🌵 Cactus: Accelerating Auto-Regressive Decoding with Constrained Acceptance Speculative Sampling

Anonymous authors
Paper under double-blind review

Abstract

Speculative sampling (SpS) has been successful in accelerating the decoding throughput of auto-regressive large language models by leveraging smaller draft models. SpS strictly enforces the generated distribution to match that of the verifier LLM. This is unnecessarily restrictive as slight variation of the verifier's distribution, such as sampling with top-k or temperature, would also be acceptable. Typical acceptance sampling (TAS) alleviates this issue by accepting more tokens using entropy-based heuristics. However, this approach distorts the verifier distribution, potentially degrading output quality when the verifier encodes critical information. In this work, we formalize the speculative sampling algorithm through the lens of constrained optimization. Based on this formulation, we propose **Cactus** (**c**onstrained **ac**cep**t**ance spec**u**lative **s**ampling), a method that guarantees controlled divergence from the verifier distribution and increasing acceptance rates. Empirical results across a wide range of benchmarks confirm the effectiveness of our approach. The code is publicly available at this anonymous link.

1 Introduction

Auto-regressive large language models (LLMs) have driven remarkable advances in machine learning and artificial intelligence (Vaswani et al., 2017; Brown et al., 2020; Kaplan et al., 2020), yet their growing size comes with steep computational costs: generating each token requires a memory-bound forward pass through hundreds of billions of parameters, which bottlenecks LLM throughput (Yuan et al., 2024). Speculative sampling (SpS) addresses this by first using a smaller draft model to propose a fixed amount of candidate tokens in multiple smaller forward passes, then verifying them in parallel with the large-scale *verifier* LLM (Stern et al., 2018; Xia et al., 2022; Leviathan et al., 2023; Chen et al., 2023). Since SpS can emit multiple tokens per large-model invocation, it substantially speeds up auto-regressive generation by alleviating the memory-bound issue.

Despite its success, SpS enforces strict distributional equivalence with the verifier, causing correct but lower-probability tokens to be rejected. In real-world applications, exact adherence to the original distribution is generally not required (Holtzman et al., 2019; Meister et al., 2020). Typical acceptance sampling (TAS Cai et al. (2024a)) mitigates this issue by accepting proposals based on entropy-driven heuristics (Hewitt et al., 2022; Meister et al., 2023). However, we show in this paper that TAS improves acceptance rates at the cost of distorting the verifier's output distribution and risking semantic drift when the verifier encodes critical information.

In this work, we reformulate speculative sampling as a constrained optimization problem, explicitly trading off acceptance rate against divergence from the verifier's distribution. Guided by this theory, we introduce Cactus (**c**onstrained **ac**ceptance spec**u**lative **s**ampling), a simple yet principled modification that enforces a hard bound on distributional divergence while enabling higher acceptance rates.

We conducted experiments on a wide range of benchmarks with multiple state-of-the-art large language models. Results show that Cactus consistently improves generation throughput compared with the lossless SpS. In addition, Cactus preserves the generation quality and diversity of the verifier model, due to its explicit divergence constraint.

Algorithm 1 Speculative sampling algorithm.

```
Require: sampling steps $m$, draft model $p$, acceptance rate $\phi$, and recover probability $g$
$t \leftarrow 1, \boldsymbol{x}_{<t} \leftarrow$ [BOS]
while not end do
    ▷ Drafting $m$ tokens
    for $i \leftarrow 0, \dots, m-1$ do
        $x_{t+i} \sim p(\cdot|\boldsymbol{x}_{<t+i})$    ▷ $\boldsymbol{x}_{<t+i}$ is concatenation of $\boldsymbol{x}_{<t}$ and $[x_t, \dots, x_{t+i-1}]$
        $u_i \sim U(0,1)$    ▷ $U(0,1)$ is the uniform distribution between $[0,1]$
    end for
    $c \leftarrow \min\{j : u_j > \phi(x_{t+j}|\boldsymbol{x}_{<t+j})\} \bigcup \{m\}$    ▷ $c$ is the length of accepted draft tokens
    $x_{t+c} \sim g(\cdot|\boldsymbol{x}_{<t+c})$    ▷ $x_c$ is always accepted
    $t \leftarrow t + c + 1$
end while
```

2 APPROACH

We first formalize speculative sampling algorithm. This enables a theoretical analysis of speculative sampling under a constrained optimization framework. Based on this analysis, we propose a new algorithm, Cactus, which provably approximates the verifier distribution q while achieving higher acceptance rates.

2.1 GENERALIZATION OF SPECULATIVE SAMPLING

Speculative sampling. The vanilla speculative sampling (SpS Chen et al. (2023)) uses a *draft model* $p(\cdot|\boldsymbol{x}_{<t})$ that has significantly less memory footprint than the *verifier model* $q(\cdot|\boldsymbol{x}_{<t})$. At a time step t, SpS repeatedly samples $m \in \mathbb{N}_+$ tokens $x_t, \dots, x_{t+m-1}$ from p in an auto-regressive manner. Each token is accepted with a probability given by the *acceptance rate* $\phi(x_{t+i}|\boldsymbol{x}_{<t+i}) = \min(1, q(x_{t+i}|\boldsymbol{x}_{<t+i})/p(x_{t+i}|\boldsymbol{x}_{<t+i}))$ for all $i \in [0, m)$. If any token x_{t+j} is rejected, then tokens $x_{t+j+1}, \dots, x_{t+m-1}$ are also discarded. As a backup, SpS resamples x_{t+j} using the *recover probability* $g(x_{t+j}|\boldsymbol{x}_{<t+j}) \propto (q(\cdot|\boldsymbol{x}_{<t+j}) - p(\cdot|\boldsymbol{x}_{<t+j}))_+$, where $(\cdot)_+$ denotes $\max(0, \cdot)$. The final accepted tokens are $x_t, \dots, x_{t+j}$. By this draft-and-verify scheme, SpS accelerates auto-regressive decoding by avoiding the need to load the large verifier model q from memory at every step. This approach has been shown effective in practice (Zhou et al., 2024; Hu et al., 2025).

Our observation. We formalize the draft-and-verify scheme as Algorithm 1. Under this setting, we can show that the algorithm produces any target distribution with an optimal acceptance rate.

Observation 1. *Given any desired target distribution h and draft model p, the acceptance rate and recovery probability are defined as*

$$\phi(x_t|\boldsymbol{x}_{<t}) = \min\left(\frac{h(x_t|\boldsymbol{x}_{<t})}{p(x_t|\boldsymbol{x}_{<t})}, 1\right) \tag{1}$$

$$\text{and} \quad g(x_t|\boldsymbol{x}_{<t}) = \frac{h(x_t|\boldsymbol{x}_{<t}) - p(x_t|\boldsymbol{x}_{<t})\phi(x_t|\boldsymbol{x}_{<t})}{\mathbb{E}_{x'\sim p}[1 - \phi(x'|\boldsymbol{x}_{<t})]} \tag{2}$$

respectively. Algorithm 1 samples from h exactly using the above ϕ and g. In addition, this ϕ is the optimal design of acceptance rate.

Proof. See Appendix A.1. □

2.2 APPROXIMATING SPS AS CONSTRAINED OPTIMIZATION

Observation 1 provides a foundation to produce an arbitrary target distribution with the optimal design. Instead of producing a fixed verifier distribution q, we utilize this observation to dynamically select a distribution h close to q while yielding higher acceptance rates based on function ϕ. This can be formulated as a constrained optimization problem.

Definition 2. For each step t, assume the drafted token has index n. Let $\boldsymbol{h} \in \mathbb{R}^{|V|-1}$ be the parameters to be optimized. The ideal h is given by $h(i|\boldsymbol{x}_{<t}) = \boldsymbol{h}_i^*$, where $\boldsymbol{h}^*$ is the solution of the following problem:

$$\max_{\boldsymbol{h}} \ \min(h_n/p(n|\boldsymbol{x}_{<t}), 1) \tag{3}$$

$$\text{s.t. } \boldsymbol{h} \in \Delta^{|V|-1} \tag{4}$$

$$D_f(\boldsymbol{h} \| q(\cdot|\boldsymbol{x}_{<t})) \leq \delta. \tag{5}$$

Here, the hyper-parameter $\delta \geq 0$ controls the closeness to the verifier model q, and D_f is any f-divergence metric used to measure the distance between q and h.

The above definition falls into the framework of constrained convex optimization, which we show has the following solution.

Theorem 3. *The optimal* $\boldsymbol{h}$ *in Definition 2 is*

$$h_i = \begin{cases} \gamma^*, & \text{if } i = n, \\ \frac{1-\gamma^*}{1-q(n|\boldsymbol{x}_{<t})} q(i|\boldsymbol{x}_{<t}), & \text{otherwise}, \end{cases} \tag{6}$$

where γ^* *is any root of the equation*

$$\delta = q(n|\boldsymbol{x}_{<t}) f\left(\frac{\gamma}{q(n|\boldsymbol{x}_{<t})}\right) + (1 - q(n|\boldsymbol{x}_{<t})) f\left(\frac{1-\gamma}{1-q(n|\boldsymbol{x}_{<t})}\right) \tag{7}$$

over the interval $[q(n|\boldsymbol{x}_{<t}), +\infty)$*, clamped into* $[q(n|\boldsymbol{x}_{<t}), 1]$*. The function* f *is the one used in the definition of* f*-divergence.*

Proof. See Appendix A.2. ☐

Theorem 3 theoretically characterizes the trade-off between closeness to the verifier model q and the acceptance rate induced by ϕ. In particular, the theorem suggests that the drafted token now has at least the same or a higher chance of being accepted (since $\gamma^* \geq q_n$). The exact probability depends on the choice of the f-divergence and the hyper-parameter δ. For other non-sampled tokens, their probabilities are scaled down proportionally so that h remains a valid distribution.

It is worth-nothing that, since the solved $\boldsymbol{h}$ in Equation (6) depends on the sampled token n, the solution is different for different sampled tokens. As a result, the effective distribution of the overall algorithm $\boldsymbol{h}_{\text{alg}}$ might have a different divergence other than δ from the target distribution q. To this end, we provide the following theorem to guarantee the controlled divergence of the effective distribution.

Theorem 4. *Let* ϕ_n *and* g_n *denote the functions that follow the solution in Theorem 3 when the sampled token is* n*. The distribution of the overall algorithm is given by*

$$\boldsymbol{h}_{alg} = \sum_{n \in [|V|]} p(n|\boldsymbol{x}_{<t}) \left[\phi_n(n)\boldsymbol{e}_n + (1 - \phi_n(n))\boldsymbol{g}_n\right], \tag{8}$$

where $\mathbf{e}_n$ *is a one-hot vector with only non-zero element at index* n*. In addition,*

$$D_f(\boldsymbol{h}_{alg} \| q(\cdot|\boldsymbol{x}_{<t})) \leq \min\{\Gamma(\delta), D_f(p(\cdot|\boldsymbol{x}_{<t}) \| q(\cdot|\boldsymbol{x}_{<t}))\} \tag{9}$$

for any $\delta \geq 0$*. Here, the function* $\Gamma : [0, +\infty) \to [0, +\infty]$ *is continuous and non-decreasing in* δ *with a value of* 0 *at* $\delta = 0$*.*

Proof. See Appendix A.3. ☐

In essence, despite the $\boldsymbol{h}$ in Equation (6) is solved specifically for the sampled token n, the divergence between the overall distribution and the target distribution is still implicitly controlled. In particular, for any target divergence $0 \leq \delta_{\text{alg}} < +\infty$ imposed on the overall algorithm, we can always find a proper $\delta \geq 0$ such that $D_f(\boldsymbol{h}_{\text{alg}} \| q) \leq \Gamma(\delta) \leq \delta_{\text{alg}}$. While Γ does not admit a closed-form expression, δ itself is a hyper-parameter. In practice, one can tune δ to achieve the desired quality-throughput trade-off. This confirms the soundness of our framework.

In fact, our framework also offers a novel theoretical interpretation of typical acceptance sampling.

Proposition 5. *Typical acceptance sampling (TAS, Cai et al. (2024a)) implicitly solves a variant of the optimization problem in Definition 2, where the f-divergence is substituted with the cross-entropy $H(\boldsymbol{h}, q(\cdot|\boldsymbol{x}_{<t}))$.*

Proof. See Appendix A.4. ☐

The suboptimality of TAS arises from the nature of cross-entropy. Specifically, the cross-entropy can be decomposed as

$$H(\boldsymbol{h}, q(\cdot|\boldsymbol{x}_{<t})) = \underbrace{D_{\mathrm{KL}}(\boldsymbol{h}\|q(\cdot|\boldsymbol{x}_{<t}))}_{\text{Mode capturing}} + \underbrace{H(\boldsymbol{h})}_{\text{Certainty}}\,. \tag{10}$$

Here, the KL divergence encourages h to focus on the mode of q (since h is the first argument), while the entropy term encourages h to be deterministic. However, the summation allows h to collapse into a deterministic distribution at the expense of increasing divergence, thereby failing to capture the full shape of q. In fact, TAS always yields h with entropy 0 while increasing the divergence by at least $H(q)$. As a result, the produced distribution may diverge significantly from the verifier model, especially when q carries high entropy and thus rich information.

2.3 CACTUS: CONSTRAINED ACCEPTANCE SPECULATIVE SAMPLING

Based on our analysis above, we propose using only the KL divergence as the measure of "distance". Specifically, this corresponds to the function $f(t) = t\log t$. Combined with our Theorem 3, γ^* is the root of

$$\Phi(\gamma) := q(n|\boldsymbol{x}_{<t}) f\left(\frac{\gamma}{q(n|\boldsymbol{x}_{<t})}\right) + (1 - q(n|\boldsymbol{x}_{<t})) f\left(\frac{1-\gamma}{1 - q(n|\boldsymbol{x}_{<t})}\right) \tag{11}$$

$$= \gamma \log\left(\frac{\gamma}{q(n|\boldsymbol{x}_{<t})}\right) + (1-\gamma)\log\left(\frac{1-\gamma}{1 - q(n|\boldsymbol{x}_{<t})}\right) \tag{12}$$

$$= \delta. \tag{13}$$

However, since Φ is a transcendental function involving terms like $x \log x$, it cannot be solved in closed form. We therefore approximate Φ by its second-order Taylor series expanded at $\gamma_0 = q(n|\boldsymbol{x}_{<t})$:

$$\Phi(\gamma) \approx \Phi(\gamma_0) + \Phi'(\gamma_0)(\gamma - \gamma_0) + \frac{\Phi''(\gamma_0)}{2}(\gamma - \gamma_0)^2. \tag{14}$$

This approximation is justified by noting that δ is typically small and γ^* remains close to $q(n|\boldsymbol{x}_{<t})$.

Corollary 6 (Cactus's solution)**.** *Let the f-divergence in Definition 2 be the KL divergence. The solution to Equation* (14) *is given by*

$$h(i|\boldsymbol{x}_{<t}) = \begin{cases} \gamma^*, & \textit{if } i = n, \\ \frac{1-\gamma^*}{1-q(n|\boldsymbol{x}_{<t})} q(i|\boldsymbol{x}_{<t}), & \textit{otherwise}, \end{cases} \tag{15}$$

where $\gamma^* = \min\left\{ q(n|\boldsymbol{x}_{<t}) + \sqrt{2\delta q(n|\boldsymbol{x}_{<t})(1 - q(n|\boldsymbol{x}_{<t}))}, 1 \right\}$.

Proof. See Appendix A.5 ☐

In other words, Cactus modifies the distribution of the verifier model by increasing the probability of the candidate token n by a small "bonus" determined jointly by $q(n|\boldsymbol{x}_{<t})$ and δ. We further show that Cactus's solution is more conservative than the exact solution when the verifier is less confident, ensuring that it strictly satisfies the divergence constraint in such cases.

Corollary 7. *When the exact solution γ^* is not greater than* 0.5 *(i.e., the token is not likely to be accepted), our approximation always satisfies the divergence constraint:*

$$D_{KL}(h\|q) \leq \delta, \tag{16}$$

where $h(n|\boldsymbol{x}_{<t})$ is given by the approximated solution in Equation (15).

Proof. See Appendix A.6. □

It is easy to see that the bonus probability attains its maximum when $q(n|\boldsymbol{x}_{<t}) = 0.5$. In practice, LLMs generally have more than 100K tokens (Dubey et al., 2024; Qwen et al., 2024), so a probability around 0.5 indicates strong model confidence in the token. However, SpS could still reject the token n solely because the draft model is overconfident (i.e., $p(n|\boldsymbol{x}_{<t})$ is large). Cactus increases the acceptance likelihood in such scenarios by modifying the verifier distribution accordingly.

Compared with TAS's criterion function, Cactus only requires reading the probability at token n rather than accessing the full vocabulary. This allows Cactus to further reduce memory access overhead, especially in large-vocabulary settings. More importantly, Cactus 's divergence is tightly controlled with minimal entropy change, whereas TAS yields only low-entropy solutions.

3 EXPERIMENTS

3.1 SETTINGS

Datasets. We evaluated Cactus on three popular benchmark datasets for large language models: (1) The **GSM8K** (Cobbe et al., 2021) dataset contains 1.3K high-quality grade school math word problems in the evaluation set, designed to assess a model's ability to apply mathematics to real-world scenarios. Following common practice in LM-Eval (Gao et al., 2024), we used 5-shot examples for each test instance. The final accuracy score is averaged over all samples. (2) The **IFEval** (Zhou et al., 2023) benchmark measures instruction-following ability. It consists of 500 ”verifiable instructions” whose outputs can be heuristically evaluated. For example, a prompt might be: “Write a blog post with 400 or more words about the benefits of sleeping in a hammock,” which can be automatically checked by counting the number of words. (3) The **GPQA** (Rein et al., 2023) diamond benchmark includes approximately 200 challenging science questions authored by domain experts, designed to test models' scientific knowledge. For instance, a sample question is: “The angular size of the event horizon of a supermassive black hole in the centre of a galaxy at a distance of $d = 10^{10}$ parsecs is measured to be $\theta = 10^{-17}$ degrees. Find the order of magnitude of the entropy of the black hole.” Following common practice (Gao et al., 2024), we include four answer choices in the prompt and have models generate the correct one.

Evaluation metrics. For all three tasks, the results are extracted from the generated text by regex matching with the corresponding format. These results are then compared with the gold labels using strict-match accuracy (i.e., 1 if the strings are identical and 0 otherwise). Final scores are obtained by averaging the accuracies over all samples. Following previous work (Dubey et al., 2024), the regex for GSM8K and GPQA is the “flexible-extract” pattern, which selects the first number in the generated sentence regardless of whether the model adheres to the few-shot examples. For IFEval, we use the “prompt-level-strict-acc” regex as defined in Qwen et al. (2024), which requires the model to strictly follow all the instructions.

In addition to task scores, we report the average acceptance length (AL) for all runs. Specifically, AL_m refers to the expected number of accepted tokens among m drafted tokens. A generally higher AL_m indicates faster generation. However, a method may artificially inflate AL by accepting low-quality draft tokens that are later revised during earlier steps of the chain of thought. Although AL remains high, this behavior can lead to lower overall throughput due to unnecessarily lengthy outputs. To present a more complete picture of generation efficiency, we also measure the number of rejected tokens during generation, which reflects both the acceptance rate and the total length of generation.

Implementation details. We used the Qwen 3 series as our main testbed for two reasons: (1) the models come in a variety of sizes, ranging from 0.6B to 14B parameters, enabling a wide range of choices of model pairs; (2) the models are trained to generate with internalized chain-of-thought reasoning (Wei et al., 2022), which makes them a natural use case for speculative sampling given the longer generation lengths (Yang et al., 2025b). For all experiments, we used the recommended generation parameters (Yang et al., 2025a), where top-p is set to 0.95, top-k equals 20, and temperature is 0.6.

Table 1: The results on three benchmarks: GSM8K, IFEval, and GPQA. We report the "strict-match" accuracy as the score with the standard regex pattern for each task. AL$_m$ indicates the number of accepted tokens when the draft length is m. Rej denotes the total number of rejected tokens throughout generation in relative scale, where we use the SpS runs as the reference (labeled as "Ref").

(a) The results of Qwen 3 8B as verifier and Qwen 3 0.6B as drafter.

		GSM8K			IFEval			GPQA		
m	Name	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Rej$^{\downarrow}$	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Rej$^{\downarrow}$	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Rej$^{\downarrow}$
	Verifier	84.31$_{\pm 0.47}$	-	-	84.66$_{\pm 0.56}$	-	-	41.07$_{\pm 1.77}$	-	-
10	SpS	83.78	4.49	Ref	84.66	2.59	Ref	40.91	3.70	Ref
	TAS	86.58	5.49	-29%	85.40	3.28	-27%	41.41	5.17	-42%
	Cactus 0.75	85.97	5.65	-34%	85.03	3.40	-31%	41.42	5.33	-47%
	Cactus 1.0	86.35	5.72	-37%	84.10	3.44	-32%	39.39	5.44	-48%
20	SpS	84.46	5.44	Ref	84.10	2.74	Ref	42.93	4.23	Ref
	TAS	85.51	7.23	-35%	84.10	3.77	-29%	38.89	6.68	-46%
	Cactus 0.75	86.66	7.50	-37%	85.95	3.76	-30%	40.01	6.89	-47%
	Cactus 1.0	86.43	7.61	-39%	84.84	4.05	-33%	39.90	7.05	-49%

(b) The results of Qwen 3 14B as verifier and Qwen 3 0.6B as drafter.

		GSM8K			IFEval			GPQA		
m	Name	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Rej$^{\downarrow}$	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Rej$^{\downarrow}$	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Rej$^{\downarrow}$
	Verifier	91.71$_{\pm 0.52}$	-	-	85.09$_{\pm 0.66}$	-	-	40.07$_{\pm 0.77}$	-	-
10	SpS	91.12	4.27	Ref	85.03	2.19	Ref	39.39	3.37	Ref
	TAS	92.65	5.24	-30%	86.14	3.00	-25%	38.89	4.99	-46%
	Cactus 0.75	92.12	5.35	-31%	86.87	3.04	-29%	44.95	5.14	-50%
	Cactus 1.0	93.10	5.44	-32%	85.96	3.03	-30%	43.43	5.16	-51%
20	SpS	91.89	5.11	Ref	84.47	2.27	Ref	40.91	3.84	Ref
	TAS	92.87	6.78	-32%	85.03	3.49	-27%	40.40	6.41	-46%
	Cactus 0.75	92.15	7.15	-36%	86.69	3.45	-30%	45.46	6.46	-46%
	Cactus 1.0	92.87	7.00	-34%	86.32	3.60	-30%	45.46	6.74	-50%

3.2 MAIN RESULTS

As shown in Table 1, speculative sampling (SpS) serves as a strong baseline that closely preserves the output distribution of the verifier model. Across all three benchmarks (GSM8K, IFEval, and GPQA), SpS maintains similar accuracies to the verifier (e.g., 84.46 vs. 84.31 on GSM8K with $m = 20$ in Table 1a, and 91.89 vs. 91.71 in Table 1b). This aligns with the theoretical claim that SpS is nearly lossless in generation quality. Additionally, the number of accepted tokens (AL$_m$) for SpS reaches 5.44 on GSM8K and 4.23 on GPQA with $m = 20$, indicating that the verifier model is invoked less frequently.

Typical acceptance sampling (TAS) outperforms SpS in terms of acceptance rate, achieving more accepted tokens and lower rejection rates. For example, on GSM8K with $m = 20$, TAS improves AL$_m$ from 5.44 to 7.23 (Table 1a) and reduces the rejection rate by 35%, which is consistent with our approximation analysis in Section 2.2. However, TAS often introduces distributional shifts that degrade performance. For instance, on GPQA in Table 1a, TAS yields lower accuracy than SpS (38.89 vs. 42.93), likely due to accepting plausible yet suboptimal tokens, especially when the verifier distribution contains fine-grained decision signals.

In contrast, our proposed method, Cactus, achieves the highest acceptance rates across all benchmarks while maintaining or improving accuracy. When $\delta = 0.75$, Cactus consistently surpasses both SpS and TAS in AL$_m$, achieving 86.66 on GSM8K with $m = 20$ (Table 1a) and 45.46 on GPQA with $m = 20$ (Table 1b), notably outperforming all baselines. When $\delta = 1.0$, Cactus further

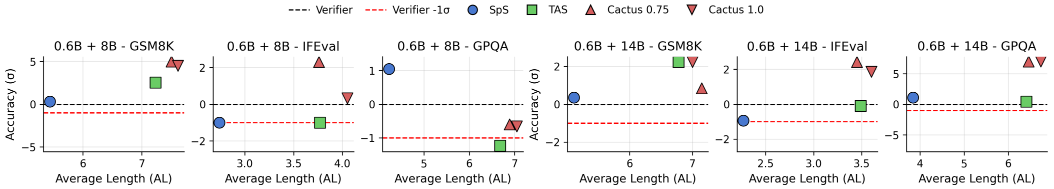

Figure 1: Accuracy-acceptance across benchmarks and model settings. The x-axis shows the average accepted length (AL), and the y-axis shows accuracy normalized by the standard deviation from the verifier.

increases AL_m to 7.61 on GSM8K with 86.43 accuracy (Table 1a), or to 7.00 with 92.87 accuracy using a larger verifier (Table 1b). Notably, unlike TAS, Cactus does not degrade performance on challenging benchmarks such as GPQA. Instead, it achieves both high acceptance rates and stable accuracy, validating its theoretical foundation in constrained optimization and demonstrating practical robustness across diverse tasks.

3.3 IN-DEPTH ANALYSES

Accuracy against acceptance rates. We visualize the accuracy-acceptance trade-off in Figure 1, where accuracy is measured in standard deviations (σ) from the verifier mean, and throughput is quantified by the average accepted length (AL). Each subplot corresponds to a specific benchmark and verifier-drafter pair. The dashed black line indicates the verifier performance, and the red dashed line marks the -1σ threshold, a commonly used indicator of statistically significant degradation.

As shown, TAS improves throughput over SpS but often suffers from accuracy drops, notably falling below the -1σ threshold on GPQA with the 8B verifier. In contrast, Cactus consistently preserves accuracy (remaining within or above the verifier confidence range) and frequently exceeds it, such as on GSM8K and IFEval with both 8B and 14B verifiers. This demonstrates that Cactus effectively improves decoding efficiency without compromising (and sometimes even enhancing) generation quality.

It is also worth noting that the improvements from Cactus are stable across tasks with different characteristics. For instance, on challenging benchmarks like GPQA, where other methods either exhibit significant degradation (e.g., TAS) or achieve limited throughput gains (e.g., SpS), Cactus substantially increases AL while maintaining accuracy above baseline. This highlights the strength of our constrained acceptance framework in balancing aggressive token acceptance with distributional fidelity.

The importance of divergence control. Our Cactus dynamically manipulates the target distribution to increase the chance of accepting the sample tokens. Since this inevitably pushes the target distribution h from the verifier q to be more similar to the draft model p, it resembles the notion of mixing the distribution q and p like interpolation. However, we argue that Cactus is superior than simple interpolation, given that it uses a principled approach which comes with a divergence guarantee. We empirically justify this argument by the following experiment.

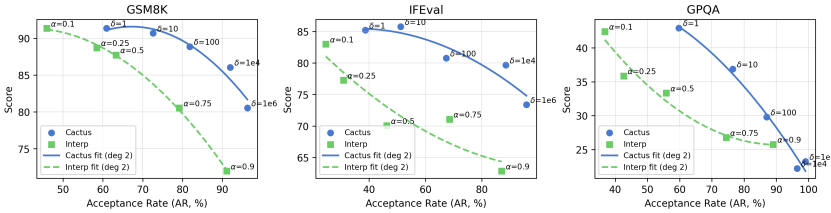

Figure 2: Score vs. acceptance rate for the 0.6B+14B Qwen3 combination without top-p/top-k sampling arguments. Solid and dashed lines are degree-2 polynomial fits.

Here, we produces data points by running grid search on δ for Cactus and interpolation rate α for interpolation, respectively. As shown, Cactus consistently outperforms interpolation at the similar acceptance rate. For example, at a similar acceptance rate of approximately 90%, Cactus achieves a score above 86 ($\delta = 1e4$) on GSM8K, whereas interpolation only reaches a score below 72 ($\alpha = 0.9$). Notably, even at a 96.3% acceptance rate, Cactus maintains a higher score (above 80), further confirming the necessity of divergence control of our method.

Throughput comparison. In Section 3.2, we used AL_m and Rej as proxies for throughput, as they are invariant to hardware and system conditions. Here, we additionally report wall-time results, all measured on A100 40GB GPUs with identical CPU and memory configurations. We used VLLM (Kwon et al., 2023) with its default compilation settings to ensure realistic inference conditions. The results on GPQA are shown in Figure 3.

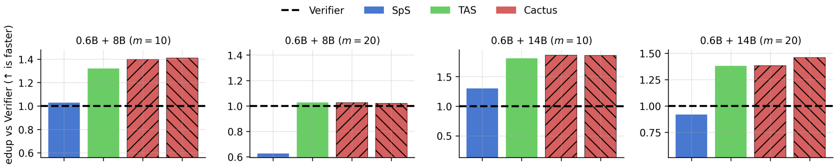

Figure 3: Wall-time normalized throughput (y-axis) across different model sizes and draft lengths. The wall time of a single verifier model is always normalized to 1.

Across all settings, Cactus remains competitive or superior to all baselines. In particular, Cactus 0.75 and 1.0 yield significant improvements in the 0.6B+14B setting, where Cactus 1.0 achieves nearly 1.9× speedup over the verifier alone with $m = 10$, while also maintaining the highest score on GPQA (see Table 1b). In contrast, TAS slightly underperforms Cactus in nearly all settings. Notably, as discussed in Section 2.2 and verified in Table 1b, TAS lacks explicit divergence control. These results highlight the benefit of Cactus's constrained acceptance strategy, which more effectively balances fidelity and efficiency than existing baselines.

Evaluating on more model series. To assess the generality of our method, we go beyond Qwen 3 and evaluate three additional model series: Gemma (2B + 9B, Team et al. (2024)), R1 (1.5B + 7B, DeepSeek-AI et al. (2025)), and LLaMA (1B + 8B, Dubey et al. (2024)). Each model pair represents a distinct series developed by different teams with varying training methodologies. Following Bachmann et al. (2025), we additionally evaluate Top-k decoding as a naive lossy baseline, where draft tokens are accepted if they fall within the top-5 candidates according to the verifier. All drafter-verifier pairs follow the same speculative decoding setup, and accuracy is measured with standard task-specific metrics. We also include SpS and TAS baselines under equivalent configurations to ensure a fair comparison. The results are presented in Figure 4.

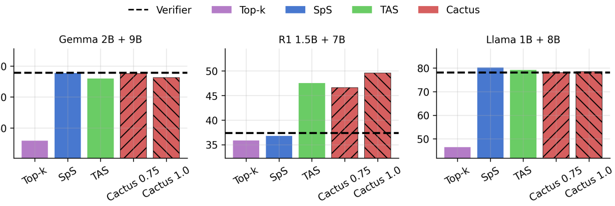

Figure 4: Evaluating on GSM8K with three model pairs.

Top-k decoding consistently underperforms the verifier model, reaffirming the importance of using principled verifier-guided sampling like Cactus. Across all settings, Cactus delivers strong and consistent performance. For R1 and Gemma, Cactus notably outperforms TAS. While SpS and TAS perform well on LLaMA, Cactus matches their accuracy and retains its robustness across models.

These results support the conclusion that Cactus generalizes well across diverse architectures and remains competitive or superior regardless of the underlying model series.

4 RELATED WORK

The draft-and-verify scheme. The line of work most closely related to this paper is the use of the draft-and-verify scheme to accelerate auto-regressive decoding. The foundation of this scheme lies in the acceptance algorithms (i.e., designing the acceptance rate and recovery probability functions in Section 2). This includes vanilla speculative sampling (Chen et al., 2023; Leviathan et al., 2023) and typical acceptance sampling (Hewitt et al., 2022; Meister et al., 2023; Cai et al., 2024a). Cactus, as it applies a different acceptance strategy, belongs to the same category. For this reason, we extensively compare it against both methods in our paper. In addition to acceptance algorithms, building specialized models for this scheme has shown to be effective (Kim et al., 2023; Liu et al., 2024a; Liao et al., 2025). For instance, Cai et al. (2024a) fine-tune multiple heads for generating subsequent tokens; Li et al. (2024b;c; 2025) propose EAGLE, which introduces an additional head for draft token generation; Bachmann et al. (2025) propose Judge Decoding, training a binary classifier to augment the acceptance rate function. However, these methods require substantial training resources, whereas Cactus is a training-free acceptance rule. We also expect that Cactus can be directly applied to that utilizes either SpS or TAS as the underlying principle. Another generalization of speculative sampling involves using multiple draft tokens or verifiers (Yang et al., 2024; Chen et al., 2024; Jeon et al., 2024). For example, Miao et al. (2023) propose SpecInfer with tree-based draft generation; TreeBoN (Qiu et al., 2024) integrates speculative sampling into best-of-N tree-search decoding. We leave the exploration on more integrated versions of multi-drafter or multi-verifier Cactus to future work.

Low-complexity attention for Transformers. Transformer models generate sequences in an auto-regressive manner (Vaswani et al., 2017). Since each token attends to all previous ones, generation time grows quadratically with sequence length (Wang et al., 2020). To address this, previous work has proposed low-complexity attention variants (Child et al., 2019; Zaheer et al., 2020; Tsai et al., 2019; Kitaev et al., 2020; Choromanski et al., 2021). These methods modify the Transformer architecture itself. Cactus can be combined with these methods since they also follow the auto-regressive paradigm. In addition to architectural changes, decoding complexity can also be reduced by manipulating the KV cache (Zhang et al., 2023; Li et al., 2024a; Cai et al., 2024b). For instance, SnapKV (Li et al., 2024a) evicts less relevant tokens from the prompt before generation; Radar (Hao et al., 2025) dynamically selects key segments using random projections. These techniques are drop-in approximations of vanilla attention and are orthogonal to speculative sampling methods like Cactus.

Minimizing overheads of Transformers. Without approximating the Transformer architecture, overheads can still be reduced to accelerate decoding. Flash Attention (Dao et al., 2022; Dao, 2023), for example, uses tiling techniques to avoid memory-bound operations, and has seen widespread adoption (Wolf et al., 2019; Kwon et al., 2023). Memory-efficient attention (Rabe and Staats, 2021) reorders computation to maintain constant memory usage regardless of context length. Another line of work applies quantization to model parameters (Lin et al., 2023; Badri and Shaji, 2023; Liu et al., 2024b). The benefits are threefold: (1) reduced memory footprint due to lower-precision data types; (2) alleviated memory bottlenecks during decoding; and (3) improved hardware efficiency via optimized kernels. All these methods can be seamlessly integrated into speculative sampling approaches, including Cactus.

5 CONCLUSION

In this paper, we propose a constrained optimization framework for analyzing and improving speculative sampling methods. Building upon this framework, we introduce Cactus, a novel training-free speculative sampling method that increases acceptance rates while maintaining a provably controlled divergence from the large verifier model. Cactus uses only basic element-wise operations, making it highly practical and lightweight for real-time inference. We empirically evaluate our method on a variety of benchmarks and confirm its effectiveness. As LLMs continue to grow in size and cost, our method provides a theoretically grounded yet practically efficient solution for scalable deployment.

ETHICS STATEMENT

We certify that all authors of this project adhere to the ICLR Code of Ethics (`https://iclr.cc/public/CodeOfEthics`). Our research does not involve human subjects, practices to dataset releases, potentially harmful content, potential conflicts of interest and sponsorship, discrimination/bias/fairness concerns, privacy and security issues, legal compliance, or research integrity issues.

REPRODUCIBILITY STATEMENT

All of our experiments use publicly accessible datasets and models. Specifically, the datasets we used can be found by the following links via HuggingFace

- GSM8K: `https://huggingface.co/datasets/openai/gsm8k`
- IFEval: `https://huggingface.co/datasets/google/IFEval`
- GPQA: `https://huggingface.co/datasets/Idavidrein/gpqa`

The models can be found by the following links

- Qwen3 0.6B: `https://huggingface.co/Qwen/Qwen3-0.6B`
- Qwen3 8B: `https://huggingface.co/Qwen/Qwen3-8B`
- Qwen3 14B: `https://huggingface.co/Qwen/Qwen3-14B`
- Gemma 2B: `https://huggingface.co/google/gemma-2-2b`
- Gemma 9B: `https://huggingface.co/google/gemma-2-9b`
- R1 1.5B: `https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B`
- R1 7B: `https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B`
- Llama 1B: `https://huggingface.co/meta-llama/Llama-3.2-1B`
- Llama 8B: `https://huggingface.co/meta-llama/Llama-3.1-8B`

In addition, our code is publicly available at the anonymous link `https://anonymous.4open.science/r/Cactus-2E4D/`.

REFERENCES

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas K Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *ICLR*, 2025. URL `https://openreview.net/forum?id=mtSSFiqW6y`.

Hicham Badri and Appu Shaji. Half-quadratic quantization of large machine learning models, November 2023. URL `https://mobiusml.github.io/hqq_blog/`.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, J. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. Henighan, R. Child, A. Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, S. Gray, B. Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, I. Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. URL `https://arxiv.org/abs/2005.14165`.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*, 2024a. URL `https://arxiv.org/abs/2401.10774`.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv: 2406.02069*, 2024b. URL `https://arxiv.org/abs/2406.02069`.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. URL `https://arxiv.org/abs/2302.01318`.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chang, and Jie Huang. Cascade speculative drafting for even faster llm inference. *NeurIPS*, 37:86226–86242, 2024. URL `https://proceedings.neurips.cc/paper_files/paper/2024/hash/9cb5b083ba4f5ca6bd05dd307a2fb354-Abstract-Conference.html`.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse Transformers. *arXiv preprint arXiv:1904.10509*, 2019. URL `https://arxiv.org/abs/1904.10509`.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021. URL `https://openreview.net/forum?id=Ua6zuk0WRH`.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv: 2110.14168*, 2021. URL `https://arxiv.org/abs/2110.14168`.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. URL `https://arxiv.org/abs/2307.08691`.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. *NeurIPS*, 35:16344–16359, 2022. URL `https://arxiv.org/abs/2205.14135`.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang

Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv: 2501.12948*, 2025. URL `https://arxiv.org/abs/2501.12948`.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, and et al. The Llama 3 herd of models. *arXiv preprint arXiv: 2407.21783*, 2024. URL `https://arxiv.org/abs/2407.21783`.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Yongchang Hao, Mengyao Zhai, Hossein Hajimirsadeghi, Sepidehsadat Hosseini, and Frederick Tung. Radar: Fast long-context decoding for any Transformers. In *ICLR*, 2025. URL `https://openreview.net/forum?id=ZTpWOwMrzQ`.

John Hewitt, Christopher D. Manning, and Percy Liang. Truncation sampling as language model desmoothing. *arXiv preprint arXiv: 2210.15191*, 2022. URL `https://arxiv.org/abs/2210.15191`.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019. URL `https://arxiv.org/abs/1904.09751`.

Yunhai Hu, Zining Liu, Zhenyuan Dong, Tianfan Peng, Bradley McDanel, and Sai Qian Zhang. Speculative decoding and beyond: An in-depth survey of techniques. *arXiv preprint arXiv:2502.19732*, 2025. URL `https://arxiv.org/abs/2502.19732`.

Wonseok Jeon, Mukul Gagrani, Raghavv Goel, Junyoung Park, Mingu Lee, and Christopher Lott. Recursive speculative decoding: Accelerating llm inference via sampling without replacement. *ICLR 2024 Workshop on Large Language Model (LLM) Agents*, 2024. URL `https://arxiv.org/abs/2402.14160`.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL `https://arxiv.org/abs/2001.08361`.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. Speculative decoding with big little decoder. *NeurIPS*, 36:39236–39256, 2023. URL `https://proceedings.neurips.cc/paper_files/paper/2023/hash/7b97adeafa1c51cf65263459ca9d0d7c-Abstract-Conference.html`.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient Transformers. In *ICLR*, 2020. URL `https://openreview.net/forum?id=rkgNKkHtvB`.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *ACM SIGOPS*, 2023. URL `https://arxiv.org/abs/2309.06180`.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from Transformers via speculative decoding. In *ICML*, pages 19274–19286, 2023. URL `https://arxiv.org/abs/2211.17192`.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. *NeruIPS*, 2024a. URL `https://arxiv.org/abs/2404.14469`.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. *ICML*, 2024b. URL `https://arxiv.org/abs/2401.15077`.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv: 2406.16858*, 2024c. URL `https://arxiv.org/abs/2406.16858`.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *NeurIPS*, 2025. URL `https://openreview.net/forum?id=4exx1hUffq`.

Baohao Liao, Yuhui Xu, Hanze Dong, Junnan Li, Christof Monz, Silvio Savarese, Doyen Sahoo, and Caiming Xiong. Reward-guided speculative decoding for efficient LLM reasoning. *arXiv preprint arXiv: 2501.19324*, 2025. URL `https://arxiv.org/abs/2501.19324`.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv preprint arXiv: 2306.00978*, 2023. URL `https://arxiv.org/abs/2306.00978v5`.

Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding via double early exiting. *arXiv preprint arXiv:2404.18911*, 2024a. URL `https://arxiv.org/abs/2404.18911`.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. SpinQuant: LLM quantization with learned rotations. *arXiv preprint arXiv: 2405.16406*, 2024b. URL `https://arxiv.org/abs/2405.16406`.

Clara Meister, Tim Vieira, and Ryan Cotterell. If beam search is the answer, what was the question? *arXiv preprint arXiv:2010.02650*, 2020. URL `https://arxiv.org/abs/2010.02650`.

Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Locally typical sampling. *TACL*, 11:102–121, 2023. URL `https://doi.org/10.1162/tacl_a_00536`.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781*, 2023. URL `https://arxiv.org/abs/2305.09781`.

Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Seungyeon Kim, Neha Gupta, Aditya Krishna Menon, and Sanjiv Kumar. Faster cascades via speculative decoding. In *ICLR*, 2025. URL `https://openreview.net/forum?id=vo9t20wsmd`.

Jiahao Qiu, Yifu Lu, Yifan Zeng, Jiacheng Guo, Jiayi Geng, Huazheng Wang, Kaixuan Huang, Yue Wu, and Mengdi Wang. TreeBoN: Enhancing inference-time alignment with speculative tree-search and best-of-n sampling. *arXiv preprint arXiv:2410.16033*, 2024. URL `https://arxiv.org/abs/2410.16033`.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv: 2412.15115*, 2024. URL `https://arxiv.org/abs/2412.15115`.

Markus N. Rabe and Charles Staats. Self-attention does not need $o(n^2)$ memory. *arXiv preprint arXiv: 2112.05682*, 2021. URL `https://arxiv.org/abs/2112.05682`.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. *arXiv preprint arXiv: 2311.12022*, 2023. URL `https://arxiv.org/abs/2311.12022`.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *NeurIPS*, 31, 2018. URL `https://arxiv.org/abs/1811.03115`.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024. URL `https://arxiv.org/abs/2403.08295`.

Vivien Tran-Thien. An optimal lossy variant of speculative decoding, 2023. URL `https://vivien000.github.io/blog/journal/a-provably-optimal-lossy-variant-of-speculative-decoding.html`.

Yao-Hung Hubert Tsai, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency, and Ruslan Salakhutdinov. Transformer dissection: An unified understanding for Transformers's attention via the lens of kernel. In *EMNLP-IJCNLP*, pages 4344–4353, 2019. URL `https://aclanthology.org/D19-1443`.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, volume 30, 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf`.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. URL `https://arxiv.org/abs/2006.04768`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In *NeurIPS*, 2022. URL `https://openreview.net/forum?id=_VjQlMeSB_J`.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv: 1910.03771*, 2019. URL `https://arxiv.org/abs/1910.03771`.

Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. *arXiv preprint arXiv:2203.16487*, 2022. URL `https://arxiv.org/abs/2203.16487`.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of ACL 2024*, pages 7655–7671, 2024. URL `https://aclanthology.org/2024.findings-acl.456`.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv: 2505.09388*, 2025a. URL `https://arxiv.org/abs/2505.09388`.

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*, 2024. URL `https://arxiv.org/abs/2401.06706`.

Wang Yang, Xiang Yue, Vipin Chaudhary, and Xiaotian Han. Speculative thinking: Enhancing small-model reasoning with large model guidance at inference time. *arXiv preprint arXiv: 2504.12329*, 2025b. URL `https://arxiv.org/abs/2504.12329`.

Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, Zhikai Li, Qingyi Gu, Yong Jae Lee, Yan Yan, Beidi Chen, Guangyu Sun, and Kurt Keutzer. Llm inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv: 2402.16363*, 2024. URL `https://arxiv.org/abs/2402.16363`.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *NeurIPS*, 33:17283–17297, 2020. URL `https://arxiv.org/abs/2007.14062`.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. *NeruIPS*, 36, 2023. URL `https://arxiv.org/abs/2306.14048`.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv: 2311.07911*, 2023. URL `https://arxiv.org/abs/2311.07911`.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024. URL `https://arxiv.org/abs/2404.14294`.

A TECHNICAL PROOFS

A.1 PROOF OF THEOREM 1

Observation 1. *Given any desired target distribution h and draft model p, the acceptance rate and recovery probability are defined as*

$$\phi(x_t|\boldsymbol{x}_{<t}) = \min\left(\frac{h(x_t|\boldsymbol{x}_{<t})}{p(x_t|\boldsymbol{x}_{<t})}, 1\right) \tag{1}$$

$$\text{and} \quad g(x_t|\boldsymbol{x}_{<t}) = \frac{h(x_t|\boldsymbol{x}_{<t}) - p(x_t|\boldsymbol{x}_{<t})\phi(x_t|\boldsymbol{x}_{<t})}{\mathbb{E}_{x'\sim p}[1-\phi(x'|\boldsymbol{x}_{<t})]} \tag{2}$$

respectively. Algorithm 1 samples from h exactly using the above ϕ and g. In addition, this ϕ is the optimal design of acceptance rate.

Proof. Let n be the selected token and $\boldsymbol{x}$ be the context. Then at each step, the resulting distribution of the algorithm is:

$$\begin{aligned}\Pr(n|\boldsymbol{x}) =& \Pr(n \sim p(\cdot|\boldsymbol{x}) \text{ and } u \le \phi(x|\boldsymbol{x})) + \sum_{i=1}^{|V|} p(i|\boldsymbol{x}) \Pr(n \sim g(\cdot|\boldsymbol{x}) \text{ and } u > \phi(i|\boldsymbol{x})) &(17)\\ =& p(n|\boldsymbol{x})\phi(n|\boldsymbol{x}) + g(n|\boldsymbol{x}) \mathop{\mathbb{E}}_{i\sim p(\cdot|\boldsymbol{x})}[1-\phi(i|\boldsymbol{x})], &(18)\end{aligned}$$

where the first term on the right hand side indicates the sampled token n is accepted. The second term means the originally sampled token is rejected, and the current token n comes from the recover probability g. Since we would like $\Pr(n|\boldsymbol{x}) = h(n|\boldsymbol{x})$, we have

$$\begin{aligned} p(n|\boldsymbol{x})\phi(n|\boldsymbol{x}) + g(n|\boldsymbol{x}) \mathop{\mathbb{E}}_{i\sim p(\cdot|\boldsymbol{x})}[1-\phi(i|\boldsymbol{x})] = h(n|\boldsymbol{x}) &\quad (19)\\ \Longleftrightarrow g(n|\boldsymbol{x}) = \frac{h(n|\boldsymbol{x}) - p(n|\boldsymbol{x})\phi(n|\boldsymbol{x})}{\mathbb{E}_{i\sim p(\cdot|\boldsymbol{x})}[1-\phi(i|\boldsymbol{x})]}, &\quad (20)\end{aligned}$$

hence proving the expression for g. Here, ϕ can be function that maps to $[0,1]$ that makes g a distribution. Since the expression of g is self-normalizing, we only need to make sure that all $g(i|\boldsymbol{x})$ are non-negative. Specifically,

$$\begin{aligned} & 0 \le g(i|\boldsymbol{x}) &(21)\\ \Longleftarrow\ & h(i|\boldsymbol{x}) - p(i|\boldsymbol{x})\phi(i|\boldsymbol{x}) \ge 0 &(\text{Image}(\phi) \subseteq [0,1])\\ \Longleftarrow\ & \phi(i|\boldsymbol{x}) \le \frac{h(i|\boldsymbol{x})}{p(i|\boldsymbol{x})}. &(22)\end{aligned}$$

Again, given $\text{Image}(\phi) \subseteq [0,1]$,

$$\phi(i|\boldsymbol{x}) \le \min\left(\frac{h(i|\boldsymbol{x})}{p(i|\boldsymbol{x})}, 1\right). \tag{23}$$

gives the optimal acceptance rate. □

A.2 PROOF OF THEOREM 3

Before proceeding to the proof of the theorem, we first show the following technical lemma.

Lemma 8 (Minimal Divergence Allocation)**.** *Let $f : \mathbb{R}_+ \to \mathbb{R}$ be convex with $f(1) = 0$. For any $\alpha \in [0,1]$ and sub-distribution $\{q(i)\}_{i\in S}$ over S with $Q := \sum_{i\in S} q(i) > 0$, the solution to:*

$$\min_{\{h(i)\}} \quad \sum_{i\in S} q(i) f\left(\frac{h(i)}{q(i)}\right) \tag{24}$$

$$\text{s.t.} \quad \sum_{i\in S} h(i) = \alpha, \quad h(i) \ge 0 \tag{25}$$

is $h^*(i) = \frac{\alpha}{Q} q(i)$ for all $i \in S$.

Proof. Let $\lambda := \frac{\alpha}{Q}$. Define $\tilde{h}(i) := \lambda q(i)$. Then:

$$\sum_{i \in S} \tilde{h}(i) = \lambda Q = \alpha \tag{26}$$

satisfies the constraints. For any feasible $h \neq \tilde{h}$, define $r(i) := \frac{h(i)}{q(i)}$. By Jensen's inequality:

$$\frac{1}{Q} \sum_{i \in S} q(i) f(r(i)) \geq f\left(\frac{1}{Q} \sum_{i \in S} q(i) r(i)\right) = f\left(\frac{\alpha}{Q}\right) \tag{27}$$

with equality iff $r(i) = \lambda$ for all $i \in S$. Thus $\tilde{h}$ is the unique minimizer. □

We can now show the theorem below.

Theorem 3. *The optimal $\boldsymbol{h}$ in Definition 2 is*

$$h_i = \begin{cases} \gamma^*, & \textit{if } i = n, \\ \frac{1-\gamma^*}{1-q(n|\boldsymbol{x}_{<t})} q(i|\boldsymbol{x}_{<t}), & \textit{otherwise}, \end{cases} \tag{6}$$

where γ^* is any root of the equation

$$\delta = q(n|\boldsymbol{x}_{<t}) f\left(\frac{\gamma}{q(n|\boldsymbol{x}_{<t})}\right) + (1 - q(n|\boldsymbol{x}_{<t})) f\left(\frac{1-\gamma}{1 - q(n|\boldsymbol{x}_{<t})}\right) \tag{7}$$

over the interval $[q(n|\boldsymbol{x}_{<t}), +\infty)$, clamped into $[q(n|\boldsymbol{x}_{<t}), 1]$. The function f is the one used in the definition of f-divergence.

Proof.

$$\max_{\boldsymbol{h}} \ \min\left(\frac{h_n}{p(n|\boldsymbol{x}_{<t})}, 1\right) \tag{28}$$

$$\text{s.t. } \boldsymbol{h} \in \Delta^{|V|-1}, \tag{29}$$

$$D_f(\boldsymbol{h} \| q(\cdot|\boldsymbol{x}_{<t})) \leq \delta. \tag{30}$$

Here, $\Delta^{|V|-1}$ denotes the probability simplex, and

$$D_f(\boldsymbol{h} \| q) = \sum_{i \in V} q(i) f\left(\frac{h(i)}{q(i)}\right) \tag{31}$$

is the f-divergence. The objective

$$\min\left(\frac{h_n}{p(n)}, 1\right) \tag{32}$$

is maximized when $\frac{h_n}{p(n)}$ is as large as possible. However, since $\min(\cdot, 1)$ caps the value at 1, the maximum achievable is 1 (when $h_n \geq p(n)$). Thus, the problem reduces to maximizing h_n under the constraints, as increasing h_n directly improves the objective until $h_n \geq p(n)$. To maximize h_n, we allocate as much probability mass to h_n as allowed by the constraints. Let $\gamma = h_n$. The remaining mass $1 - \gamma$ must be distributed over $i \neq n$. By Lemma 8, the optimal allocation for $i \neq n$ is:

$$h(i) = \frac{1-\gamma}{1-q(n)} q(i), \tag{33}$$

where $\frac{1-\gamma}{1-q(n)}$ ensures $\sum_{i \neq n} h(i) = 1 - \gamma$. Thus, for $i \neq n$:

$$h(i) = \frac{1-\gamma}{1-q(n)} q(i), \tag{34}$$

where $\frac{1-\gamma}{1-q(n)}$ ensures $\sum_{i \neq n} h(i) = 1 - \gamma$.

Substitute $h_n = \gamma$ and $h(i) = \frac{1-\gamma}{1-q(n)} q(i)$ into $D_f(\boldsymbol{h}\|q)$:

$$D_f = q(n) f\left(\frac{\gamma}{q(n)}\right) + \sum_{i \neq n} q(i) f\left(\frac{1-\gamma}{1-q(n)}\right). \tag{35}$$

Simplify the second term using $\sum_{i \neq n} q(i) = 1 - q(n)$:

$$D_f = q(n) f\left(\frac{\gamma}{q(n)}\right) + (1 - q(n)) f\left(\frac{1-\gamma}{1-q(n)}\right). \tag{36}$$

The constraint $D_f \leq \delta$ becomes an equality at optimality (since increasing γ further would violate the constraint). Thus, γ^* solves:

$$q(n) f\left(\frac{\gamma}{q(n)}\right) + (1 - q(n)) f\left(\frac{1-\gamma}{1-q(n)}\right) = \delta. \tag{37}$$

Finally, since γ^* may exceed 1 (when δ is set too large to attain), it is truncated into $[q(n), 1]$ as a proper probability value. $\square$

A.3 PROOF OF THEOREM 4

Theorem 4. *Let ϕ_n and g_n denote the functions that follow the solution in Theorem 3 when the sampled token is n. The distribution of the overall algorithm is given by*

$$\boldsymbol{h}_{alg} = \sum_{n \in [|V|]} p(n|\boldsymbol{x}_{<t}) \left[\phi_n(n)\boldsymbol{e}_n + (1 - \phi_n(n))\boldsymbol{g}_n\right], \tag{8}$$

where $\mathbf{e}_n$ is a one-hot vector with only non-zero element at index n. In addition,

$$D_f(\boldsymbol{h}_{alg}\|q(\cdot|\boldsymbol{x}_{<t})) \leq \min\{\Gamma(\delta), D_f(p(\cdot|\boldsymbol{x}_{<t})\|q(\cdot|\boldsymbol{x}_{<t}))\} \tag{9}$$

for any $\delta \geq 0$. Here, the function $\Gamma : [0, +\infty) \to [0, +\infty]$ is continuous and non-decreasing in δ with a value of 0 at $\delta = 0$.

Proof. We work at a single step at t and suppress the context $\boldsymbol{x}_{<t}$. Fix p and q on a finite alphabet. For each drafted index n, let h_n be any target with $D_f(h_n\|q) \leq \delta$. The conditional output is

$$\boldsymbol{r}_n = \phi_n(n)\boldsymbol{e}_n + (1 - \phi_n(n))\boldsymbol{g}(h_n),$$

and the algorithm's one-step output is

$$\boldsymbol{h}_{\text{alg}} = \sum_n p(n)\boldsymbol{r}_n.$$

Let

$$\mathcal{H}_\delta := \Big\{(h_n)_n : D_f(h_n\|q) \leq \delta \ \forall n, \ \ q(i) = 0 \Rightarrow h_n(i) = 0\Big\}, \tag{38}$$

$$F\big((h_n)_n\big) := D_f\big(\boldsymbol{h}_{\text{alg}}\|q\big). \tag{39}$$

Define

$$\Gamma(\delta) \ := \sup_{(h_n) \in \mathcal{H}_\delta} F\big((h_n)_n\big) \ \in [0, \infty], \tag{40}$$

and note that Γ depends only on (p, q, f) and the budget δ. By construction,

$$D_f\big(\boldsymbol{h}_{\text{alg}}\|q\big) \ \leq \ \Gamma(\delta) \qquad \text{for every feasible family } (h_n) \in \mathcal{H}_\delta. \tag{41}$$

It is straight-forward to show that $D_f\big(\boldsymbol{h}_{\text{alg}}\|q\big) \leq D_f(p\|q)$ given that the all-acceptance distribution is simply p. Thus it remains to show that Γ has the claimed shape: non-decreasing, $\Gamma(0) = 0$, and continuous in the extended-real sense.

Basic properties of Γ. *(i)* $\Gamma(0) = 0$. If $\delta = 0$ then $h_n = q$ for all n, so $\boldsymbol{h}_{\text{alg}} = q$ and thus $\Gamma(0) = D_f(q\|q) = 0$.

(ii) Monotonicity. If $\delta_2 \geq \delta_1$ then $\mathcal{H}_{\delta_1} \subseteq \mathcal{H}_{\delta_2}$, so $\Gamma(\delta_1) \leq \Gamma(\delta_2)$ by definition of the supremum.

(iii) Continuity. We show right- and left-continuity. On a finite alphabet, the set of probability distributions is compact (a simplex), and with support alignment the feasible set $\mathcal{H}_\delta$ is closed (as the preimage of $[0, \delta]$ under the continuous function $\max_n D_f(\cdot\|q)$) and thus compact. The map $(h_n)_n \mapsto \boldsymbol{h}_{\text{alg}}$ is continuous (operations involved are continuous on their domains), hence F is continuous.

We first show the right-continuity. Let $\delta_k \downarrow \delta$. For each k pick $(h_n^{(k)})_n \in \mathcal{H}_{\delta_k}$ with $F((h_n^{(k)})_n) \geq \Gamma(\delta_k) - \varepsilon_k$, where $\varepsilon_k \downarrow 0$. Since the alphabet is finite, the feasible families live in a finite product of simplices, which is compact; therefore, there exists a subsequence (not relabeled) such that $h_n^{(k)} \to h_n^\star$ for each n. By continuity of $D_f(\cdot\|q)$, $D_f(h_n^\star\|q) = \lim_k D_f(h_n^{(k)}\|q) \leq \lim_k \delta_k = \delta$, so $(h_n^\star)_n \in \mathcal{H}_\delta$. Continuity of F gives

$$\limsup_{k\to\infty} \Gamma(\delta_k) \;\leq\; \lim_{k\to\infty} \big(F((h_n^{(k)})_n) + \varepsilon_k\big) \;=\; F((h_n^\star)_n) \;\leq\; \Gamma(\delta).$$

Monotonicity gives $\Gamma(\delta) \leq \liminf_{k\to\infty} \Gamma(\delta_k)$, hence $\lim_{k\to\infty} \Gamma(\delta_k) = \Gamma(\delta)$.

We then show the left-continuity. Let $\delta_k \uparrow \delta$ and fix $\varepsilon > 0$. Choose $(h_n^\star)_n \in \mathcal{H}_\delta$ with $F((h_n^\star)_n) \geq \Gamma(\delta) - \varepsilon$. For $\theta \in (0, 1)$ define $h_{n,\theta} := (1-\theta)h_n^\star + \theta q$. By convexity of $D_f(\cdot\|q)$ in its first argument,

$$D_f(h_{n,\theta}\|q) \leq (1-\theta)D_f(h_n^\star\|q) + \theta D_f(q\|q) \leq (1-\theta)\delta < \delta,$$

so $(h_{n,\theta})_n \in \mathcal{H}_{(1-\theta)\delta}$. By continuity of F, for sufficiently small $\theta > 0$ we have

$$F((h_{n,\theta})_n) \geq F((h_n^\star)_n) - \varepsilon \geq \Gamma(\delta) - 2\varepsilon.$$

For all large k with $\delta_k > (1-\theta)\delta$, monotonicity gives

$$\Gamma(\delta_k) \;\geq\; \Gamma((1-\theta)\delta) \;\geq\; F((h_{n,\theta})_n) \;\geq\; \Gamma(\delta) - 2\varepsilon.$$

Thus $\liminf_{k\to\infty} \Gamma(\delta_k) \geq \Gamma(\delta)$, and since monotonicity gives $\limsup_{k\to\infty} \Gamma(\delta_k) \leq \Gamma(\delta)$, we have $\lim_{k\to\infty} \Gamma(\delta_k) = \Gamma(\delta)$.

In conclusion, by definition of Γ, for every feasible family $(h_n) \in \mathcal{H}_\delta$,

$$D_f\big(\boldsymbol{h}_{\text{alg}}\|q\big) \leq \Gamma(\delta),$$

with Γ non-decreasing, continuous on $[0, \infty)$, and $\Gamma(0) = 0$. This proves the theorem. ☐

A.4 PROOF OF PROPOSITION 5

Proposition 5. *Typical acceptance sampling (TAS, Cai et al. (2024a)) implicitly solves a variant of the optimization problem in Definition 2, where the f-divergence is substituted with the cross-entropy $H(\boldsymbol{h}, q(\cdot|\boldsymbol{x}_{<t}))$.*

Proof. Given the optimization problem:

$$\begin{aligned} \max_{\boldsymbol{h}} \quad & \min\left(\frac{h_n}{p(n)}, 1\right) \\ \text{s.t.} \quad & \boldsymbol{h} \in \Delta^{|V|-1}, && (42) \\ & H(\boldsymbol{h}, q) \leq H(q) + \delta, && (43) \end{aligned}$$

where $H(q)$ is the entropy. The optimal solution concentrates mass on $\{n, m\}$. Equation (43) is equivalent to

$$\sum_i [q(i) - h(i)] \log \frac{1}{q(i)} \geq -\delta. \tag{44}$$

To maximize $h_n = \gamma$, we must minimize the LHS of (44). Based on Lemma 9, the resulting distribution is always two-point distribution. Let $m = \arg\max_i q(i)$. For fixed $h_n = \gamma$, the optimal allocation places all remaining mass on m:

$$h(i) = \begin{cases} \gamma, & i = n \\ 1-\gamma, & i = m \\ 0, & \text{otherwise} \end{cases} \tag{45}$$

Substituting the optimal form into (43):

$$\begin{aligned} \gamma \log \frac{1}{q(n)} + (1-\gamma)\log\frac{1}{q(m)} &\leq H(q) + \delta \\ \gamma\left(\log\frac{1}{q(n)} - \log\frac{1}{q(m)}\right) &\leq H(q) + \delta - \log\frac{1}{q(m)} \\ \gamma &\leq \frac{H(q) + \delta - \log\frac{1}{q(m)}}{\log\frac{q(m)}{q(n)}} \end{aligned} \tag{46}$$

Sine γ is a probability, its maximum is reached when

$$\begin{aligned} \gamma = 1 &\iff \log\frac{q(m)}{q(n)} \leq H(q) + \delta - \log\frac{1}{q(m)} \\ &\iff q(n) \geq \exp\left(-H(q)\right)\exp(-\delta), \end{aligned} \tag{47}$$

which is the acceptance rate used in TAS.

It should be noted that our theory here is used to reveal the soundness of the TAS acceptance function, without aiming to replicate the exact TAS algorithm. However, based on our framework, one can derive the exact TAS algorithm by adding an $H(\boldsymbol{h}) = 0$ constraint and an ϵ threshold to the cross-entropy limit, which we omitted for simplicity. ☐

In the proof above, we invoked the following technical lemma.

Lemma 9. *For any $\gamma \in [0,1]$, the minimal value of $\sum_{i\neq n}[q(i) - h(i)]\log\frac{1}{q(i)}$ is achieved when:*

$$h(m) = 1-\gamma, \quad h(i) = 0\ \forall i \neq n, m. \tag{48}$$

We provide the proof below.

Proof. Let $h(i) = \alpha_i(1-\gamma)$ for $i \neq n$, where $\sum_i \alpha_i = 1$. Then:

$$\sum_{i\neq n}[q(i) - \alpha_i(1-\gamma)]\log\frac{1}{q(i)} \tag{49}$$

is minimized when α_i concentrates on $m = \arg\max q(i)$, since $\log\frac{1}{q(i)}$ is minimized at $i = m$. ☐

A.5 PROOF OF THEOREM 6

Corollary 6 (Cactus's solution)**.** *Let the f-divergence in Definition 2 be the KL divergence. The solution to Equation* (14) *is given by*

$$h(i|\boldsymbol{x}_{<t}) = \begin{cases} \gamma^*, & \textit{if } i = n, \\ \frac{1-\gamma^*}{1-q(n|\boldsymbol{x}_{<t})}q(i|\boldsymbol{x}_{<t}), & \textit{otherwise}, \end{cases} \tag{15}$$

where $\gamma^* = \min\left\{q(n|\boldsymbol{x}_{<t}) + \sqrt{2\delta q(n|\boldsymbol{x}_{<t})(1-q(n|\boldsymbol{x}_{<t}))}, 1\right\}$.

Proof. We first compute the derivatives of Φ at γ_0:

$$\Phi(\gamma_0) = \Phi'(\gamma_0) = 0, \tag{50}$$

$$\text{and } \Phi''(\gamma_0) = \frac{1}{q(n|\boldsymbol{x}_{<t})(1-q(n|\boldsymbol{x}_{<t}))}. \tag{51}$$

The unique root in $[q(n|\boldsymbol{x}_{<t}), +\infty)$ is then

$$\gamma_0 + \sqrt{\frac{2\delta}{\Phi''(\gamma_0)}} = q(n|\boldsymbol{x}_{<t}) + \sqrt{2\delta q(n|\boldsymbol{x}_{<t})(1-q(n|\boldsymbol{x}_{<t}))}.$$

We clip this value to the interval $[q(n|\boldsymbol{x}_{<t}), 1]$ to ensure validity as a probability. □

A.6 PROOF OF COROLLARY 7

Corollary 7. *When the exact solution γ^* is not greater than* 0.5 *(i.e., the token is not likely to be accepted), our approximation always satisfies the divergence constraint:*

$$D_{KL}(h\|q) \le \delta, \tag{16}$$

where $h(n|\boldsymbol{x}_{<t})$ is given by the approximated solution in Equation (15).

Proof. Let $q := q\big(n \mid \boldsymbol{x}_{<t}\big)$ for brevity and define the quadratic-approximate root

$$\hat{\gamma} := q + \sqrt{2\delta\, q(1-q)}. \tag{52}$$

Because $\Phi'(\gamma) = \log\frac{\gamma}{q} - \log\frac{1-\gamma}{1-q}$, we have $\Phi'(\gamma) > 0$ for every $\gamma \in (q, 1)$; hence Φ is strictly increasing on $[q, 1]$ and the equation $\Phi(\gamma) = \delta$ admits a unique root $\gamma^\star \in (q, 1]$.

Taylor's theorem with the Lagrange remainder, expanded at $\gamma_0 = q$, gives, for some $\xi \in (q, \gamma)$,

$$\Phi(\gamma) = \underbrace{\frac{\Phi''(q)}{2}\,(\gamma-q)^2}_{=:T_2(\gamma)} + \frac{\Phi'''(\xi)}{6}\,(\gamma-q)^3. \tag{53}$$

For the Bernoulli KL,

$$\Phi''(\gamma) = \frac{1}{\gamma(1-\gamma)}, \qquad \Phi'''(\gamma) = -\frac{1-2\gamma}{\gamma^2(1-\gamma)^2}. \tag{54}$$

Whenever $\gamma \le \frac{1}{2}$, the factor $1-2\gamma$ is non-negative and therefore $\Phi'''(\xi) \le 0$. It follows that

$$\Phi(\gamma) \le T_2(\gamma) = \frac{(\gamma-q)^2}{2q(1-q)}, \qquad \forall \gamma \in (q, \tfrac{1}{2}]. \tag{$*$}$$

Choose $\hat{\gamma}$ such that $T_2(\hat{\gamma}) = \delta$, this yields the expression given above. If $\hat{\gamma} \le \frac{1}{2}$ or equivalently

$$\delta \le \frac{(1/2-q)^2}{2q(1-q)} \tag{55}$$

then the above inequality gives $\Phi(\hat{\gamma}) < \delta$. Since Φ is strictly increasing, we obtain

$$\hat{\gamma} < \gamma^\star. \tag{56}$$

This result ensures that our approximation never overestimate γ when the verifier model is not confident about the current sampled token. □

B ADDITIONAL EXPERIMENTS

Mentored decoding. A blog post proposed Mentored decoding (Tran-Thien, 2023), which uses binary search to generate a target distribution $\tilde{q}$ such that $D_{\mathrm{KL}}(q\|\tilde{q}) \leq \delta$ is met. Compared with Cactus, there are two major differences: (1) Mentored decoding allows sampled tokens to be accepted even when the verifier has zero probability, violating the principle of adhering to the verifier's mode; (2) more importantly, the solution is found via a numerical optimization procedure, significantly slowing down the decoding speed and defying the purpose of high-throughput decoding. We conduct additional experiments to compare Cactus and Mentored decoding (using $\delta = 1$ as recommended).

As shown in Table 2, Mentored decoding has the least acceptance rate gain at the cost of increasing the per-step generation time. For example, on GSM8K, the overall wall time is even longer than that of the naive SpS method by 20

Speculative cascading. More recently, Narasimhan et al. (2025) proposed speculative cascading (SpecCas), which dynamically decides if the sampled token will be verified by the large model based on the difference between the two distributions. Essentially, it is mathematically equivalent to mixing the draft and verifier distributions as the target distribution at different steps. We therefore conduct experiments with SpecCas (the [OPT] variant and $\alpha = 0.1$ for better quality).

The results in Table 2 show that SpecCas significantly increases the acceptance rate and the decoding speed. However, its generation quality is not as good as that of other methods, even when we choose hyperparameters to favor higher generation quality. On the other hand, we also ran experiments with $\delta = 10$ for Cactus. With a similar wall-time acceleration on GSM8K and GPQA, Cactus 's generation quality is considerably higher. We hypothesize that this is due to the lack of explicit divergence control in SpecCas, whereas the other methods (especially Cactus) guarantee controlled "distances." Given that the primary focus of this paper is to introduce a new, principled method, we leave a deeper investigation of these methods to future work.

Table 2: The results with Qwen 3 14B as verifier and Qwen 3 0.6B as drafter.

		GSM8K			IFEval			GPQA		
m	Name	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Wall$^{\downarrow}$	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Wall$^{\downarrow}$	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Wall$^{\downarrow}$
10	SpS	91.12	4.27	1.00x	85.03	2.19	1.00x	39.39	3.37	1.00x
	Mentored	91.66	4.51	1.20x	61.37	2.88	0.96x	40.91	4.31	0.93x
	SpecCas	88.40	6.42	0.85x	69.50	5.02	0.54x	32.83	6.27	0.68x
	TAS	92.65	5.24	0.86x	86.14	3.00	0.82x	38.89	4.99	0.72x
	Cactus 1	93.10	5.44	0.87x	85.96	3.03	0.78x	43.43	5.16	0.70x
	Cactus 10	92.72	5.73	0.83x	84.66	3.41	0.74x	39.40	5.71	0.69x

Scaling to larger models. To evaluate the scalability of our method under more memory-intensive conditions, we conduct experiments on a larger model pair: Qwen 3 1.7B (drafter) and 32B (verifier). This setting involves significantly higher parameter counts than the reported 14B maximum in the main table, serving to verify performance where memory bottlenecks are typically more prominent. We maintain the standard speculative decoding setup with a draft length of $m = 10$ and report both accuracy and acceptance length (AL).

Table 3: The results of Qwen 3 32B as verifier and Qwen 3 1.7B as drafter on three benchmarks: GSM8K, IFEval, and GPQA. We report the "strict-match" accuracy and the acceptance length (AL).

		GSM8K		IFEval		GPQA	
m	Name	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Score$^{\uparrow}$	AL$_m^{\uparrow}$	Score$^{\uparrow}$	AL$_m^{\uparrow}$
10	SpS	**95.30**	5.03	83.36	2.61	40.40	3.73
	TAS	94.10	7.02	83.73	4.16	40.40	6.12
	Ours ($\delta = 1$)	94.40	**7.13**	**85.21**	**4.47**	**41.92**	**6.36**

As shown in Table 3, Cactus demonstrates superior efficiency (achieving the longest acceptance lengths) across all three benchmarks. In terms of task performance, it notably surpasses TAS and SpS on IFEval and GPQA, while remaining a comparable result on GSM8K. These findings confirm that the effectiveness of Cactus naturally extends to larger models, delievering consistent improvements in acceptance rates while maintaining the accuracy.

Evaluations on Spec-Bench. To provide a more comprehensive assessment of Cactus across diverse scenarios, we conduct evaluations on Spec-Bench (Xia et al., 2024), a unified benchmark designed to test speculative decoding methods across multiple distinct domains, including multi-turn conversation (MT-Bench), translation (WMT), summarization (CNN/DM), question answering (natural questions), mathematical reasoning (GSM8K), and retrieval-augmented generation (RAG). This broad coverage ensures that the observed speedups are not limited to specific task types but are consistent across varied real-world applications. We use the Qwen 3 14B model as the verifier and the 0.6B model as the drafter, maintaining a temperature of 0.6.

Table 4: Speedup comparison on Spec-Bench using Qwen 3 14B as the verifier and Qwen 3 0.6B as the drafter. We report the speedup ratio relative to standard autoregressive decoding. "Accepted" denotes the mean number of accepted tokens per step.

	MT Bench	Trans.	Summ.	QA	Math	RAG	AL_{10}	Overall
SpS	2.01×	1.40×	1.92×	1.85×	1.83×	1.86×	3.20	1.81×
Cactus ($\delta = 1$)	**2.09**×	**1.40**×	**2.04**×	**1.95**×	**1.86**×	**1.92**×	**3.29**	**1.88**×

The results are summarized in Table 4. Cactus is tested without any hyper-parameter tuning ($\delta = 1$). However, it immediately yields acceleration over the SpS baseline. In addition, Cactus consistently outperforms SpS across different domains, achieving an overall speedup of 1.88× (+88% gain over autoregressive decoding). This significant reduction in compute cost is achieved without additional training. It is worth noting that these speeds are measured using the HuggingFace Transformers framework (Wolf et al., 2019), which is less optimized for speculative sampling. We anticipate that the real-world performance gains would be even larger with a better implementation such as vLLM (Kwon et al., 2023), as indicated by our other experiments.

Impact of draft model size. We employ same-family models to ensure aligned tokenization, consistent with standard practice (Leviathan et al., 2023; Chen et al., 2023). To investigate the impact of drafter capacity, we evaluate Cactus on GSM8K using a Qwen 3 14B verifier with varying drafter sizes (Table 5).

Table 5: Ablation on GSM8K using Qwen 3 14B verifier with different drafter sizes ($\delta = 1$).

Draft Size	Score	AL	Rej
Verifier (Oracle)	91.71	-	-
0.6B	**93.10**	5.44	-32%
1.7B	92.50	6.78	-60%
4B	92.57	**7.76**	**-76%**

Increasing the drafter size to 4B significantly boosts the mean accepted length (AL) to 7.76 and reduces rejection rates by 76%, while maintaining high task accuracy. These results confirm that Cactus effectively scales with stronger drafters, translating improved draft quality into greater decoding efficiency.

C CASE STUDY

In this section, we discuss whether the choice of δ affects qualitative measures such as reasoning ability. We gather the results of the first sample from GSM8k, where δ is set to different values when running Cactus with the Qwen3 0.6B + 14B model pair.

From the case study in Table 6, we can see that the reasoning is poor and lengthy when δ is large (more divergence allowed). Consequently, the result is wrong due to the low-quality chain-of-thought. This confirms that the divergence control in Cactus helps maintain qualitative measures.

D BROADER IMPACT AND FUTURE DIRECTIONS

Broader impact. By improving the inference efficiency of large language models without sacrificing output quality, our method reduces computational costs and energy consumption. This contributes to more sustainable AI deployment, broadens access to high-performance language models, and supports environmentally conscious machine learning practices. Additionally, Cactus can enable faster, lower-cost applications in education, healthcare, and low-resource settings.

Future directions. Our goal in this paper is to introduce and analyze the draft–verify framework, not to exhaustively optimize every dimension of the system. Accordingly, we scope out several extensions and leave them for future exploration by the community: (1) *Model scale.* We capped evaluation at 14B parameters to keep the methodology clear and costs tractable. Pushing to substantially larger backbones could reveal scaling behavior (e.g., effects on acceptance rates, latency, and robustness) and is best investigated in follow-on work, including studies of scaling laws and distributed inference. (2) *Model training.* We emphasize a training-free method to highlight the mechanism itself. While targeted tuning (e.g., LoRA for the draft, verifier calibration, joint distillation) may further improve proposal quality and reduce disagreement error, such engineering is orthogonal to our core contribution and thus deferred. (3) *Memory usage.* Draft-and-verify introduces extra memory for the draft model and caches. Techniques like quantization, weight sharing, cache reuse, selective offloading, and early-exit heuristics could lower this footprint, but a thorough treatment would distract from the main result; we leave these optimizations to future work. (4) *Leveraging ensemble effects.* In our main experiments, we observe that Cactus often performs better than the verifier model. For example, Cactus surpasses the verifier's accuracy by 2 standard deviations on both IFEval and GPQA. We hypothesize that this is because Cactus enables a "healthy" ensemble effect by combining two model distributions. Leveraging ensemble effects in speculative sampling could be explored in future work.

E THE USE OF LARGE LANGUAGE MODELS

Throughout this paper (with this paragraph being an exemption), we use large language models to help identify grammar errors. Specifically, we prompt ChatGPT to "Revise grammar errors with minimal changes of the original text", followed by the latex source code of each paragraph. In addition, we use ChatGPT and DeepSeek R1 to triple-check all technical proofs. The code for plotting all the figures is initially generated by ChatGPT, which is further revised by the authors according to the authors' taste. We certify that the originality and scientific contributions of our method do not come from any large language models.

Table 6: Generations for different values of δ.

Task	Question: Janets ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers’ market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers’ market?
$\delta = 1$	<think>\nOkay , let me try to figure out how much Janet makes each day at the farmers’ market. Hmm, so the problem says she has ducks that lay 16 eggs per day. Then she eats three for breakfast and bakes muffins with four eggs each day. Then she sells the remaining eggs for $2 each. Alright, let’s break this down step by step.\n \nFirst , total eggs per day: 16 eggs. Then she uses some for breakfast and baking. Let’s see, for breakfast, she eats 3 eggs. Then for baking, she uses 4 eggs. So total eggs used each day would be 3 + 4 = 7 eggs. Therefore, the remaining eggs would be 16 - 7 = 9 eggs. Then she sells those 9 eggs at $2 each. So, how much does she make? 9 eggs multiplied by $2, which is 18 dollars.\n \nWait , let me check again. If she uses 3 eggs for breakfast and 4 for baking, that’s 7. 16 minus 7 is indeed 9. Then 9 times 2 is 18. Yeah, that seems right. So her daily earnings would be $18. Hmm, is there any possibility I made a mistake here? Maybe I should re-read the problem to make sure I didn’t miss anything.\n \nThe problem says she lays 16 eggs per day. Then she eats 3 for breakfast and bakes muffins each day with four. So the total used is 3+4=7. Subtract that from 16, which leaves 9. Multiply by $2 for selling, 9*2=18. Yep, that’s correct. I think that’s the answer.\n </think>\n \nJanet s ducks lay 16 eggs per day. She uses 3 eggs for breakfast and 4 eggs for baking muffins, totaling 7 eggs used. The remaining eggs are $16 - 7 = 9\$. She sells these at $2 per egg, earning $9 \\times 2 = 18\$. \n \n ######## 18
$\delta = 10^6$	<think>\nOkay , so let\’s try to figure out how much Janet makes every day at the farmers\’ market. Let me start by breaking down the problem step by step. **[lengthy CoT omitted]** so total eggs used is 4 * 4 = 16. Then, after eating 3 for breakfast, leftover is 16 - 3 = 13, then she sells them at 2 each: 13 * 2 = $26. \n \nYes , that seems to make sense. Therefore, the answer should be 26. But let me check if the question is asking "how much money does she make every day at the farmers\’ market?" So yes, selling the remainder at $2 per egg. Therefore, answer is $26. \n \nTherefore , the answer is 26, which in the previous problems they had 5. This is a different case. So I think that\’s correct.\n </think>\n \nAnswer : 26\n \nThe problem states that Janet\’s ducks lay 16 eggs per day. She eats 3 eggs for breakfast and 4 eggs to make muffins daily. After these expenses, the remaining eggs are $ 16 - 3 - 4 = 9 $, which are sold at $2 per egg. Therefore, the total money she earns at the farmers\’ market is $ 9 \\times 2 = 18 $. \n \nHowever , the phrase "bakes muffins for her friends every day with four" might imply that she uses 4 eggs per muffin, meaning she makes 4 muffins. If she uses 4 eggs each muffin, the total eggs consumed would be $ 4 \\times 4 = 16 $. Since she eats 3, the remaining eggs are $ 16 - 3 = 13 $, which are sold for $ 13 \\times 2 = 26 $. Hence, the correct answer is **26**. \n \n ######## 26