Mamba Drafters for Speculative Decoding

Paper

Daewon Choi[1] Seunghyuk Oh[1] Saket Dingliwal[2] Jihoon Tack[1] Kyuyoung Kim[1] Woomin Song[1, 2]
Seojin Kim[3] Insu Han[1] Jinwoo Shin[1] Aram Galstyan[2] Shubham Katiyar[2] Sravan Babu Bodapati[2]

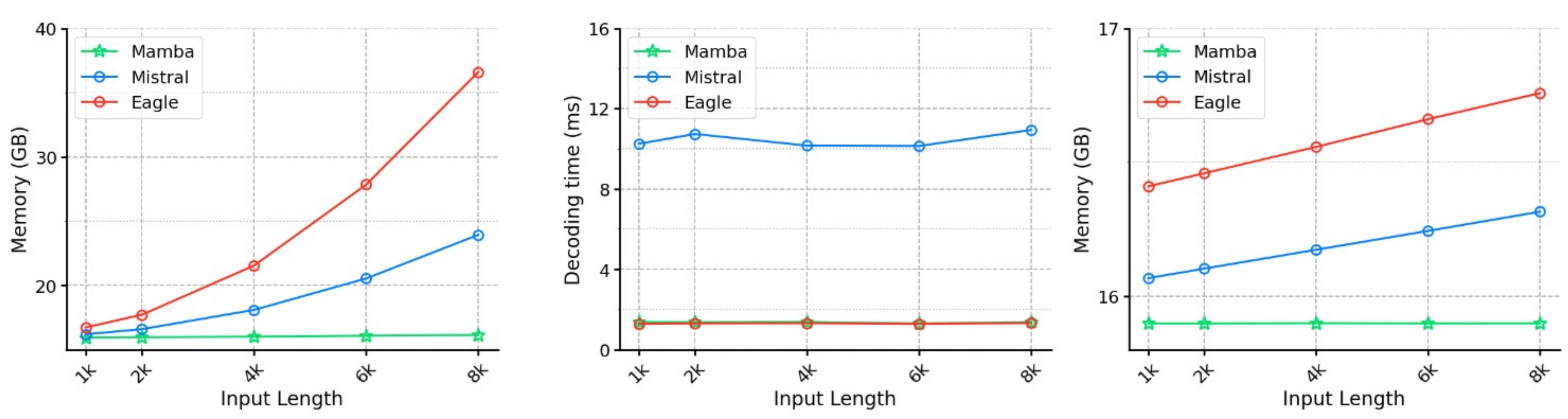
[1]KAIST [2]Amazon AGI [3]Seoul National University

Contact: daeone0920@kaist.ac.kr

TL;DR: Fast, memory-efficient, even effective-
Mamba is the ideal drafter for speculative decoding.

Why Mamba for Speculative Decoding ?

1. Efficiency of Mamba as a drafter



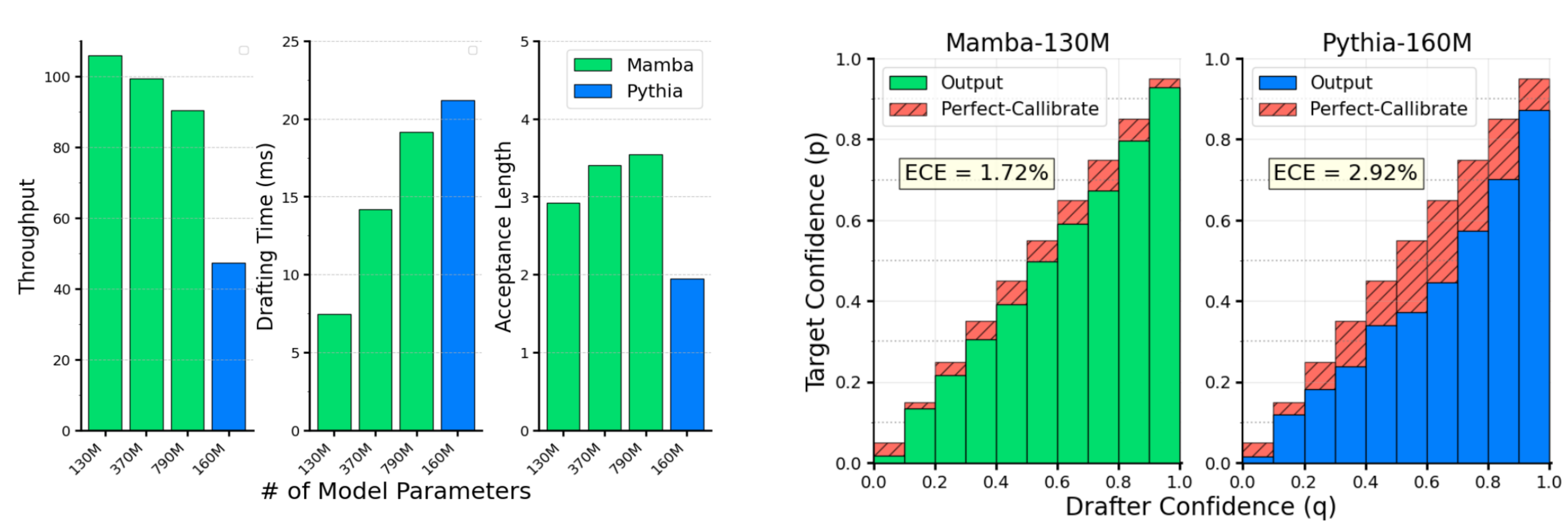
(a) Encoding Memory (b) Decoding Time (c) Decoding Memory

* Mamba / Mistral / EAGLE

External Transformer drafter Self-speculation with a single-layer Transformer

✓ Mamba enables fast and memory-efficient decoding regardless of context length. (Unlike Transformer-based drafters)

2. Effectiveness of Mamba as a drafter



✓ Mamba drafters achieve higher acceptance length than the Transformer (Better alignment with the distribution of larger Transformers, e.g., lower ECE)

✓ Smaller Mamba offers significantly faster drafting speed compared to larger sizes. (With a slightly lower acceptance length -> achieve higher throughput!)

Tree-Structured Drafting with Mamba

Method1) Efficient tree-structured drafting with batch generation

For generating multiple candidates per generation during the drafting step,

- Mamba: copying the current single state to predict the next token.
- Transformers: copying the current sequence length of the KV cache.

⟶ Duplication overheads for batch generation is low in Mamba !

Given a tree configuration $\mathcal{T} = (N_1, N_2, ..., N_\gamma)$,
(γ : draft length
N_i : # of new nodes obtained by sampling from each node at the i^{th} generation)

✓ Batch generation of a total batch size, i.e., $\mathcal{B}_i = N_1 \times N_2 \times \cdots \times N_i$
✓ Possible input size is deterministic, i.e.,$(\mathcal{B}_1, 1), (\mathcal{B}_2, 1), ..., (\mathcal{B}_\gamma, 1)$

Pre-allocation of memory / Graph caching is applicable for acceleration 🔥

Method2) Test-time dynamic tree search using multi-armed bandit

Thanks to fast drafting, Mamba can leverage different tree configurations.

⟶ Formalize the tree search problem as a multi-armed bandit (MAB) at test-time!

Given a tree configuration set $\mathcal{S} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_K\}$, choose $\mathcal{T}_{k^*}^{(t)}$ at round t

✓ Optimization (using UCB) ✓ Reward

$$k^* = \underset{k \in \{1,..,K\}}{\arg\max} \underbrace{\boxed{\hat{r}_k^{(t)}}}_{\text{Reward}} + \lambda_{\text{UCB}} \sqrt{\frac{2 \ln t}{n_k^{(t)}}}$$

$$r_k^{(t)} := -\left(\frac{1}{\underbrace{\boxed{N_{\text{accept}}}}_{\text{\# of accepted tokens}}} + \lambda_\gamma \frac{\overbrace{\boxed{\gamma^{(\mathcal{T}_k)}}}^{\text{Draft length of tree}}}{N_{\text{accept}}} \right) \cdot I$$

At test-time, an acceptance ↑ & draft length ↓ tree is likely to be selected.

Experiments

Mamba drafter outperform Transformer drafters of all sizes, and are even comparable to state-of-the-art self-speculation method !

Target	Drafter Method	Drafter Size	Greedy (Temp=0) XSum	Greedy CNN-DM	Greedy GSM-8k	Sampling (Temp=1) XSum	Sampling CNN-DM	Sampling GSM-8k
Pythia-6.9B	No drafter	–	53.30	49.29	54.69	52.51	45.33	53.81
	Pythia	70M	47.31 (1.52)	46.99 (1.54)	57.36 (1.68)	41.86 (1.67)	45.30 (1.76)	47.96 (1.77)
		160M	50.05 (2.23)	49.53 (2.26)	67.89 (2.72)	46.67 (2.28)	47.17 (2.30)	55.40 (2.63)
		410M	70.53 (4.62)	70.08 (4.73)	75.97 (4.64)	53.50 (3.60)	56.64 (3.80)	63.64 (4.01)
	Ours	130M	**138.80** (4.55)	**131.97** (4.38)	**149.46** (4.57)	**108.68** (3.53)	**105.01** (3.53)	**119.67** (3.73)
Mistral-7B	No drafter	–	51.15	49.55	50.31	53.49	47.40	52.92
	Mistral	160M	61.55 (3.13)	61.04 (3.05)	49.38 (2.21)	53.91 (2.74)	50.50 (2.68)	62.29 (2.94)
	Ours	130M	**76.71** (2.39)	**65.23** (2.13)	**77.50** (2.25)	**79.18** (2.73)	**70.95** (2.65)	**82.63** (2.73)

Pre-trained models on language modeling tasks

Target	Drafter Method	Drafter External?	Greedy (Temp=0) MT-bench	Greedy Alpaca	Greedy Human-Eval	Sampling (Temp=1) MT-bench	Sampling Alpaca	Sampling Human-Eval
Pythia-6.9B	No drafter	–	54.51	55.28	54.76	53.89	54.72	54.21
	Pythia	✓	70.71 (3.10)	60.77 (2.65)	109.51 (4.68)	65.73 (3.03)	62.07 (2.82)	109.52 (4.25)
	EAGLE	✗	125.61 (3.85)	**117.17** (3.53)	122.44 (4.71)	87.01 (2.67)	78.58 (2.40)	83.05 (2.97)
	Ours	✓	**128.21** (3.91)	114.08 (3.41)	**172.38** (5.41)	**110.20** (3.65)	**108.54** (3.51)	**143.55** (4.82)
Mistral-7B	No drafter	–	52.97	53.58	52.30	52.39	53.02	52.34
	Mistral	✓	67.47 (3.04)	61.40 (2.73)	100.23 (4.53)	57.19 (2.84)	51.05 (2.40)	80.94 (3.92)
	EAGLE	✗	**107.16** (3.22)	94.03 (2.79)	**132.69** (3.98)	**94.03** (2.90)	**86.60** (2.63)	**122.11** (3.78)
	Ours	✓	102.48 (3.16)	**96.83** (2.96)	118.04 (3.69)	88.68 (2.95)	82.75 (2.71)	87.81 (2.94)

Instruction-tuned models on instruction following tasks

Method	External?	Single-Document QA 1k	2k	4k	8k	Multi-Document QA 1k	2k	4k	8k	Peak Memory (GB) 1k	2k	4k	8k
No drafter	-	31.02	27.89	24.35	19.30	28.17	24.22	19.01	14.83	15	16	20	36
Mistral	✓	25.30 (2.43)	23.28 (2.37)	19.48 (2.24)	15.23 (2.21)	24.64 (2.53)	21.06 (2.48)	16.49 (2.44)	12.18 (2.39)	31	33	38	59
EAGLE	✗	53.13 (2.73)	**47.00** (2.81)	**37.27** (2.76)	**26.12** (2.71)	42.48 (2.60)	35.38 (2.61)	25.10 (2.68)	17.36 (2.64)	32	34	42	72
Ours	✓	**55.09** (2.91)	45.65 (2.77)	36.32 (2.77)	24.92 (2.80)	**47.91** (2.94)	**36.40** (2.86)	**26.27** (2.88)	**17.77** (2.87)	31	32	36	52

Long context scenario

Discussion points

Q1. Can Mamba be used with different target model without re-training?

Method	Setup	Accept length	Throughput
EAGLE	Pythia → Pythia	2.59	94.75
	Mistral → Pythia	N/A	N/A
Ours	Pythia → Pythia	3.08	112.69
	Mistral → Pythia	2.45	93.20

Mamba achieves throughput comparable to EAGLE, which is specifically trained for the target model.

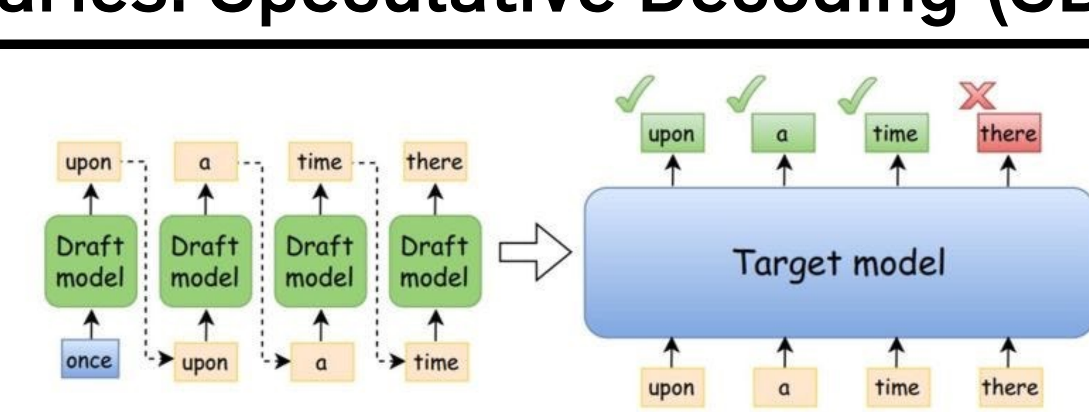
Q2. Effects of tree-drafting

Search?	MT-bench	Alpaca	HumanEval	Avg.
✗	124.99	**116.12**	149.15	130.09
✓	**128.21**	114.08	**172.38**	**138.22**

Q3. Effects of test-time tree search

Tree?	Accept length	Latency	Throughput
✗	3.08	6.62	112.69
✓	3.91	8.30	127.37

Preliminaries: Speculative Decoding (SD)



Step1: Autoregressive generation Step2: Parallel verification

SD is to generate candidate tokens with an efficient drafter and verifying them in parallel with the target model.