SJD-PAC: Accelerating Speculative Jacobi Decoding via Proactive Drafting and Adaptive Continuation

Supplementary Material

A. Proofs of Correctness

In this section, we provide the theoretical guarantee that SJD-PAC is lossless. That is, the distribution of the generated sequence converges exactly to the target distribution $p(x)$, identical to standard autoregressive sampling. We prove this by demonstrating that neither Adaptive Continuation (AC) nor Proactive Drafting (PD) introduces bias into the verified sequence.

A.1. Preliminaries

Let $p(x_i \mid x_{<i})$ denote the target distribution at position i given prefix $x_{<i}$, and $q(x_i \mid x_{<i})$ denote the draft distribution. Standard speculative decoding relies on the property that for a random variable X sampled via the rejection sampling scheme:

$$\begin{aligned} P(\mathsf{X}=x) &= q(x)\cdot\min\left(1,\frac{p(x)}{q(x)}\right) \\ &\quad + \mathcal{N}\cdot\frac{\max\left(0,p(x)-q(x)\right)}{\mathcal{N}} \\ &= p(x), \end{aligned} \tag{1}$$

where $\mathcal{N}=\sum_z \max\left(0,p(z)-q(z)\right)$ is the normalization factor for the residual distribution. We refer to this standard result as the **Rejection Sampling Lemma**.

A.2. Proof of Correctness for Adaptive Continuation (AC)

Theorem 1. *AC preserves the target distribution $p(x)$ for all tokens accepted in the subsequent iteration.*

Proof. Let X^t denote the draft sequence generated by the AC mechanism during iteration t. At iteration $t+1$, we verify the draft tokens x_j^t against the target distribution $p(x)=p(x\mid X_{<j}^{t+1})$. Let $q(x)$ denote the distribution used to generate the draft token at iteration t. Although q may be derived from stale context, it is fixed and known during the verification at $t+1$.

To prove that the marginal probability of outputting a token matches the target distribution $p(x)$, we analyze the rejection sampling mechanism step-by-step. The probability of outputting a specific token x is the sum of the probability of it being accepted from the draft and the probability of it being resampled after a rejection.

Let $P(\mathsf{X}_j^{t+1}=x)$ be the total probability that token x is generated at position j. We express this as:

$$P(\mathsf{X}_j^{t+1}=x)=P(\text{Acc. }x)+P(\text{Rej.})\cdot P(\text{Resamp. }x) \tag{2}$$

Probability of Acceptance: The draft proposes x with probability $q(x)$. The acceptance probability is defined as $\alpha(x)=\min\left(1,\frac{p(x)}{q(x)}\right)$. Thus, the joint probability of proposing and accepting x is:

$$\begin{aligned} P(\text{Acc. }x) &= q(x)\cdot\min\left(1,\frac{p(x)}{q(x)}\right) \\ &= \min\Big(q(x),p(x)\Big). \end{aligned} \tag{3}$$

Probability of Rejection. The probability of rejecting the draft (summing over all possible vocabulary tokens v) corresponds to the probability mass where the draft exceeds the target (or conversely, the missing mass):

$$\begin{aligned} P(\text{Rej.}) &= 1-\sum_{v\in V}P(\text{Acc. }v) \\ &= \sum_{v\in V}p(v)-\sum_{v\in V}\min\Big(q(v),p(v)\Big) \\ &= \sum_{v\in V}\Big(p(v)-\min\Big(q(v),p(v)\Big)\Big) \\ &= \sum_{v\in V}\max\Big(0,p(v)-q(v)\Big). \end{aligned} \tag{4}$$

Probability of Resampling. Upon rejection, we sample from the residual distribution $p_{\text{res}}(x)$. This distribution is defined as the normalized difference between the target and the draft:

$$\begin{aligned} P(\text{Resamp. }x) &= p_{\text{res}}(x) \\ &= \frac{\max\Big(0,p(x)-q(x)\Big)}{\sum_{v\in V}\max\Big(0,p(v)-q(v)\Big)}. \end{aligned} \tag{5}$$

Observe that the denominator is the normalization constant, which equals $P(\text{Rej.})$ derived in Eq. (4).

Combining terms. Substituting these back into Eq. (2) for total probability:

$$\begin{aligned} P(\mathsf{X}_j^{t+1}=x) &= \min\Big(q(x),p(x)\Big) \\ &\quad + P(\text{Rej.})\frac{\max\Big(0,p(x)-q(x)\Big)}{P(\text{Rej.})} \\ &= \min\Big(q(x),p(x)\Big) \\ &\quad + \max\Big(0,p(x)-q(x)\Big) \end{aligned} \tag{6}$$

We consider two cases to solve this summation for any token x:

- **Case 1**, $p(x) \geq q(x)$: $\min(q(x), p(x)) + \max(0, p(x) - q(x)) = q(x) + (p(x) - q(x)) = p(x)$.
- **Case 2**, $p(x) < q(x)$: $\min(q(x), p(x)) + \max(0, p(x) - q(x)) = p(x) + 0 = p(x)$.

In both cases, the marginal probability $P(\mathsf{X}_j^{t+1} = x)$ is exactly $p(x)$. Crucially, this equality holds independently of the quality or parameters of the draft distribution $q(x)$, provided that $q(x)$ is a valid probability distribution over the vocabulary. □

A.3. Proof of Correctness for Proactive Drafting (PD)

Theorem 2. *PD, which involves the sequential verification of K candidates sampled without replacement and is supplemented by sampling from the residual distribution upon total rejection, guarantees exact recovery of the target distribution $p(x)$.*

Proof. We prove this theorem by induction on the number of candidates K.

Definitions. Let $p(x)$ denote the target probability distribution and $q(x)$ denote the draft distribution. PD generates a set of candidates by sampling iteratively from q without replacement. For the k-th candidate c_k, let $q^{(k)}(x)$ be the draft distribution adjusted for previous samples $\mathcal{C}_{<k} = \{c_1, \ldots, c_{k-1}\}$, and $p^{(k)}(x)$ be the residual target distribution after previous rejections (with $p^{(1)} = p$). The adjusted draft distribution is:

$$q^{(k)}(x) = \begin{cases} \frac{q(x)}{1 - \sum_{y \in \mathcal{C}_{<k}} q(y)} & \text{if } x \notin \mathcal{C}_{<k} \\ 0 & \text{if } x \in \mathcal{C}_{<k} \end{cases} \tag{7}$$

The residual transition follows the Rejection Sampling Lemma established in Eq. (1):

$$p^{(k+1)}(x) = \frac{\max\left(0,\ p^{(k)}(x) - q^{(k)}(x)\right)}{\sum_v \max\left(0,\ p^{(k)}(v) - q^{(k)}(v)\right)}. \tag{8}$$

Base Case ($K = 0$). No candidates are proposed. The algorithm samples directly from the residual $p^{(1)}$, which is initialized as $p^{(1)} = p$. The output distribution is trivially correct.

Inductive Step. Assume that at step k, the target distribution is correctly captured by the residual $p^{(k)}$. We verify that the procedure at step k preserves this distribution. Specifically, the probability of rejecting candidate c_k is the complement of the acceptance mass:

$$\begin{aligned} P(\text{Rej. } c_k) &= \sum_x q^{(k)}(x) \left(1 - \min\left(1, \frac{p^{(k)}(x)}{q^{(k)}(x)}\right)\right) \\ &= \sum_x \max\left(0,\ p^{(k)}(x) - q^{(k)}(x)\right). \end{aligned} \tag{9}$$

We now show that the marginal probability of accepting a token u at step k, or rejecting it and subsequently sampling it from the next residual $p^{(k+1)}$, equals the probability of sampling u from the current target $p^{(k)}$.

$$\begin{aligned} p^{(k)}(u) \stackrel{?}{=} & \underbrace{q^{(k)}(u) \cdot \min\left(1, \frac{p^{(k)}(u)}{q^{(k)}(u)}\right)}_{\text{Accept } u \text{ at step } k} \\ & + \underbrace{P(\text{Rej. } c_k) \cdot p^{(k+1)}(u)}_{\text{Sample } u \text{ from residual}} \end{aligned} \tag{10}$$

Substituting the definition of $p^{(k+1)}(u)$ and the rejection probability from Eq. (9):

$$\begin{aligned} \text{RHS} &= \min\left(q^{(k)}(u),\ p^{(k)}(u)\right) \\ &\quad + p^{(k+1)}(u) \sum_x \max\left(0,\ p^{(k)}(x) - q^{(k)}(x)\right) \\ &= \min\left(q^{(k)}(u),\ p^{(k)}(u)\right) \\ &\quad + \frac{\max\left(0,\ p^{(k)}(u) - q^{(k)}(u)\right)}{\sum_x \cdots} \sum_x \cdots \\ &= \min\left(q^{(k)}(u),\ p^{(k)}(u)\right) \\ &\quad + \max\left(0,\ p^{(k)}(u) - q^{(k)}(u)\right) \\ &= p^{(k)}(u). \end{aligned} \tag{11}$$

By induction, since the procedure preserves the target distribution at each step k, and the final sampling upon total rejection is drawn from $p^{(K+1)}$, the overall Proactive Drafting process exactly recovers $p(x)$. □

B. Overhead Analysis

In this section, we analyze the memory consumption and computational latency introduced by SJD-PAC compared to the vanilla AR baseline.

GPU Memory Consumption. We evaluate the memory footprint on Lumina-mGPT [14]. The baseline implementation requires approximately 18.7 GB of GPU memory. Implementing SJD-PAC with a decoding window length of $L = 64$ results in a memory usage of 18.7 GB, demonstrating negligible memory overhead.

Although SJD-PAC employs a significantly larger decoding window ($L = 64$) compared to vanilla AR ($L = 1$), the memory cost remains stable. This is because the KV cache is inherently required to store the history of the generated sequence. In our approach, the accepted tokens within the window are immediately converted into persistent KV cache entries. Consequently, the memory used by the window effectively acts as pre-allocated space for the upcoming KV cache, rather than an add-on burden.

Computational Latency. We break down the runtime costs per step to analyze the temporal overhead. The breakdown is summarized in Tab. 4. For the vanilla AR model, a single forward pass takes approximately 40.8 ms, with miscellaneous overheads totaling roughly 1.0 ms. With SJD-PAC ($L = 64$), the model forward pass increases slightly to 44.3 ms. The additional components specific to our algorithm include tree verification and generation (3.2 ms) and KV cache maintenance (2.2 ms).

Table 4. Runtime breakdown of a single decoding step in milliseconds. Comparison between Vanilla AR and SJD-PAC with window length $L = 64$.

Operation	**Vanilla AR**	**SJD-PAC**
Model Forward	40.8 ms	44.3 ms
Tree Verification & Generation	-	3.2 ms
KV Cache Update & Backtracking	-	2.2 ms
Other Overheads	1.0 ms	1.0 ms

Despite the overhead of SJD-PAC, verification is efficient due to the high parallelizability enabled by the static tree structure and independent positional operations. Moreover, as current KV cache overheads arise mainly from non-contiguous memory access, future optimizations for contiguous layouts are expected to further mitigate latency.