BLOCK VERIFICATION FOR SPECULATIVE COUPLED DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive image and video models generate long sequences of visual tokens, making token-by-token decoding slow. Speculative Coupled Decoding (SCD) accelerates generation without a separate draft model by reusing draft tokens across Jacobi iterations. Its token-wise verifier stops at the first rejection, whereas Block Verification can accept longer prefixes using the same target-model evaluation. We replace SCD's verifier with Block Verification. However, verification changes the distribution of the tokens left for reuse, so applying SCD's updates with their original stored distributions can bias the output. We derive the exact post-verification distributions and use them to update the tokens before reuse. The method requires only a small code change to SCD, with no training or extra target-model passes per iteration. We prove that the resulting decoder is lossless: it preserves the target-model sampling distribution exactly. Across two image generation models and one video generation model, our method reduces mean target-model calls by 2.1%–20.7% and mean generation time by 1.6%–19.9% compared with SCD at matching window sizes.

1 INTRODUCTION

Modern autoregressive (AR) models represent outputs as sequences of discrete tokens. This applies in the case of AR image and video generation, which entails hundreds or thousands of sequential model evaluations for generating every image or frame.

Speculative decoding reduces the number of sequential target-model calls by drafting several future tokens, scoring their prefixes in parallel, and committing a prefix that preserves the target distribution (Leviathan et al., 2023; Chen et al., 2023). Speculative Jacobi Decoding (SJD) forms the draft through consecutive Jacobi iterations rather than from a separately trained model (Teng et al., 2024). Speculative Coupled Decoding (SCD) uses coupling to keep more draft tokens unchanged between Jacobi iterations (So et al., 2025). We build on its maximal-coupling variant, which we refer to as SCD throughout this paper unless otherwise stated.

SCD improves how the Jacobi draft is formed, but still uses token-wise modified rejection sampling (MRS) for verification. Token-wise verification stops at the first rejection. Block Verification considers prefixes of every length and selects the longest accepted prefix, using the same draft and target probabilities. Sun et al. (2025) show that it preserves the target distribution and maximizes expected acceptance length in a standard speculative-decoding step with a single draft path.

SCD reuses the unused draft suffix as the next Jacobi state, whereas ordinary speculative decoding discards it. The verifier's output reveals information about these tokens, so their conditional distributions can differ from their stored draft distributions. Using the original draft distributions in the next MRS update can therefore introduce bias.

Our contributions are:

- We introduce SCD with Block Verification (Algorithm 1), derive the draft-token distributions needed for the MRS updates after verification, and prove that the resulting decoder is lossless. The method requires only a small code change to SCD, with no separate draft model, training, or extra target-model passes per iteration.

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.

- We evaluate the method on two image generation models and one video generation model. It reduces mean target-model calls and generation time relative to SCD at every tested window size.

Figure 1 illustrates the update.

2 BACKGROUND

2.1 AUTOREGRESSIVE AND SPECULATIVE DECODING

Let $\mathcal{V}$ be the token vocabulary, c the committed context, and $O_{1:n}$ the sequence of not yet committed tokens. Ordinary autoregressive (AR) generation samples each token from the target model p_θ conditioned on the preceding tokens, giving the joint distribution

$$\mathbb{P}(O_{1:n} = o_{1:n} \mid c) = \prod_{i=1}^{n} p_\theta(o_i \mid c, o_{1:i-1}). \tag{1}$$

A decoder whose joint output distribution matches Equation (1) is considered *lossless* w.r.t. ordinary AR sampling.

Standard speculative decoding (Leviathan et al., 2023; Chen et al., 2023) reduces the number of sequential target-model passes by using a cheaper model to draft a sequence of L tokens, which are then evaluated simultaneously in a single target-model pass. The decoder then verifies the draft from left to right using modified rejection sampling (MRS). Verification returns the accepted draft tokens followed by one additional token: a correction token if a draft token is rejected, or a bonus token if the entire draft is accepted.

To verify each token, speculative decoding uses *Modified Rejection Sampling* (MRS): given a token $X \sim Q$ from a draft distribution Q and a target-model distribution P, MRS accepts X and sets $Y = X$ with probability

$$\min\left\{\frac{P(X)}{Q(X)}, 1\right\}. \tag{2}$$

In standard speculative decoding, Q and P condition on the same preceding tokens. If MRS rejects the draft token, it samples a correction token

$$Y \sim \mathrm{Normalize}\big([P - Q]_+\big). \tag{3}$$

Here $[z]_+ = \max\{z, 0\}$ is applied at each token, and Normalize scales these weights to sum to 1. For nonnegative weights w with positive sum, $\mathrm{Normalize}(w)(x) = w(x)/\sum_{z\in\mathcal{V}} w(z)$. The probability of token x being sampled and accepted is therefore

$$Q(x) \cdot \min\left\{\frac{P(x)}{Q(x)}, 1\right\} = \min\{P(x), Q(x)\}.$$

The remaining target probability mass, $P(x) - \min\{P(x), Q(x)\} = [P(x) - Q(x)]_+$, is referred to as the *residual mass*. Normalizing it gives the *residual distribution*, used to sample the correction token. The rejection probability is

$$r = 1 - \sum_{z\in\mathcal{V}} \min\{P(z), Q(z)\} = \sum_{z\in\mathcal{V}} [P(z) - Q(z)]_+.$$

Thus, when $r > 0$,

$$\begin{aligned}\mathbb{P}(Y = x) &= \underbrace{\min\{P(x), Q(x)\}}_{\text{accept } x} + \underbrace{r}_{\text{reject}} \underbrace{\frac{[P(x) - Q(x)]_+}{r}}_{\text{draw } x \text{ after rejection}} \\ &= \min\{P(x), Q(x)\} + \big(P(x) - \min\{P(x), Q(x)\}\big) \\ &= P(x).\end{aligned}$$

If $r = 0$, then $P = Q$ and every draft token is accepted. We write $(Y, K) \leftarrow \mathrm{MRS}(P, Q, X)$, where $K = 1$ records acceptance and $K = 0$ rejection. Algorithm 3 in Appendix C gives the procedure.

At the first rejection, verification stops and the rest of the draft is discarded. If all L draft tokens are accepted, the bonus token is sampled from the target model conditioned on the full draft. This gives a lossless decoder that appends between one and $L+1$ tokens per target-model pass.

MRS produces $Y \sim P$ and maximizes the probability of retaining the draft token (Jacob et al., 2020, Section 4.1):

$$\mathbb{P}(K=1) = \mathbb{P}(Y=X) = \sum_{x\in\mathcal{V}} \min\{Q(x), P(x)\}. \tag{4}$$

Such an update is called a *maximal coupling* of Q and P.

2.2 Speculative Jacobi and Coupled Decoding

Speculative Jacobi Decoding (SJD) uses the target model itself to repeatedly update a window of L tentative future tokens (Teng et al., 2024). In each iteration, one parallel target-model pass scores the whole window. The decoder then commits an accepted prefix followed by a correction or bonus token, updates the remaining draft tokens, and fills the newly opened positions for the next iteration.

Let c^t be the committed context at iteration t, and X_i^t the tentative token at window position i. One causal target-model pass computes

$$P_i^t(\cdot) = p_\theta(\cdot \mid c^t, X_{<i}^t), \qquad i = 1, \ldots, L+1. \tag{5}$$

Each P_i^t uses the earlier tokens $X_{<i}^t$ from the current draft. These tokens are already available before any updates, so the model can compute all $L+1$ distributions in parallel. They are then used to update the tentative tokens for the next pass. The first L distributions score the draft tokens; P_{L+1}^t provides the bonus token, if all are accepted. Each draft token also has a stored draft distribution p_i^t. MRS uses it together with P_i^t to decide whether to keep the existing token or sample a replacement.

SJD verifies the current draft token by token using MRS. For positions beyond the correction token, it independently samples updated tentative tokens $\widetilde{X}_i^t \sim P_i^t$ for the next draft. This can change an existing token even when its stored draft distribution and the new target distribution are similar. Later draft tokens are sampled using the earlier token values as context. Changing an earlier token can therefore make those later tokens less likely under the target model, increasing the chance of rejection in the next iteration.

Instead, *Speculative Coupled Decoding* (SCD) with *maximal coupling* applies MRS at every draft position (So et al., 2025):

$$(\widetilde{X}_i^t, K_i) \leftarrow \mathrm{MRS}(P_i^t, p_i^t, X_i^t), \qquad i = 1, \ldots, L. \tag{6}$$

By the MRS property above, the updated token has distribution P_i^t, with the largest possible probability of retaining X_i^t. These updates use independent randomness and run at all positions; the first rejection determines which tokens are committed.

If the first rejection occurs at position $\tau+1$, the decoder commits $X_{1:\tau}^t$ and the correction token $\widetilde{X}_{\tau+1}^t$. For every position $i \geq \tau+2$, the decoder stores $\widetilde{X}_i^t$ together with its distribution P_i^t, even if MRS kept the original token. Each token and its stored distribution are then shifted left by $\tau+1$ positions. An independent sample from P_{L+1}^t extends the shifted suffix, and any remaining positions are filled with random tokens. If all L draft tokens are accepted, that sample is committed as a bonus token and the entire window is refilled. Thus SCD uses the same MRS updates to verify the current draft and form the next one. Figure 1(a) compares SCD's verifier with ours; panel (b) illustrates how our Block Verification variant commits tokens and shifts the updated suffix.

2.3 Block Verification

Token-wise verification stops accepting draft tokens at the first rejection. Block Verification considers prefixes of every length, which can increase the average number of draft tokens accepted per target-model pass (Sun et al., 2025).

For one iteration, write the draft as $X_{1:L}$, its draft distributions as Q_j, and its target distributions as

$$P_j(\cdot) = p_\theta(\cdot \mid c, X_{1:j-1}). \tag{7}$$

In SCD, $Q_j = p_j^t$ and $P_j = P_j^t$.

We use Algorithm 2 of Sun et al. (2025) as the BlockVerify subroutine in our decoder. For each length $j = 1, \ldots, L$, it decides whether to accept the entire prefix $X_{1:j}$ using an independent uniform random draw. It selects the longest accepted prefix, with length τ (or $\tau = 0$ if no prefix is accepted). Rejecting a shorter prefix does not prevent a longer prefix, including those same tokens, from being accepted; Figure 1(a) shows an example. The verifier returns $X_{1:\tau}$ followed by a correction token if $\tau < L$, or a bonus token sampled from P_{L+1} if the full draft is accepted. Block Verification is lossless: it preserves the output distribution of ordinary autoregressive sampling (Sun et al., 2025, Theorem 1). Algorithm 2 gives the pseudocode.

Our method, introduced in Section 4, replaces SCD's token-wise MRS verifier with Block Verification while retaining its reuse of draft tokens beyond the correction token. To determine their distribution after verification, we need the conditional probability that the verifier returns at least j draft tokens. Block Verification computes $\rho_0, \ldots, \rho_L$ recursively:

$$\rho_0 = 1, \qquad \rho_j = \min\left\{1, \rho_{j-1}\frac{P_j(X_j)}{Q_j(X_j)}\right\}. \tag{8}$$

Sun et al. (2025, Lemma 3) show that, given the context c and draft prefix $X_{1:j}$, ρ_j is the probability that the verifier returns at least j draft tokens. In our decoder, the stored draft distributions are themselves random, so we also condition on them. If each draft token X_j has distribution Q_j given $(c, Q_{1:j}, X_{<j})$, then

$$\mathbb{P}(\tau \geq j \mid c, Q_{1:j}, X_{1:j}) = \rho_j, \qquad 1 \leq j \leq L. \tag{9}$$

Their proof also applies with this additional conditioning because each draft token still follows its specified distribution. Verification also supplies one correction or bonus token, so ρ_j is the probability of producing at least $j + 1$ tokens to append to the generated sequence.

3 Related Work

Speculative verification. Speculative decoding verifies drafts from a faster model while preserving the target model's output distribution (Leviathan et al., 2023; Chen et al., 2023). SpecTr uses optimal transport to select among multiple draft candidates at each token position (Sun et al., 2023). Block Verification considers every prefix of a single draft sequence and returns the longest accepted prefix (Sun et al., 2025). Traversal Verification traverses a draft tree from its leaves toward the root, considering acceptance of the entire sequence from the root to each node (Weng et al., 2026).

Parallel and Jacobi decoding. Song et al. (2021) formulate feedforward computation, including autoregressive sampling, as a system of nonlinear equations that can be solved with parallel Jacobi updates. Santilli et al. (2023) apply Jacobi and Gauss–Seidel iterations to greedy decoding for machine translation without retraining the model. Lookahead Decoding generates and verifies n-grams from Jacobi iterations without a separate draft model (Fu et al., 2024). CLLMs are obtained by fine-tuning the target model to predict the converged output from intermediate Jacobi states (Kou et al., 2024).

Jacobi decoding for visual generation. SJD combines Jacobi updates with speculative verification for stochastic text-to-image generation (Teng et al., 2024). SCD's maximal-coupling variant uses MRS to keep draft tokens unchanged as often as possible between iterations (So et al., 2025). SJD++ reuses tokens beyond the correction token when the ratio of their probability under the current prefix to that under the previous prefix exceeds a threshold, and resamples the others (Teng et al., 2025). SJD-PAC continues token-wise rejection sampling beyond the first rejection to form the next draft, and builds a shallow tree of draft branches rooted at the corrected token (Kang et al., 2026). We replace SCD's token-wise verifier with Block Verification. To preserve losslessness when reusing the suffix, we derive each token's distribution after verification and use it in the MRS update.

4 BLOCK VERIFICATION FOR SCD

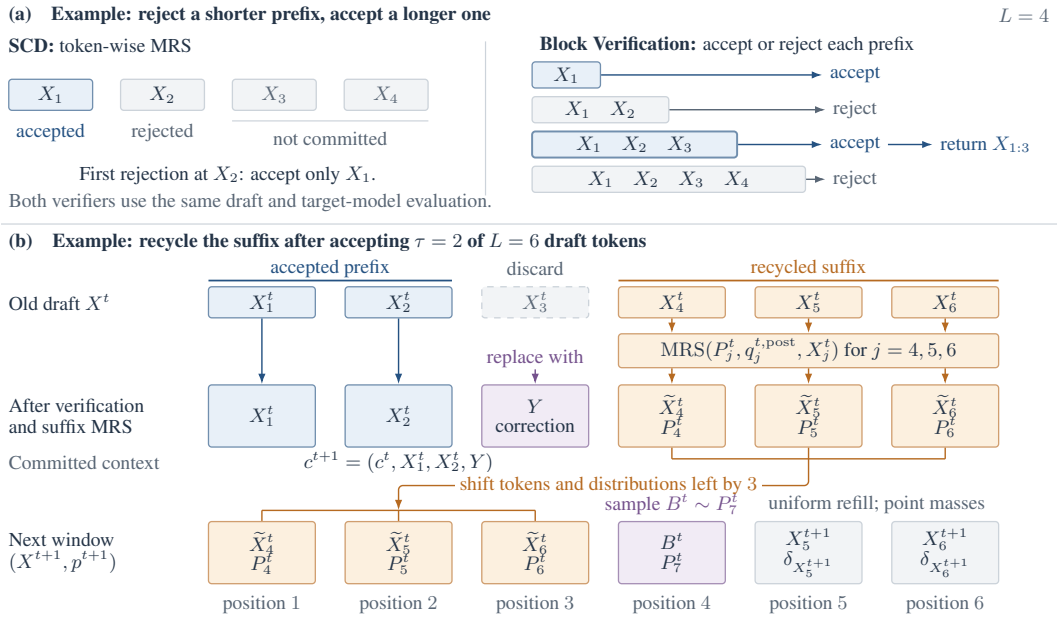

Figure 1: **Block Verification inside SCD.** (a) Example for $L = 4$: token-wise verification accepts only X_1, whereas Block Verification accepts $X_{1:3}$ despite rejecting $X_{1:2}$. Each verifier then appends a correction token (not shown). (b) Example: reusing a six-token draft after accepting two tokens. After committing $X_{1:2}^t$ and the correction token Y, MRS updates $X_{4:6}^t$ using their distributions after verification. The updated tokens and their stored distributions $P_{4:6}^t$ shift three positions left; an independent sample $B^t \sim P_7^t$ and uniformly sampled tokens complete the next draft. Each box in the next window shows a token above its stored draft distribution; δ_x assigns probability one to x.

Our method (Algorithm 1) replaces SCD's token-wise MRS verification with Block Verification, as illustrated in Figure 1(a). Each iteration makes one target-model pass, computing $P_1^t, \ldots, P_{L+1}^t$ from the current context and draft. Block Verification selects the longest accepted prefix, of length τ, and returns one additional token Y. The decoder appends $X_{1:\tau}^t$ followed by Y to the output: Y is a correction token if $\tau < L$, or a bonus token drawn from P_{L+1}^t if all draft tokens are accepted.

If decoding continues, we reuse the draft tokens at positions $j \geq \tau + 2$, after position $\tau + 1$ occupied by Y (Figure 1(b)). As in SCD, MRS keeps or replaces each of them in the *coupling update* (line 10), but uses the corrected input distribution $q_j^{t,\text{post}}$ instead of the stored p_j^t. Section 4.1 explains why this correction is needed, and Section 4.2 derives it.

4.1 WHY THE COUPLING UPDATE NEEDS CORRECTED PROBABILITIES

It is tempting to keep SCD's MRS update unchanged when replacing its verifier with Block Verification, but using p_j^t can then give an updated distribution different from P_j^t.

Consider a fair coin whose result is sent to another function only when it is tails. The receiving function gets only tails, although every toss was fair. The selection changes the proportions among the results reaching the function; it does not change any individual result. Block Verification makes a similar selection: a token is reused only after every prefix containing it has been rejected. These rejections can be more likely for some token values than others.

MRS needs to know how often each token value reaches it, because that determines how often it should keep that value. The MRS guarantee applies when its input distribution describes the tokens left for reuse. The stored p_j^t describes the token before verification, so those probabilities may no longer describe the inputs reaching MRS.

For a concrete example, consider position 4 in Figure 1(b). Suppose the vocabulary is $\mathcal{V} = \{a, b\}$, the stored draft distribution is $p_4^t = (0.98, 0.02)$, the target distribution is $P_4^t = (0.99, 0.01)$, and $\rho_3 = 98/99$. Here ρ_3 is the probability that verification accepts at least the first three draft tokens. By Equations (8) and (9), the probabilities of accepting through position 4 are

$$\text{if } X_4^t = a: \quad \rho_4 = \min\left\{1, \frac{98}{99} \cdot \frac{0.99}{0.98}\right\} = 1,$$
$$\text{if } X_4^t = b: \quad \rho_4 = \min\left\{1, \frac{98}{99} \cdot \frac{0.01}{0.02}\right\} = \frac{49}{99}.$$

Block Verification accepts a with probability 1, but b only with probability $49/99$. Hence, only b can be rejected by Block Verification at position 4. Therefore, any token at position 4 that reaches the coupling update must be b. We denote the distribution of tokens reaching the coupling update at position j by $q_j^{t,\text{post}}$. Here, $q_4^{t,\text{post}} = (0, 1)$.

Probabilities at position 4	a	b
Before verification (p_4^t)	98%	2%
Tokens left for reuse, before the coupling update ($q_4^{t,\text{post}}$)	0%	100%
Required after the coupling update (P_4^t)	99%	1%

Naive update. Using $p_4^t = (0.98, 0.02)$ incorrectly assumes that the token entering MRS is b with probability 2%. Using the acceptance rule in line 2 of Algorithm 3, MRS would keep the input token b with probability $\min\{1, P_4^t(b)/p_4^t(b)\} = \min\{1, 0.01/0.02\} = 1/2$. Otherwise, MRS would draw from the residual distribution $\text{Normalize}([P_4^t - p_4^t]_+)$ (line 5 of Algorithm 3). Here $[P_4^t - p_4^t]_+ = (0.01, 0)$, so the new token is always a. Since the input must be b, the updated token would be a with probability 50% and b with probability 50%, instead of the required 99% for a and 1% for b.

Our update. With $q_4^{t,\text{post}} = (0, 1)$, MRS keeps the input token b with probability $\min\{1, P_4^t(b)/q_4^{t,\text{post}}(b)\} = \min\{1, 0.01/1\} = 0.01$. On rejection, the residual weights are $[P_4^t - q_4^{t,\text{post}}]_+ = (0.99, 0)$, so the new token is again a. The updated token is therefore a with probability 0.99 and b with probability 0.01, as required by P_4^t.

4.2 The distribution entering the coupling update

We now derive the distribution of the tokens that reach the coupling update. We condition on $H_j = (c^t, p_{1:j}^t, X_{<j}^t)$: the context, stored draft distributions through position j, and earlier draft tokens. The stored distribution describes the token before verification:

$$p_j^t(x) = \mathbb{P}(X_j^t = x \mid H_j). \tag{10}$$

Let R_j be the event that token X_j^t reaches the coupling update. The coupling update calls $\text{MRS}(P_j^t, q_j^{t,\text{post}}, X_j^t)$ (Algorithm 3) with the target distribution P_j^t, the input token's distribution $q_j^{t,\text{post}}$, and the existing token X_j^t. The input distribution $q_j^{t,\text{post}}$ must describe the tokens that reach this update:

$$q_j^{t,\text{post}}(x) := \mathbb{P}(X_j^t = x \mid H_j, R_j). \tag{11}$$

For a token to reach the update, every prefix containing position j must first be rejected ($\tau < j$). Write $\rho_j(x)$ for ρ_j when $X_j^t = x$. By Equation (9), these prefixes are all rejected with probability $1 - \rho_j(x)$. For $p_j^t(x) > 0$, the probability that the token is x and is not accepted is therefore

$$\begin{aligned}\mathbb{P}(X_j^t = x, \tau < j \mid H_j) &= \underbrace{p_j^t(x)}_{\text{token is } x} \cdot \underbrace{(1 - \rho_j(x))}_{\text{not accepted, given } x} \\ &\overset{(8)}{=} p_j^t(x) - \min\{p_j^t(x), \rho_{j-1} P_j^t(x)\} \\ &= [p_j^t(x) - \rho_{j-1} P_j^t(x)]_+. \end{aligned} \tag{12}$$

The final expression also holds when $p_j^t(x) = 0$.

When $\tau = j - 1$, X_j^t is replaced by the correction token and does not reach the coupling update. The decoder also skips the update once it has produced all output tokens. Let C_j be the probability that an unaccepted token reaches the coupling update:

$$C_j := \mathbb{P}(R_j \mid H_j, \tau < j).$$

To compute $q_j^{t,\mathrm{post}}(x)$, we divide the probability that the token is x and reaches the coupling update by the probability that it reaches the update:

$$\begin{aligned} q_j^{t,\mathrm{post}}(x) &= \frac{\mathbb{P}(X_j^t = x, R_j \mid H_j)}{\mathbb{P}(R_j \mid H_j)} \\ &= \frac{\mathbb{P}(X_j^t = x, R_j \mid H_j)}{\sum_{z \in \mathcal{V}} \mathbb{P}(X_j^t = z, R_j \mid H_j)}. \end{aligned}$$

We now rewrite the numerator $\mathbb{P}(X_j^t = x, R_j \mid H_j)$ using the event that the token was not accepted ($\tau < j$):

$$\mathbb{P}(X_j^t = x, R_j \mid H_j) = \mathbb{P}(X_j^t = x, \tau < j \mid H_j) \cdot \mathbb{P}(R_j \mid H_j, \tau < j).$$

Substituting Equation (12) for the first factor and C_j for the second, then cancelling C_j, gives:

$$\begin{aligned} q_j^{t,\mathrm{post}}(x) &= \frac{C_j \cdot [p_j^t(x) - \rho_{j-1} P_j^t(x)]_+}{C_j \cdot \sum_{z \in \mathcal{V}} [p_j^t(z) - \rho_{j-1} P_j^t(z)]_+} \\ &= \frac{[p_j^t(x) - \rho_{j-1} P_j^t(x)]_+}{\sum_{z \in \mathcal{V}} [p_j^t(z) - \rho_{j-1} P_j^t(z)]_+}. \end{aligned} \tag{13}$$

If the denominator in Equation (13) is zero, the token cannot reach the coupling update, so this distribution is never needed. For the example in Section 4.1, the numerator of Equation (13) is zero for a and positive for b, so $q_4^{t,\mathrm{post}} = (0, 1)$.

We use this distribution in the coupling update, $\mathrm{MRS}(P_j^t, q_j^{t,\mathrm{post}}, X_j^t)$, and write $\widetilde{X}_j^t$ for the returned token. Given H_j and R_j, the input has distribution $q_j^{t,\mathrm{post}}$, so the MRS guarantee from Section 2.1 gives

$$\mathbb{P}(\widetilde{X}_j^t = x \mid H_j, R_j) = P_j^t(x), \qquad x \in \mathcal{V}. \tag{14}$$

We store P_j^t with the updated token for the next iteration.

4.3 Complete decoder and losslessness

When decoding continues, we construct the next draft from the updated tokens and new samples. The tokens returned by the coupling update shift left by $\tau + 1$ positions, each stored with P_j^t. Since P_j^t was computed on the old draft, it need not equal the target distribution under the new prefix; the next iteration verifies the token using that prefix.

If all draft tokens were accepted ($\tau = L$), they and the bonus token $Y \sim P_{L+1}^t$ have all been committed to the output. No draft tokens remain to reuse, so we fill every position of the next draft with a new uniformly sampled token. Otherwise, Y is the correction token; to complete the next draft, we draw a separate bonus token $B^t \sim P_{L+1}^t$, place it after the shifted tokens, and store it with P_{L+1}^t. Any remaining positions are filled with uniformly sampled tokens. Following SCD, each uniformly sampled token x is stored with the point mass δ_x, which assigns probability one to x. The initial draft is formed using the same random refill rule.

Theorem 1 (Losslessness). *Let $O_{1:N}$ be the N tokens generated by Algorithm 1 from context c_0 using target model p_θ. Then, for every $o_{1:N} \in \mathcal{V}^N$,*

$$\mathbb{P}(O_{1:N} = o_{1:N} \mid c_0) = \prod_{i=1}^{N} p_\theta(o_i \mid c_0, o_{1:i-1}). \tag{15}$$

Algorithm 1 Speculative Coupled Decoding with Block Verification

Require: AR model p_θ; initial context c_0; window length L; continuation length $N \geq 1$

$c \leftarrow c_0$, $t \leftarrow 0$
For $j = 1, \ldots, L$, draw $X_j^0 \sim \mathrm{Unif}(\mathcal{V})$ and set $p_j^0 \leftarrow \delta_{X_j^0}$
while $|c| < |c_0| + N$ **do**
  $P_j^t \leftarrow p_\theta(\cdot \mid c, X_{<j}^t)$ for $j = 1, \ldots, L+1$ in parallel ▷ One target pass
  $(\tau, Y, \rho_{0:L}) \leftarrow \mathrm{BlockVerify}(X_{1:L}^t, p_{1:L}^t, P_{1:L+1}^t)$ ▷ Algorithm 2
  Append $(X_{1:\tau}^t, Y)$ to c ▷ Accepted tokens and correction or bonus token
  if $|c| \geq |c_0| + N$, **return** the first $|c_0| + N$ tokens of c
  for $j = \tau + 2, \ldots, L$ **in parallel do**
    $q_j^{t,\mathrm{post}} \leftarrow \mathrm{Normalize}\Big(\big[p_j^t - \rho_{j-1} P_j^t\big]_+\Big)$ ▷ Input distribution, Equation (13)
    $\widetilde{X}_j^t \leftarrow$ token returned by $\mathrm{MRS}(P_j^t, q_j^{t,\mathrm{post}}, X_j^t)$ ▷ Coupling update
    $(X_{j-\tau-1}^{t+1}, p_{j-\tau-1}^{t+1}) \leftarrow (\widetilde{X}_j^t, P_j^t)$ ▷ Shift token and stored distribution
  if $\tau < L$, draw $B^t \sim P_{L+1}^t$ and set $(X_{L-\tau}^{t+1}, p_{L-\tau}^{t+1}) \leftarrow (B^t, P_{L+1}^t)$ ▷ Bonus token draw
  For $\ell = L - \tau + 1, \ldots, L$, draw $X_\ell^{t+1} \sim \mathrm{Unif}(\mathcal{V})$ and set $p_\ell^{t+1} \leftarrow \delta_{X_\ell^{t+1}}$ ▷ Random refill
  $t \leftarrow t + 1$

Proof. We prove two properties by induction over iterations: the committed tokens follow the target model, and every token in the next draft follows its stored distribution.

At initialization, Equation (10) holds because each stored distribution assigns probability one to its token. Suppose it holds at iteration t. Block Verification then gives the correct distribution for the tokens committed in this iteration (Section 2.3): given H_j and that the iteration emits at least j tokens, its jth token has distribution P_j^t for $j \leq L$. This accounts for both accepting a draft token and drawing a correction token. If all L draft tokens are accepted, the additional bonus token is drawn from P_{L+1}^t.

By Equation (14), the coupling update returns a token distributed as P_j^t. This remains true given H_j, the verification outcome, and earlier updated tokens, since these additional outcomes give no further information about the reused token's value. After shifting, each updated token still follows its stored distribution in the next draft. The bonus token B^t, drawn independently from P_{L+1}^t, also follows its stored distribution; each random refill token is determined by its stored point mass. Storing these distributions with the tokens therefore establishes Equation (10) for the next iteration whenever decoding continues.

Each iteration emits at least one token, so all N output tokens are produced. Regardless of which iteration emits token i, its preceding tokens are $(c_0, o_{<i})$. The Block Verification guarantee therefore gives conditional distribution $p_\theta(\cdot \mid c_0, o_{<i})$. Multiplying these conditional next-token probabilities gives the theorem. □

Cost. Each iteration computes the same $L + 1$ target distributions as SCD. Section 5 evaluates decoding efficiency.

5 Experiments

We test whether replacing SCD's verifier with Block Verification reduces target-model calls and generation time.

5.1 Setup

We compare SCD with Block Verification (SCD + BV, ours) against SCD, with AR and SJD as additional baselines.

We evaluate Lumina-mGPT-7B (Liu et al., 2024) and Janus-Pro-7B (Chen et al., 2025) for image generation, and Cosmos-1.0-Autoregressive-4B (Agarwal et al., 2025) for video continuation. Both

image models use the same 1,000 prompts from MS-COCO (Lin et al., 2014). For Cosmos, we use the first 9 frames of each of 500 RealEstate10K clips (Zhou et al., 2018) to generate 24 additional frames. For each model, all methods use the same inputs and generation settings.

We test window sizes $W \in \{16, 32, 64, 128, 256, 512\}$ for all three models, where a full window contains $L = W - 1$ draft tokens in Algorithm 1. Each generation runs on one H100 80GB GPU with batch size one.

We report two metrics: generation speedup relative to AR and block efficiency. Speedup is the mean AR generation time divided by the method's mean generation time. Block efficiency measures the average number of tokens produced per target-model call, including the correction or bonus token (Sun et al., 2025). Higher block efficiency means more tokens per call, reducing the calls needed to generate a sequence. Appendix B gives further experimental details.

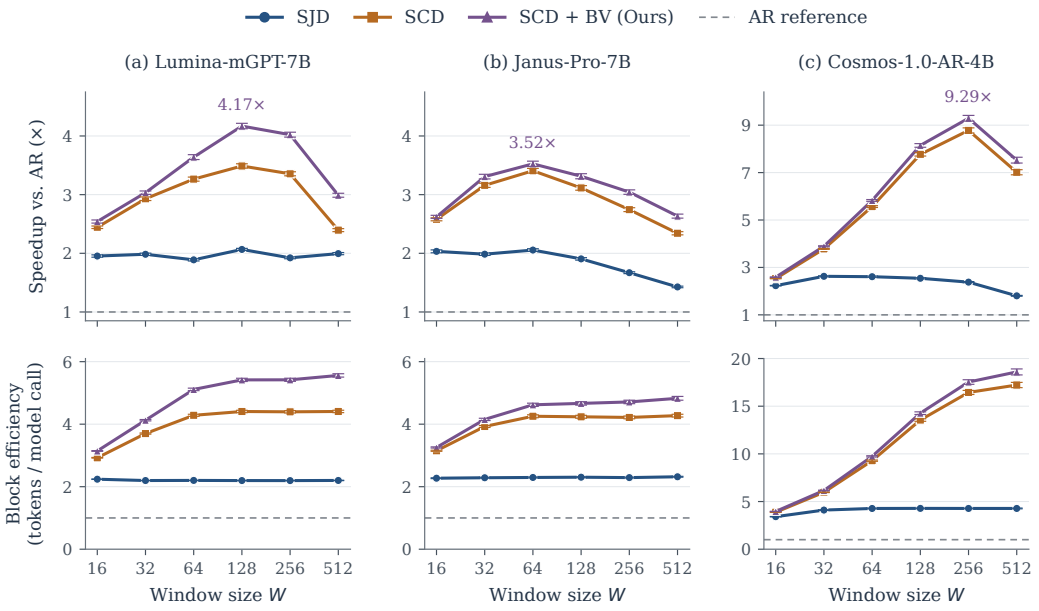

Figure 2: Decoding efficiency across window sizes for 1,000 inputs per image model and 500 inputs for Cosmos. Top: generation speedup relative to AR (ratio of mean generation times). Bottom: average tokens produced per target-model call (block efficiency), including correction and bonus tokens. Higher is better in both rows. Error bars show 95% bootstrap confidence intervals for each setting, obtained by resampling inputs. Dashed lines show the AR baseline: one token per target-model call and $1\times$ speedup. Vertical scales differ between the image models and Cosmos.

5.2 Results

Figure 2 shows that SCD + BV produces more tokens per target-model call and generates outputs faster than SCD in all 18 combinations of model and window size. At matching window sizes, it reduces the mean number of target-model calls by 2.1%–20.7% and mean generation time by 1.6%–19.9%. Comparing each method at its fastest tested window, SCD + BV reduces mean generation time relative to SCD by 16.3% on Lumina, 3.3% on Janus, and 5.5% on Cosmos.

The reduction in target-model calls relative to SCD grows with window size for all three models. However, both SCD and SCD + BV are fastest at $W = 128$ for Lumina, $W = 64$ for Janus, and $W = 256$ for Cosmos. Larger windows reduce the number of target-model calls, but each forward pass processes more tokens. Beyond a certain window size, generation speedup declines despite the further reduction in calls.

AI USE STATEMENT

We used generative AI tools for literature discovery, research ideation and execution, developing proofs, and drafting and editing the manuscript. The authors take responsibility for the paper's content.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

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.

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.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In *Forty-first International Conference on Machine Learning*, 2024.

Pierre E Jacob, John O'leary, and Yves F Atchadé. Unbiased markov chain monte carlo methods with couplings. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(3):543–600, 2020.

Jialiang Kang, Han Shu, Wenshuo Li, Yingjie Zhai, and Xinghao Chen. Sjd-pac: Accelerating speculative jacobi decoding via proactive drafting and adaptive continuation. *arXiv preprint arXiv:2603.18599*, 2026.

Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. Cllms: Consistency large language models. *arXiv preprint arXiv:2403.00835*, 2024.

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, 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.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12336–12355, 2023.

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*, 2025.

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.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36: 30222–30242, 2023.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae H Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. In *International Conference on Learning Representations*, volume 2025, pp. 24965–24994, 2025.

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.

Yepeng Weng, Qiao Hu, Xujie Chen, Li Liu, Dianwen Mei, Huishi Qiu, Jiang Tian, et al. Traversal verification for speculative tree decoding. *Advances in Neural Information Processing Systems*, 38:77899–77924, 2026.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

A Block Verification Details

For completeness, Algorithm 2 presents BlockVerify (Algorithm 2 of Sun et al. (2025)), adapted to this paper's notation.

Algorithm 2 BlockVerify in the notation of this paper

Require: Draft $X_{1:L}$; draft distributions $Q_{1:L}$; target distributions $P_{1:L+1}$

$\rho_0 \leftarrow 1$
for $j = 1, \ldots, L$ **do**
  $\rho_j \leftarrow \min\{1, \rho_{j-1} P_j(X_j)/Q_j(X_j)\}$
for $j = 1, \ldots, L-1$ **do**
  $A_{j+1} \leftarrow \sum_x [\rho_j P_{j+1}(x) - Q_{j+1}(x)]_+$
  $h_j^{\text{block}} \leftarrow A_{j+1}/(A_{j+1} + 1 - \rho_j)$, using 0 when the ratio is $0/0$
$h_L^{\text{block}} \leftarrow \rho_L$; $\tau \leftarrow 0$
for $j = 1, \ldots, L$ **do**
  Draw $\eta_j \sim \text{Unif}(0, 1)$ independently
  if $\eta_j \leq h_j^{\text{block}}$ **then**
    $\tau \leftarrow j$ ▷ continue to longer prefixes
if $\tau < L$ **then**
  Draw $Y \sim \text{Normalize}([\rho_\tau P_{\tau+1} - Q_{\tau+1}]_+)$
else
  Draw $Y \sim P_{L+1}$
return $(\tau, Y, \rho_{0:L})$

B Experimental Details

Our implementation builds on the SCD codebase (So et al., 2025), reusing most of its code with changes to verification and suffix updates.

Verification and MRS use the normalized distributions after applying the sampling settings, including temperature, top-k, top-p, and classifier-free guidance (CFG). For both image models, the conditional and unconditional CFG branches are batched together for every method, including AR. Lumina uses the `Alpha-VLLM/Lumina-mGPT-7B-768` checkpoint in BF16 with CFG scale 3, image top-$k = 2000$, and text top-$k = 10$. Janus uses `deepseek-ai/Janus-Pro-7B` in BF16 with CFG scale 5 and image top-$k = 1000$, generating 576 image tokens. Both image models use temperature 1 and seed 1.

Cosmos uses the `RealEstate10K_Mini/test_256` split of `kiwhansong/DFoT` and generates 24 continuation frames (7,680 tokens) at 640×1024 resolution. Cosmos runs in BF16 with temperature 1, top-$p = 0.8$, and no CFG.

C Modified Rejection Sampling

Algorithm 3 Modified Rejection Sampling: $\text{MRS}(P, Q, X)$

Require: Target distribution P; input distribution Q; token $X \sim Q$
Ensure: Token $Y \sim P$ and acceptance indicator K

Draw $U \sim \text{Unif}(0, 1)$
if $U < \min\{1, P(X)/Q(X)\}$ **then**
  return $(X, 1)$ ▷ Keep the input token
else
  Draw $Y \sim \text{Normalize}([P - Q]_+)$ ▷ Sample from the residual distribution
  return $(Y, 0)$