SJD-VP: SPECULATIVE JACOBI DECODING WITH VERIFICATION PREDICTION FOR AUTOREGRESSIVE IMAGE GENERATION

Anonymous authors
Paper under double-blind review

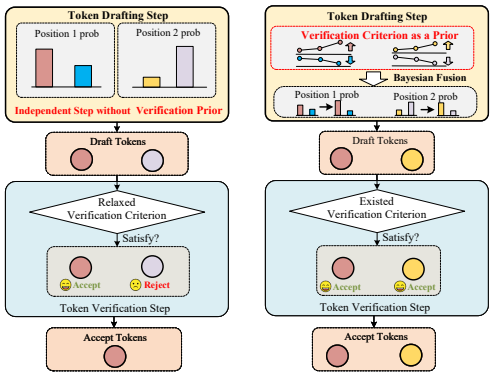
ABSTRACT

Speculative Jacobi Decoding (SJD) has emerged as a promising method for accelerating autoregressive image generation. Despite its potential, existing SJD often suffers from the low acceptance rate issue of speculative tokens due to token selection ambiguity. Recent works attempt to mitigate this issue primarily from the perspective of revising token verification, but fail to fully exploit the dynamics of Jacobi decoding. In this paper, we conduct an in-depth analysis and find that tokens whose probabilities exhibit an increasing trend in the Jacobi dynamic are more likely to match the verification-accepted and correct token. Based on this, we propose a novel Speculative Jacobi Decoding with Verification Prediction (SJD-VP). The key idea is to leverage the change in token probabilities across Jacobi dynamics to guide the token drafting process, favoring tokens whose probabilities increase. This effectively predicts which tokens are likely to pass subsequent verification, boosting the acceptance rate. In particular, our SJD-VP is plug-and-play and can be seamlessly integrated into existing SJD variants. Extensive experiments on standard benchmarks demonstrate that our SJD-VP method consistently accelerates autoregressive decoding while maintaining image generation quality.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Autoregressive image generation models have achieved remarkable success in producing high-quality images (Brown et al., 2020; Achiam et al., 2023; Touvron et al., 2023), enabling applications such as high-fidelity image synthesis (Van Den Oord et al., 2017; Razavi et al., 2019; Chen et al., 2020), controllable image editing (Esser et al., 2021; Chang et al., 2022), text-to-image generation (Ramesh et al., 2021; Yu et al., 2022), image inpainting and outpainting (Yu et al., 2018; Ramesh et al., 2022), and complex multi-modal reasoning tasks (Alayrac et al., 2022; Liu et al., 2023; Team et al., 2023). However, sequential sampling leads to slow inference (Van Den Oord et al., 2017; Leviathan et al., 2023). To address this issue, Speculative Jacobi Decoding (SJD) (Teng et al., 2024) has emerged as a promising technique that generates and validates multiple tokens in parallel through iterative refinement. Despite its potential, existing SJD approaches (Jang et al., 2025) often provide limited speedup due to low speculative acceptance rates, constraining their efficiency.



(a) Existing method **(b)** Our SJD-VP

Figure 1: Illustration of the structural decoupling in existing SJD and our SJD-VP. In (a), existing SJD methods draft tokens from current drafting distribution without considering verification preference, leading to frequent token rejection during decoding. In contrast, (b) SJD-VP uses the verification criterion as a prior and models probability trajectories to favor consecutive probability growth, thereby better aligning token drafting with verification.

Recently, existing studies point out that the main reason causing such low acceptance rates is token semantic ambiguity issue, i.e., image tokens often exhibit inherent redundancy and diversity, meaning multiple tokens can convey similar semantics. Motivated by this observation, existing SJD variants attempt to improve the acceptance rate during speculative Jacobi decoding by revising the verification criterion, i.e., relaxing the verification criterion to tolerate semantically similar tokens (Jang et al., 2025; So et al., 2025b), extending verification granularity from individual tokens to subsequence to alleviate semantic ambiguity (Zhang et al., 2026; Yu et al., 2026), or broadening the verification scope to entire windows (Teng et al., 2025; So et al., 2025a). Although these methods have shown superior performance, as shown in Figure 1(a), they still suffer from another key limitation: the drafting step does not explicitly predict whether the generated tokens are likely to satisfy the subsequent verification criterion. This lack of explicit verification prediction leads to an inherent misalignment between the drafted tokens and the verification criterion, making it difficult to generate the draft tokens satisfying the deterministic verification criterion. In other words, the drafting step does not take into account whether the generated draft tokens are likely to be accepted by the verification step, resulting in a low hit rate of verification-accepted tokens and thus limiting the overall acceleration potential. This naturally raises a key question: can the deterministic verification criterion be leveraged to enable Verification Prediction, i.e., guiding the draft process to generate draft tokens that are more likely to pass verification and thereby further improving efficiency?

To answer this question, in this paper, we conduct an in-depth analysis (see Section 3.1 for more details) on these verification-accepted tokens, i.e., those tokens satisfying the deterministic verification criteria. Then, we reveal a key phenomenon, i.e., **most verification-accepted tokens show a consecutive probability growth along the Jacobi dynamics before performing the verification step and such phenomenon can be consistently observed in nearly 90% of all verification-accepted tokens.** This phenomenon means that such consecutive probability growth pattern can be effectively leveraged to enable the verification prediction, i.e., guiding the generation of draft tokens which are more likely to pass verification and thus improve the overall acceptance rate. However, a natural follow-up question arises: Are the tokens predicted based on this growth pattern actually the correct tokens? To figure out this point, we conducted further experiments on these correct tokens and observe that **this consecutive probability growth is highly specific to correct tokens, whereas incorrect tokens typically exhibit fluctuating or decreasing probabilities.** This indicates that tokens following such a consecutive probability growth are also more likely to be the correct tokens.

The above two-point analysis provides a key insight: *by drafting tokens along the probability increasing trend observed from Jacobi dynamics, we can not only increase their likelihood of passing subsequent verification but also guide them toward the correct tokens, thereby simultaneously accelerating decoding and improving the quality of the generated images.* Based on this insight, we propose a novel speculative Jacobi decoding with verification prediction (called SJD-VP) for accelerating autoregressive image generation. As illustrated in Figure 1(b), our core idea is to regarding the verification prior estimated from token probability growth patterns observed from Jacobi dynamics as a Bayesian prior and viewing the current drafting probability as the conditional likelihood, and then leverage the Bayesian manner to derive a posterior as the final drafting probability, thereby achieving faster decoding while simultaneously improving the generation quality. In particular, our SJD-VP is very simple and plug-and-play, which can be seamlessly integrated into existing SJD variants by few modifications in token drafting step. Specifically, we first execute *Verification Prior Estimation* to estimate a verification prior from historical probability dynamic trajectories, identifying tokens that show consecutive probability growth. Then, we perform *Bayesian Fusion* to derive a posterior drafting distribution by fusing this verification prior with the current drafting distribution. Finally, we draft tokens from this posterior distribution, thereby effectively ensuring that the drafted step not only align with the verification step but are also significantly higher quality.

Our main contributions can be summarized as follows:

- We identify that overlooking the prior of the verification criterion is a key factor constraining the efficiency of SJD and its variants. We reveal that this prior can be leveraged to predict which tokens are more likely to be accepted during the drafting step, offering a new perspective to further enhance decoding efficiency for autoregressive image generation.
- We propose **SJD-VP**, a plug-and-play token drafting strategy that integrates *verification prediction* into the decoding step via a Bayesian framework. By fusing the verification prior with the current drafting probability to predict verification results, it improves the

acceptance rate of draft tokens. Importantly, our method is complementary to most existing SJD variants, and can be integrated with them to achieve further improvements.

- Extensive experiments demonstrate that SJD-VP consistently accelerates decoding while maintaining image quality. Furthermore, it significantly boosts the performance of SJD variants, thereby robustly validating effectiveness and general flexibility of SJD-VP.

2 Related Work

2.1 Autoregressive Image Generation

Autoregressive image generation models an image as a sequence of visual tokens and predicts them sequentially, capturing fine-grained dependencies for high-fidelity synthesis. Existing methods can be roughly divided into two groups: **1) Continuous-token autoregressive models.** These methods autoregress over pixel values or continuous latent features without discrete codebooks, including PixelRNN (Van Den Oord et al., 2016), PixelCNN (Van den Oord et al., 2016), iGPT (Chen et al., 2020), and the DALL·E 2 prior (Ramesh et al., 2022). **2) Discrete-token autoregressive models.** These methods quantize images into discrete token sequences and perform generation in the resulting token space. Representative approaches include VQ-VAE and VQ-GAN (Van Den Oord et al., 2017; Esser et al., 2021), DALL·E (Ramesh et al., 2021), and Parti (Yu et al., 2022). Recent multimodal frameworks further unify text and image tokens within a shared vocabulary for cross-modal generation, such as Janus and Lumina-mGPT (Liu et al., 2024; Wu et al., 2025). Despite strong generation quality and controllability, these methods remain constrained by slow sequential decoding. Therefore, this paper focuses on accelerating the autoregressive decoding process.

2.2 Autoregressive Generation Acceleration

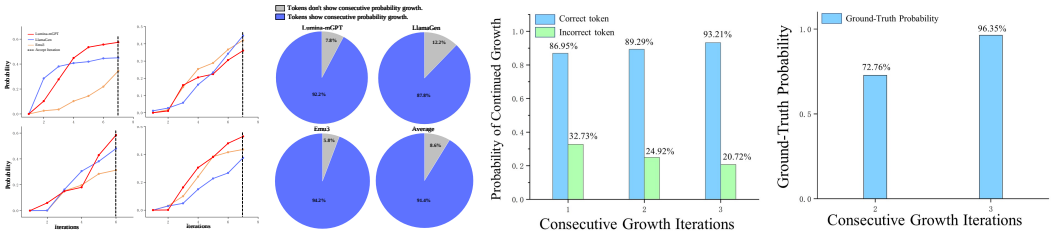
Autoregressive generation acceleration aims to reduce the latency of sequential token decoding, which is particularly costly for high-resolution image synthesis. Existing approaches can be roughly divided into two groups: **1) Model-based acceleration methods.** These methods improve generation efficiency by redesigning model architectures or introducing auxiliary predictors during training. Representative works explore hierarchical token prediction, distillation from large AR transformers, and hybrid AR–parallel generation modules (Wang et al., 2025; Ren et al., 2025; Yu et al., 2024), respectively, enabling faster inference while preserving generation quality. **2) Inference-based acceleration methods.** These methods accelerate generation at inference time without modifying or retraining the model. Speculative decoding (Leviathan et al., 2023) proposes draft tokens in parallel and verifies them with the base model, while Speculative Jacobi Decoding (SJD) (Teng et al., 2024) extends this paradigm to discrete autoregressive image generation through parallel speculative updates. However, its acceleration is limited by low acceptance rates. Existing studies address this limitation by revising the verification criterion. For example, LANTERN (Jang et al., 2025), GSD (So et al., 2025b) and CSD (Wang et al., 2026) relax the verification criterion to improve token acceptance; SJD-PV (Yu et al., 2026) and SJD-SV (Zhang et al., 2026) extend verification from individual tokens to subsequences, jointly evaluating multiple tokens rather than verifying them independently; SJD++ (Teng et al., 2025) and SCD (So et al., 2025a) extend the verification rule to the whole decoding window, allowing more valid draft tokens to be retained.

Nevertheless, as discussed in Section 1, these existing methods still suffer from an inherent misalignment issue between drafting and the verification, i.e., drafting tokens without considering verification criterion prior. In this paper, we propose SJD-VP to address this issue by introducing verification prediction prior into the drafting step for improving token acceptance. Since our method does not modify the verification criterion, it complements existing methods for further acceleration.

3 Methodology

3.1 Motivation Analysis

Analysis on Verification-Accepted Tokens. To answer the above question, in this section, we conduct an in-depth analysis on these verification-accepted tokens (i.e., tokens satisfying the deterministic verification criteria). Specifically, we first take the latest AR image models (i.e., Lumina-



(a) Probability trajectories. (b) Token statistics. (c) Growth-token proportion. (d) Prediction accuracy.

Figure 2: Motivation analysis of verification-accepted tokens. (a) Probability trajectories of representative tokens. (b) Statistical analysis of their probability evolution. (c) Probability of continued growth for correct and incorrect tokens. (d) Prediction accuracy of correct tokens under consecutive n-step probability growth.

mGPT (Liu et al., 2024), LlamaGen (Sun et al., 2024), and Emu3 (Wang et al., 2024)) with Speculative Jacobi Decoding as the base model. Then, we randomly select four text prompts from the PartiPrompts dataset and visualize the probability trajectories of verification-accepted tokens across multiple Jacobi dynamics. The results are shown in Figure 2(a). From the results, we can see that verification-accepted tokens usually exhibit consecutive probability growth before being accepted. This means that "consecutive probability growth" in the Jacobi dynamics of decoding can be regarded as a prediction prior for verification-accepted tokens.

To further assess whether "consecutive probability growth" can serve as a reliable prediction prior for the verification step, Figure 2(b) presents a statistical analysis of the proportion of verification-accepted tokens that demonstrate consecutive probability growth before being accepted. From the results, we observe that in Lumina-mGPT, LlamaGen, and Emu3, approximately 92.2%, 87.8%, and 94.2% (with an average of 91.4%) of verification-accepted tokens demonstrate this consecutive probability growth pattern. Such consistently high proportions across different models indicate that consecutive probability growth is a strong and model-agnostic characteristic associated with successful verification. This suggests that *"consecutive probability growth" can serve as an effective prediction prior for the verification step, providing a basis for guiding the drafting process toward high-acceptance tokens and improving speculative decoding efficiency.*

Analysis on Correct Tokens. However, a natural follow-up question is: *Are the tokens predicted based on the above characteristic of "consecutive probability growth" actually correct tokens?* (i.e., the correct token refers to the reference token generated by the original AR decoding without speculative decoding.) To investigate this point, we conduct experiments on correct and incorrect tokens, as shown in Figure 2(c). Specifically, given that a token has exhibited probability increases for n consecutive steps, we compute the probability that its likelihood will continue to increase at the next step. From the results, we can see that 1) correct tokens (see blue bars) consistently demonstrate a much higher continuation probability exceeding 85% for all values of n; whereas 2) the continuation probability of incorrect tokens (see green bars) rapidly decreases as n grows. These results confirm that consecutive probability growth is a distinctive and reliable characteristic of correct tokens.

Finally, to further evaluate whether the prediction prior of "consecutive probability growth" can directly guide correct token drafting, Figure 2(d) measures the probability that a token selected solely based on consecutive probability growth for n iterations corresponds to the correct token. This probability is already relatively high (72.76%) when consecutive growth first emerges ($n = 2$), and increases substantially to 96.35% as the growth persists ($n = 3$). This means that *token drafting based on the prior of consecutive probability growth not only enables verification prediction to improve efficiency but also guides drafting toward correct tokens to preserve generation quality.*

3.2 The Proposed SJD-VP Method

As discussed in Section 1, existing methods usually regard drafting and verification as two independent processes, which leads to misalignment between drafted tokens and the verification criterion. To this end, we propose a novel speculative Jacobi decoding with verification prediction (SJD-VP), which regards the verification prior (i.e., the character "consecutive probability growth") estimated

from the verification criterion as a Bayesian prior and the current drafting probability as the conditional likelihood, and then leverages Bayesian fusion to derive a posterior as the final drafting probability, thereby accelerating autoregressive generation. In particular, our method only revises the token drafting step, which is plug-and-play and can be seamlessly integrated into existing SJD methods. Next, we take the existing SJD as an example to introduce our SJD-VP, while adaptation to other SJD variants will be explained at the end of the section. Specifically, for each token drafting step, we first perform the **Verification Prior Estimation** phase to construct a historical reference $\bar{p}_t(x)$ and leverage it to estimate the verification prior $S_t(x)$ with the character "consecutive probability growth". Then, we perform the **Bayesian Verification Fusion** phase to fuse the prior with the current drafting probability to derive a posterior probability $p'_t(x)$. Finally, we sample the token from $p'_t(x)$ to draft tokens. We have provided detailed pseudocode of SJD-VP in Algorithm 1 of Appendix A.3. Next, we detail these two phases.

Verification Prior Estimation As discussed in Section 3.1, verification-accepted tokens exhibit a pattern of consecutive probability growth. Therefore, we can estimate verification prior by capturing such growth trend between the current probability and its historical probability. Specifically, let $p_t(x)$ denote the probability of a candidate token x at the t-th Jacobi iteration. We maintain a position-aware history buffer of length L to record the recent probability trajectories $\{p_{t-L}(x), \ldots, p_{t-1}(x)\}$. Then, we employ an Exponentially Weighted Aggregation (EWA) method to synthesize a robust historical reference probability $\bar{p}_t(x)$ that aggregates historical Jacobi dynamics probability to mitigate volatile fluctuations between Jacobi iterations while remaining sensitive to recent drafting probability. That is,

$$\bar{p}_t(x) = \frac{\sum_{k=1}^{L} \gamma^{k-1} p_{t-k}(x)}{\sum_{k=1}^{L} \gamma^{k-1}}, \tag{1}$$

where $\gamma \in (0, 1)$ is the decay factor of EWA.

With the above historical probability $\bar{p}_t(x)$, we first calculate a prediction score $S_t(x)$ by computing the gain between current probability $p_t(x)$ and historical probability $\bar{p}_t(x)$. Considering the mathematical stability, we perform this gain computation in the log-space. That is,

$$S_t(x) = \log p_t(x) - \log \bar{p}_t(x). \tag{2}$$

However, the verification prior estimated by this comparison does not explicitly validate whether a token shows consecutive probability growth. To address this issue, we further introduce a **Consecutive Growth Mask** $M_t(x)$ to mark whether a token x shows consecutive probability growth:

$$M_t(x) = \begin{cases} 1, & \text{if } p_{t-k}(x) > p_{t-k-1}(x), \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where the condition must hold for all $k \in 0, \ldots, N-1$. By applying this mask, we obtain the **Verification Prior** $M_t(x) \cdot S_t(x)$, which filters out tokens that fail to exhibit the required probability growth. The resulting Verification Prior is then used for Bayesian Verification Fusion.

Bayesian Verification Fusion Until now, we have obtained two sources of information, i.e., the verification prior $M_t(x)S_t(x)$ and the current observed probability $p_t(x)$. To achieve more accurate probability estimation for drafting tokens, we regard the above verification prior $M_t(x)S_t(x)$ as a Bayesian prior in the log space and the current probability $p_t(x)$ as a conditional likelihood, and then estimate the Bayesian posterior probability $p'_t(x)$ as our final sampling probability. In particular, we perform such Bayesian fusion only on the Top-k candidates (denoted as $\mathcal{C}_t$) for robustness, thereby avoiding potential interference from low-probability tail tokens. Similarly, we perform such operation in the log space for mathematical stability, and obtain the posterior distribution by normalizing the following weights:

$$p'_t(x) \propto \begin{cases} \exp(\log p_t(x) + M_t(x) \cdot S_t(x)), & \text{if } x \in \mathcal{C}_t, \\ p_t(x), & \text{otherwise.} \end{cases} \tag{4}$$

Finally, we draft token x_t from the posterior probability $p'_t(x)$. That is, $x_t \sim p'_t$.

3.3 Adaptation to Other SJD Variants

Our method is plug-and-play, which can be directly integrated into other SJD variants(e.g., LANTERN (Jang et al., 2025), GSD (So et al., 2025b) and CSD (Wang et al., 2026)), by directly replacing their original token draft step with our token draft step with verification prediction.

3.4 Theoretical Analysis Of SJD-VP

The goal of our SJD-VP is to improve the acceptance rate of draft tokens during verification. Next, we analyze SJD-VP from the perspective of proposal–verification distribution alignment. Formally, following (So et al., 2025b), we use the Total Variation Distance (D_{TV}) between the draft distribution p_t and verification distribution q to characterize their distributional mismatch, i.e.,

$$D_{\text{TV}}(p_t, q) = \frac{1}{2} \sum_x |p_t(x) - q(x)|. \tag{5}$$

Therefore, we analyze whether SJD-VP can reduce the Total Variation Distance, i.e., whether $D_{TV}(p'_t, q) < D_{TV}(p_t, q)$, where p'_t is the posterior probability estimation by our SJD-VP.

The Derivation of $D_{TV}(p'_t, q) = \frac{1}{2} \sum_x |p'_t(x) - q(x)|$. In particular, the posterior estimation p'_t can be understood as a correction of the original draft distribution $p_t(x)$, i.e.,

$$p'_t(x) = p_t(x) + \Delta p(x), \tag{6}$$

where $\Delta p(x)$ denotes the probability shift applied to each token based on the verification prior of SJD-VP, with the goal of guiding the draft distribution p_t toward the verification distribution q. Formally, the probability shift $\Delta p(x)$ introduced by SJD-VP can be decomposed into three factors: the predicted probability shift direction $\hat{y}(x) \in \{-1, +1\}$, the token-specific weight $\omega(x) \geq 0$, and the overall shift magnitude $m > 0$. Specifically, $\hat{y}(x)$ determines whether the probability of token x should be increased $(+1)$ or decreased (-1), $\omega(x)$ specifies the probability shift weight for each token, while m denotes its overall magnitude of probability shift. Therefore, the small probability shift $\Delta p(x)$ can be expressed as: $\Delta p(x) = m \cdot \omega(x)\hat{y}(x)$ and then Eq. 5 becomes:

$$\begin{aligned} D_{\text{TV}}(p'_t, q) &= \frac{1}{2} \sum_x |p'_t(x) - q(x)| \\ &= \frac{1}{2} \sum_x |p_t(x) + m \cdot \omega(x)\hat{y}(x) - q(x)|. \end{aligned} \tag{7}$$

Then, we apply a first-order Taylor expansion at $m = 0$ to simplify $D_{\text{TV}}(p'_t, q)$:

$$\begin{aligned} D_{\text{TV}}(p'_t, q) &\approx D_{\text{TV}}(p_t, q) + m \frac{d}{dm} D_{\text{TV}}(p'_t, q)\Big|_{m=0} \\ &\approx D_{\text{TV}}(p_t, q) + \frac{m}{2} \sum_x \text{sign}(p_t(x) - q(x)) \cdot \omega(x)\hat{y}(x). \end{aligned} \tag{8}$$

Analysis of $D_{TV}(p'_t, q)$. According to Eq. 8, proving $D_{TV}(p'_t, q) < D_{TV}(p_t, q)$ under the first-order approximation is equivalent to showing that $\frac{m}{2} \sum_x \text{sign}(p_t(x) - q(x))\omega(x)\hat{y}(x)$ is negative. For each token, $\text{sign}(p_t(x) - q(x)) = +1$ indicates that its draft probability is higher than the verification probability and should be decreased, whereas $\text{sign}(p_t(x) - q(x)) = -1$ indicates that it should be increased. Thus, the correct shift direction $y(x)$ should be opposite to $\text{sign}(p_t(x) - q(x))$, which we define as

$$y(x) = \text{sign}(q(x) - p_t(x)), \tag{9}$$

Then, substituting $-y(x) = \text{sign}(p_t(x) - q(x))$ into the Taylor expansion in Eq. 8, we establish the relationship of $D_{\text{TV}}(p'_t, q)$ with the correct shift direction $y(x)$:

$$D_{\text{TV}}(p'_t, q) \approx D_{\text{TV}}(p_t, q) - \frac{m}{2} \sum_x \omega(x) y(x) \hat{y}(x). \tag{10}$$

Table 1: Weighted directional accuracy of SJD-VP across different models on PartiPrompts.

Model	Lumina-mGPT	LlamaGen	Emu3	Janus-Pro-7B	Average
Q_w (↑)	72.24%	70.23%	75.93%	79.55%	**74.49%**

When SJD-VP predicts the shift direction correctly, $y(x)\hat{y}(x) = +1$; otherwise, it equals -1. Therefore, the sign of $\sum_x y(x)\omega(x)\hat{y}(x)$ in Eq. 10 is determined by the total shift weight of correct predictions, $\sum_{\hat{y}(x)=y(x)} \omega(x)$, and that of incorrect predictions, $\sum_{\hat{y}(x)\neq y(x)} \omega(x)$. To compare the magnitudes of the shift weights associated with correct and incorrect predictions, we define the weighted directional accuracy Q_w as

$$Q_w = \frac{\sum_{\hat{y}(x)=y(x)} \omega(x)}{\sum_{\hat{y}(x)=y(x)} \omega(x) + \sum_{\hat{y}(x)\neq y(x)} \omega(x)} = \frac{\sum_{\hat{y}(x)=y(x)} \omega(x)}{\sum_x \omega(x)}. \tag{11}$$

According to the definition of Q_w, the total shift weights of correct and incorrect predictions are $\sum_{\hat{y}(x)=y(x)} \omega(x) = Q_w \sum_x \omega(x)$ and $\sum_{\hat{y}(x)\neq y(x)} \omega(x) = (1 - Q_w) \sum_x \omega(x)$, respectively. Substituting these two parts into $\sum_x y(x)\omega(x)\hat{y}(x)$ in Eq. 10, we obtain

$$\sum_x \omega(x)y(x)\hat{y}(x) = \sum_{\hat{y}(x)=y(x)} \omega(x) - \sum_{\hat{y}(x)\neq y(x)} \omega(x) = Q_w \sum_x \omega(x) - (1 - Q_w) \sum_x \omega(x) = (2Q_w - 1) \sum_x \omega(x) \tag{12}$$

Substituting Eq. 12 into Eq. 10, we obtain

$$D_{\mathrm{TV}}(p'_t, q) \approx D_{\mathrm{TV}}(p_t, q) - \frac{m}{2}(2Q_w - 1) \sum_x \omega(x). \tag{13}$$

Eq. 13 indicates that $Q_w > 0.5$ guarantees a negative first-order change in D_{TV}. To verify this, we evaluate the weighted directional accuracy Q_w of SJD-VP on PartiPrompts across four model architectures, i.e., Lumina-mGPT, LlamaGen, Emu3, and Janus-Pro-7B. As reported in Table 1, SJD-VP achieves an average Q_w of 74.49%, with four models consistently exceeding 0.5. These results support our theoretical analysis that SJD-VP tends to reduce D_{TV} under the first-order approximation.

4 Experiments

4.1 Experimental Settings

Dataset We evaluate SJD-VP on two standard benchmarks: (1) **Parti-prompts** (Yu et al., 2022), containing 1,600 diverse open-domain prompts; and (2) **MS-COCO 2017** (Lin et al., 2014), comprising 118K image-caption pairs. For both, we strictly follow the preprocessing and evaluation protocols established in prior works (Teng et al., 2024; So et al., 2025b). All experiments use identical sampling conditions to ensure fair comparison.

Baselines and Metrics. We compare SJD-VP against the base autoregressive model **Lumina-mGPT** (Liu et al., 2024) and representative recent decoding methods for autoregressive image generation. All methods use the same backbone and hyperparameters for fair comparison. We use five standard evaluation metrics. **Latency** (in seconds per image) and **NFE** (number of function evaluations) quantify decoding efficiency, while **Acceleration** measures the relative speed-up over the base Lumina-mGPT model. For generation quality, we adopt **FID** to assess perceptual fidelity and diversity, and **CLIP-Score** to evaluate text–image semantic alignment.

Implementation Details All experiments use a single NVIDIA A100 GPU with 80 GB memory. Our implementation is based on PyTorch and HuggingFace Transformers, with mixed-precision inference (FP16) enabled for all methods to ensure efficiency and fairness. Unless otherwise specified, we adopt the same hyperparameter configuration as GSD. Specifically, the temperature is set to 1.0, and top-k sampling is applied with 2000. A classifier-free guidance scale of 3.0 is used to balance fidelity and diversity, and the maximum generation length is fixed at 8192. For SJD-VP, the historical memory length is $L = 3$, with decay factor $\gamma = 0.8$.

Table 2: Results of quantitative evaluation on the Parti-prompt and MS-COCO 2017 benchmarks.

Configuration	Latency (↓)	NFE (↓)	Acceleration (↑)		FID (↓)	CLIP-Score (↑)
			Latency	NFE		
Parti-prompt						
Lumina-mGPT (Liu et al., 2024)	79.37s	2392	1.00×	1.00×	–	32.09
Jacobi Decoding (Song et al., 2021)	82.21s	2300.0	0.97x	1.04×	–	32.09
SJD-PV (Yu et al., 2026)	34.72s	941.41	2.28x	2.54x	–	32.09
SJD (Teng et al., 2024)	36.07s	1035.3	2.20x	2.31×	–	32.09
SJD-VP	**27.00s**	**720.69**	**2.94x**	**3.32x**	–	**32.15**
LANTERN(Jang et al., 2025)	31.40s	**6**36.66	2.53x	3.76x	–	32.10
LANTERN-VP	**26.97s**	**580.47**	**2.94x**	**4.12x**	–	**32.11**
GSD (So et al., 2025b)	33.36s	**8**98.97	2.38x	2.66x	–	32.11
GSD-VP	**31.10s**	**671.60**	**2.55x**	**3.56x**	–	**32.13**
SJD++ (W=96) (Teng et al., 2025)	26.41s	373.75	3.01x	6.40×	–	32.07
SJD-VP++ (W=96)	**24.72s**	**349.67**	**3.21x**	**6.84x**	–	32.06
CSD (Wang et al., 2026)	33.27s	896.55	2.38x	2.67x	–	32.08
CSD-VP	**27.68s**	**649.10**	**2.86x**	**3.68x**	–	32.09
SJD-SV (Zhang et al., 2026)	30.48s	826.32	2.60x	2.89×	–	32.13
SJD-SV-VP	**26.74s**	**624.24**	**2.97x**	**3.83×**	–	32.12
MS-COCO 2017						
Lumina-mGPT (Liu et al., 2024)	86.55s	2379	1.00×	1.00×	30.79	31.31
Jacobi Decoding (Song et al., 2021)	85.64s	2312	1.01x	1.03x	30.78	31.31
SCD (So et al., 2025a)	33.28s	814.5	2.60x	2.92x	30.73	31.32
SJD-PV (Yu et al., 2026)	35.92s	945.2	2.41x	2.52x	31.55	31.32
SJD (Teng et al., 2024)	40.10s	1058.6	2.16x	2.25×	30.78	31.31
SJD-VP	**26.95s**	**729.70**	**3.21x**	**3.26x**	**30.77**	**31.42**
LANTERN(Jang et al., 2025)	33.80s	655.37	2.56x	3.63x	31.72	31.32
LANTERN-VP	**26.71s**	**590.33**	**3.24x**	**4.03x**	**31.70**	**31.43**
GSD (So et al., 2025b)	34.12s	925.89	2.54x	2.57x	31.50	31.33
GSD-VP	**30.78s**	**689.58**	**2.81x**	**3.45x**	**31.43**	**31.46**
SJD++ (W=96) (Teng et al., 2025)	27.65s	369.41	3.13×	6.44×	31.48	31.27
SJD-VP++ (W=96)	**25.81s**	**345.76**	**3.35x**	**6.88x**	31.48	31.26
CSD (Wang et al., 2026)	34.10s	922.87	2.54x	2.58x	31.66	31.31
CSD-VP	**27.70s**	**650.51**	**3.12x**	**3.66x**	31.68	31.31
SJD-SV (Zhang et al., 2026)	32.92s	866.8	2.63x	2.74×	30.76	31.33
SJD-SV-VP	**28.21s**	**664.37**	**3.07x**	**3.58×**	30.79	31.31



Figure 3: Qualitative Experiment across Five Diverse Prompt Categories.

4.2 Experimental Results

Quantitative Evaluation In Table 2, we compare SJD-VP with recent speculative decoding methods on Parti-prompt and MS-COCO 2017. SJD-VP and its variants consistently reduce latency and NFE while maintaining comparable generation quality across both benchmarks. These results demonstrate the effectiveness and broad applicability of our verification prediction mechanism, as well as its compatibility with existing speculative decoding methods.

Qualitative Evaluation To evaluate different aspects of generation performance, in Figure 3, we present images generated by our method and baseline methods, conditioned on text prompts across five diverse categories: (1) **Counting**, assessing numerical accuracy and object fidelity; (2) **Portrait realism**, evaluating facial details and spatial consistency; (3) **Fast motion**, focusing on edge sharpness and temporal coherence; (4) **Indoor multi-object layout**, examining spatial relationships, occlusion handling, and material realism; and (5) **Outdoor night street**, assessing reflections, lighting effects, and overall scene coherence. Please refer to the Appendix A.1 for the specific text prompts. From Figure 3, we can see that our methods consistently produce sharper details and fewer artifacts compared to baselines. Crucially, this improvement is achieved with an average **1.3× NFE reduction**, confirming that SJD-VP improves both decoding efficiency and image quality.

Table 3: Ablation of Bayesian Fusion.

Drafting Strategy p_t	Drafting Strategy S_t	Drafting Strategy p'_t	Latency	NFE	CLIP
✓			36.07 s	1035.3	32.09
	✓		32.13 s	899.7	32.10
		✓	**27.00 s**	**720.7**	**32.15**

Table 4: Ablation of Mask Growth Steps.

Growth Steps (N)	Latency	NFE	CLIP
$N = 0$	30.35 s	831.2	32.09
$N = 1$	28.21 s	763.9	32.11
$N = 2$	27.67 s	744.1	32.12
$N = 3$	**27.00 s**	**720.7**	**32.15**
$N = 4$	28.83 s	785.5	32.10

Table 5: Evaluation across Various Models.

Method	Latency	NFE	CLIP
SJD (LlamaGen)	19.86 s	569.0	28.16
SJD-VP (LlamaGen)	**19.26 s**	**455.39**	**28.18**
SJD (Emu3)	191.78 s	3571.1	32.13
SJD-VP (Emu3)	**169.49 s**	**2511.2**	**32.14**
SJD (Janus-Pro-7B)	5.18 s	283.0	32.30
SJD-VP (Janus-Pro-7B)	**4.00 s**	**204.8**	32.25

Table 6: Ablation of Jacobi Window Sizes W.

Method	Latency	NFE	CLIP
SJD ($W = 16$)	36.07 s	1035.3	32.09
SJD-VP ($W = 16$)	**27.00 s**	**720.6**	**32.10**
SJD ($W = 32$)	36.01 s	1032.4	32.08
SJD-VP ($W = 32$)	**21.68 s**	**537.6**	**32.08**
SJD ($W = 64$)	36.07 s	1035.2	32.10
SJD-VP ($W = 64$)	**18.71 s**	**401.7**	**32.11**

4.3 Ablation Study

Does Bayesian Fusion yield superior performance? To answer this question, we generate draft tokens based on p_t, S_t, and p'_t, corresponding to the original SJD, prediction score, and Bayesian Fusion, respectively. As shown in Table 3, the conditional prior (S_t) surpasses the baseline SJD (p_t), demonstrating that the prior effectively captures useful growth patterns. However, it still falls short of Bayesian Fusion (p'_t). This indicates that while S_t provides accurate structural guidance, it lacks semantic context. Therefore, Bayesian Fusion is necessary to align these growth trends with the semantically valid distribution p_t, ensuring optimal performance.

How does the number of consecutive growth steps N impact performance? To answer this, we analyze the sensitivity of the mask which forces SJD-VP to focus on tokens with an N-step consecutive growth. In Table 4, the results exhibit an initial performance improvement followed by a decline. Within an optimal range (e.g., $N \leq 3$), increasing N enhances the reliability of the prior, as the multi-step constraint effectively filters out tokens that are not in a state of consecutive probability growth. However, further tightening the constraint (e.g., $N = 4$) renders it overly strict. This leads to a scarcity of tokens that satisfy the continuous growth condition, thereby limiting the effective scope of the prior guidance and causing a regression in efficiency.

How does SJD-VP perform across various autoregressive image generation models? To evaluate our approach across various model architectures, we compare SJD-VP against baseline SJD on three base models: LlamaGen (Sun et al., 2024), Emu3 (Wang et al., 2024) and Janus–Pro (Chen et al., 2025). As shown in Table 5, SJD-VP achieves substantial speedups on all three models. Generation quality remains stable across models, with CLIP scores virtually unchanged. This shows our verification prediction strategy is applicable and effective for accelerating decoding inference.

How does the Jacobi window size W affect performance? To investigate the influence of the Jacobi window size W on the efficiency of SJD-VP, we conduct ablations with $W \in \{16, 32, 64\}$. As shown in Table 6, all three window configurations benefit from SJD-VP. Notably, larger window sizes yield superior performance, as the expanded sampling range enables more effective verification prior. When $W = 64$, SJD-VP achieves dramatic improvements in both latency and NFE while maintaining comparable generation quality. This demonstrates that increased window sizes pair exceptionally well with our approach, delivering optimal efficiency and generation quality.

5 Conclusions

In this paper, we propose a novel Speculative Jacobi Decoding framework with verification prediction that improves decoding efficiency while maintaining generation quality of AR image generation by aligning the drafting step with the verification step within the SJD decoding process. In particular, our method is plug-and-play, which can be seamlessly integrated into existing SJD variants, by simply replacing their draft steps while keeping all other components unchanged. Results show that our method achieves a superior efficiency–quality trade-off compared to state-of-the-art approaches.

AI USE DISCLOSURE

Generative AI tools were used to assist with polishing of the main text and supplementary materials. All AI-assisted content was reviewed and verified by the authors. All experimental results, mathematical claims, and conclusions were independently checked by the authors.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11305–11315. IEEE, 2022.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pp. 1691–1703. PMLR, 2020.

Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Doohyuk Jang, Sihwan Park, June Yong Yang, Yeonsung Jung, Jihun Yun, Souvik Kundu, Sung-Yub Kim, and Eunho Yang. Lantern: Accelerating visual autoregressive models with relaxed speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 37871–37900, 2025.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International conference on machine learning*, pp. 19274–19286. PMLR, 2023.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Dongyang Liu, Shitian Zhao, Le Zhuo, Weifeng Lin, Yi Xin, Xinyue Li, Qi Qin, Yu Qiao, Hongsheng Li, and Peng Gao. Lumina-mgpt: Illuminate flexible photorealistic text-to-image generation with multimodal generative pretraining. *arXiv preprint arXiv:2408.02657*, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Beyond next-token: Next-x prediction for autoregressive visual generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15781–15791. IEEE, 2025.

Junhyuk So, Hyunho Kook, Chaeyeon Jang, and Eunhyeok Park. Speculative coupled decoding for training-free lossless acceleration of autoregressive visual generation. *arXiv preprint arXiv:2510.24211*, 2025a.

Junhyuk So, Juncheol Shin, Hyunho Kook, and Eunhyeok Park. Grouped speculative decoding for autoregressive image generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15375–15384. IEEE, 2025b.

Yang Song, Chenlin Meng, Renjie Liao, and Stefano Ermon. Accelerating feedforward computation via parallel nonlinear equation solving. In *International Conference on Machine Learning*, pp. 9791–9800. PMLR, 2021.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Yao Teng, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. *arXiv preprint arXiv:2410.01699*, 2024.

Yao Teng, Zhihuan Jiang, Han Shi, Xian Liu, Xuefei Ning, Guohao Dai, Yu Wang, Zhenguo Li, and Xihui Liu. Sjd++: Improved speculative jacobi decoding for training-free acceleration of discrete auto-regressive text-to-image generation. *arXiv preprint arXiv:2512.07503*, 2025.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Mingcheng Wang, Junbo Qiao, Yunchen Li, Lingfu Jiang, Wei Li, Jie Hu, Jiao Xie, Zhou Yu, Xinghao Chen, Guixu Zhang, et al. Csd: Content-aware speculative decoding for efficient image generation. *arXiv preprint arXiv:2606.27829*, 2026.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

Yuqing Wang, Shuhuai Ren, Zhijie Lin, Yujin Han, Haoyuan Guo, Zhenheng Yang, Difan Zou, Jiashi Feng, and Xihui Liu. Parallelized autoregressive visual generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12955–12965. IEEE, 2025.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12966–12977. IEEE, 2025.

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 5505–5514. IEEE, 2018.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

Lijun Yu, José Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion-tokenizer is key to visual generation. In *International Conference on Learning Representations*, volume 2024, pp. 765–783, 2024.

Zhehao Yu, Baoquan Zhang, Bingqi Shan, Xinhao Liu, Dongliang Zhou, Guotao Liang, Guangming Ye, and Yunming Ye. Sjd-pv: Speculative jacobi decoding with phrase verification for autoregressive image generation. *arXiv preprint arXiv:2603.06666*, 2026.

Baoquan Zhang, Bingqi Shan, Shihao Fang, Kenghong Lin, Xutao Li, and Yunming Ye. Sjd-sv: Speculative jacobi decoding with semantics verification for autoregressive image generation. *arXiv preprint arXiv:2609.13245*, 2026.

A Appendix

A.1 Detail Experiment Settings and Results Interpretation

Prompt of qualitative experiment All qualitative results in both the main paper and this supplementary material are generated using the following prompts:

- **Counting / numeracy:** *2 blue spheres*
- **Portrait realism:** *female portrait*
- **Fast motion:** *splash frozen*
- **Indoor object layout:** *laptop*, *plant*
- **Outdoor night street:** *rainy Tokyo street*

Results Interpretation of GSD Acceleration We observe in Table 2 that SJD-VP exhibits a discrepancy between NFE and latency improvements when accelerating GSD. Specifically, while NFE shows noticeable reduction, the corresponding latency decrease is less pronounced. Through detailed analysis, we identify that this discrepancy stems from GSD's inherent sorting overhead during verification, which significantly increases the per-NFE execution time. In contrast, other baseline methods do not incur such sorting costs during their verification phases. Therefore, the mismatch between NFE and latency reduction observed in GSD is a well-founded phenomenon that reflects the architectural characteristics of the method.

A.2 Statistical Protocol of Figure 2

We further clarify the statistical protocol used in the motivation analysis of Figure 2. The *verification-accepted token* and the *correct token* are defined independently. A verification-accepted token refers to a token that is accepted by the SJD verification process. In contrast, the correct token refers to the reference token produced by the original autoregressive decoding without speculative decoding.

Importantly, the reference AR trajectory is not generated independently from the beginning. For each analyzed SJD state, we first fix the prefix that has already been committed before the current speculative window. Starting from this shared prefix, the original AR model performs standard autoregressive decoding to obtain the reference continuation at the corresponding token positions. Therefore, the SJD trajectory and the reference AR trajectory are aligned under the same committed prefix. The "correct" label in Figure 2 indicates agreement with this reference AR continuation rather than an additional human semantic annotation.

Figure 2(b) analyzes verification-accepted tokens and measures the proportion of them that exhibit consecutive probability growth before acceptance. For Figure 2(c), candidate tokens are divided into correct and incorrect groups according to whether their token identities match the corresponding reference AR token. Conditioned on a candidate having exhibited probability increases for n consecutive steps, we measure the probability that its probability continues to increase at the next Jacobi iteration.

For Figure 2(d), among candidate tokens satisfying the n-step consecutive-growth condition, the prediction accuracy is computed as

$$\mathrm{Acc}(n) = \frac{\{\text{growth candidates matching the reference AR token}\}}{\{\text{candidates satisfying the } n\text{-step growth condition}\}}. \tag{14}$$

Thus, Figure 2(b) examines how frequently the growth pattern appears among verification-accepted tokens, while Figure 2(c)–(d) examine whether the same pattern is also informative of the reference token produced by the original AR decoding.

Unless otherwise specified in the figure, the statistics in Figure 2(c)–(d) are collected using Lumina-mGPT on 300 randomly sampled prompts from PartiPrompts over all valid token positions. Only

candidate-token instances with sufficient Jacobi history to evaluate the corresponding consecutive-growth condition are included. When additional model names are explicitly shown in Figure 2, the corresponding statistics are computed separately for those models.

For each valid token position, all vocabulary tokens are treated as candidate tokens. As n increases, the consecutive-growth condition becomes more restrictive, and therefore the number of candidates included in the denominator naturally decreases.

A.3 Detailed Pseudocode of SJD-VP

In this section, we provide the detailed pseudocode of SJD-VP, as shown in Algorithm 1.

A.3.1 Verification Interface of SJD-VP

Drafting and Verification. SJD-VP modifies only the distribution used to draft candidate tokens. Specifically, candidate tokens are sampled from the posterior drafting distribution p'_t. After a candidate is drafted, the subsequent verification follows the original SJD procedure using the original drafting probability p_t and verification probability q. Therefore, p'_t participates only in token drafting and does not replace p_t in the original SJD verifier.

For completeness, under the standard SJD verification rule, a drafted token x is accepted with probability

$$a_{p,q}(x) = \min\left(1, \frac{q(x)}{p_t(x)}\right). \tag{15}$$

If the token is rejected, the rejected position is resampled from the calibrated residual distribution

$$r_{p,q}(x) = \frac{\max\left(0, q(x) - p_t(x)\right)}{\sum_{x'} \max\left(0, q(x') - p_t(x')\right)}. \tag{16}$$

The remaining positions after the first rejected token follow the original SJD sampling procedure. SJD-VP does not modify these verification and rejection-handling rules; it only changes which candidates are more likely to be proposed before verification.

Scope of the Sampling Guarantee. Although SJD-VP retains the original SJD verification rule and its probability inputs, it changes the proposal distribution from p_t to p'_t. Consequently, the exact distribution-preservation guarantee of standard SJD, which is established for its original matched proposal and verification procedure, does not directly transfer to SJD-VP. We therefore do not claim strict preservation of the original autoregressive sampling distribution. Instead, SJD-VP is designed as an empirical acceleration method, and its generation quality is directly evaluated using FID, CLIP-Score, and qualitative comparisons.

A.4 Derivation of Eq. 8

In this section, we provide the detailed derivation of the first-order Taylor expansion in Eq. 8, which is omitted in the main text.

Starting from Eq. 7, the Total Variation Distance can be written as

$$D_{\mathrm{TV}}(p'_t, q) = \frac{1}{2}\sum_x \left|p_t(x) - q(x) + m\omega(x)\hat{y}(x)\right|. \tag{17}$$

For simplicity, we define

$$u_x(m) = p_t(x) - q(x) + m\omega(x)\hat{y}(x). \tag{18}$$

For the generic case where $p_t(x) \neq q(x)$ for all coordinates with nonzero shift weight $\omega(x)$, applying the chain rule gives

$$\begin{aligned} \frac{d}{dm} D_{\mathrm{TV}}(p'_t, q) &= \frac{1}{2}\sum_x \frac{d}{dm}|u_x(m)| \\ &= \frac{1}{2}\sum_x \mathrm{sign}(u_x(m))\omega(x)\hat{y}(x). \end{aligned} \tag{19}$$

Substituting $u_x(m)$ and evaluating the derivative at $m = 0$, we obtain

$$\frac{d}{dm} D_{\mathrm{TV}}(p'_t, q)\Big|_{m=0} = \frac{1}{2} \sum_x \mathrm{sign}(p_t(x) - q(x))\omega(x)\hat{y}(x). \tag{20}$$

Finally, applying the first-order Taylor expansion at $m = 0$ gives

$$\begin{aligned} D_{\mathrm{TV}}(p'_t, q) &\approx D_{\mathrm{TV}}(p_t, q) + m \left.\frac{d}{dm} D_{\mathrm{TV}}(p'_t, q)\right|_{m=0} \\ &\approx D_{\mathrm{TV}}(p_t, q) + \frac{m}{2} \sum_x \mathrm{sign}(p_t(x) - q(x))\omega(x)\hat{y}(x), \end{aligned} \tag{21}$$

which gives Eq. 8 in the main text.

Boundary Case. When $p_t(x) = q(x)$ for a coordinate with $\omega(x) > 0$, the absolute-value term is non-differentiable at $m = 0$. Since the shift magnitude satisfies $m \geq 0$, we can instead consider the right directional derivative:

$$\frac{d}{dm} \left|p_t(x) - q(x) + m\omega(x)\hat{y}(x)\right|\Big|_{m=0^+} = \omega(x). \tag{22}$$

Therefore, the general right derivative at $m = 0$ can be written as

$$\begin{aligned} \left.\frac{d}{dm} D_{\mathrm{TV}}(p'_t, q)\right|_{m=0^+} = \frac{1}{2} &\sum_{x:p_t(x)\neq q(x)} \mathrm{sign}(p_t(x) - q(x))\omega(x)\hat{y}(x) \\ &+ \frac{1}{2} \sum_{x:p_t(x)=q(x)} \omega(x). \end{aligned} \tag{23}$$

Thus, Eq. 8 corresponds to the generic case considered in the main text, where coordinates satisfying $p_t(x) = q(x)$ have zero shift weight. If a nonzero shift is applied at such a coordinate, it contributes a non-negative boundary term and therefore makes the condition for reducing D_{TV} more restrictive rather than artificially favoring the analysis.

A.5 Derivation of Eq. 12

In this section, we provide the detailed derivation of Eq. 12, which is simplified in the main text.

Under the generic case considered above, $y(x) = \mathrm{sign}(q(x) - p_t(x)) \in \{-1, +1\}$ for all coordinates with nonzero shift weight. Starting from the term in Eq. 10, we separate the tokens according to whether their shift directions are predicted correctly or incorrectly:

$$\begin{aligned} \sum_x \omega(x)y(x)\hat{y}(x) = &\sum_{\hat{y}(x)=y(x)} \omega(x)y(x)\hat{y}(x) \\ &+ \sum_{\hat{y}(x)\neq y(x)} \omega(x)y(x)\hat{y}(x). \end{aligned} \tag{24}$$

When the shift direction is predicted correctly, $\hat{y}(x) = y(x)$ and therefore $y(x)\hat{y}(x) = +1$. Conversely, when the shift direction is predicted incorrectly, $\hat{y}(x) \neq y(x)$ and therefore $y(x)\hat{y}(x) = -1$. Thus, the above equation becomes

$$\begin{aligned} \sum_x \omega(x)y(x)\hat{y}(x) &= \sum_{\hat{y}(x)=y(x)} \omega(x) \cdot (+1) \\ &\quad + \sum_{\hat{y}(x)\neq y(x)} \omega(x) \cdot (-1) \\ &= \sum_{\hat{y}(x)=y(x)} \omega(x) - \sum_{\hat{y}(x)\neq y(x)} \omega(x). \end{aligned} \tag{25}$$

According to the definition of Q_w in Eq. 11, the total shift weights associated with correct and incorrect predictions are

$$\sum_{\hat{y}(x)=y(x)} \omega(x) = Q_w \sum_x \omega(x), \tag{26}$$

and

$$\sum_{\hat{y}(x)\neq y(x)} \omega(x) = (1 - Q_w) \sum_x \omega(x), \tag{27}$$

respectively.

Substituting these two expressions into the above equation gives

$$\begin{aligned} \sum_x \omega(x) y(x) \hat{y}(x) &= Q_w \sum_x \omega(x) - (1 - Q_w) \sum_x \omega(x) \\ &= (2Q_w - 1) \sum_x \omega(x), \end{aligned} \tag{28}$$

which gives Eq. 12 in the main text.

Ideal Directional Interpretation. The above analysis is motivated by a simple ideal case. Let $\Delta p(x) = p'_t(x) - p_t(x)$. If every nonzero probability shift satisfies

$$\Delta p(x)\big(q(x) - p_t(x)\big) \geq 0, \tag{29}$$

and the shift does not overshoot the verification probability,

$$|\Delta p(x)| \leq |q(x) - p_t(x)|, \tag{30}$$

then every adjusted coordinate moves monotonically toward the verification distribution. Consequently,

$$D_{\mathrm{TV}}(p'_t, q) \leq D_{\mathrm{TV}}(p_t, q). \tag{31}$$

In practice, SJD-VP predicts this ideal shift direction from probability trajectories and therefore does not require every token to be adjusted correctly. The weighted directional accuracy Q_w measures whether correctly directed probability shifts dominate in a weighted sense.

Scope of the Analysis. The theoretical analysis in the main text is a first-order local analysis around $m = 0$. Therefore, $Q_w > 0.5$ indicates that the probability shift is favorable for reducing D_{TV} under the first-order approximation in the generic case, rather than providing a global guarantee for an arbitrarily large probability shift. For a finite update, an excessively large shift may overshoot the verification distribution, which is why the above exact directional interpretation additionally requires the no-overshooting condition.

Moreover, reducing $D_{\mathrm{TV}}(p'_t, q)$ characterizes improved alignment between the posterior drafting distribution and the verification distribution. Since SJD-VP predicts the favorable shift direction probabilistically rather than perfectly for every token, the empirical Q_w results in Table 1 provide evidence that the realized probability shifts are predominantly aligned with this direction.

Relation between Proposal Reweighting and Acceptance. The TV analysis in Section 3.4 characterizes the alignment between the posterior drafting distribution p'_t and the verification distribution q. Since SJD-VP changes the proposal distribution while retaining the original SJD verifier, the actual acceptance rate under the unchanged verifier should be distinguished from the TV distance.

For a fixed decoding state, let

$$a_{p,q}(x) = \min\left(1, \frac{q(x)}{p_t(x)}\right) \tag{32}$$

denote the acceptance probability of token x under the original SJD verifier. The expected acceptance rates of the original SJD proposal and the SJD-VP proposal are

$$A_{\mathrm{SJD}} = \sum_x p_t(x) a_{p,q}(x), \tag{33}$$

and

$$A_{\mathrm{VP}} = \sum_x p'_t(x) a_{p,q}(x), \tag{34}$$

respectively. Therefore,

$$A_{\mathrm{VP}} - A_{\mathrm{SJD}} = \sum_x \left(p'_t(x) - p_t(x)\right) a_{p,q}(x). \tag{35}$$

This expression shows that acceptance improves when the proposal reweighting assigns more probability mass, in aggregate, to candidates with higher acceptance probability under the original verifier. In the ideal directional case, probability mass is shifted from tokens satisfying $p_t(x) > q(x)$ toward tokens satisfying $p_t(x) < q(x)$. For $p_t(x) < q(x)$, the original SJD verifier accepts the token with probability one, whereas for $p_t(x) > q(x)$ the acceptance probability is $q(x)/p_t(x) < 1$. Therefore, such an ideal mass transfer simultaneously moves the proposal distribution toward q and increases the expected acceptance rate.

In practice, SJD-VP predicts this favorable shift direction from historical probability dynamics and does not guarantee that every individual shift follows the ideal direction. Accordingly, the TV analysis and Q_w in the main text should be interpreted as evidence of proposal–verification distribution alignment, rather than as an exact equivalence between TV reduction and acceptance-rate improvement. The actual acceleration benefit is evaluated directly by the end-to-end latency and NFE results in Section 4.

A.5.1 Computation of Weighted Directional Accuracy.

We further clarify how the weighted directional accuracy Q_w in Table 1 is computed. For each evaluated drafting state, we first obtain the normalized posterior drafting distribution p'_t produced by SJD-VP and compute the actual probability shift

$$\Delta p(x) = p'_t(x) - p_t(x). \tag{36}$$

The realized shift direction is defined as

$$\hat{y}(x) = \mathrm{sign}(\Delta p(x)), \tag{37}$$

while the reference direction toward the verification distribution is

$$y(x) = \mathrm{sign}(q(x) - p_t(x)). \tag{38}$$

For the empirical evaluation, we directly use the magnitude of the actual normalized probability shift, $|\Delta p(x)|$, as the shift weight. Accordingly, the model-level weighted directional accuracy is computed as

$$Q_w = \frac{\sum_s \sum_x |\Delta p_s(x)| \mathbf{1}\left[\Delta p_s(x)\big(q_s(x) - p_s(x)\big) > 0\right]}{\sum_s \sum_x |\Delta p_s(x)|}, \tag{39}$$

where s indexes the evaluated drafting states over prompts and token positions. The numerator therefore accumulates the probability-shift magnitude whose realized direction agrees with the direction toward the verification distribution, while the denominator accumulates the total probability-shift magnitude. Coordinates with zero probability shift contribute zero weight.

This computation directly evaluates whether the realized probability updates introduced by SJD-VP are predominantly aligned with the verification distribution. It is also consistent with the decomposition $\Delta p(x) = m\omega(x)\hat{y}(x)$ used in the main-text analysis, where the shift magnitude can be absorbed into the corresponding empirical weight. The reported average in Table 1 is the arithmetic mean of the model-level Q_w values.

The role of Q_w also clarifies why arbitrary proposal reweighting is insufficient. A reweighting strategy improves proposal–verification alignment only when the probability shifts are predominantly directed toward q, corresponding to $Q_w > 0.5$ under our first-order analysis. Static sharpening does not explicitly use verification-related historical dynamics and therefore has no guarantee of satisfying this condition. In contrast, SJD-VP achieves $Q_w > 0.5$ across all evaluated models, indicating that its realized probability shifts are predominantly aligned with the verification-favorable direction.

Algorithm 1 The Proposed SJD-VP Method

Require: Pretrained AR model θ, prefix c, speculative window W, history buffer $\mathcal{H}$, history length L, decay factor γ, Top-K ratio ρ
Ensure: Generated sequence X

while c is not terminated **do**
  Obtain current token distributions p_t from the SJD iteration
  // Phase 1: Verification Prior Estimation
  Retrieve historical distributions $\{p_{t-L}, \ldots, p_{t-1}\}$ from $\mathcal{H}$
  Compute historical reference distribution $\bar{p}_t$ from $\{p_{t-1}, \ldots, p_{t-L}\}$ by Eq. (1)
  Compute probability-growth score $S_t(x) \leftarrow \log p_t(x) - \log \bar{p}_t(x)$ by Eq. (2)
  Compute consecutive-growth mask $M_t(x)$ by Eq. (3)
  // Phase 2: Bayesian Verification Fusion
  Select candidate set $\mathcal{C}_t \leftarrow \mathrm{TopK}_\rho(p_t)$
  for each token $x \in \mathcal{V}$ **do**
    if $x \in \mathcal{C}_t$ **then**
      $w_t(x) \leftarrow \exp(\log p_t(x) + M_t(x) S_t(x))$
    else
      $w_t(x) \leftarrow p_t(x)$
    end if
  end for
  Normalize w_t to obtain posterior distribution p'_t following Eq. (4)
  Draft speculative tokens from p'_t
  // Token Verification
  Verify drafted tokens with θ following standard SJD verification
  Accept verified tokens and update prefix c
  Update history buffer $\mathcal{H}$ with p_t
end while
$X \leftarrow c$
return X

A.6 Supplementary Ablation Study

In this section, we provide additional ablations on the EWA decay factor, Top-k filtering strategy, and computational overhead of SJD-VP.

How does the EWA Decay Factor γ impact Verification Prediction? To answer this question, we evaluate EWA Decay Factor γ across a range of values. The results shown in Table 7 indicate that performance degrades at both extremes. A lower γ (e.g., 0.2, 0.4) results in rapid memory decay, making the predictor overly sensitive to recent iterations. Conversely, a higher γ (e.g., 1.0) introduces excessive lag, causing the prediction to drift behind the actual trend due to high inertia. The optimal balance is achieved at $\gamma = 0.8$, where the predictor retains sufficient history while remaining responsive to new growth patterns.

Table 7: Ablation of Decay Factor γ on PartiPrompt.

γ	Latency	NFE	CLIP
1.0	28.75 s	781.6	32.12
0.8	**27.00 s**	**720.7**	**32.15**
0.6	27.51 s	737.8	32.11
0.4	27.92 s	752.0	32.09
0.2	28.04 s	756.9	32.12

How does the Top-k filtering ratio impact performance? To answer this, we investigate the influence of the Top-k filtering ratio on performance of SJD-VP. As shown in Table 8, stricter filtering yields a dual benefit. First, it significantly reduces redundant computational overhead by strategically limiting the sampling scope to only the most promising candidates, directly leading to lower latency. Second, it enhances robustness by aggressively pruning low-probability tail noise, preventing potential interference from irrelevant tokens. Consequently, the setting of $k = 10\%$ achieves the optimal balance, delivering the fastest inference speed without compromising generation quality.

Table 8: Ablation of the Top-k filtering strategy on PartiPrompt.

Filter Top-k	Latency	NFE	CLIP
w/o Filter	32.19 s	904.8	32.09
$k = 70\%$	30.02 s	833.6	32.10
$k = 50\%$	28.47 s	759.0	32.10
$k = 30\%$	27.51 s	739.4	32.11
$k = 10\%$	**27.00 s**	**720.7**	**32.15**

What is the computational overhead of SJD-VP? To assess the specific cost of our proposed modules, we measure the additional computational burden imposed by different methods. As reported in Table 9, SJD-VP incurs a minimal per-step overhead of 1.026 ms, confirming that our Bayesian Fusion and Prior Estimation modules are lightweight.

Furthermore, we analyze the behavior of GSD. While GSD effectively reduces NFE, it introduces a significantly higher overhead of 1.967 ms per step—nearly double that of SJD-VP. This is likely attributed to its internal reliance on logits sorting and re-ranking operations, which are computationally more demanding. This finding clarifies the discrepancy between NFE reduction and actual speedup. This observation also provides empirical support for the analysis presented in Section A.1.

Table 9: Computational Cost of SJD-VP Analysis.

	SJD-VP	GSD	GSD-VP
Per-step overhead	1.026ms	1.967 ms	2.926 ms