# STITCH-AND-REPAIR SPECULATIVE DECODING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Speculative decoding accelerates language model inference by having a small draft model propose a block of tokens that a large target model verifies in parallel. A rejection discards the rest of the draft. Preparing continuations in advance helps, but only when they anticipate the target's sampled correction. We introduce STITCH-AND-REPAIR, a training-free method that makes these misses both rarer and cheaper, recycling rejected computation instead of restarting from scratch. In addition to deterministic candidates, STITCH-AND-REPAIR prepares a stochastic wildcard root which, unlike a deterministic candidate, can be coupled to the target's correction. After a rejection, it maximally couples speculative decoding's residual correction distribution with the wildcard's distribution, so the correction lands on prepared work more often while both marginals stay unchanged. On a match, the prepared continuation becomes the next proposal; on a miss, STITCH-AND-REPAIR bridges the correction to a retained donor continuation through a single fresh draft token. All reused tokens are re-verified within the target's regular verification pass, exactly preserving the target distribution with no additional target forward passes. Across Llama-3.1-8B, DeepSeek-R1-Distill-Llama-8B, and Qwen3-8B targets on HumanEval, UltraFeedback, AlpacaEval, and GSM8K, STITCH-AND-REPAIR improves end-to-end throughput over speculative decoding with continuations prepared in advance by 4.7–8.4% on pooled workloads and by up to 12.2% on individual datasets. It is also 20.2–37.3% faster than a tuned EAGLE-3 on the two targets with official EAGLE-3 checkpoints. Against a control that prepares identical work but samples the correction independently, coupling alone improves throughput by 2.2% on average and by up to 5.3%. Rejected speculation need not be wasted: coupling routes corrections to prepared work, and bridging reconnects the rest to retained work.

## 1 INTRODUCTION

Speculative decoding (Stern et al., 2018; Leviathan et al., 2023; Chen et al., 2023) accelerates language model generation by having a small draft model propose a block of tokens that a large target model verifies in parallel, advancing several tokens per target call while preserving the target distribution. Its efficiency hinges on what happens at a rejection. When the target rejects a draft token, standard speculative decoding samples a correction from the residual distribution, discards the rest of the draft, and restarts drafting from scratch after the correction. Recent work hides part of this cost by preparing continuations in parallel with verification (Kumar et al., 2026): if the target's correction happens to be one of the prepared roots, drafting can resume immediately from prepared work. But the correction is random, and a fixed set of prepared roots pays off only when that sample happens to land on one of them. Whenever the correction falls elsewhere, the prepared work is discarded along with the rejected draft, and the decoder pays for fresh sequential draft steps on the critical path.

In this work, we introduce STITCH-AND-REPAIR[1], a training-free method that makes prepared-root misses both rarer and cheaper. Our key observation is that a fixed root is hit only by chance, but a randomly drawn root can be coordinated with the correction: by drawing the two with shared randomness, we can make them land on the same token whenever their distributions overlap. STITCH-AND-REPAIR therefore combines both kinds of roots: fixed roots at the draft's most likely alternatives cover the outcomes it already expects, and a *stochastic wildcard* root, drawn from the rest of the draft distribution, covers the long tail that no small fixed set can reach. After a rejection,

[1] Borrowed from *stitch-and-repair* shops, which mend a torn garment with stitches instead of replacing it.



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.

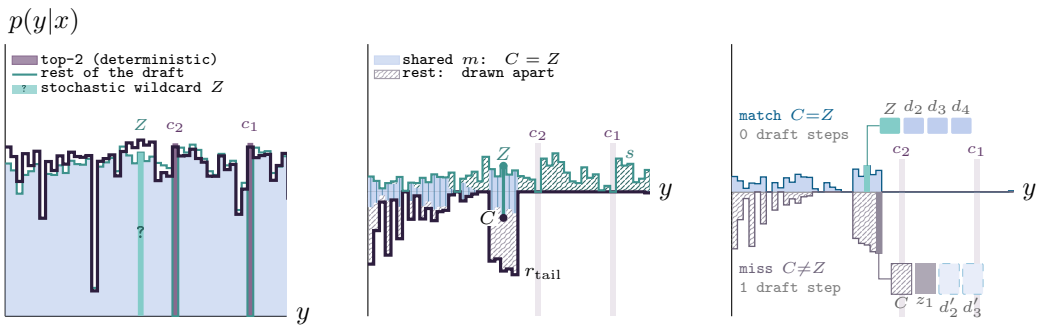


Figure 1: **Overview of STITCH-AND-REPAIR.** (a) *In parallel with target verification* of the current block, the draft prepares two fixed roots $c_1, c_2$ (its most likely alternatives to the drafted token) and one stochastic wildcard root $Z \sim s$, drawn from the rest of the draft distribution, each with a successor block. (b) After the first rejection, STITCH-AND-REPAIR maximally couples the residual $r_j$ with $s$. Because $s$ is zero on $c_1, c_2$, only the tail $r_{\text{tail}}$ of $r_j$ outside them can be shared. The marginals remain unchanged, but the shared mass $m(y) = \min\{r_j(y), s(y)\}$ makes $C = Z$ more likely. (c) A match $C = Z$ unlocks an entire prepared block with no new draft steps, while unmatched correction mass falls back to one fresh bridge token $z_1$ followed by a donor tail (dashed). Strip thickness shows each path's probability: a single-token routing gain can yield multi-token reuse at no extra draft cost.

STITCH-AND-REPAIR samples the correction from a maximal coupling with the wildcard, leaving the correction's distribution exactly unchanged (Algorithm 2). On a prepared-root match, the block already drafted after the matched root becomes the next proposal at no additional draft cost. On a miss, STITCH-AND-REPAIR does not restart from scratch: it drafts a single fresh bridge token after the true correction and completes the proposal with a donor tail prepared at the same position.

Reusing prepared work must not bias the output, and we guarantee this in two ways. First, the coupling changes only how the correction and the wildcard are drawn together, never the distribution of the correction itself. Second, reused tokens are never accepted outright: they are treated as ordinary speculative proposals and checked by the target in its regular verification pass, using the probabilities with which they were actually proposed (Algorithm 1). Hence, STITCH-AND-REPAIR samples exactly from the target distribution and adds no target forward passes; draft-side preparation overlaps with verification, and a miss drafts only a single bridge token on the critical path.

We evaluate STITCH-AND-REPAIR with Llama-3.1-8B, DeepSeek-R1-Distill-Llama-8B, and Qwen3-8B targets on HumanEval, UltraFeedback, AlpacaEval, and GSM8K. Against speculative decoding with continuations prepared in advance, STITCH-AND-REPAIR improves end-to-end throughput by 4.7–8.4% on pooled workloads and by up to 12.2% on individual datasets, and it is 20.2–37.3% faster than a tuned EAGLE-3 on the two targets with official checkpoints. A controlled ablation that keeps the prepared work identical and replaces the coupling with independent sampling isolates the coupling itself, which contributes 2.2% on average and up to 5.3%. We further analyze bridging and find that, although donor tokens are accepted less often than freshly drafted ones, they add accepted tokens at no extra draft cost. Our code will be made publicly available online.

To summarize, our contributions are as follows:

- **Coupled correction routing.** We introduce a stochastic wildcard root that can be coupled to the target's correction, and show that a maximal coupling makes the correction land on prepared work as often as possible without changing its distribution.
- **Bridging fallback.** When the correction still misses, STITCH-AND-REPAIR drafts a single bridge token and completes the proposal with a donor tail prepared at the same position, so a miss costs one fresh draft step instead of a full sequential redraft.
- **Exactness at no extra target cost.** Every reused token is reverified as an ordinary speculative proposal within the target's regular verification pass, so STITCH-AND-REPAIR samples exactly from the target distribution without any additional target forward passes.

- **Empirical gains.** Across three models and four datasets, STITCH-AND-REPAIR improves end-to-end throughput by 4.7–8.4% over speculative decoding with advance preparation and is 20.2–37.3% faster than a tuned EAGLE-3 on two targets; with identical prepared work, coupling alone contributes 2.2% on average and up to 5.3%.

## 2 RELATED WORK

**Speculative decoding and better drafts.** Speculative decoding builds on blockwise parallel decoding (Stern et al., 2018) and was introduced in its exact sampling form by Leviathan et al. (2023) and Chen et al. (2023); related parallel decoding methods guess several tokens and refine or verify them in parallel without a separate draft model (Santilli et al., 2023; Fu et al., 2024). Much follow-up work improves the proposal itself (Xia et al., 2024): through distilled or trained drafters (Zhou et al., 2024; Li et al., 2024a;b; 2025; Chen et al., 2026a), self-speculation with the target's own layers (Zhang et al., 2024; Elhoushi et al., 2024), additional decoding heads (Cai et al., 2024; Ankner et al., 2024), and tree-structured candidates (Miao et al., 2024; Chen et al., 2024; Svirschevski et al., 2024). These methods raise acceptance *before* verification. STITCH-AND-REPAIR is orthogonal to them: it keeps the drafter unchanged and alters only how the correction is sampled and what follows.

**Speculating in advance.** Drafting and verification need not alternate: PEARL (Liu et al., 2025) and AMUSD (McDanel, 2024) keep drafting while the target verifies, and speculative speculative decoding (Kumar et al., 2026) goes further, predicting likely verification outcomes and preparing a speculation for each of them; if the actual outcome is among the prepared ones, the next speculation is available immediately. We build on this setting and target its failure mode: when the correction falls outside the prepared set, the prepared work is wasted and decoding falls back to ordinary drafting. STITCH-AND-REPAIR makes these misses *rarer*, by coupling the correction to a stochastic wildcard root, and *cheaper*, by stitching it to retained work through a single fresh bridge token.

**Couplings and verification rules.** Speculative sampling is itself a coupling construction: for a single token, its acceptance rule realizes a maximal coupling between draft and target, so the output equals the draft token with probability equal to the overlap of the two distributions. SpecTr (Sun et al., 2023) generalizes this view with optimal transport to couple several draft candidates with the target, later work characterizes optimal multi-draft verification (Khisti et al., 2025; Hu et al., 2025), block verification accepts draft blocks jointly rather than token by token (Sun et al., 2025), and Daliri et al. (2025) use communication-free couplings to make the output independent of the drafter for a fixed random seed. In all of these, the coupling is between the draft's proposal and the target's sample, while the correction after a rejection is drawn independently. STITCH-AND-REPAIR applies the same maximal-coupling construction one step later, between that correction and a stochastic root prepared in advance, so the correction lands on work that already exists.

**Reusing rejected computation and copied continuations.** Several recent methods recover value from rejected speculation by reusing its *representations*. ReTrace (Lin et al., 2026) conditions the next draft block on the hidden representations of rejected suffixes, Carryover Drafting (Koo et al., 2026) feeds the target's hidden states at rejected positions back into the drafter as additional context, and Chen et al. (2026b) design a hidden-state drafter whose states remain reusable after verification failures. A second line copies *tokens*: model-free methods draft by copying continuations from the prompt, the generation history, or a datastore (He et al., 2024; Oliaro et al., 2025), Ouroboros (Zhao et al., 2024) lengthens drafts by reusing phrases, and Token Recycling (Luo et al., 2025) retrieves candidate tokens cached from earlier decoding steps. STITCH-AND-REPAIR leaves the drafter unchanged, unlike most representation-based methods, and unlike copying methods it reuses the model's own retained speculation after a correction, reconnecting to it through a fresh bridge token and verifying each recycled token as a fixed, point-mass proposal so that the output stays exact. It is complementary to both lines of work and could in principle be combined.

## 3 STITCH-AND-REPAIR: COUPLING AND BRIDGING

We sample a continuation from a target model $p$ given a prefix $\boldsymbol{x}$, using a smaller draft model $q$. STITCH-AND-REPAIR changes only how the correction is sampled and what is proposed after it. It keeps two kinds of speculative work available across a correction, namely continuations prepared in advance for likely corrections and continuations retained from earlier speculation. A rejection

breaks the speculation, and STITCH-AND-REPAIR *repairs* it by *stitching* the correction back onto this work, which gives the method its name. The stitch is made in one of two ways. *Coupling* makes the correction land on a prepared root as often as possible, so that the continuation prepared after it can be reused directly (Section 3.4); when the correction misses, *bridging* drafts a single fresh token that connects it to retained work (Section 3.6). All reused tokens are reverified by the target like any other proposal (Section 3.7). Algorithm 1 gives the full procedure, and Figure 1 illustrates each step.

**Notation.** We denote by bold letters token sequences and by plain letters single tokens; $\boldsymbol{y}_{<i}$ and $\boldsymbol{y}_{i:k}$ are subsequences, $\|$ is concatenation, and $[K] = \{1, \dots, K\}$. Lowercase letters such as $p$, $q$, $r_j$, and $s$ denote distributions over tokens, while $\Pr$ and $\mathbb{E}$ range over events and random variables.

### 3.1 BACKGROUND: VERIFICATION WITH DECLARED PROPOSALS

Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) proposes a block of tokens $\tilde{\boldsymbol{y}}_{1:K}$ and verifies them in parallel with a single target call. We write each proposed token together with the distribution it was generated from, its *declared proposal* $Q_i$. The target accepts $\tilde{y}_i$ with probability

$$a_i = \min\left(1, \frac{p_i(\tilde{y}_i)}{Q_i(\tilde{y}_i)}\right), \qquad p_i = p(\cdot \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_{<i}),$$

and at the first rejection $j$ it samples a correction from the residual

$$r_j(y) \propto \big[p_j(y) - Q_j(y)\big]_+;$$

if all $K$ tokens are accepted, it samples a bonus token from $p_{K+1}$. This procedure returns exact samples from $p$ for *any* declared proposal, provided each $\tilde{y}_i$ is in fact drawn from $Q_i$ independently of the randomness used to verify it. A freshly drafted token declares the draft conditional $Q_i = q(\cdot \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_{<i})$. A token copied deterministically from earlier work declares the point mass $Q_i = \mathbf{1}[\cdot = d]$: it is accepted with probability $p_i(d)$, and on rejection the residual is $p_i$ restricted to $y \neq d$. STITCH-AND-REPAIR reuses both kinds of token without any change to the verification rule: prepared tokens are drafted fresh after their root, and tokens copied from retained work declare point masses.

### 3.2 PREPARING ROOTS

Following Kumar et al. (2026), the draft prepares continuations for likely corrections while the target verifies the current block, so this work is off the critical path. It does so at every position $j \in [K]$ that could become the first rejection. Let $q_j = q(\cdot \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_{<j})$ be the draft distribution at $j$. Since a rejected token can never be the correction ($r_j(\tilde{y}_j) = 0$), STITCH-AND-REPAIR prepares three roots among the alternatives to $\tilde{y}_j$, two fixed and one stochastic:

- two *fixed* roots $c_1, c_2$, the two most likely alternatives to $\tilde{y}_j$ under $q_j$;
- one *stochastic wildcard* root $Z \sim s$, drawn from the rest of the draft distribution:

$$s(y) \propto q_j(y)\, \mathbf{1}\big[y \notin \{\tilde{y}_j, c_1, c_2\}\big].$$

For each root $\rho$, the draft generates a successor block $D^{(j,\rho)} = d_{1:K}$ of $K$ tokens. These blocks form a bank $\mathcal{B}$ of retained work: a block is reused when the correction equals its root (Section 3.5), and otherwise supplies donor tails for bridging (Section 3.6). The wildcard uses only draft-side randomness, so it is independent of the target's verification randomness, as Proposition 1 requires.

### 3.3 MATCH OR MISS

After a rejection at position $j$, the correction $C \sim r_j$ either lands on one of the prepared roots $c_1, c_2, Z$ (constituting a *match*, denoted by $H$) or it does not (constituting a *miss*, denoted by $H^c$). Two quantities describe the proposal that follows: its acceptance length $V$, the number of tokens the target accepts, and its drafting cost $T$, the number of fresh draft steps needed before it is ready. Then

$$\mathbb{E}[V] = \Pr(H)\,\mathbb{E}[V \mid H] + \big(1 - \Pr(H)\big)\,\mathbb{E}[V \mid H^c], \qquad \mathbb{E}[T] = 1 - \Pr(H), \tag{1}$$

since a match needs no fresh draft steps and a miss needs one. Coupling (Section 3.4) raises $\Pr(H)$ without changing the distribution of $C$. Because a match makes an entire successor block available at once, raising the probability of this single-token event both lowers $\mathbb{E}[T]$ and shifts weight toward the branch with the longer acceptance length. Bridging (Section 3.6) keeps the cost of a miss at one step: instead of redrafting all $K$ tokens, it drafts a single bridge token and then reuses retained work.

**Algorithm 1** Stitch-and-Repair speculative decoding

**Require:** Prefix $\boldsymbol{x}$, target model $p$, draft model $q$, block length $K$

$\mathcal{P} \leftarrow \varnothing$ ▷ pending reused proposal
**while** not finished **do**
  **if** $\mathcal{P} = \varnothing$ **then**
    Draft $\tilde{\boldsymbol{y}}_{1:K}$, $\tilde{y}_i \sim q(\cdot \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_{<i})$; set $Q_i \leftarrow q(\cdot \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_{<i})$
  **else**
    $(\tilde{\boldsymbol{y}}_{1:K}, Q_{1:K}) \leftarrow \mathcal{P}$; $\mathcal{P} \leftarrow \varnothing$ ▷ reused tokens are ordinary proposals
  **end if**
  **in parallel:** $\mathcal{B} \leftarrow \textsc{Prepare}(\boldsymbol{x}, \tilde{\boldsymbol{y}})$ ▷ roots $c_1, c_2, Z$ per position, plus all-accepted branches
  Verify $\tilde{\boldsymbol{y}}$: accept $\tilde{y}_i$ w.p. $\min\{1, p_i(\tilde{y}_i)/Q_i(\tilde{y}_i)\}$; first rejection at $j$ ($K{+}1$ if none)
  Append $\tilde{\boldsymbol{y}}_{<j}$ to $\boldsymbol{x}$
  **if** $j \le K$ **then**
    $r_j \leftarrow [p_j - Q_j]_+ \,/\, \|[p_j - Q_j]_+\|_1$; $C \leftarrow \textsc{CoupledCorrection}(r_j, \mathcal{B})$ ▷ Alg. 2
  **else**
    Sample a bonus token $C \sim p_{K+1}$ ▷ uncoupled
  **end if**
  Append $C$ to $\boldsymbol{x}$
  **if** $\mathcal{B}$ holds a block $D^{(j,C)} = d_{1:K}$ **then** ▷ prepared-root match: 0 draft steps
    $\mathcal{P} \leftarrow (d_{1:K}$, their draft laws)
  **else if** $j \le K$ **then** ▷ miss: bridge, 1 draft step
    Draft $z_1 \sim q(\cdot \mid \boldsymbol{x})$; take a donor $d'_{1:K}$ at position $j$, with $d'_1 = z_1$ if possible
    $\mathcal{P} \leftarrow \big((z_1, d'_{2:K}),\ (q(\cdot \mid \boldsymbol{x}), \mathbf{1}[\cdot = d'_2], \ldots, \mathbf{1}[\cdot = d'_K])\big)$ ▷ point masses
  **end if**
**end while**

### 3.4 Coupled correction

A fixed root is matched only when the correction happens to fall on it, so the two fixed roots are matched with probability $r_j(c_1) + r_j(c_2)$. The wildcard is different: because it is random, its joint distribution with the correction can be chosen freely. Instead of sampling $C \sim r_j$ independently of $Z$, Stitch-and-Repair samples $C$ from a *maximal coupling* of $r_j$ and $s$ (Algorithm 2). Let $m(y) = \min\{r_j(y), s(y)\}$. Given the prepared $Z$, the correction is set to $C = Z$ with probability $m(Z)/s(Z)$; otherwise $C$ is drawn from the leftover distribution $(r_j - m)/(1 - \sum_y m(y))$.

**Proposition 1** (Exact maximal coupling)**.** *If $Z \sim s$ is drawn independently of the verification randomness, the correction $C$ returned by Algorithm 2 satisfies $C \sim r_j$ and*

$$\Pr(C = Z) = \sum_y \min\{r_j(y), s(y)\},$$

*which is the largest value of $\Pr(C = Z)$ over all couplings of $r_j$ and $s$.*

*Proof sketch.* For every $y$,

$$\Pr(C = y) = s(y)\,\frac{m(y)}{s(y)} + \Big(1 - \sum_{y'} m(y')\Big)\,\frac{r_j(y) - m(y)}{1 - \sum_{y'} m(y')} = r_j(y).$$

The leftover distributions $r_j - m$ and $s - m$ have disjoint supports, so $C = Z$ only through the first branch, which gives $\Pr(C = Z) = \sum_y m(y)$. For any coupling, $\Pr(C = Z = y) \le \min\{r_j(y), s(y)\}$, and summing over $y$ gives the upper bound. □

Under independent sampling, the wildcard would be matched with probability only $\sum_y r_j(y)\, s(y)$, which never exceeds $\sum_y m(y)$. The fixed roots need no special treatment. Since $s$ is zero on $\tilde{y}_j, c_1, c_2$, their residual mass stays in the leftover branch, and a correction that lands on $c_1$ or $c_2$ there is an ordinary prepared-root match. The overall match probability is therefore

$$\Pr(H) = r_j(c_1) + r_j(c_2) + \sum_{y \notin \{\tilde{y}_j, c_1, c_2\}} \min\{r_j(y), s(y)\}.$$

**Algorithm 2** Coupled correction sampling

**Require:** Residual $r$; bank $\mathcal{B}$ with wildcard $Z \sim s$, drawn during preparation
1: **if** $\mathcal{B}$ has no wildcard for this position **then**
2: **return** a sample $C \sim r$
3: **end if**
4: $m(y) \leftarrow \min\{r(y), s(y)\}$ for all $y$ ▷ $s = 0$ on $\tilde{y}_j, c_1, c_2$
5: Draw $u \sim \mathrm{Unif}[0, 1]$
6: **if** $u \leq m(Z)/s(Z)$ **then**
7: **return** $C \leftarrow Z$ ▷ prepared-root match
8: **end if**
9: **return** a sample $C \sim (r - m) \,/\, \big(1 - \sum_y m(y)\big)$

### 3.5 Reuse on a match

If the correction $C$ equals a prepared root $\rho$, Stitch-and-Repair retrieves $D^{(j,\rho)}$ and proposes the entire block $d_{1:K}$. These tokens were sampled from the draft conditioned on the correct prefix $\boldsymbol{x}\|\tilde{\boldsymbol{y}}_{<j}\|C$, so their declared proposals are the draft conditionals they were sampled from, exactly as for a fresh draft. They are verified in the next regular target call, and no fresh draft steps are needed.

### 3.6 Bridging on a miss

If $C$ matches no prepared root, Stitch-and-Repair does not restart drafting from scratch. It drafts a single *bridge* token $z_1 \sim q(\cdot \mid \boldsymbol{x}, \tilde{\boldsymbol{y}}_{<j}, C)$ under the corrected prefix and completes the proposal with a donor tail taken from a branch $D' = d'_{1:K}$ prepared at the same position. We call $D'$ *aligned* with the bridge when its first successor matches it, $d'_1 = z_1$. Stitch-and-Repair prefers an aligned donor and otherwise falls back to an eligible branch at that position. The proposal is $(z_1, d'_{2:K})$. Only the aligned token is shared: the donor was drafted after a different root, so its tail comes from a stale context. This affects how often donor tokens are accepted, not exactness. The bridge is an ordinary fresh draft token. The donor tokens are copied, so each is declared as a point mass $\mathbf{1}[\cdot = d'_i]$ and accepted with probability $p_i(d'_i)$ (Section 3.1). Donor tokens are typically accepted less often than freshly drafted ones, but they require no additional draft steps.

### 3.7 Exactness and cost

**Theorem 1** (Exactness)**.** *Algorithm 1 generates sequences distributed according to the target $p$.*

*Proof sketch.* Let $h$ be the history up to a rejection at position $j$. By Proposition 1, $\Pr(C = c \mid h) = \sum_z s(z)\ \Pr(C = c \mid h, Z = z) = r_j(c)$. Now condition on $h$, the wildcard $Z$, and $C = c$. Every token of the next proposal is drawn from its declared proposal independently of the target's verification randomness: a reused successor block was drafted after its root with draft-side randomness only, so it is independent of $C$ given $(h, Z)$; the bridge is a fresh draft token; and donor tokens are point masses determined by the bank. The standard argument then gives $\Pr(Y \mid h, C = c, Z) = p(Y \mid h, c)$ for the continuation $Y$ that follows. Since the right-hand side does not depend on $Z$,

$$\Pr(C = c, Y \mid h) = r_j(c)\, p(Y \mid h, c),$$

which is exactly the joint law of standard speculative decoding. Applying this argument at each successive rejection shows by induction that the whole output is distributed according to $p$. □

**Cost.** All reused tokens are checked inside the target's regular verification pass, so Stitch-and-Repair adds no target forward passes. On the draft side, preparing roots and successor blocks uses additional draft compute, but it runs in parallel with verification. The only drafting left on the critical path after a rejection is a single bridge step on a miss, in place of a full $K$-token redraft.

Table 1: End-to-end throughput (tokens/s, higher is better) of standard speculative decoding (SD), SSD (Kumar et al., 2026), and STITCH-AND-REPAIR. The two Llama-based targets use Llama-3.2-1B as the draft, and Qwen3-8B uses Qwen3-0.6B. Gain is relative to the stronger baseline on each workload. Pooled rows aggregate an equal workload drawn from all four datasets.

| | Llama-3.1-8B | | | | R1-Distill-Llama-8B | | | | Qwen3-8B | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | SD | SSD | Ours | Gain | SD | SSD | Ours | Gain | SD | SSD | Ours | Gain |
| HumanEval | 440.2 | 504.1 | 544.1 | +7.9% | 391.5 | 414.8 | 434.7 | +4.8% | 402.6 | 654.4 | 683.9 | +4.5% |
| UltraFeedback | 420.7 | 428.4 | 480.8 | +12.2% | 367.4 | 366.2 | 385.0 | +4.8% | 381.7 | 487.6 | 525.0 | +7.7% |
| AlpacaEval | 433.6 | 417.3 | 441.0 | +1.7% | 375.5 | 373.1 | 393.6 | +4.8% | 384.6 | 509.5 | 549.8 | +7.9% |
| GSM8K | 449.7 | 471.2 | 507.6 | +7.7% | 406.8 | 448.9 | 462.3 | +3.0% | 411.2 | 659.5 | 717.3 | +8.8% |
| Pooled | 435.8 | 452.6 | 490.5 | **+8.4%** | 384.7 | 398.1 | 416.6 | **+4.7%** | 394.6 | 566.7 | 607.9 | **+7.3%** |

# 4 EXPERIMENTS

We evaluate whether STITCH-AND-REPAIR improves end-to-end throughput over strong speculative decoding baselines (Sections 4.2 and 4.3), how much of the gain comes from the coupling itself (Section 4.4), and what bridging recovers when the correction misses prepared work (Section 4.5).

## 4.1 SETUP

**Models and data.** We use three target-draft pairs: Llama-3.1-8B-Instruct and DeepSeek-R1-Distill-Llama-8B, both drafted by Llama-3.2-1B, and Qwen3-8B drafted by Qwen3-0.6B; we refer to the targets as Llama, DeepSeek, and Qwen. We evaluate on four workloads spanning code, instruction following, and reasoning: HumanEval, UltraFeedback, AlpacaEval, and GSM8K. For each workload, we use 160 prompts split into two disjoint cohorts of 80, and generate at most 256 new tokens per prompt. All models sample at temperature 1.0.

**Metric and details** We report end-to-end throughput in generated tokens per second over a whole run. Timing covers prefill, decoding, speculative preparation, inter-GPU communication, request replacement, and pipeline drain, and excludes model loading and CUDA graph capture. Pooled numbers weight the four datasets equally: they are the harmonic mean of the per-dataset throughputs, which is the throughput of a run that generates the same number of tokens from each dataset. Each entry combines the two 80-prompt cohorts, run with across two random seeds. All experiments run on two NVIDIA RTX A6000 48 GB GPUs, with the target and draft on separate GPUs, following the setup by Kumar et al. (2026). All experiments used a batch size of eight for maximum occupancy.

**Baselines.** We compare against standard speculative decoding (SD) (Leviathan et al., 2023; Chen et al., 2023) and speculative speculative decoding (SSD) (Kumar et al., 2026), which prepares speculations in parallel with verification and is the setting STITCH-AND-REPAIR builds on. We also compare against EAGLE-3 (Li et al., 2025), a strong trained drafter, which we tune separately (Section 4.3). We do not compare against Make Every Draft Count (Chen et al., 2026b), whose code is not publicly available, or Token Recycling (Luo et al., 2025), which is designed for greedy decoding rather than the exact stochastic sampling we study. Apart from EAGLE-3, which uses its own drafter, all methods share the same target model, draft model, and block length.

**Independent-sampling control.** To isolate the coupling, we also run a control that uses the same prepared roots, successor blocks, and bridging as STITCH-AND-REPAIR, but samples the correction from $r_j$ independently of the wildcard, as in standard speculative decoding. Both variants are exact and prepare identical work, so they differ only in how often the correction lands on it: the wildcard is matched with probability $\sum_y r_j(y)\, s(y)$ under the control and $\sum_y \min\{r_j(y), s(y)\}$ under coupling (Section 3.4). Any difference in throughput between them is therefore due to coupling (Section 4.4).

## 4.2 MAIN RESULTS

Table 1 reports throughput against SD and SSD. STITCH-AND-REPAIR is the fastest method on every model and dataset. Over the stronger of the two baselines, its pooled gains are 8.4% on Llama, 4.7% on DeepSeek, and 7.3% on Qwen, reaching 12.2% on a single dataset (UltraFeedback

Table 2: End-to-end throughput (tokens/s, higher is better) of EAGLE-3 and STITCH-AND-REPAIR under matched settings (batch size 8, temperature 1, raw prompts). EAGLE-3 uses the official checkpoint for each target with its speculation structure tuned on held-out data. Gain is the speedup of STITCH-AND-REPAIR over EAGLE-3. SD and SSD results are in Table 1.

| | Llama-3.1-8B | | | DeepSeek-R1-Distill-Llama-8B | | |
|---|---|---|---|---|---|---|
| Dataset | EAGLE-3 | Ours | Gain | EAGLE-3 | Ours | Gain |
| HumanEval | 381.8 | **544.1** | +42.5% | 351.7 | **434.7** | +23.6% |
| UltraFeedback | 342.8 | **480.8** | +40.3% | 334.4 | **385.0** | +15.1% |
| AlpacaEval | 350.7 | **441.0** | +25.8% | 335.9 | **393.6** | +17.2% |
| GSM8K | 355.9 | **507.6** | +42.6% | 366.1 | **462.3** | +26.3% |
| Pooled | 357.2 | **490.5** | **+37.3%** | 346.6 | **416.6** | **+20.2%** |

on Llama). The gains also hold where preparing speculation in advance does not help on its own: on AlpacaEval with Llama, and on UltraFeedback and AlpacaEval with DeepSeek, standard SD is faster than SSD, yet STITCH-AND-REPAIR is faster than both. Over standard SD, the pooled gains are 12.6%, 8.3%, and 54.1%; on Qwen, where SSD is much faster than SD, the two gains compound.

### 4.3 COMPARISON WITH EAGLE-3

Table 2 compares STITCH-AND-REPAIR with EAGLE-3 on Llama-3.1-8B and DeepSeek-R1-Distill-Llama-8B. We omit Qwen3-8B because the EAGLE-3 authors release no checkpoint for it; third-party heads exist, but they are trained with different data and recipes. For both targets, we use the official checkpoint and tune its speculation structure on held-out data. Even with this tuning, STITCH-AND-REPAIR is faster than EAGLE-3 on every dataset for both targets: by 37.3% on the pooled workload (25.8–42.6% per dataset) for Llama, and by 20.2% (15.1–26.3%) for DeepSeek. For Llama, EAGLE-3 is also slower than standard SD with a 1B draft model. Both comparisons use raw prompts rather than the chat formatting of the EAGLE-3 evaluation, so they measure the methods under identical inputs rather than reproducing the published setting. The two approaches are complementary: EAGLE-3 changes the drafter, whereas STITCH-AND-REPAIR leaves the drafter unchanged and alters only how the correction is sampled and what is proposed after it.

### 4.4 ABLATION: THE EFFECT OF COUPLING

Table 3: Throughput gain from coupling: STITCH-AND-REPAIR versus the independent-sampling control.

| Dataset | Llama | DeepSeek | Qwen |
|---|---|---|---|
| HumanEval | +5.3% | +1.8% | +5.3% |
| UltraFeedback | +2.7% | +0.2% | +2.6% |
| AlpacaEval | +3.3% | +0.2% | +2.2% |
| GSM8K | +1.8% | +1.4% | +0.3% |

Table 3 isolates coupling by comparing STITCH-AND-REPAIR with the control, which prepares identical work and differs only in how the correction is sampled. The gain from coupling alone is positive in all twelve settings, averaging 2.2% and reaching 5.3% (HumanEval on Llama and Qwen). Because the two variants share all prepared work and bridging, this gain comes from routing more corrections onto prepared roots; the rest of the gain of STITCH-AND-REPAIR over SSD comes from what the two variants share. The effect is smallest on DeepSeek (0.2–1.8%), likely because of the drafter: distillation from DeepSeek-R1 shifts the target's distribution away from that of the Llama draft, so the wildcard distribution, which is built from the draft, overlaps less with residual, therefore leaving coupling with less gain to exploit (Proposition 1).

### 4.5 WHAT BRIDGING RECOVERS

We next ask what bridging recovers when the correction misses. On Llama, we collect 6,182 misses from HumanEval (2,333) and UltraFeedback (3,849), using 128 prompts per dataset, seeds 96–98, and 64 generated tokens per prompt. For each miss, we compare the bridged proposal, a fresh bridge token followed by a donor tail, with redrafting the rest of the proposal from scratch after the bridge. Figure 2 summarizes the comparison, which we break down below.

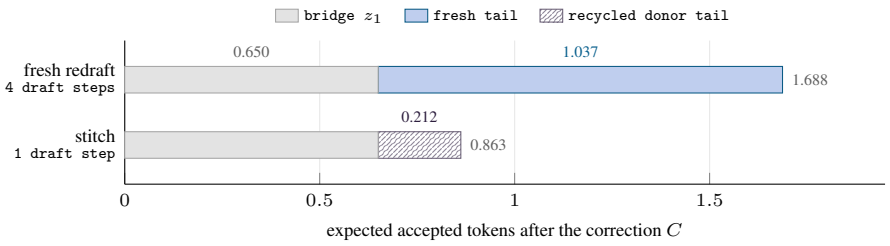


Figure 2: Expected accepted tokens per miss for the bridged proposal and for fresh redrafting, over 6,182 misses on Llama (HumanEval and UltraFeedback). Both variants draft the bridge token $z_1$ freshly, and 65.0% of misses accept it. After the bridge, the donor tail adds 0.212 accepted tokens, 20.5% of the 1.037 added by a fresh tail, but it needs no further draft steps, whereas fresh redrafting needs three more. The difference, 0.825 tokens per miss, has a 95% CI of $[0.808, 0.843]$.

**Donor tails are cheap but weaker.** Both variants draft the bridge token freshly, so they agree on the first position, where 65.0% of misses accept it. After that they diverge: a fresh tail adds 1.037 accepted tokens, whereas the donor tail adds 0.212, or 20.5% of the fresh value. Fresh redrafting therefore yields 0.825 more accepted tokens per miss (95% CI $[0.808, 0.843]$), but it needs $K = 4$ sequential draft steps instead of one. Relative to drafting only the bridge, the donor tail adds 33% more accepted tokens at no additional draft cost.

**The first donor token is the bottleneck.** Dividing consecutive survival probabilities, the probability that the first $i$ tokens are all accepted, gives per-position acceptance. Fresh tokens are accepted at a steady 70–75%. Donor tokens are accepted at only about 20% at the first recycled position but at about 45% at the positions after it: once the first donor token is accepted, the rest of the tail tends to follow. The first donor token is most exposed to the stale context, since the donor was drafted after a different root; once it is accepted, the donor and the actual prefix share more recent context. Donor tokens also face a ceiling that fresh tokens do not: a point mass at $d'_i$ is accepted with probability $p_i(d'_i) \leq \max_y p_i(y)$. Better donor selection should therefore target the first donor token.

**Donor availability.** In 88.4% of misses an aligned donor is available, and they retain 23.3% of the fresh tail value. The remaining 11.6% fall back to another branch at the same position, whose tail retains only 0.1%. Raising donor coverage is a concrete, training-free target for future work.

## 5 Conclusion

We introduced Stitch-and-Repair, a training-free method that reuses speculative work across a rejection. Because the correction is random, a stochastic wildcard root can be maximally coupled to it, so the correction lands on prepared work more often without changing its distribution; when it still misses, a single fresh draft token bridges it to retained work. All reused tokens are verified as ordinary proposals, so the method is exact and adds no target forward passes. Across three targets and four datasets, Stitch-and-Repair improves pooled throughput by 4.7–8.4% over the stronger of SD and SSD, and it is 20.2–37.3% faster than a tuned EAGLE-3 on the two targets with official checkpoints; coupling alone contributes a gain that is positive in all twelve settings.

## 6 Limitations and future work

Two quantities bound the gains. The wildcard comes from the draft's remaining mass, while the residual concentrates where the draft under-predicts the target; this caps the match rate and likely explains the small coupling gain on DeepSeek, and wildcards that better anticipate the residual could raise it. Donor tokens are accepted far less often than fresh drafts, especially at the first position, and 11.6% of misses have no aligned donor, which points to better donor selection and coverage. Stitch-and-Repair also needs spare draft capacity for preparation, and our evaluation covers only 8B-scale targets at batch size eight on one hardware configuration.

### AI USE STATEMENT

In this work, we used generative AI tools to implement methods: LLM-based coding assistants wrote parts of the implementation of STITCH-AND-REPAIR and of the experimental pipeline. We also used them to help formulate mathematical claims, to assist in writing the proof sketches, to provide feedback on the methodology and experiments, and to help interpret results. The idea is our own, but we did use generative AI tools to brainstorm next steps; we have not used them to generate synthetic data, and translation, dataset cleaning, and qualitative data analysis are not applicable to this work. Additionally, we used generative AI tools to identify related work, draft parts of the paper, edit it for readability, and format tables. We have reviewed all AI-assisted work. The implementation was developed against an extensive test suite, and all code was tested and verified by the authors. Every section drafted with AI assistance was revised and edited by the authors, and every AI-suggested claim, proof step, and interpretation was checked by the authors before inclusion. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES

Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. In *First Conference on Language Modeling*, 2024.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*, 2024.

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.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=Oz335dV48X.

Yuetao Chen, Xuliang Wang, Xinzhou Zheng, Ming Li, Peng Wang, and Hong Xu. Make every draft count: Hidden state based speculative decoding, 2026b. URL https://arxiv.org/abs/2602.21224.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yu-Hsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable and robust speculative decoding. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Majid Daliri, Christopher Musco, and Ananda Theertha Suresh. Coupling without communication and drafter-invariant speculative decoding. In *2025 IEEE International Symposium on Information Theory (ISIT)*, 2025.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24, 2024.

Zhenyu He, Zexuan Zhong, Tianle Cai, Jason Lee, and Di He. REST: Retrieval-based speculative decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, 2024.

Zhengmian Hu, Tong Zheng, Vignesh Viswanathan, Ziyi Chen, Ryan A. Rossi, Yihan Wu, Dinesh Manocha, and Heng Huang. Towards optimal multi-draft speculative decoding, 2025. URL https://arxiv.org/abs/2502.18779.

Ashish J Khisti, MohammadReza Ebrahimi, Hassan Dbouk, Arash Behboodi, Roland Memisevic, and Christos Louizos. Multi-draft speculative sampling: Canonical decomposition and theoretical limits. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=N1L5TgtkAw.

Jahyun Koo, Sunghyeon Woo, Jaeeun Kil, Jeongtae Lee, Sungjae Lee, Kyomin Jung, and Minsub Kim. Carryover drafting: Recycling rejected states for speculative decoding, 2026. URL https://arxiv.org/abs/2609.14717.

Tanishq Kumar, Tri Dao, and Avner May. Speculative speculative decoding. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=aL1Wnml9Ef.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *ICML*, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning*, 2024a.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Empirical Methods in Natural Language Processing*, 2024b.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=4exx1hUffq.

Luxi Lin, Zhanpeng Zeng, Shuang Peng, Songwei Liu, and Rongrong Ji. Retrace: Rejected-trajectory conditioning for speculative decoding, 2026. URL https://arxiv.org/abs/2608.29748.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, Winston Hu, and Xiao Sun. PEARL: Parallel speculative decoding with adaptive draft length. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xianzhen Luo, Yixuan Wang, Qingfu Zhu, Zhiming Zhang, Xuanyu Zhang, Qing Yang, and Dongliang Xu. Turning trash into treasure: Accelerating inference of large language models with token recycling. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Bradley McDanel. Amusd: Asynchronous multi-device speculative decoding for llm acceleration. *2025 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2024. URL https://api.semanticscholar.org/CorpusID:273532247.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, 2024.

Gabriele Oliaro, Zhihao Jia, Daniel F Campos, and Aurick Qiao. Suffixdecoding: Extreme speculative decoding for emerging AI applications. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. 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)*, 2023.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Ziteng Sun, Uri Mendlovic, Yaniv Leviathan, Asaf Aharoni, Jae Hun Ro, Ahmad Beirami, and Ananda Theertha Suresh. Block verification accelerates speculative decoding. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=frsg32u0rO.

Ruslan Svirschevski, Avner May, Zhuoming Chen, Beidi Chen, Zhihao Jia, and Max Ryabinin. Specexec: Massively parallel speculative decoding for interactive LLM inference on consumer devices. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=JAhNsZ9dvG.

Heming Xia, Zhe Yang, Qingxiu Dong, Peiyi Wang, Yongqi Li, Tao Ge, Tianyu Liu, Wenjie Li, and Zhifang Sui. Unlocking efficiency in large language model inference: A comprehensive survey of speculative decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, August 2024.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2024.

Weilin Zhao, Yuxiang Huang, Xu Han, Wang Xu, Chaojun Xiao, Xinrong Zhang, Yewei Fang, Kaihuo Zhang, Zhiyuan Liu, and Maosong Sun. Ouroboros: Generating longer drafts phrase by phrase for faster speculative decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rsY6J3ZaTF.