Experimental Setup

- Models: Vicuna, LLaMA2-Chat, Mixtral
- Benchmarks: MT-bench, HumanEval, GSM8K, Alpaca
- Tasks: dialogue, code, math, instructions

EAGLE achieves **3–4.5× speedup** at temperature 0 with average acceptance length **3.8–4.5 tokens**. Performance drops under higher temperature (T=1), but still maintains **2–3× acceleration**. 13B models consistently show the best balance of speedup and acceptance

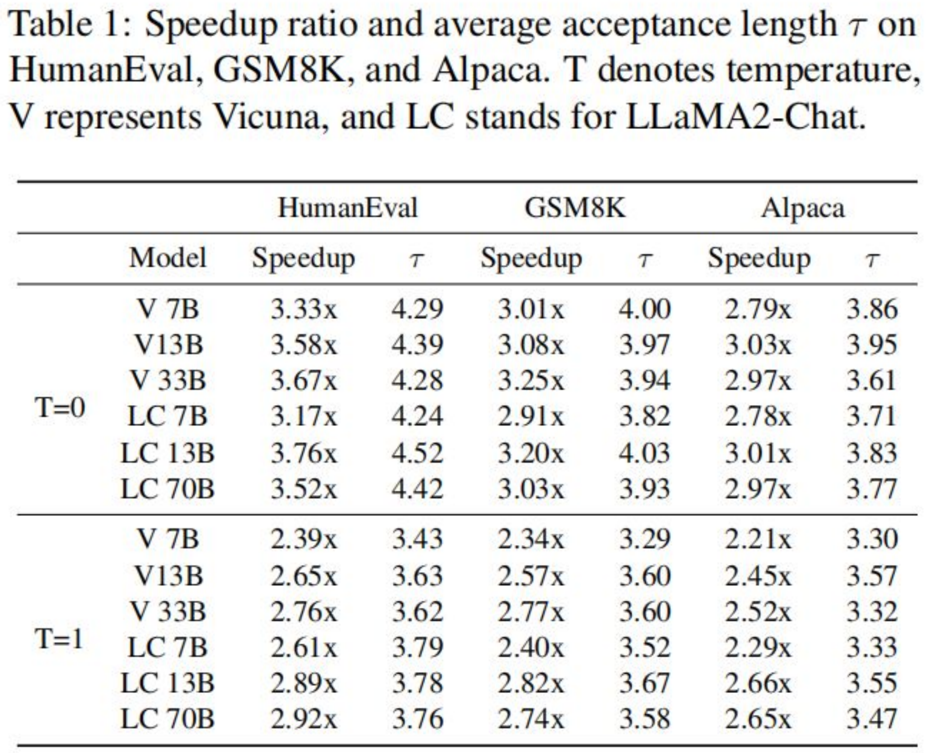


Table 1: Speedup ratio and average acceptance length τ on HumanEval, GSM8K, and Alpaca. T denotes temperature, V represents Vicuna, and LC stands for LLaMA2-Chat.

		HumanEval		GSM8K		Alpaca	
	Model	Speedup	τ	Speedup	τ	Speedup	τ
T=0	V 7B	3.33x	4.29	3.01x	4.00	2.79x	3.86
	V13B	3.58x	4.39	3.08x	3.97	3.03x	3.95
	V 33B	3.67x	4.28	3.25x	3.94	2.97x	3.61
	LC 7B	3.17x	4.24	2.91x	3.82	2.78x	3.71
	LC 13B	3.76x	4.52	3.20x	4.03	3.01x	3.83
	LC 70B	3.52x	4.42	3.03x	3.93	2.97x	3.77
T=1	V 7B	2.39x	3.43	2.34x	3.29	2.21x	3.30
	V13B	2.65x	3.63	2.57x	3.60	2.45x	3.57
	V 33B	2.76x	3.62	2.77x	3.60	2.52x	3.32
	LC 7B	2.61x	3.79	2.40x	3.52	2.29x	3.33
	LC 13B	2.89x	3.78	2.82x	3.67	2.66x	3.55
	LC 70B	2.92x	3.76	2.74x	3.58	2.65x	3.47

Main Results

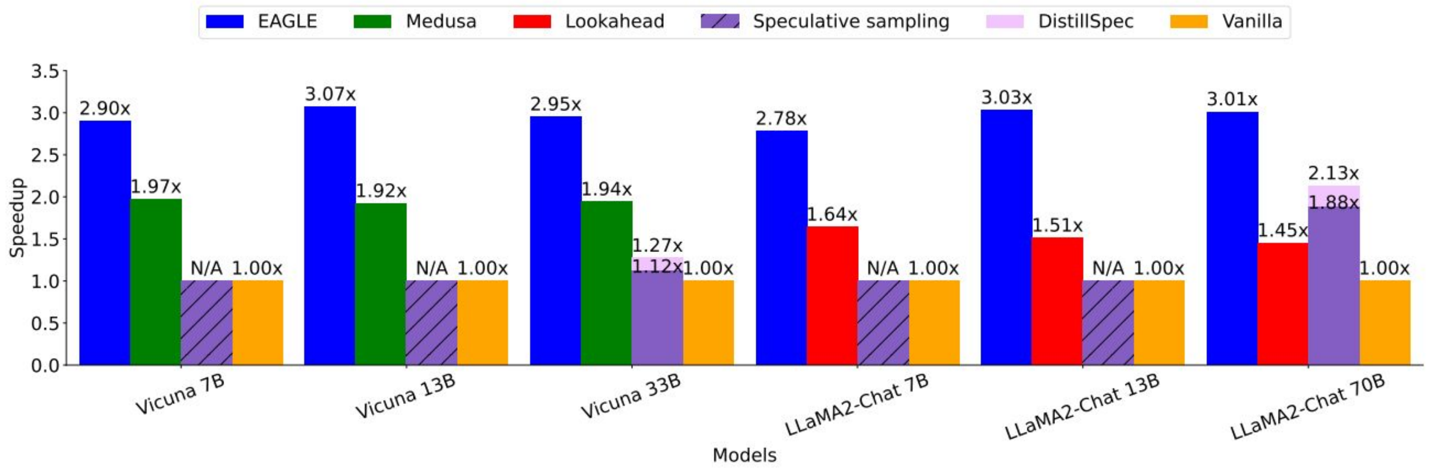


Figure 1: Speedup ratio of Vicuna and LLaMA2-Chat inference latency on the MT-bench for greedy (temperature=0) settings. Speedup ratio of Medusa and Lookahead are copied from their original technical reports. With speculative sampling, there is a lack of suitable draft models to accelerate the 7B model. Employing a 7B model as the draft model for a 13B model results in slow speeds due to the high overhead of the 7B model, rendering it less efficient than vanilla autoregressive decoding. These scenarios are marked as N/A. *In this paper, we only compare with speculative sampling based methods that do not need to finetune the backbone models, ensuring the output text distribution remains constant.*

Main Results

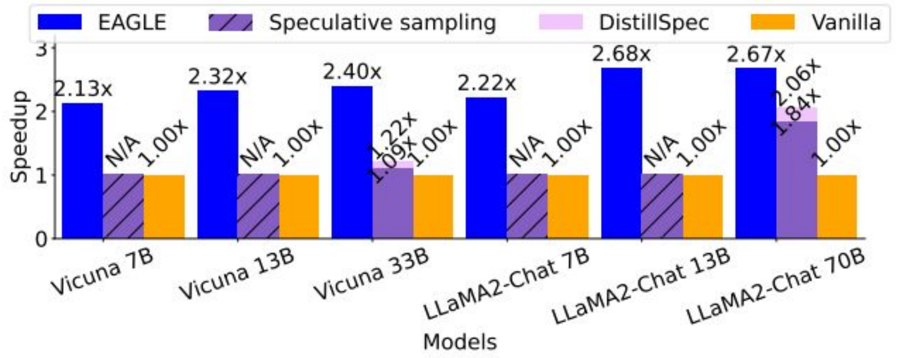


Figure 2: Speedup ratio on the MT-bench for non-greedy (temperature=1) settings. Lookahead is confined to greedy decoding, and the non-greedy generation of Medusa does not guarantee lossless performance. Therefore, EAGLE is not compared with these methods.

Across all models, we achieve **2.7×–3.1× speedups** with an **average acceptance length of 3–4 tokens per pass**, **consistently outperforming Medusa and Lookahead**.

- Speedup: 2.7×–3.1× across all models
- Avg. acceptance length: 3–4 tokens per pass
- Outperforms Medusa & Lookahead consistently

Training Efficiency

- **Fewer than 1 billion trainable parameters**:
  The draft autoregression head is lightweight compared to the full model, so the training cost is modest.

- **Only about 70,000 dialogue samples needed**:
  Training can be done on a relatively small dataset such as ShareGPT conversations, without requiring massive corpora.

- **1–2 days on 4×A100 GPUs for LLaMA2-Chat 70B**:
  Even for a very large target model, the draft head can be trained within a couple of days on a small GPU cluster.

Strengths

- **Lossless acceleration**
  Output distribution is provably preserved — same quality as target LLM.

- **General applicability**
  Works across many LLM families and sizes without modifying the target model.

- **Compatible with other methods**
  Can stack with quantization or compilation (e.g., gpt-fast) for extra speedup.

Limitations

- Gains drop with large batch sizes
- Smaller improvement on MoE models (e.g., Mixtral)
- Still requires training a draft module

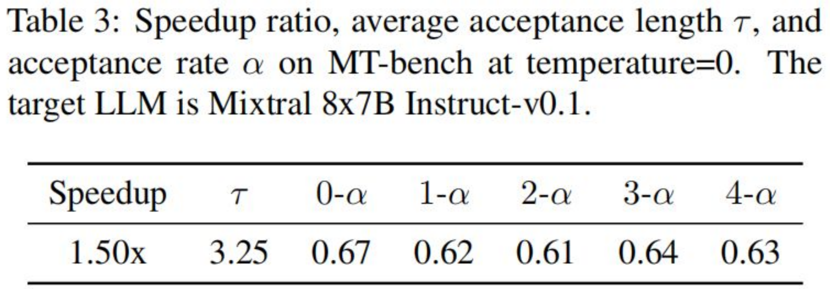


Table 3: Speedup ratio, average acceptance length τ, and acceptance rate α on MT-bench at temperature=0. The target LLM is Mixtral 8x7B Instruct-v0.1.

Speedup	τ	0-α	1-α	2-α	3-α	4-α
1.50x	3.25	0.67	0.62	0.61	0.64	0.63

Conclusion

- Speculative sampling is shifted from the token level to the feature level, making drafting smoother and more predictable.
- The uncertainty of feature sequences is resolved by conditioning on shifted tokens.
- A lightweight draft head combined with tree attention enables efficient multi-branch drafting.
- EAGLE achieves 2–4× faster inference while fully preserving the target LLM’s output distribution.

QA

Learning Harmonized Representations for Speculative Sampling

Lefan Zhang, Xiaodan Wang, Yanhua Huang*, Ruiwen Xu Xiaohongshu Inc., Shanghai, China
{lefan,xiaodan2,yanhuahuang,ruiwenxu}@xiaohongshu.com

arXiv preprint, 2024

Presenter: Ethan Morton, Qiyang Li, Lian Lian

Executive Summary

- HASS is a new speculative sampling method for LLM decoding, addressing both objective misalignment and context misalignment in draft model training.
- Proposes **harmonized objective distillation**, focusing draft model training on the top-K tokens with highest probability from the target LLM, thereby boosting acceptance rates.
- Proposes **harmonized context alignment** to eliminate context inconsistency between training and decoding, mitigating exposure bias and error accumulation.
- Outperforms prior work (notably EAGLE-2) on LLaMA2-Chat 7/13B and LLaMA3-Instruct 8/70B across MT-bench, HumanEval, and GSM8K: achieves 2.81x–4.05x speedup, with 8%-20% higher acceleration than EAGLE-2.

Speculative Sampling: Background and Motivation

- Large Language Models (LLMs) decode auto-regressively, resulting in limited concurrency and slow generation.
- **Speculative sampling** accelerates decoding by using a lightweight draft model to propose a batch of tokens, which is then verified in parallel by the target LLM.
- Performance hinges on (1) low draft model cost and (2) alignment between draft and target model distributions.
- Prior approaches use target LLM states (e.g., hidden states, KV cache) in draft input, but suffer from context misalignment and mismatched training vs. decoding objectives.

Problems in Prior Draft Model Training

- **Context Misalignment**: During training, draft models always see target LLM hidden states; during decoding, they must generate some features themselves, leading to feature ”exposure bias”.
- **Objective Misalignment**: Draft models are usually distilled on full-vocabulary cross-entropy, yet during decoding, only high-probability tokens (top-K or top-P) matter for acceptance; this leads to inefficient draft proposals.
- Both issues widen the distribution gap, decreasing acceptance length τ and limiting wall-clock speedup.
- Figure on next slide illustrates the context misalignment between training and decoding using EAGLE as an example.

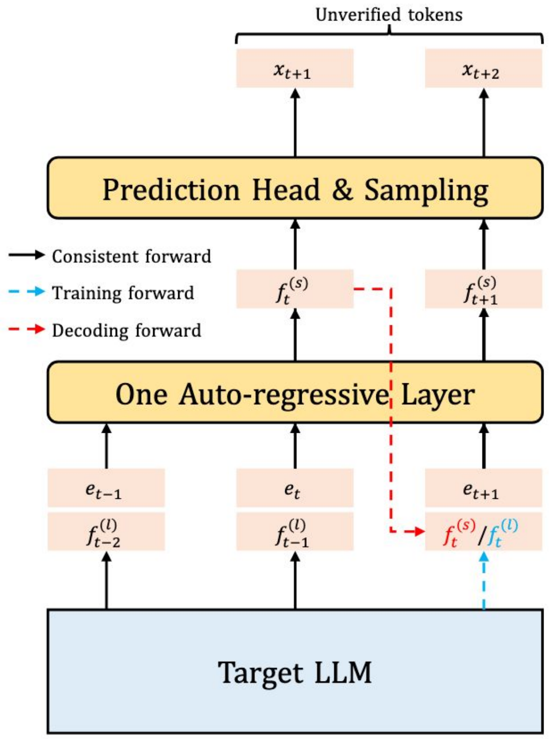


Figure: Context misalignment: During training, the draft model sees target LLM states at all steps; during decoding, it must rely on its own past outputs, leading to drift and mismatch.

Speculative Sampling and Acceptance Length

- In speculative sampling:
  - Draft model $M^{(s)}$ predicts a sequence of L draft tokens auto-regressively. Target LLM $M^{(l)}$ evaluates all L tokens in parallel.
  - A subset of tokens is accepted to preserve the target LLM's distribution (modified rejection sampling).
- Key metric: **Acceptance length** τ — number of draft tokens accepted per cycle. Speedup is directly proportional to τ .
- Efficient speculative sampling requires draft alignment only on high-probability tokens, not across the full vocabulary.
- Prior distillation focused on full-vocabulary likelihood, suboptimal for acceptance length.

Overview of the HASS Method

- **HASS** (HArmonized Speculative Sampling) introduces:
  - **Harmonized Objective Distillation:** Draft model is trained to focus on top-K likely tokens from the target LLM, inspired by ranking distillation from recommender systems.
  - **Harmonized Context Alignment:** Training procedure mimics real decoding context by chaining draft model outputs over several steps, mixing in only available target LLM features.
- Both components improve alignment where it matters — on likely tokens in realistic inference contexts — boosting acceptance length and speedup.

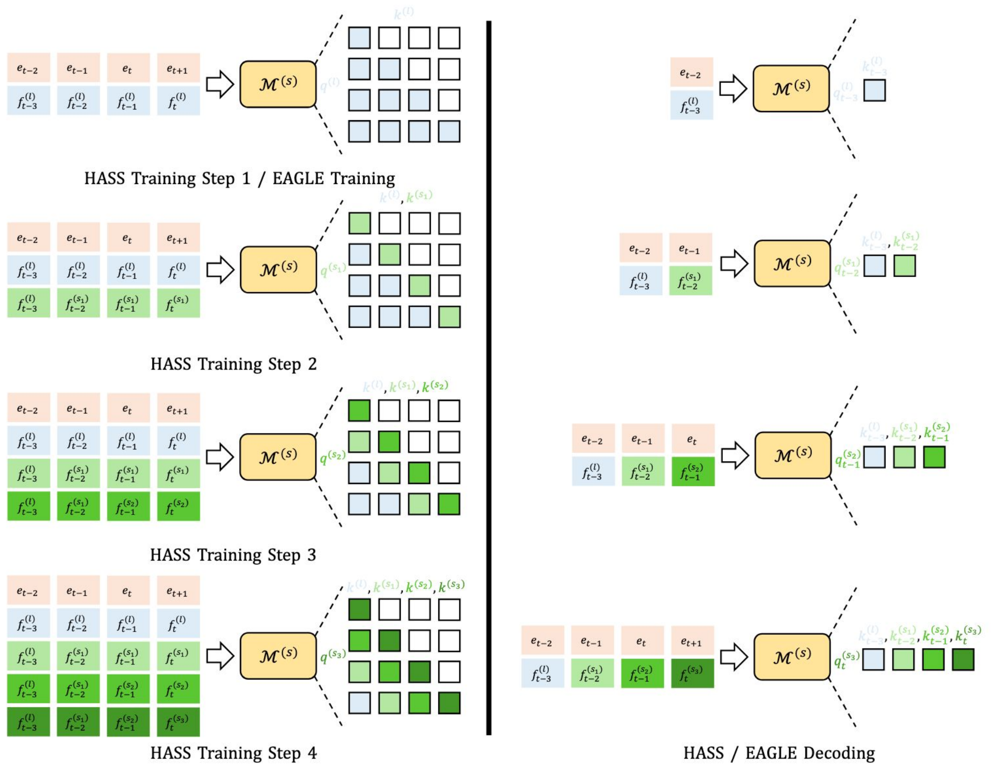

HASS Training Step 1 / EAGLE Training
HASS Training Step 2
HASS Training Step 3
HASS Training Step 4
HASS / EAGLE Decoding
$\mathcal{M}^{(s)}$
e_{t-2} e_{t-1} e_t e_{t+1}
$f_{t-3}^{(l)}$ $f_{t-2}^{(l)}$ $f_{t-1}^{(l)}$ $f_t^{(l)}$
$f_{t-2}^{(s_1)}$ $f_{t-1}^{(s_1)}$ $f_t^{(s_1)}$
$f_{t-1}^{(s_2)}$ $f_t^{(s_2)}$ $f_t^{(s_3)}$
$q^{(l)}$ $q^{(s_1)}$ $q^{(s_2)}$ $q^{(s_3)}$
$k^{(l)}, k^{(s_1)}, k^{(s_2)}, k^{(s_3)}$
$q_{t-3}^{(l)}$ $q_{t-2}^{(s_1)}$ $q_{t-1}^{(s_2)}$ $q_t^{(s_3)}$
$k_{t-3}^{(l)}, k_{t-2}^{(s_1)}, k_{t-1}^{(s_2)}, k_t^{(s_3)}$

Figure: Harmonized context alignment in HASS: During training, query/key-value chains mix target LLM features (superscript (l)) with draft model features (superscript (s_j)) to simulate decoding context over n steps.

Harmonized Objective Distillation: Top-K Loss

$$L_{\text{Top-K}} = -\sum_{x \in \hat{\Omega}} q(x) \log p(x)$$

- $q(x)$ and $p(x)$: target LLM and draft model predicted distributions, respectively.
- $\hat{\Omega}$: set of K tokens with highest probability under target LLM; loss focuses distillation on these probable next tokens.
- **Motivation**: Encourages recall on important tokens for acceptance, not full-vocabulary accuracy. Top-K set can be computed efficiently using target hidden states (when using EAGLE interface).
- Outperforms alternatives (e.g., BiLD, Recall@k losses) in acceptance length across all tasks and LLMs tested.

Harmonized Context Alignment: Training

- Training is divided into *n* steps; at each, simulate the draft model operating with progressively more self-generated (inaccurate) context features.
- At training step *j*:
  - Query: previous draft model output $f_t^{(s_{j-1})}$.
  - Key/Value: mixture of target LLM features ($f_{1:t-j+1}^{(l)}$) and draft model features($f_{t-j+2:t}^{(s_{1:j-1})}$).
- Adapts attention masks so the draft model must generate next-feature inputs like during real decoding.
- **Result**: Draft model learns to compensate for compounded self-generated context, reducing error accumulation and exposure bias.

Implementation and Practical Details

- Evaluated with LLaMA2-Chat 7B/13B and LLaMA3-Instruct 8B/70B as target LLMs.
- Tasks: MT-bench (dialogue), HumanEval (code), GSM8K (math reasoning).
- Metrics:
    - **Speedup ratio**: Wall-clock time vs. vanilla decoding on NVIDIA H800.
    - **Acceptance length** τ : Accepted tokens per draft cycle.
- Draft alignment steps $n = 3$ and Top-K parameter $K = 10$ found to be empirically effective and efficient.
- No fine-tuning of target LLM; method is lossless with respect to output quality.

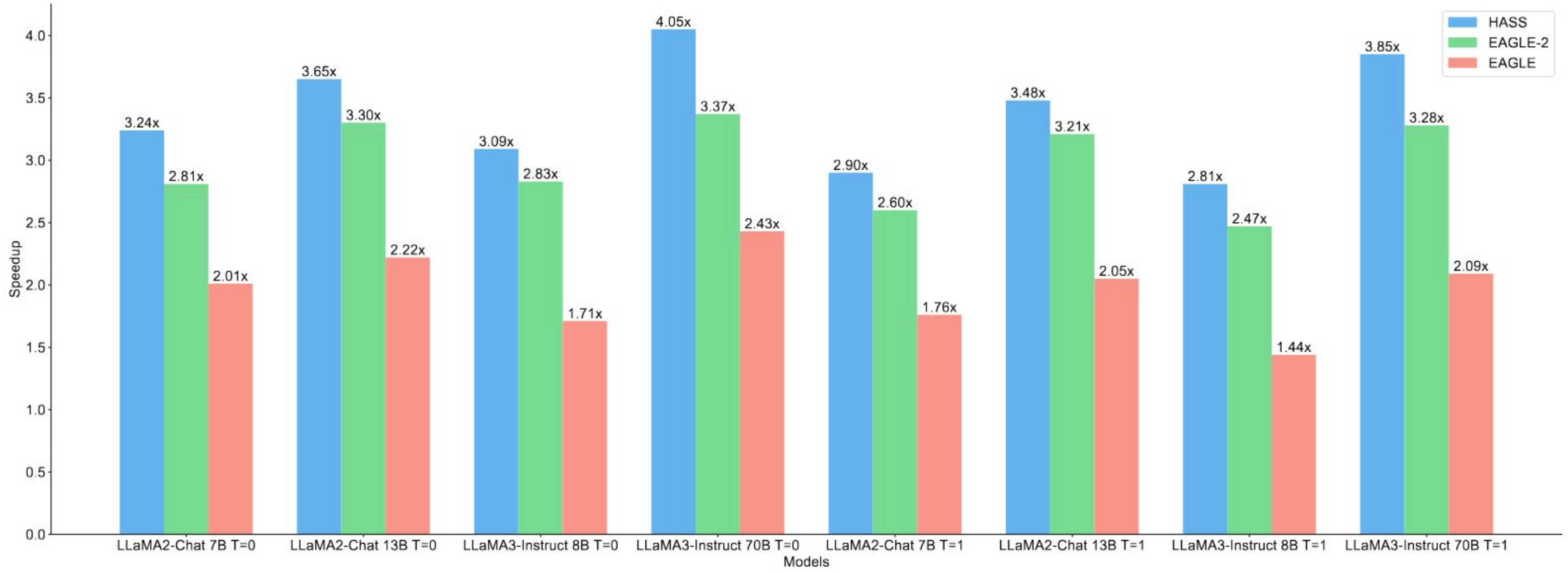


Figure: Speedup ratios of HASS and recent baselines on LLaMA2-Chat 7/13B, LLaMA3-Instruct 8/70B across MT-bench, HumanEval, GSM8K ($T = 0, 1$). HASS provides 2.81x–4.05x speedup, consistently outperforming EAGLE-2 by 8%–20%.

Results: Acceptance Length and Speedup

- HASS achieves top performance on **acceptance length** τ and actual **speedup ratio**:
  - LLaMA2-Chat 7B: HASS $\tau = 5.15$ (EAGLE-2: 4.61), speedup 3.24x (EAGLE-2: 2.81x).
  - LLaMA2-Chat 13B: HASS $\tau = 5.58$ (EAGLE-2: 5.16), speedup 3.65x (EAGLE-2: 3.30x).
  - LLaMA3-Instruct 8B: HASS $\tau = 5.08$, speedup 3.09x.
  - LLaMA3-Instruct 70B: HASS $\tau = 5.21$, 4.05x.
- Comparable trends for both $T = 0$ (greedy) and $T = 1$ (sampling).
- HASS is robust across tasks (dialogue, code, math); outperforms baselines on all evaluated LLMs.

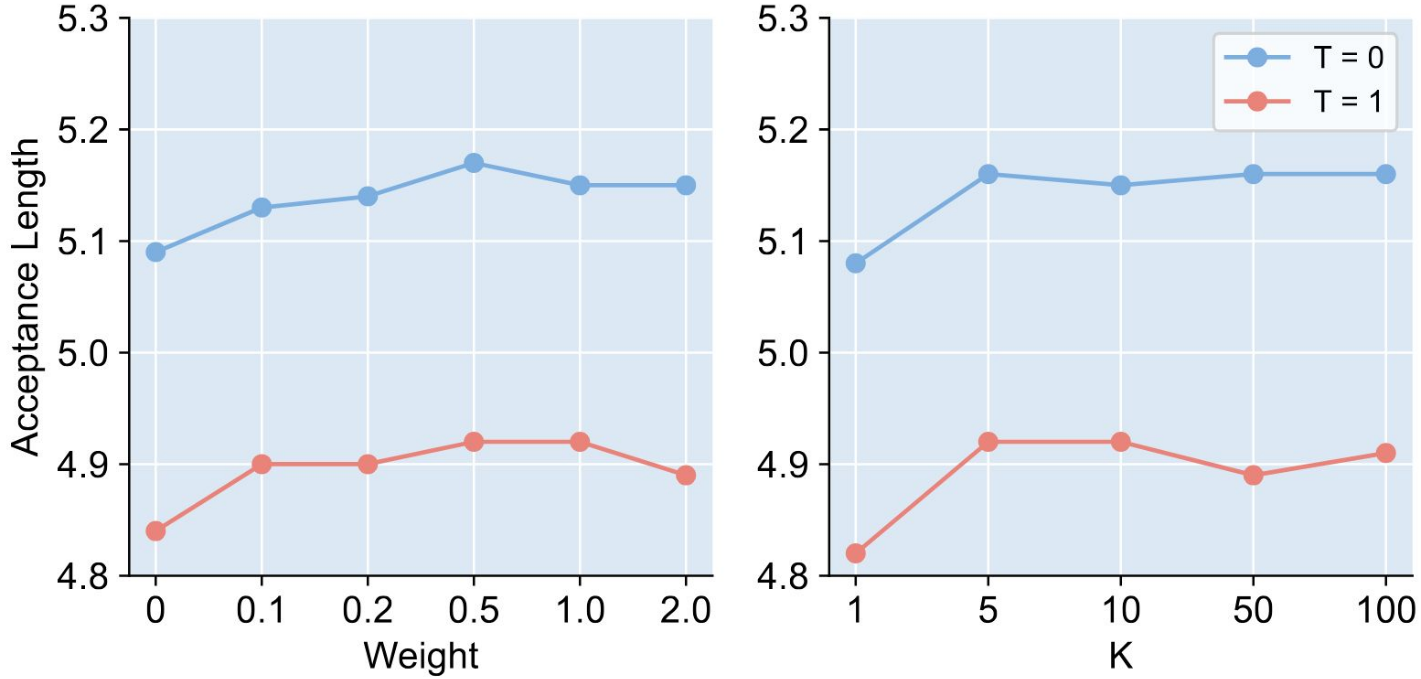


Figure: Ablation on Top-K parameters: Acceptance length increases with Top-K loss ($w > 0$); $K = 5$ and $w = 0.5$ is optimal. Too small K underperforms as draft ignores other likely tokens.

Ablation: Loss Functions for Harmonized Objective Distillation

- Evaluated several loss functions for draft distillation:
  - Top-P Loss (probability mass cutoff)
  - Normed Top-K Loss (with/without softmax)
  - Bi-directional Top-K Loss
  - Recall@k Surrogate Loss
  - BiLD Loss (logit ranking)
- Results (LLaMA2-Chat 7B, MT-bench):
  - Top-K: $\tau = 4.92$ (best)
  - Top-P: $\tau = 4.90$
  - BiLD: $\tau = 4.90$
  - All others τ between 4.85–4.90
- Top-K loss gives strongest consistent benefit, especially at higher temperature ($T = 1$).

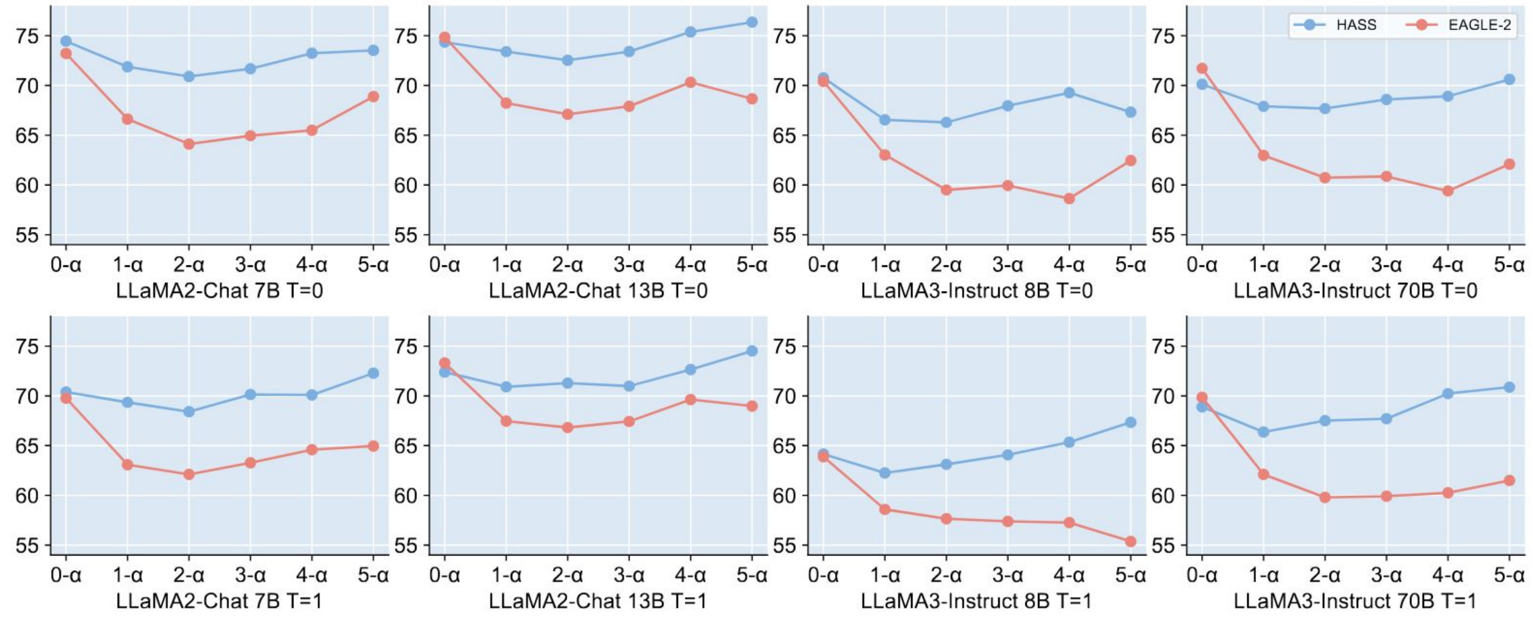


Figure: Acceptance rate per speculation step: HASS vs. EAGLE-2. HASS substantially improves acceptance rate in later steps, indicating reduced error accumulation.

Ablation: Steps in Context Alignment

- Varying the number of alignment steps (n) in HASS:
  - Best at $n = 3$ or 4. For LLaMA2-Chat 7B: $n = 3$: $\tau = 5.15$; $n = 4$: $\tau = 5.16$. Too
  - many steps ($n = 5$) can degrade performance slightly, possibly due to limited draft model capacity.
- Loss reweighting across steps (β): Assigning higher importance to early steps increases first-step acceptance rate and overall τ.
- HASS remains efficient, as acceleration gains converge for small n.

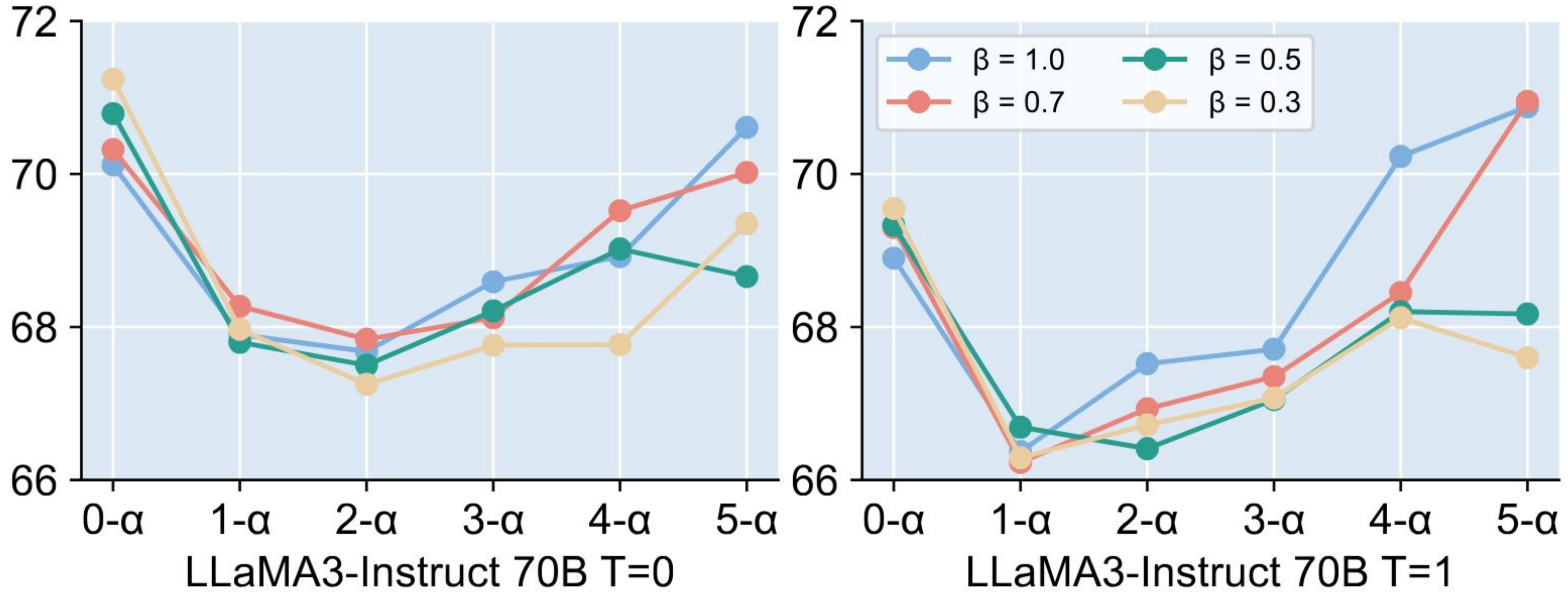


Figure: Ablation on loss reweighting: Emphasizing early steps ($\beta = 0.5$) increases acceptance at the first step and lifts overall acceptance length.

Conclusion and Impact

- HASS delivers substantial wall-clock acceleration for LLM decoding via harmonized draft training.
- Innovations: Focus draft on high-likelihood tokens (Top-K loss); simulate decoding context with multi-step context alignment.
- Outperforms leading baselines (EAGLE-2: 8–20% greater speedup) across LLaMA2/3, dialogue, code, math, and translation tasks.
- Opens new research: tailoring distillation loss and efficient context simulation. Code/models open-sourced at https://github.com/HArmonizedSS/HASS.

Q/A