# RECOVERING OFF-POLICY SUPERVISION FOR SPECULATIVE DECODING

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

## ABSTRACT

Block drafters for speculative decoding are commonly trained on corpora written by external models, where a single off-policy token invalidates supervision for all subsequent slots in a block. Existing approaches discard these divergent slots, resulting in severe supervision loss. To resolve this problem while preserving the training corpus, we propose a rollout-based training framework that recovers full supervision through two complementary components. The first component, Anchor-Label Relabelling (ALR), replaces corpus labels with distributions from greedy target rollouts, restoring valid supervision across all predicted slots. The second component, In-Rollout Anchors (IRA), places draft blocks directly inside these rollouts to expose the drafter to target-generated context, reusing precomputed rollout features at no additional target cost. Across fixed vision-language and text corpora, our framework increases greedy accepted length by up to 36.5% over DFlash and consistently outperforms erasing baselines. Notably, a single epoch of our method surpasses the best multi-epoch erase schedules and matches the acceptance length of training on target-regenerated responses. These results show that our framework provides an effective and compute-efficient approach for training speculative drafters on fixed corpora without modifying the original text.

## 1 INTRODUCTION

In autoregressive language models, speculative decoding reduces generation latency while preserving the target distribution in exact arithmetic (Leviathan et al., 2023; Chen et al., 2023). A small drafter proposes several candidate tokens, and the primary model to be accelerated (the target) verifies them in a single forward pass. The resulting speedup depends on the accepted length, defined as the number of tokens committed in each verification round, and on the relative cost of drafting and verification. Recent work increases accepted length by improving agreement between the drafter and the target, and by training with verification (Cai et al., 2024; Li et al., 2024a; 2025; Zhou et al., 2023; An et al., 2026). The same framework now also serves vision-language models (Kang et al., 2025; Tong et al., 2026), whose instruction-tuning corpora typically consist of responses written by a stronger proprietary model (Liu et al., 2023a; Chen et al., 2024a;b). Such a corpus is off-policy for the target, and its responses often differ from the continuations the target itself would produce.

Block drafters are especially sensitive to this mismatch (An et al., 2025; Chen et al., 2026). Given an anchor token, a block drafter predicts all slots of a block in parallel, and each slot is trained on the corpus token at the same position. The label of a later slot therefore depends on the intermediate corpus tokens, which the drafter does not observe. If one of these tokens differs from the greedy choice of the target, every subsequent label in the block continues a prefix that the target would not generate at inference.

PARD-2 (An et al., 2026) addresses this problem by down-weighting such slots. Each slot is weighted by the probability that the target assigns to the preceding corpus tokens in the block, and this weight decays toward zero once the prefix departs from the target distribution. Supervision is thus retained where the corpus agrees with the target and erased where it diverges. Regenerating the corpus with the target avoids this loss of supervision (Kim & Rush, 2016; Cai et al., 2024; Li et al., 2025; An et al., 2025; Chen et al., 2026). However, regeneration requires a separate generation stage before training and replaces the original responses, which may need to be kept, for example when only an audited or licensed release can be used for training. In imitation learning, this mismatch is



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.

framed as covariate shift, and the standard remedy is to train on newly collected trajectories (Ross et al., 2010; Agarwal et al., 2023; Gu et al., 2023; Liu et al., 2023b). This approach likewise replaces the underlying training text.

In this paper, we show that a block drafter learns more from greedy continuations of the target than from erasing the off-policy slots of a fixed corpus. The first component of our method, Anchor-Label Relabelling (ALR), leaves the corpus unchanged and updates only the training labels. From each anchor, the target continues the corpus prefix greedily, and each slot is labeled with the target distribution conditioned on the prefix and preceding rollout tokens, erasing none of them. As in standard training, the drafter receives the anchor token and the target features of the corpus prefix. The rollout is computed within the training step by reusing the cached forward pass of the target, avoiding the need to generate or store full responses in advance. For ALR in isolation, a short rollout depth is sufficient.

ALR alone still leaves a residual gap to a drafter trained on responses regenerated by the target at matched lengths. A position-wise analysis shows no measurable gap over the first 16 generated tokens, with the deficit emerging only in later positions. From that point on, the recent context of the drafter at inference consists of text generated by the target, whereas a block anchored strictly in the corpus never observes such context during training. The second component of our method, In-Rollout Anchors (IRA), supplies this target-generated context. IRA keeps the total number of blocks fixed and relocates half of them into the rollouts of the other half. Each relocated block is anchored on a token generated by the target and conditioned on features already computed during the rollout, requiring no additional forward pass of the target. Thus, ALR corrects the labels, while IRA reduces the discrepancy between training and inference contexts.

We evaluate both components on two fixed vision-language corpora and one fixed text corpus. Under greedy decoding, ALR + IRA improves accepted length by up to 36.5% over DFlash trained on the same data and by up to 8.54% over the erasing baseline. It also achieves a higher accepted length than the erasing baseline on every evaluated domain and benchmark. These improvements hold for a smaller vision-language target and for a text-only target on seven benchmarks, under both greedy and sampled decoding. With the other training settings matched, a single epoch of ALR + IRA outperforms the best erase schedule we evaluated on either vision-language corpus, at less than twice the cost of one erase epoch. After three epochs of training, ALR + IRA achieves greedy decoding speedups of 2.82× on ALLaVA and 2.77× on ShareGPT4V over autoregressive decoding, compared with 2.62× for erase on both corpora. On ALLaVA, its accepted length is within 0.4% of drafters trained on responses regenerated by the target, even though the original corpus is kept fixed. The gain of IRA over ALR is largest at positions in held-out captions where an independent drafter fails to predict the first token. Ultimately, our findings show that target-generated context complements label relabelling, demonstrating that training-time target rollouts provide a practical alternative to response regeneration when the original corpus must be preserved.

## 2 Related Work on Draft Training

PARD-2 (An et al., 2026) aligns parallel draft training with consecutive token acceptance through confidence-adaptive loss weights. Each weight is the product of the target's probabilities for the preceding corpus tokens within the block. Our erase comparator applies this weighting to knowledge distillation on the fixed corpus. Its labels remain target distributions conditioned on corpus continuations, while unlikely corpus tokens reduce supervision for later slots. ALR obtains these distributions along a greedy target rollout from the same corpus prefix, and IRA also supplies rollout features as drafter context. This extends target alignment to the text that determines both the labels and the drafter's context, while preserving the original corpus.

The EAGLE family generates draft tokens autoregressively using target features (Li et al., 2024a;b). EAGLE-3 and HASS train on hidden states generated by the drafter to reduce the mismatch between training and multi-step drafting (Li et al., 2025; Zhang et al., 2024). Parallel drafters predict several future positions in one forward pass (Stern et al., 2018; Cai et al., 2024; Xiao et al., 2024; An et al., 2025), while semi-autoregressive methods combine sequential and parallel drafting (Gao et al., 2025). We use DFlash (Chen et al., 2026), a small block-diffusion drafter conditioned on target features. Our method keeps this architecture unchanged while obtaining supervision and context from the target's own continuations.

Knowledge distillation aligns a drafter's output distributions with those of the target (Hinton et al., 2015; Zhou et al., 2023). On-policy distillation uses student-generated continuations (Agarwal et al., 2023; Gu et al., 2023), while online speculative decoding adapts the drafter using feedback collected during serving (Liu et al., 2023b). Model-generated responses also support sequence-level and reasoning distillation (Kim & Rush, 2016; Lee et al., 2026), and several draft-training methods use target-generated responses (Cai et al., 2024; Goel et al., 2024; Zafrir et al., 2025; Hong et al., 2025). Regenerating the corpus provides target-written supervision and context, but replaces the original responses. When these responses must be preserved, computing target distributions along them still leaves supervision conditioned on continuations that may differ from the target's own. Our ALR + IRA addresses this mismatch with short target rollouts from corpus prefixes during training, supplying both supervision and context without rewriting the corpus.

For vision-language targets, prior work has developed both visual processing and training strategies. ViSpec (Kang et al., 2025) compresses image tokens with a vision adaptor, DREAM (Hu et al., 2025) fuses target features through cross-attention, and MSD (Lin et al., 2025) processes visual and text tokens differently and gradually introduces multimodal data after text-only training. ViSpec, DREAM, and MASSV (Ganesan et al., 2025) also train on target-generated responses. Our ALR + IRA changes training labels and anchor context through short target rollouts on a fixed corpus, without introducing image-specific components.

## 3 Relabelling Anchor Blocks with the Target's Own Rollout

For a response sequence $x$ from the training corpus and an index $a$, an *anchor block* spans $K+1$ slots beginning at position $a$. Slot 0 contains the anchor token $x_a$, and the drafter predicts slots 1 through $K$ in parallel conditioned on the target hidden representations up to position $a$. Under the standard corpus objective, slot $k$ receives the one-hot target $x_{a+k}$. Following DFlash (Chen et al., 2026), we set $K = 15$ with a five-layer architecture that receives target features from layers 1, 9, 17, 25, and 33, sampling 128 anchors per training sequence. All drafters are initialized from the publicly released text-only DFlash head corresponding to the target base model without architectural changes. Appendix A details these checkpoints and additional training hyperparameters. The compared objectives vary the target distribution and loss weight assigned to each slot, while IRA additionally relocates half of the draft blocks into rollout trajectories. Figure 1 illustrates label construction across erase, ALR, and ALR + IRA. We denote the target next-token distribution by $p_T$, omitting prompt and image conditioning tokens from the notation for simplicity.

### 3.1 Labels from the corpus prefix

Every objective beyond the corpus baseline replaces the one-hot target with a distribution $\pi_{a,k}$ and assigns a slot weight $\omega_{a,k}$,

$$\mathcal{L}(\theta) = \frac{\sum_{a\in\mathcal{A}}\sum_{k=1}^{K}\omega_{a,k}\,\ell\big(\pi_{a,k},\, q_\theta^k(\cdot \mid a)\big)}{\sum_{a\in\mathcal{A}}\sum_{k=1}^{K}\omega_{a,k}}, \qquad w_k = e^{-(k-1)/\gamma}, \tag{1}$$

where $\mathcal{A}$ denotes the set of anchors in a batch, $q_\theta^k(\cdot \mid a)$ is the drafter prediction for slot $k$ at anchor $a$, and $\ell$ evaluates soft cross-entropy against the target distribution restricted to the top 8 tokens with the residual probability mass grouped into a tail bin. Normalization by the total weight allows gating $\omega_{a,k}$ to redistribute supervision across slots without shrinking the overall loss scale. All distillation objectives incorporate the decay envelope $w_k$ into $\omega_{a,k}$, setting $\gamma = 2$ by default. For the fixed-corpus baseline, we apply knowledge distillation (KD) (Zhou et al., 2023) using $\pi_{a,k} = p_T(\cdot \mid x_{<a+k})$ and $\omega_{a,k} = w_k$. This formulation leaves labels conditioned on corpus continuations $x_{a+1}, \ldots, x_{a+k-1}$ that the target may not generate. During inference, a block is drafted over text generated by the target itself, and slot $k$ contributes to the accepted length only when the target accepts all preceding candidate tokens.

**Erasing off-policy slots.** Under the erase objective, we retain the distillation distribution and scale the slot weight by a survival gate,

$$g_{a,k} = \prod_{1\le j<k} p_T\big(x_{a+j} \mid x_{<a+j}\big), \qquad \omega_{a,k} = w_k\, g_{a,k}, \tag{2}$$

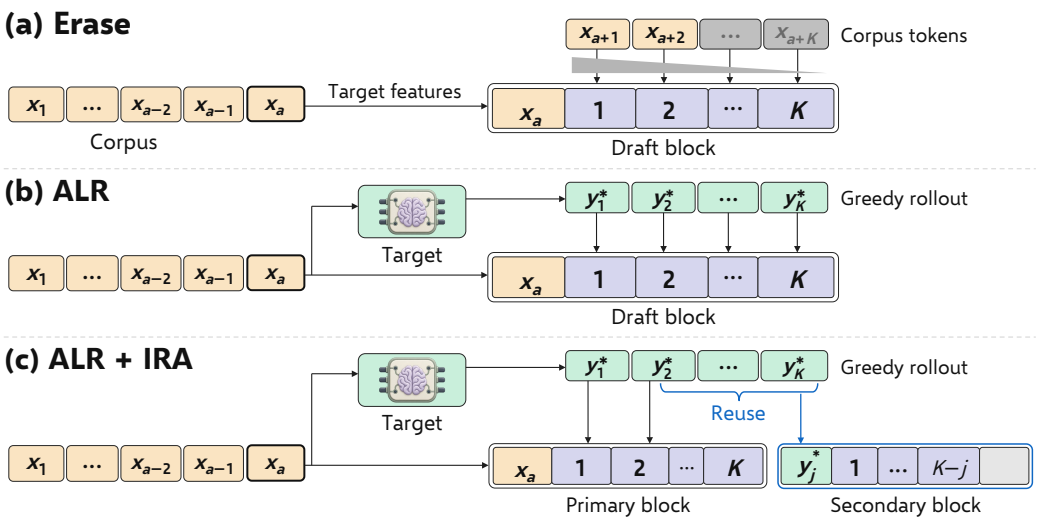


Figure 1: Label construction from a fixed corpus. Amber and green denote corpus and target-rollout tokens, respectively. Purple cells are draft slots, and black downward arrows supply the target distributions predicting the tokens above them. Erase down-weights corpus-conditioned labels, whereas ALR uses rollout-conditioned labels. IRA retains the corpus context and reuses rollout features and labels in a secondary block without another target pass. Its gray tail is unsupervised.

adapting the confidence-adaptive survival weighting of PARD-2 (An et al., 2026) to distillation. The gate equals 1 at slot 1, decreases monotonically along the block, and down-weights slots whose preceding corpus tokens have low probability under the target. A hard variant replaces each factor with the indicator $\mathbf{1}[x_{a+j} = \hat{y}_{a+j}]$, where $\hat{y}_{a+j}$ denotes the target greedy prediction after $x_{<a+j}$, discarding all subsequent slots upon the first mismatch. We adopt the soft survival formulation as the primary erase comparator throughout this work. In both variants, supervision remains conditioned on the corpus continuation, with only the loss weights changed.

### 3.2 Labels from the target's rollout

Anchor-label relabelling (ALR) retains the corpus context while substituting the corpus continuation with greedy rollouts generated by the target. Starting from each anchor, the target extends $x_{\le a}$ greedily, providing slot $k$ with the target distribution conditioned on the preceding $k-1$ rollout tokens,

$$y_i^* = \arg\max_v p_T\big(v \mid x_{\le a}, y^*_{<i}\big), \qquad \pi_{a,k} = p_T\big(\cdot \mid x_{\le a}, y^*_{<k}\big), \tag{3}$$

where the rollout depth $R$ counts the greedy tokens fed back to the target, yielding up to $R+1$ slot distributions. We set $R = K-1$ to provide supervision across all $K$ slots whenever the target does not terminate early. In-rollout anchors introduced below require this full depth and are never evaluated with shorter rollouts. Consequently, experimental configurations with $R < K-1$ evaluate ALR in isolation, leaving slots beyond $R+1$ unconstrained. Slots following early sequence termination are masked, while remaining slots retain $\omega_{a,k} = w_k$, ensuring that full-depth ALR preserves supervision across every active position.

On a greedy target rollout, the hard variant of the gate in Eq. (2) equals 1 across all active slots. The soft gate can still decrease because greedy tokens need not have probability one, so ALR omits it. Slot 1 receives identical supervision under KD, erase, and ALR, as $y^*_{<1}$ is empty. For subsequent slots, each label aligns with the greedy target trajectory verified during greedy decoding. Since slot $k$ contributes to the accepted length only when slots 1 through $k-1$ match $y^*_1, \ldots, y^*_{k-1}$, the ALR distribution matches the verification distribution whenever slot $k$ contributes to the accepted length. Drafter inputs, consisting of the anchor token and target representations up to $a$, remain unchanged from the standard corpus objective.

Table 4b compares ALR against a drafter trained with the same KD loss on target-regenerated responses truncated to the corpus lengths. On COCO captions, ALR matches this regeneration refer-

ence over the initial 16 generated tokens before exhibiting a performance deficit that subsequently plateaus. Every anchor in a regenerated response follows context generated by the target in preceding tokens. In contrast, an ALR block encounters target-generated tokens only in its supervision labels, as drafter features condition strictly on the corpus prefix up to $a$. We attribute this gap to the absence of target-generated context, motivating a secondary anchor block described next.

**In-rollout anchors.** In-rollout anchors (IRA) preserve ALR's anchor draw and total block count $m = \min(v, 128)$, where $v$ is the number of valid anchor positions in the sample, while partitioning the blocks into two subsets. The $\lceil m/2 \rceil$ primary blocks follow the standard ALR formulation in Eq. (3). Each of the $\lfloor m/2 \rfloor$ secondary blocks is positioned within a primary rollout at an offset $j \sim \mathcal{U}\{2, \ldots, K-1\}$. Its conditioning context concatenates the target representations of the corpus prefix up to $a$ with the intermediate representations stored for $y^*_1, \ldots, y^*_{j-1}$. Slot 0 is assigned the target-generated token $y^*_j$, and slot $k$ inherits the primary block label at position $j + k$,

$$\pi^{(j)}_{a,k} = \pi_{a,j+k} = p_T\big(\cdot \mid x_{\le a}, y^*_{<j+k}\big), \qquad \omega^{(j)}_{a,k} = w_k \, \mathbf{1}[\, j + k \le K \,]. \tag{4}$$

A secondary block provides an anchor whose immediate prefix consists of target-generated text, supervising up to $K - j$ slots, averaging seven slots prior to sequence termination. This design incurs no additional forward passes over the target model, as representations for $y^*_{<j}$ and target distributions past index $j$ are already computed during the primary rollout. Only the $\lceil m/2 \rceil$ primary blocks initiate rollout generation, compared to all $m$ blocks in ALR. By reusing rollout features and distributions from primary blocks, IRA introduces secondary anchors without increasing target forward computation.

## 4 Accepted Length and Speedup on a Fixed Corpus

**Experimental setup.** For vision-language experiments, we adopt Qwen3-VL-8B-Instruct (Bai et al., 2025) as the target model and pair it with a DFlash block drafter (Chen et al., 2026) initialized from the public text-only DFlash head. We train drafters on either the `allava_laion` split of ALLaVA (Chen et al., 2024a) or ShareGPT4V captions (Chen et al., 2024b), matching both corpora to an identical count of image-text pairs. For the text-only target Qwen3-4B (Yang et al., 2025), we train all four methods on the same UltraChat subset (Ding et al., 2023), keeping all three training corpora strictly fixed. We evaluate vision-language drafting on COCO captioning (Lin et al., 2014), TextVQA (Singh et al., 2019), and DocVQA (Mathew et al., 2021), and assess text-only drafting across seven benchmarks spanning instruction following, mathematical reasoning, and code generation.

**Evaluation metrics.** We report mean acceptance length $\tau$ (MAT), counting accepted draft tokens plus one target token per verification step, alongside wall-clock speedup over autoregressive decoding (AR). To compute aggregate acceptance lengths, we take the geometric mean across evaluation tasks within each training seed and then average across seeds. For vision-language benchmarks, we evaluate the first 100 prompts per domain, capping generation at 256 new tokens across all tasks. We examine both greedy decoding ($T$=0) and stochastic sampling ($T$=1). We provide detailed evaluation protocols in Appendices B and D. Given that bf16 tree verification trajectories can vary slightly across GPU architectures, we measure acceptance length for the primary vision-language target on RTX A6000 hardware and evaluate the smaller vision-language and text-only targets on H100 GPUs. We measure all speedups on H100 GPUs. In Appendix G, we document training replicates and sensitivity analyses, and in Appendix E, we define the formal comparison rule for vision-language evaluations.

### 4.1 Accepted Length Across Targets

Table 1 summarizes accepted lengths and decoding speedups following three epochs of training on each fixed corpus. Under greedy decoding ($T$=0), ALR + IRA attains the highest accepted length across all three vision-language configurations. On both 8B corpora, ALR consistently improves upon the soft survival-weighted erase baseline from PARD-2 (An et al., 2026), with IRA providing an additional boost that reaches an 8.54% margin over erase on ALLaVA. Both steps clear the

Table 1: Speedup over autoregressive decoding (AR) and mean acceptance length $\tau$ after three epochs on each fixed training corpus. Vision values summarize COCO captioning, TextVQA and DocVQA. Text rows identify evaluation benchmarks. Aggregates are geometric means across benchmarks, averaged over available replicates.

| *Vision* | | | | | | |
|---|---|---|---|---|---|---|
| Target | Qwen3-VL-8B | | | | Qwen3-VL-4B | |
| Train corpus | ALLaVA | | ShareGPT4V | | ALLaVA | |
| Method | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| | | | $T$=0 | | | |
| DFlash | 2.09× | 2.88 | 2.01× | 2.80 | 2.19× | 2.92 |
| Erase | 2.62× | 3.54 | 2.62× | 3.57 | 2.62× | 3.56 |
| ALR | 2.68× | 3.70 | **2.80**× | 3.67 | 2.65× | 3.63 |
| ALR + IRA | **2.82**× | **3.84** | 2.77× | **3.83** | **2.77**× | **3.81** |
| | | | $T$=1 | | | |
| DFlash | 2.04× | 2.79 | 1.90× | 2.71 | 2.04× | 2.78 |
| Erase | 2.49× | 3.39 | 2.54× | 3.39 | 2.45× | 3.33 |
| ALR | 2.55× | 3.50 | 2.54× | 3.48 | 2.53× | 3.41 |
| ALR + IRA | **2.68**× | **3.65** | **2.58**× | **3.60** | **2.67**× | **3.53** |

| *Text benchmarks: Qwen3-4B* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | DFlash | | Erase | | ALR | | ALR + IRA | |
| Benchmark | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| | | | | $T$=0 | | | | |
| MT-Bench | 1.82× | 2.50 | 2.07× | 2.85 | 2.09× | 2.94 | **2.13**× | **2.99** |
| Alpaca | 1.56× | 2.17 | 1.88× | 2.52 | 1.87× | 2.58 | **1.92**× | **2.62** |
| GSM8K | 2.79× | 3.96 | 3.16× | 4.48 | 3.38× | 4.76 | **3.55**× | **4.83** |
| AIME24 | 2.61× | 3.69 | 2.85× | 4.06 | 2.91× | 4.20 | **2.97**× | **4.27** |
| AIME25 | 2.76× | 4.07 | 3.01× | 4.34 | 2.94× | 4.53 | **3.14**× | **4.66** |
| HumanEval | 2.98× | 4.22 | 3.46× | 4.92 | 3.56× | 4.98 | **3.63**× | **5.02** |
| LiveCodeBench | 2.33× | 3.25 | 2.64× | 3.72 | 2.62× | 3.81 | **2.73**× | **3.84** |
| Avg. (7 tasks) | 2.35× | 3.32 | 2.67× | 3.75 | 2.70× | 3.87 | **2.80**× | **3.93** |
| | | | | $T$=1 | | | | |
| MT-Bench | 1.79× | 2.42 | 1.97× | 2.72 | 2.06× | 2.79 | **2.15**× | **2.83** |
| Alpaca | 1.52× | 2.16 | 1.72× | 2.47 | 1.78× | 2.54 | **1.80**× | **2.57** |
| GSM8K | 2.71× | 3.78 | 3.01× | 4.26 | 3.19× | 4.50 | **3.29**× | **4.55** |
| AIME24 | 2.41× | 3.35 | 2.49× | 3.54 | 2.58× | **3.66** | **2.59**× | 3.65 |
| AIME25 | 2.40× | 3.49 | 2.62× | 3.77 | 2.46× | 3.85 | **2.66**× | **3.96** |
| HumanEval | 2.89× | 3.97 | **3.22**× | 4.57 | 3.19× | 4.63 | 3.20× | **4.66** |
| LiveCodeBench | 2.20× | 3.00 | 2.44× | 3.38 | 2.42× | 3.47 | **2.52**× | **3.49** |
| Avg. (7 tasks) | 2.23× | 3.10 | 2.45× | 3.46 | 2.48× | 3.56 | **2.55**× | **3.60** |

registered comparison threshold on these datasets. As reported in Appendix G, soft gating outperforms matched ungated distillation, establishing erase as a competitive baseline. The relative ordering across methods remains consistent within each domain of both 8B corpora (Table 8), where DocVQA exhibits the narrowest gap. Under stochastic sampling ($T$=1), the aggregate ranking is preserved even though the training labels derive from greedy target rollouts. Figure 2 illustrates these benchmark-level gains over erase under both temperature settings. Furthermore, Table 15 documents a 5.90% aggregate accepted-length improvement over erase for ALR + IRA under chain verification at $T$=0 on ALLaVA, demonstrating that the gain does not rely on tree branching.

**ALR + IRA outperforms DFlash and the best erase schedules.** ALR + IRA improves accepted length over DFlash trained on identical data by up to 36.5% at $T$=0 and 33.1% at $T$=1 across both 8B settings (Table 1). Across training horizons, the combined framework also surpasses the strongest evaluated erase schedule on each corpus at $T$=0 (Table 12). On ShareGPT4V, where the top baseline reference corresponds to single-epoch erase, ALR + IRA maintains a 5.09% lead.

**The gain repeats with a second target.** These improvements generalize to an alternative target model, where ALR + IRA exceeds erase by 7.13% in accepted length at $T$=0 using Qwen3-VL-4B-Instruct (Table 1). On DocVQA, ALR in isolation falls below erase at both decoding temperatures,

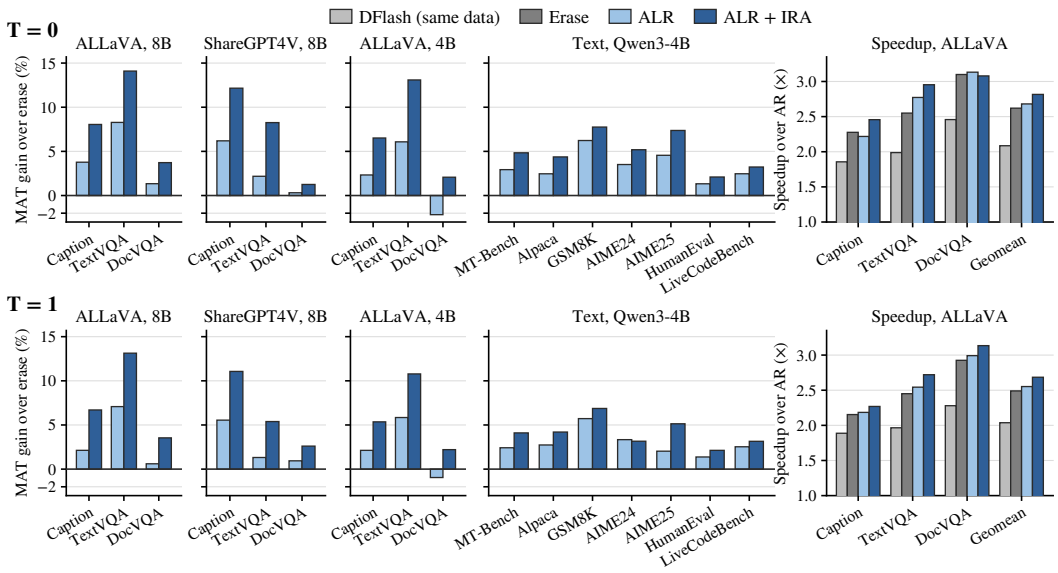


Figure 2: MAT gain over erase across vision-language and text benchmarks, and ALLaVA decoding speedup over autoregressive decoding after three training epochs. The top and bottom rows show $T{=}0$ and $T{=}1$, respectively. Gains use means across training replicates.

whereas ALR + IRA rises above it (Figure 2). This reversal confirms the distinct value of IRA in regimes where relabelling alone does not overcome erase.

### 4.2 Decoding speedup

Figure 2 shows that ALR + IRA decodes faster than erase on ALLaVA at both temperatures: 2.82× versus 2.62× over AR at $T{=}0$, and 2.68× versus 2.49× at $T{=}1$. At $T{=}0$, captioning and TextVQA drive the speed gain while DocVQA is nearly tied despite higher accepted length. At $T{=}1$, ALR + IRA is faster than erase in all three domains. Table 1 shows the speed advantage also holds on ShareGPT4V under sampling, and Table 16 reports a 4.1% advantage over its one-epoch erase reference at $T{=}0$. Appendix I reports the separate EAGLE-3 comparison at both temperatures.

**The gains extend to a text-only target.** Table 1 shows that ALR + IRA has the highest aggregate accepted length and decoding speedup across the seven text benchmarks at both temperatures. Its accepted-length gain over erase reaches 4.96% at $T{=}0$ and 4.10% at $T{=}1$. Table 11 confirms aggregate speed gains over erase of 4.62% at $T{=}0$ and 3.89% at $T{=}1$ under repeated A100 timing on a fixed text subset.

## 5 What the Recovered Supervision Costs and Where It Acts

Table 2 pairs training time with accepted length for one-epoch ALR and ALR + IRA and for C, the best erase schedule measured on each vision-language corpus. This comparison accounts for target-rollout compute alongside data exposure. Appendix F gives the full epoch, envelope and rollout-depth sweeps.

**One epoch exceeds the best erase schedules.** Table 2 shows that a single epoch of ALR + IRA exceeds the best erase schedule on both corpora, while costing less than two epochs of erase. On ALLaVA, it achieves this higher accepted length in 41% less training time than the best erase schedule.

Table 2: One-epoch training and the best erase schedules C at $T{=}0$. ALR uses $R{=}4$, and ALR + IRA uses $R{=}14$. Mean ± SD.

| Method | Epochs | Time/C | $\tau$ |
|---|---|---|---|
| *ALLaVA* | | | |
| Erase | 1 | 0.33 | $3.533_{\pm 0.006}$ |
| ALR | 1 | 0.42 | $3.649_{\pm 0.014}$ |
| ALR + IRA | 1 | 0.59 | $\mathbf{3.728}_{\pm 0.005}$ |
| Erase (C) | 3 | 1.00 | $3.539_{\pm 0.010}$ |
| *ShareGPT4V* | | | |
| Erase (C) | 1 | 1.00 | $3.640_{\pm 0.002}$ |
| ALR | 1 | 1.28 | $3.695_{\pm 0.011}$ |
| ALR + IRA | 1 | 1.77 | $\mathbf{3.778}_{\pm 0.003}$ |

**Additional training preserves the advantage.** Figure 3 shows that ALR + IRA improves from one to three epochs, while training erase for six epochs does not close the gap. At both recorded epoch counts, ALR + IRA achieves higher accepted length than full-depth ALR in less training time. Its secondary blocks reuse features and labels from the primary rollouts. This reuse reduces rollout computation while providing target-generated context. The trajectories suggest that recovering supervision is more effective than additional passes over the fixed corpus.

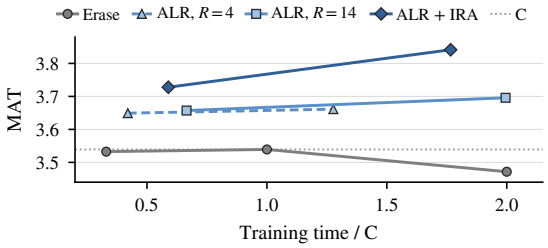


Figure 3: ALLaVA training cost at $T$=0. Points show one and three epochs, plus six for erase. Time is relative to C, whose MAT the dotted line marks.

**Short rollouts suffice for ALR.** Table 3 shows that $R = 4$ is the shortest tested ALR depth that is non-inferior to $R = 14$ after three epochs on ALLaVA. It also meets the comparison criterion on ShareGPT4V and trains faster than $R = 14$ on both corpora, supporting its use for the low-cost ALR setting in Table 2. The criterion is defined in Appendix E, and Table 13 gives the full sweep. The main three-epoch comparison keeps $R = 14$ for both ALR and ALR + IRA. IRA needs that depth because it draws secondary anchors at offsets up to $K - 1$ inside the rollout.

Table 3: ALR depth after three epochs at $T$=0. Mean ± SD.

| Rollout depth $R$ | MAT ↑ |
|---|---|
| *ALLaVA* | |
| 14 | $3.696_{\pm 0.005}$ |
| 6 | $3.682_{\pm 0.004}$ |
| 4 | $3.661_{\pm 0.023}$ |
| 2 | $3.597_{\pm 0.017}$ |
| *ShareGPT4V* | |
| 14 | $3.673_{\pm 0.007}$ |
| 4 | $3.659_{\pm 0.014}$ |

**ALR + IRA matches length-matched regeneration on a fixed corpus.** Table 4-(a) shows that, after three epochs on fixed ALLaVA, our ALR + IRA comes within 0.4% of both target-regeneration references in aggregate accepted length. Response regeneration aligns the drafter's training text with the target (Cai et al., 2024; Zhou et al., 2023); our two references use the same distillation loss and control supervised response length by truncation or a loss window. This agreement suggests that rollout labels and in-rollout context can recover the accepted-length benefit of length-controlled regeneration without replacing the training corpus.

Table 4: Length-matched target regeneration on the same ALLaVA prompts after three epochs at $T$=0. (a) Overall MAT with Qwen3-VL-8B. (b) MAT gains over ALR by round-start token offset on COCO captions.

| | (a) Overall | (b) COCO gain over ALR (%) | | | |
|---|---|---|---|---|---|
| Method | MAT ↑ | 0–15 | 16–63 | 64–127 | 128+ |
| ALR | 3.696 | 0.00 | 0.00 | 0.00 | 0.00 |
| ALR + IRA | 3.841 | +1.32 | +5.60 | +5.15 | +3.57 |
| *Target regeneration* | | | | | |
| Truncated | 3.833 | +1.49 | +5.61 | +4.51 | +4.72 |
| Loss window | **3.856** | +3.60 | +7.21 | +5.12 | +5.65 |

**IRA gains beyond changing slot weights.** Figure 4-(a) shows that ALR + IRA achieves 4.10% higher accepted length than the slot-matched control on ALLaVA after three epochs. The control matches IRA's block count, offsets and slot weights while keeping all blocks at corpus anchors, yet performs similarly to ALR. This supports placing anchors inside target rollouts to train the drafter on both recent target-generated context and its continuation labels.

**IRA's gain concentrates where drafting is hard.** Figure 4-(b)–(c) groups drafting offsets in held-out captions, disjoint from the main evaluation, by the acceptance length of an independent drafter trained on the other corpus. This drafter is the compared arms' common parent. On both corpora, IRA's gain is largest where that drafter misses the first token, whereas the seed control stays flat and the gain is smaller where it accepts five or more tokens. Appendix G gives additional controls.

**IRA's benefit grows beyond the opening tokens.** Table 4-(b) shows that IRA's gain over ALR is larger at later positions in COCO captions. The truncated regeneration reference shows a related pattern: its advantage over ALR grows after the first 16 caption tokens and then levels off. Together, these patterns support exposing the drafter to the target-generated context that accumulates during decoding. The positional rise also persists when all drafters are scored on fixed text, while an ALR seed control shows no such rise, as detailed in Appendix G.

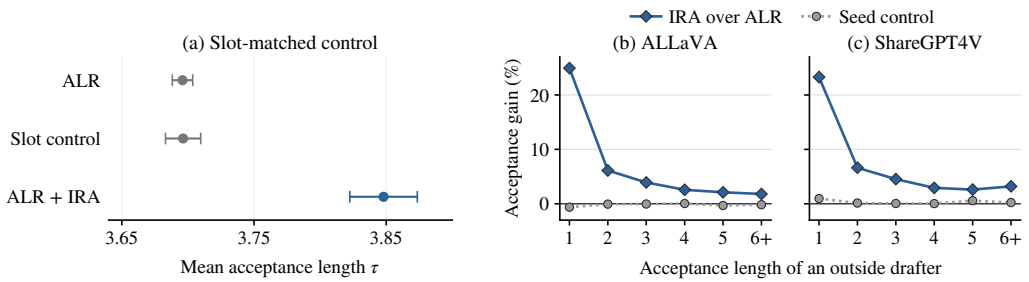


Figure 4: IRA controls. (a) Slot-matched control on ALLaVA at $T{=}0$. Error bars show sample SD across matched training replicates. (b,c) IRA's acceptance-length gain over ALR on held-out captions, grouped by an independent drafter's acceptance length, with an ALR seed control.

**Rollout labels improve acceptance under either weighting rule.** Table 5 shows that rollout-conditioned labels improve aggregate accepted length under both choices of survival weighting on the fixed text corpus. The four arms keep the training recipe and corpus anchors fixed, separating label construction from the use of the gate. Survival weighting helps corpus-conditioned labels but adds little to rollout-conditioned labels at either decoding temperature. The gain therefore persists when the weighting rule is retained, supporting the use of the target's continuation to construct later-slot supervision.

Table 5: Label source and survival weighting after three epochs on fixed UltraChat. MAT aggregates seven text benchmarks. Entries report mean ± SD across matched training replicates.

| Label source | Soft gate | $T{=}0$ | $T{=}1$ |
|---|---|---|---|
| Corpus (KD) | Off | 3.586 ± 0.0004 | 3.306 ± 0.005 |
| Corpus (erase) | On | 3.758 ± 0.015 | 3.483 ± 0.027 |
| Target rollout | On | 3.886 ± 0.001 | 3.583 ± 0.007 |
| Target rollout (ALR) | Off | 3.884 ± 0.008 | 3.580 ± 0.035 |

## 6 Conclusion

In this work, we demonstrated that block drafters trained on fixed off-policy corpora benefit substantially more from supervision along target greedy rollouts than from pruning divergent slots. Our framework recovers this missing supervision through two complementary mechanisms. Anchor-Label Relabelling (ALR) derives valid target distributions along greedy rollouts across all draft positions, while In-Rollout Anchors (IRA) position secondary blocks within these trajectories to expose the drafter to target-generated context. By reusing intermediate representations computed during the primary rollout, IRA supplies this recent context without incurring additional forward passes over the target model. Across two fixed vision-language corpora, the integrated framework improves greedy accepted length by up to 36.5% over DFlash trained on identical data and consistently outperforms the strongest evaluated erase schedules. These empirical gains translate into concrete decoding speedups, generalize to a smaller vision-language target, and transfer directly to text-only benchmarks. Crucially, on ALLaVA, ALR + IRA matches the acceptance length of training on target-regenerated responses, confirming that rollout-based training provides a practical and compute-efficient alternative when the original training corpus must remain intact. Ultimately, our findings establish target rollouts as a robust foundation for off-policy draft training, and we plan to extend this framework to broader drafter architectures in future work.

## Reproducibility Statement

Every DFlash-family objective we compare is defined in Section 3. Appendix A gives the shared vision-language training configuration, public warm-start checkpoints, corpus selection procedures, and rollout implementation and cost. Appendix B lists the DFlash-family vision-language decoder settings and prompt sets. Appendix D gives the text-only training and evaluation protocol, and Appendix G records the training replicates behind the main comparisons. Appendix E specifies the device-consistency checks and comparison rule for the DFlash-family vision-language contrasts.

## Ethics Statement

This work trains draft models for speculative decoding using public text and image-text corpora and public language and vision-language models. It involves no human-subject experiments or new data collection. Speculative decoding preserves the target distribution in exact arithmetic, and the method inherits the target's biases and failure modes (Lee et al., 2022). The vision-language training costs are reported in Section 5, and the text experiment protocol is given in Appendix D.

## AI use statement

A large language model assisted with LaTeX tables, and language polishing. The authors reviewed and verified all outputs and take full responsibility for the paper.

## References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stańczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes, 2023. URL http://arxiv.org/abs/2306.13649.

Zihao An, Huajun Bai, Ziqiong Liu, Dong Li, and Emad Barsoum. PARD: Accelerating LLM Inference with Low-Cost PARallel Draft Model Adaptation, 2025. URL https://arxiv.org/abs/2504.18583.

Zihao An, Taichi Liu, Ziqiong Liu, Dong Li, Ruofeng Liu, and Emad Barsoum. PARD-2: Target-Aligned Parallel Draft Model for Dual-Mode Speculative Decoding, 2026. URL https://arxiv.org/abs/2605.08632.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL Technical Report, 2025. URL https://arxiv.org/abs/2511.21631.

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, 2024. URL http://arxiv.org/abs/2401.10774.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating Large Language Model Decoding with Speculative Sampling, 2023. URL http://arxiv.org/abs/2302.01318.

Guiming Hardy Chen, Shunian Chen, Ruifei Zhang, Junying Chen, Xiangbo Wu, Zhiyi Zhang, Zhihong Chen, Jianquan Li, Xiang Wan, and Benyou Wang. ALLaVA: Harnessing GPT4V-Synthesized Data for Lite Vision-Language Models, 2024a. URL http://arxiv.org/abs/2402.11684.

Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block Diffusion for Flash Speculative Decoding, 2026. URL https://arxiv.org/abs/2602.06036.

Lin Chen, Jinsong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. ShareGPT4V: Improving Large Multi-modal Models with Better Captions, 2024b. URL https://doi.org/10.1007/978-3-031-72643-9_22.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri,

Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code, 2021. URL https://arxiv.org/abs/2107.03374.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training Verifiers to Solve Math Word Problems, 2021. URL https://arxiv.org/abs/2110.14168.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.183. URL https://aclanthology.org/2023.emnlp-main.183/.

Mugilan Ganesan, Shane Segal, Ankur Aggarwal, Nish Sinnadurai, Sean Lie, and Vithursan Thangarasa. MASSV: Multimodal Adaptation and Self-Data Distillation for Speculative Decoding of Vision-Language Models, 2025. URL https://doi.org/10.18653/v1/2025.findings-emnlp.656.

Xiangxiang Gao, Weisheng Xie, Yiwei Xiang, and Feng Ji. Falcon: Faster and Parallel Inference of Large Language Models Through Enhanced Semi-Autoregressive Drafting and Custom-Designed Decoding Tree, 2025. URL https://doi.org/10.1609/aaai.v39i22.34566.

Raghavv Goel, Mukul Gagrani, Wonseok Jeon, Junyoung Park, Mingu Lee, and Christopher Lott. Direct Alignment of Draft Model for Speculative Decoding with Chat-Fine-Tuned LLMs, 2024. URL http://arxiv.org/abs/2403.00858.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: On-Policy Distillation of Large Language Models, 2023. URL http://arxiv.org/abs/2306.08543.

Geoffrey E. Hinton, O. Vinyals, and J. Dean. Distilling the Knowledge in a Neural Network, 2015. URL https://arxiv.org/abs/1503.02531.

Fenglu Hong, Ravi Raju, Jonathan Lingjie Li, Bo Li, Urmish Thakker, Avinash Ravichandran, Swayambhoo Jain, and Changran Hu. Training Domain Draft Models for Speculative Decoding: Best Practices and Insights, 2025. URL http://arxiv.org/abs/2503.07807.

Yunhai Hu, Tianhua Xia, Zining Liu, Rahul Raman, Xingyu Liu, BO BAO, Eric Sather, Vithursan Thangarasa, and Sai Qian Zhang. DREAM: Drafting with Refined Target Features and Entropy-Adaptive Cross-Attention Fusion for Multimodal Speculative Decoding, 2025. URL https://doi.org/10.52202/085713-5586.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code, 2024. URL https://arxiv.org/abs/2403.07974.

Jialiang Kang, Han Shu, Wenshuo Li, Yingjie Zhai, and Xinghao Chen. ViSpec: Accelerating Vision-Language Models with Vision-Aware Speculative Decoding, 2025. URL https://doi.org/10.52202/085713-3852.

Yoon Kim and Alexander M. Rush. Sequence-Level Knowledge Distillation, 2016. URL https://doi.org/10.18653/v1/d16-1139.

Jungseob Lee, Midan Shim, Suhyune Son, Chanjun Park, Yujin Kim, and Heuiseok Lim. There is no rose without a thorn: Finding weaknesses on BlenderBot 2.0 in terms of Model, Data and User-Centric Approach, 2022. URL https://arxiv.org/abs/2201.03239.

Jungseob Lee, Seungyoon Lee, Suhyune Son, Dongyub Jude Lee, Sungbin Han, Sugyeong Eo, and Heuiseok Lim. Answer-Conditioned Chains of Thought Degrade Verifiable-Reasoning Distillation in Large Language Models, 2026. URL https://arxiv.org/abs/2607.14552.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative Sampling Requires Rethinking Feature Uncertainty, 2024a. URL http://arxiv.org/abs/2401.15077.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster Inference of Language Models with Dynamic Draft Trees, 2024b. URL https://doi.org/10.18653/v1/2024.emnlp-main.422.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-3: Scaling up Inference Acceleration of Large Language Models via Training-Time Test, 2025. URL https://doi.org/10.52202/085713-4562.

Luxi Lin, Zhihang Lin, Zhanpeng Zeng, and Rongrong Ji. Speculative Decoding Reimagined for Multimodal Large Language Models, 2025. URL http://arxiv.org/abs/2505.14260.

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, 2014. URL https://doi.org/10.1007/978-3-319-10602-1_48.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual Instruction Tuning, 2023a. URL https://doi.org/10.52202/075280-1516.

Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online Speculative Decoding, 2023b. URL http://arxiv.org/abs/2310.07177.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. DocVQA: A Dataset for VQA on Document Images, 2021. URL https://doi.org/10.1109/wacv48630.2021.00225.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning, 2010. URL https://arxiv.org/abs/1011.0686.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA Models That Can Read, 2019. URL https://arxiv.org/abs/1904.08920.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise Parallel Decoding for Deep Autoregressive Models, 2018. URL http://arxiv.org/abs/1811.03115.

Yujia Tong, Tian Zhang, Yunyang Wan, Kaiwei Lin, Jingling Yuan, and Chuang Hu. SAGE: Accelerating Vision-Language Models via Entropy-Guided Adaptive Speculative Decoding, 2026. URL https://arxiv.org/abs/2602.00523.

Zilin Xiao, Hongming Zhang, Tao Ge, Siru Ouyang, Vicente Ordóñez, and Dong Yu. ParallelSpec: Parallel Drafter for Efficient Speculative Decoding, 2024. URL http://arxiv.org/abs/2410.05589.

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

Ofir Zafrir, Igor Margulis, Dorin Shteyman, Shira Guskin, and Guy Boudoukh. FastDraft: How to Train Your Draft, 2025. URL https://doi.org/10.18653/v1/2025.findings-acl.1156.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning Harmonized Representations for Speculative Sampling, 2024. URL http://arxiv.org/abs/2408.15766.

Yifan Zhang and Team Math-AI. American Invitational Mathematics Examination (AIME) 2025. Math-AI dataset release on Hugging Face, 2025. URL https://math-ai-org.github.io/aime25/.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623. Curran Associates, Inc., 2023. doi: 10.52202/075280-2020. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91f18a1287b398d378ef22505bf41832-Paper-Datasets_and_Benchmarks.pdf.

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, 2023. URL http://arxiv.org/abs/2310.08461.

## A Training Configuration

Table 6 lists the shared vision-language training configuration. These arms differ only in the objective, number of epochs, rollout depth $R$ or envelope $\gamma$ named in the comparison. Appendix D gives the text-only configuration.

**Rollout implementation.** The rollout runs inside the training step and reuses the target's forward pass over the corpus sample. That pass fills a key-value cache, and the rollout then takes $R$ greedy steps, each of which runs all anchors of a sample through the target's decoder layers as one batch. The query of an anchor attends under a single softmax to the cached corpus keys up to its anchor, which all anchors share, and to its own rollout keys so far, which are kept in a separate per-anchor buffer. The corpus keys are therefore never copied per anchor, and grouped-query attention forms its scores per key-value head without repeating them. Rollout positions continue the anchor's text position, with the three M-RoPE axes advancing together. Each step reads the labels from the target's final normalised hidden state through its output head, keeping the 8 most probable tokens and the remaining mass, and greedy ties go to the lower token id. For IRA, the same steps also store the outputs of the five target layers the drafter reads for $y^*_1, \ldots, y^*_{13}$, which become the context of the secondary blocks. On four H100 GPUs, an epoch on the ALLaVA rows takes 0.75 GPU-hours for erase, 1.49 for ALR, 0.95 for ALR at $R{=}4$ and 1.32 for ALR + IRA.

## B Evaluation Protocol

Table 7 lists the DFlash-family vision-language evaluation protocol. The images of both training corpora come from LAION, CC and SBU, and none of the evaluation image sources is shared with either training corpus.

Table 6: Shared vision-language training configuration.

| Setting | Value |
|---|---|
| | *Target* |
| Model | Qwen3-VL-8B-Instruct |
| Second target | Qwen3-VL-4B-Instruct, text decoder hidden 2560, 36 layers, tied embeddings |
| Text decoder hidden size | 4096 |
| Text decoder layers | 36 |
| Attention heads | 32 |
| Vocabulary | 151,936 |
| | *Drafter* |
| Architecture | DFlash block drafter, 5 layers |
| Target feature layers | 1, 9, 17, 25, 33 |
| Block | 16 slots, slot 0 the anchor, slots 1 to 15 predicted in parallel |
| Warm start | `z-lab/Qwen3-8B-DFlash-b16`, text-only; `z-lab/Qwen3-4B-DFlash-b16` for the 4B target |
| Architecture after warm start | kept unchanged |
| | *Data* |
| ALLaVA dataset | `FreedomIntelligence/ALLaVA-4V`, instruction file of the `allava_laion` subset |
| ALLaVA content | LAION images, GPT-4V-style responses |
| ALLaVA selection | rows shuffled with a fixed seed, then the first 26,196 that are single-turn, have their image in the first released image chunk and a response of at least 40 characters |
| ShareGPT4V source | GPT-4V captions of `cap100k` whose images come from LLaVA-Pretrain (LAION, CC, SBU), 30,000 rows |
| ShareGPT4V selection | sorted by id, shuffled with a fixed seed, first 26,196 kept to match the ALLaVA row count |
| Maximum length | 4096 tokens |
| Image size | 50,176 to 200,704 pixels |
| Vision tokens per image | 16 to 64 after merging |
| | *Optimisation* |
| Hardware | 4 H100 80 GB GPUs, FSDP |
| Batch size | 6 per GPU |
| Learning rate | $1.4697 \times 10^{-3}$, the same for every objective |
| Optimiser | AdamW on fp32 copies of the bf16 weights, $\beta = (0.9, 0.999)$, no weight decay |
| Schedule | cosine annealing after a linear warmup over 4% of the steps |
| Gradient-norm clip | 1.0 |
| Epochs | 3, unless a row names another schedule |
| Steps for 3 epochs, ALLaVA | 3,261 |
| Steps for 3 epochs, ShareGPT4V | 3,273 |
| Anchors per sample | 128 |
| Precision | bf16 |
| Attention | sdpa |
| | *Loss* |
| KD loss | soft cross-entropy against the target's top-8 probabilities plus one bucket for the remaining mass, weighted mean over slots |
| KD loss scale $\lambda$ | 1.013441, matching the cross-entropy loss scale at initialisation |
| Slot envelope | $w_k = e^{-(k-1)/\gamma}$ |
| Envelope $\gamma$ | 2, except one erase schedule at 0.5 |
| Rollout depth $R$ | 14 for ALR and ALR + IRA |
| Depth sweep, ALR alone | $R \in \{2, 4, 6\}$ |
| Secondary-anchor offset | $j \sim \mathcal{U}\{2, \ldots, 14\}$ |

## C Vision-Language Results by Domain

Table 8 disaggregates the vision-language results of Table 1 by evaluation domain and reports training-seed variation. ALLaVA and ShareGPT4V name training corpora, while COCO, TextVQA and DocVQA name evaluation datasets.

Table 7: DFlash-family vision-language evaluation protocol.

| Setting | Value |
|---|---|
| *Prompts* | |
| Captioning | COCO val2017, detailed captioning |
| Captioning prompt | "Describe this image in detail." |
| TextVQA | validation split |
| DocVQA | validation split |
| Prompts per domain | the first 100, of which 99 are scored after one warm-up |
| Maximum new tokens | 256 |
| Temperature | 0, and 1 for the sampling readings |
| *Decoder* | |
| Verification | tree, one target forward pass per round |
| Tree budget | 63 drafted tokens |
| Candidates per slot | the drafter's 8 most probable tokens |
| Tree construction | best-first by path log-probability, prefix-closed; a path can reach all 15 slots |
| Mean acceptance length $\tau$ (MAT) | accepted draft tokens plus the one token the target adds in each verification round |
| Chain decoding | one robustness reading: the top token of all 15 slots, verified as one chain |
| Attention | sdpa |
| Precision | bf16 |
| *Hardware* | |
| Batch size | one prompt at a time, for both the drafter and the baseline |
| MAT | one RTX A6000 per evaluation; one H100 for the 4B target |
| Speed | one H100 assigned to each evaluator |
| Speed baseline | autoregressive decoding timed in the same run |

## D TEXT-ONLY EVALUATION

We train DFlash, erase, ALR and ALR + IRA for three epochs on the same fixed UltraChat subset with Qwen3-4B as the target and thinking disabled. The selection contains 26,196 conversations, of which 25,555 pass the minimum assistant-token filter. All methods start from `z-lab/Qwen3-4B-DFlash-b16` and use two training replicates, effective batch size 24, learning rate $1.4697 \times 10^{-3}$, maximum training length 1,024, 128 anchors, block size 16, $\gamma = 2$ and KD scale 1.0. Corpus selection and data order stay fixed. The four-H100 training configuration uses batch size 6 per GPU.

Evaluation covers seven benchmarks at $T{=}0$ and $T{=}1$, with at most 256 generated tokens. It includes all 80 MT-Bench first turns (Zheng et al., 2023), 1,319 GSM8K test questions (Cobbe et al., 2021) and 164 HumanEval tasks (Chen et al., 2021), plus the four benchmarks described below. The full-benchmark evaluations use one H100, bf16, SDPA and the public DFlash chain decoder. At $T{=}1$, target sampling preserves the target distribution in exact arithmetic but does not force identical outputs across methods. We measure decoding efficiency rather than answer accuracy or code pass rate. Acceptance length pools verification rounds within each task. The Text panel of Table 1 reports task means and their aggregate. Tables 9 and 10 add standard deviations across the two training seeds. The aggregate first takes the geometric mean across all seven tasks within each seed, then averages across seeds.

Within each text benchmark and temperature, all methods use the same AR timing reference. Model loading and tokenization are excluded.

**Additional text benchmarks.** The additional benchmarks comprise 80 Alpaca prompts distributed with EAGLE-3 (Li et al., 2025), all 30 AIME24 problems from the 2024 I and II contests,[1] 30 AIME25 problems (Zhang & Team Math-AI, 2025) and 400 LiveCodeBench release_v1 tasks (Jain et al., 2024). Dataset revisions and prompt formatting are fixed across all four methods and

[1] We use the math-ai/aime24 test split at the fixed revision `83a7f387`.

Table 8: Vision-language mean acceptance length $\tau$ by evaluation domain after three training epochs. Subscripts show training-seed standard deviations. Avg. is the geometric mean across domains within each seed, averaged across seeds.

| Method | COCO | TextVQA | DocVQA | Avg. |
|---|---|---|---|---|
| *Qwen3-VL-8B trained on ALLaVA, $T$=0* | | | | |
| DFlash | $2.563_{\pm 0.006}$ | $2.655_{\pm 0.010}$ | $3.501_{\pm 0.010}$ | $2.878_{\pm 0.007}$ |
| Erase | $3.124_{\pm 0.006}$ | $3.375_{\pm 0.005}$ | $4.204_{\pm 0.037}$ | $3.539_{\pm 0.010}$ |
| ALR | $3.242_{\pm 0.007}$ | $3.655_{\pm 0.024}$ | $4.261_{\pm 0.016}$ | $3.696_{\pm 0.005}$ |
| ALR + IRA | $3.376_{\pm 0.007}$ | $3.851_{\pm 0.011}$ | $4.361_{\pm 0.065}$ | $3.841_{\pm 0.021}$ |
| *Qwen3-VL-8B trained on ALLaVA, $T$=1* | | | | |
| DFlash | $2.435_{\pm 0.009}$ | $2.601_{\pm 0.013}$ | $3.436_{\pm 0.005}$ | $2.792_{\pm 0.002}$ |
| Erase | $2.952_{\pm 0.007}$ | $3.196_{\pm 0.012}$ | $4.115_{\pm 0.021}$ | $3.386_{\pm 0.009}$ |
| ALR | $3.015_{\pm 0.012}$ | $3.422_{\pm 0.024}$ | $4.140_{\pm 0.026}$ | $3.495_{\pm 0.007}$ |
| ALR + IRA | $3.150_{\pm 0.006}$ | $3.615_{\pm 0.015}$ | $4.261_{\pm 0.026}$ | $3.647_{\pm 0.005}$ |
| *Qwen3-VL-8B trained on ShareGPT4V, $T$=0* | | | | |
| DFlash | $2.542_{\pm 0.009}$ | $2.691_{\pm 0.004}$ | $3.216_{\pm 0.028}$ | $2.802_{\pm 0.003}$ |
| Erase | $3.403_{\pm 0.010}$ | $3.326_{\pm 0.023}$ | $4.023_{\pm 0.036}$ | $3.571_{\pm 0.016}$ |
| ALR | $3.614_{\pm 0.003}$ | $3.398_{\pm 0.009}$ | $4.035_{\pm 0.017}$ | $3.673_{\pm 0.007}$ |
| ALR + IRA | $3.817_{\pm 0.024}$ | $3.600_{\pm 0.024}$ | $4.073_{\pm 0.040}$ | $3.825_{\pm 0.029}$ |
| *Qwen3-VL-8B trained on ShareGPT4V, $T$=1* | | | | |
| DFlash | $2.395_{\pm 0.003}$ | $2.599_{\pm 0.007}$ | $3.183_{\pm 0.000}$ | $2.706_{\pm 0.003}$ |
| Erase | $3.132_{\pm 0.000}$ | $3.207_{\pm 0.031}$ | $3.874_{\pm 0.014}$ | $3.388_{\pm 0.015}$ |
| ALR | $3.306_{\pm 0.020}$ | $3.249_{\pm 0.004}$ | $3.910_{\pm 0.021}$ | $3.476_{\pm 0.015}$ |
| ALR + IRA | $3.478_{\pm 0.004}$ | $3.379_{\pm 0.019}$ | $3.975_{\pm 0.030}$ | $3.602_{\pm 0.017}$ |
| *Qwen3-VL-4B trained on ALLaVA, $T$=0* | | | | |
| DFlash | $2.619_{\pm 0.005}$ | $2.704_{\pm 0.008}$ | $3.529_{\pm 0.003}$ | $2.924_{\pm 0.006}$ |
| Erase | $3.185_{\pm 0.007}$ | $3.347_{\pm 0.016}$ | $4.215_{\pm 0.044}$ | $3.555_{\pm 0.004}$ |
| ALR | $3.259_{\pm 0.002}$ | $3.550_{\pm 0.025}$ | $4.124_{\pm 0.051}$ | $3.627_{\pm 0.024}$ |
| ALR + IRA | $3.392_{\pm 0.002}$ | $3.785_{\pm 0.017}$ | $4.303_{\pm 0.019}$ | $3.809_{\pm 0.011}$ |
| *Qwen3-VL-4B trained on ALLaVA, $T$=1* | | | | |
| DFlash | $2.476_{\pm 0.000}$ | $2.554_{\pm 0.001}$ | $3.383_{\pm 0.007}$ | $2.776_{\pm 0.002}$ |
| Erase | $2.947_{\pm 0.004}$ | $3.105_{\pm 0.024}$ | $4.046_{\pm 0.014}$ | $3.333_{\pm 0.011}$ |
| ALR | $3.010_{\pm 0.011}$ | $3.287_{\pm 0.042}$ | $4.008_{\pm 0.047}$ | $3.410_{\pm 0.024}$ |
| ALR + IRA | $3.105_{\pm 0.005}$ | $3.440_{\pm 0.018}$ | $4.135_{\pm 0.024}$ | $3.535_{\pm 0.002}$ |

Table 9: Text-only generation speedup over autoregressive decoding and mean accepted length $\tau$ on Qwen3-4B. All methods train for three epochs on the same fixed UltraChat corpus. Subscripts show standard deviations across training seeds.

| | MT-Bench | | GSM8K | | HumanEval | |
|---|---|---|---|---|---|---|
| Method | Speedup | $\tau$ | Speedup | $\tau$ | Speedup | $\tau$ |
| $T$=0 | | | | | | |
| DFlash | $1.82_{\pm 0.07}\times$ | $2.50_{\pm 0.01}$ | $2.79_{\pm 0.05}\times$ | $3.96_{\pm 0.01}$ | $2.98_{\pm 0.07}\times$ | $4.22_{\pm 0.02}$ |
| Erase | $2.07_{\pm 0.04}\times$ | $2.85_{\pm 0.01}$ | $3.16_{\pm 0.08}\times$ | $4.48_{\pm 0.01}$ | $3.46_{\pm 0.10}\times$ | $4.92_{\pm 0.00}$ |
| ALR | $2.09_{\pm 0.01}\times$ | $2.94_{\pm 0.02}$ | $3.38_{\pm 0.09}\times$ | $4.76_{\pm 0.00}$ | $3.56_{\pm 0.13}\times$ | $4.98_{\pm 0.06}$ |
| ALR + IRA | $2.13_{\pm 0.02}\times$ | $2.99_{\pm 0.00}$ | $3.55_{\pm 0.06}\times$ | $4.83_{\pm 0.07}$ | $3.63_{\pm 0.00}\times$ | $5.02_{\pm 0.06}$ |
| $T$=1 | | | | | | |
| DFlash | $1.79_{\pm 0.03}\times$ | $2.42_{\pm 0.05}$ | $2.71_{\pm 0.05}\times$ | $3.78_{\pm 0.01}$ | $2.89_{\pm 0.02}\times$ | $3.97_{\pm 0.07}$ |
| Erase | $1.97_{\pm 0.04}\times$ | $2.72_{\pm 0.04}$ | $3.01_{\pm 0.09}\times$ | $4.26_{\pm 0.01}$ | $3.22_{\pm 0.02}\times$ | $4.57_{\pm 0.10}$ |
| ALR | $2.06_{\pm 0.03}\times$ | $2.79_{\pm 0.02}$ | $3.19_{\pm 0.08}\times$ | $4.50_{\pm 0.03}$ | $3.19_{\pm 0.07}\times$ | $4.63_{\pm 0.00}$ |
| ALR + IRA | $2.15_{\pm 0.04}\times$ | $2.83_{\pm 0.01}$ | $3.29_{\pm 0.05}\times$ | $4.55_{\pm 0.02}$ | $3.20_{\pm 0.11}\times$ | $4.66_{\pm 0.01}$ |

both training seeds. The target, decoder, temperatures and 256-token output limit remain the same. Table 10 reports seed variation for these four benchmarks.

Table 10: Generation speedup over autoregressive decoding and mean acceptance length $\tau$ on additional text benchmarks. Subscripts show standard deviations across two training seeds.

| | $T$=0 | | $T$=1 | |
|---|---|---|---|---|
| Method | Speedup | $\tau$ | Speedup | $\tau$ |
| | | *Alpaca* | | |
| DFlash | $1.56_{\pm 0.02}\times$ | $2.17_{\pm 0.00}$ | $1.52_{\pm 0.04}\times$ | $2.16_{\pm 0.01}$ |
| Erase | $1.88_{\pm 0.01}\times$ | $2.52_{\pm 0.01}$ | $1.72_{\pm 0.09}\times$ | $2.47_{\pm 0.01}$ |
| ALR | $1.87_{\pm 0.07}\times$ | $2.58_{\pm 0.02}$ | $1.78_{\pm 0.13}\times$ | $2.54_{\pm 0.01}$ |
| ALR + IRA | $1.92_{\pm 0.04}\times$ | $2.62_{\pm 0.00}$ | $1.80_{\pm 0.08}\times$ | $2.57_{\pm 0.00}$ |
| | | *AIME24* | | |
| DFlash | $2.61_{\pm 0.02}\times$ | $3.69_{\pm 0.03}$ | $2.41_{\pm 0.03}\times$ | $3.35_{\pm 0.05}$ |
| Erase | $2.85_{\pm 0.04}\times$ | $4.06_{\pm 0.01}$ | $2.49_{\pm 0.04}\times$ | $3.54_{\pm 0.07}$ |
| ALR | $2.91_{\pm 0.07}\times$ | $4.20_{\pm 0.04}$ | $2.58_{\pm 0.14}\times$ | $3.66_{\pm 0.05}$ |
| ALR + IRA | $2.97_{\pm 0.11}\times$ | $4.27_{\pm 0.05}$ | $2.59_{\pm 0.02}\times$ | $3.65_{\pm 0.04}$ |
| | | *AIME25* | | |
| DFlash | $2.76_{\pm 0.14}\times$ | $4.07_{\pm 0.04}$ | $2.40_{\pm 0.04}\times$ | $3.49_{\pm 0.13}$ |
| Erase | $3.01_{\pm 0.03}\times$ | $4.34_{\pm 0.02}$ | $2.62_{\pm 0.05}\times$ | $3.77_{\pm 0.06}$ |
| ALR | $2.94_{\pm 0.04}\times$ | $4.53_{\pm 0.08}$ | $2.46_{\pm 0.03}\times$ | $3.85_{\pm 0.06}$ |
| ALR + IRA | $3.14_{\pm 0.01}\times$ | $4.66_{\pm 0.06}$ | $2.66_{\pm 0.04}\times$ | $3.96_{\pm 0.01}$ |
| | | *LiveCodeBench* | | |
| DFlash | $2.33_{\pm 0.01}\times$ | $3.25_{\pm 0.05}$ | $2.20_{\pm 0.06}\times$ | $3.00_{\pm 0.04}$ |
| Erase | $2.64_{\pm 0.06}\times$ | $3.72_{\pm 0.01}$ | $2.44_{\pm 0.00}\times$ | $3.38_{\pm 0.02}$ |
| ALR | $2.62_{\pm 0.11}\times$ | $3.81_{\pm 0.03}$ | $2.42_{\pm 0.09}\times$ | $3.47_{\pm 0.01}$ |
| ALR + IRA | $2.73_{\pm 0.13}\times$ | $3.84_{\pm 0.02}$ | $2.52_{\pm 0.15}\times$ | $3.49_{\pm 0.01}$ |

Table 11: Aggregate generation speedup over AR on 220 fixed prompts from seven text benchmarks, measured on A100. Gain compares ALR + IRA with erase. Brackets give paired-prompt 95% bootstrap intervals.

| Method | $T$=0 | $T$=1 |
|---|---|---|
| Erase | 2.66× | 2.48× |
| ALR + IRA | 2.79× | 2.58× |
| Gain over erase (%) | 4.62 [3.80, 5.42] | 3.89 [2.52, 5.28] |

**Repeated timing.** Table 11 tests the aggregate speed advantage with fresh AR references and balanced execution order. We select 32 prompts per benchmark by identifier hash and all 30 from each AIME set, giving 220 fixed prompts. The target, decoder and output limit remain unchanged. After warm-up, each prompt runs on the same A100 for all methods. For each prompt and temperature, two trained checkpoint pairs and three timing repeats cover all six AR/erase/ALR + IRA execution orders. Sampling randomness stays fixed across timing repeats. We average total generation milliseconds per token over repeats and then prompts within each task. Speedups are geometric means of task latency ratios within each checkpoint pair, averaged over pairs. Gain uses the erase/ALR + IRA ratio directly. The 95% intervals use 10,000 paired prompt resamples stratified by task, retaining all methods, checkpoints, temperatures and repeats together. They condition on the measured hardware and trained checkpoint pairs.

**Label controls.** Table 5 uses the same text recipe with a separate matched pair of training replicates. Gates use probabilities along the selected continuation.

## E Evaluation Determinism

**Device consistency.** The main 8B DFlash-family acceptance-length comparisons and per-prompt analyses use RTX A6000 evaluations; the 4B comparison uses H100, as does the separate EAGLE-3 comparison for both drafter families. An evaluation is bitwise repeatable for the same checkpoint

and GPU model, including across hosts, but bf16 near-ties can change the greedy verification path across GPU models. Evaluating identical 8B checkpoints on A100 and RTX A6000 gives relative MAT shifts from $-1.66\%$ to $+0.77\%$ (mean $-0.21\%$, sd 0.46%), which informs the comparison band below.

**Replicate variance.** The comparison rule uses a pooled seed standard deviation of 0.35%, based on the estimate of 0.349% from four RTX A6000 settings. Across all settings evaluated on RTX A6000, the pooled standard deviation is 0.355%, and the standard deviation across the two data orders is 0.16%.

**Comparison rule.** We compare relative MAT differences between DFlash-family vision-language methods using a prespecified rule. A tolerance of 1.66% corresponds to the largest observed cross-device MAT shift with identical checkpoints. The lower and upper thresholds lie two standard deviations of the difference below and above this tolerance. They are 0.96% and 2.36% with two training replicates for each method, and 1.09% and 2.23% with three. An improvement is supported above the upper threshold. Differences between the thresholds are inconclusive, and those below the lower threshold do not meet the improvement criterion. For non-inferiority of a cheaper setting, we reflect the thresholds below zero. With three training replicates for each method, a difference above $-1.09\%$ meets the non-inferiority criterion, one below $-2.23\%$ does not, and intermediate values remain inconclusive.

**Prompt resampling.** Tables 1 and 2 provide the drafter results and erase references used in our prompt-resampling analysis. We resampled the 99 scored prompts of each domain with replacement 10,000 times, applying each resample to both arms and every seed. The 95% intervals for ALR + IRA's relative gain over erase are 7.71–9.36% on ALLaVA, 4.39–5.79% on ShareGPT4V and 6.36–7.91% with the smaller target. All three intervals remain above zero.

## F Training Schedules and Rollout Depths

Table 12 reports the epoch and envelope sweep, grouping methods by training corpus and epoch count. Table 13 holds the epoch count fixed within each group to isolate ALR's rollout depth. Table 3 summarizes its three-epoch aggregate results. These sweeps select the erase references and the short-rollout ALR setting in Table 2. The three-epoch results in Table 1 instead keep $R = 14$ for both ALR and ALR + IRA. ALLaVA times use a standalone reference, with the short-rollout ALR time calibrated against a rollout-depth control measured under the same load. ShareGPT4V ratios use numerator and denominator measurements from runs with the same node-sharing conditions.

Table 2 reports the one-epoch comparison and the best erase references. The best erase schedule uses three epochs on ALLaVA and one on ShareGPT4V, although differences among erase schedules remain below the preregistered resolution threshold. Figure 3 extends the ALLaVA comparison across training epochs.

## G Replicates and Robustness

**Training seeds.** For the primary vision-language target, the DFlash-family ALLaVA settings use three training replicates, except for erase at six epochs, erase with $\gamma = 0.5$ in Table 12 and both regeneration references in Table 4, which use two. Every ShareGPT4V setting, smaller-target setting and text-only method also uses two replicates. Within each setting, the seeds vary training randomness, including anchor sampling, from a common initialization. Corpus selection and data order remain fixed across these replicates. Table 1 reports means. Tables 2, 3, 8, 9, 12 and 13 also give sample standard deviations of acceptance length.

**Common-seed check.** Table 14 shows that restricting the ALLaVA comparisons to their two common training replicates changes ALR + IRA's gain over the highest measured erase schedule by about 0.1 percentage points at $T{=}0$. The depth comparison still identifies $R = 4$ as sufficient and $R = 2$ as insufficient, so neither conclusion depends on unequal replicate counts.

Table 12: Mean acceptance length $\tau$ under different training durations at $T$=0. Rollouts use $R = 14$. All rows use $\gamma = 2$ unless stated otherwise. Subscripts show training-seed standard deviations, and Avg. is the geometric mean across domains.

| Method | Epochs | COCO | TextVQA | DocVQA | Avg. |
|---|---|---|---|---|---|
| *ALLaVA* | | | | | |
| Erase | 1 | $3.106_{\pm 0.014}$ | $3.381_{\pm 0.028}$ | $4.197_{\pm 0.036}$ | $3.533_{\pm 0.006}$ |
| ALR | 1 | $3.199_{\pm 0.006}$ | $3.593_{\pm 0.027}$ | $4.255_{\pm 0.020}$ | $3.657_{\pm 0.017}$ |
| ALR + IRA | 1 | $3.263_{\pm 0.011}$ | $3.732_{\pm 0.006}$ | $4.254_{\pm 0.030}$ | $3.728_{\pm 0.005}$ |
| Erase | 3 | $3.124_{\pm 0.006}$ | $3.375_{\pm 0.005}$ | $4.204_{\pm 0.037}$ | $3.539_{\pm 0.010}$ |
| Erase ($\gamma$=0.5) | 3 | $3.090_{\pm 0.009}$ | $3.341_{\pm 0.004}$ | $4.115_{\pm 0.016}$ | $3.489_{\pm 0.006}$ |
| ALR | 3 | $3.242_{\pm 0.007}$ | $3.655_{\pm 0.024}$ | $4.261_{\pm 0.016}$ | $3.696_{\pm 0.005}$ |
| ALR + IRA | 3 | $3.376_{\pm 0.007}$ | $3.851_{\pm 0.011}$ | $4.361_{\pm 0.065}$ | $3.841_{\pm 0.021}$ |
| Erase | 6 | $3.083_{\pm 0.010}$ | $3.283_{\pm 0.003}$ | $4.134_{\pm 0.034}$ | $3.472_{\pm 0.014}$ |
| *ShareGPT4V* | | | | | |
| Erase | 0.5 | $3.302_{\pm 0.009}$ | $3.408_{\pm 0.014}$ | $4.151_{\pm 0.029}$ | $3.601_{\pm 0.017}$ |
| Erase | 1 | $3.362_{\pm 0.018}$ | $3.409_{\pm 0.004}$ | $4.208_{\pm 0.013}$ | $3.640_{\pm 0.002}$ |
| ALR + IRA | 1 | $3.658_{\pm 0.012}$ | $3.542_{\pm 0.032}$ | $4.163_{\pm 0.014}$ | $3.778_{\pm 0.003}$ |
| Erase | 3 | $3.403_{\pm 0.010}$ | $3.326_{\pm 0.023}$ | $4.023_{\pm 0.036}$ | $3.571_{\pm 0.016}$ |
| ALR | 3 | $3.614_{\pm 0.003}$ | $3.398_{\pm 0.009}$ | $4.035_{\pm 0.017}$ | $3.673_{\pm 0.007}$ |
| ALR + IRA | 3 | $3.817_{\pm 0.024}$ | $3.600_{\pm 0.024}$ | $4.073_{\pm 0.040}$ | $3.825_{\pm 0.029}$ |

Table 13: Mean acceptance length $\tau$ for ALR at different rollout depths $R$, with $T$=0. Subscripts show training-seed standard deviations, and Avg. is the geometric mean across domains.

| Method | COCO | TextVQA | DocVQA | Avg. |
|---|---|---|---|---|
| *ALLaVA, one epoch* | | | | |
| ALR | | | | |
| $R$=14 | $3.199_{\pm 0.006}$ | $3.593_{\pm 0.027}$ | $4.255_{\pm 0.020}$ | $3.657_{\pm 0.017}$ |
| $R$=6 | $3.194_{\pm 0.009}$ | $3.587_{\pm 0.022}$ | $4.250_{\pm 0.019}$ | $3.652_{\pm 0.001}$ |
| $R$=4 | $3.195_{\pm 0.006}$ | $3.591_{\pm 0.028}$ | $4.236_{\pm 0.018}$ | $3.649_{\pm 0.014}$ |
| *ALLaVA, three epochs* | | | | |
| ALR | | | | |
| $R$=14 | $3.242_{\pm 0.007}$ | $3.655_{\pm 0.024}$ | $4.261_{\pm 0.016}$ | $3.696_{\pm 0.005}$ |
| $R$=6 | $3.230_{\pm 0.010}$ | $3.629_{\pm 0.018}$ | $4.258_{\pm 0.008}$ | $3.682_{\pm 0.004}$ |
| $R$=4 | $3.232_{\pm 0.016}$ | $3.606_{\pm 0.039}$ | $4.211_{\pm 0.063}$ | $3.661_{\pm 0.023}$ |
| $R$=2 | $3.205_{\pm 0.002}$ | $3.489_{\pm 0.026}$ | $4.161_{\pm 0.032}$ | $3.597_{\pm 0.017}$ |
| *ShareGPT4V, one epoch* | | | | |
| ALR | | | | |
| $R$=4 | $3.524_{\pm 0.005}$ | $3.460_{\pm 0.020}$ | $4.139_{\pm 0.019}$ | $3.695_{\pm 0.011}$ |
| *ShareGPT4V, three epochs* | | | | |
| ALR | | | | |
| $R$=14 | $3.614_{\pm 0.003}$ | $3.398_{\pm 0.009}$ | $4.035_{\pm 0.017}$ | $3.673_{\pm 0.007}$ |
| $R$=4 | $3.606_{\pm 0.005}$ | $3.384_{\pm 0.019}$ | $4.016_{\pm 0.019}$ | $3.659_{\pm 0.014}$ |

**Data order, sampling and decoder.** Figure 2 compares greedy and sampled decoding using the same methods and benchmarks.

Table 15 shows that ALR's gain in accepted length over erase persists across data orders on ALLaVA and that sampling at $T$=1 preserves the ordering of erase, ALR and ALR + IRA. ALR + IRA remains 5.90% above erase when each block is verified as a single chain, showing that its improvement does not require tree branching. The mean gain across data orders and the gains under sampling and chain verification exceed the comparison thresholds in Appendix E. ALR + IRA also outperforms the slot-matched control using either all available replicates or the matched subset in Figure 4a.

**Displaced-block control.** Figure 4a compares ALR + IRA with a control that retains corpus anchors for the blocks IRA would move into target rollouts. The control matches IRA's block count,

Table 14: Relative MAT differences on ALLaVA at $T{=}0$ using all available training replicates or two matched replicates. Erase uses its highest measured schedule. Rollout depths are compared after three epochs.

| Comparison | All seeds (%) | Common seeds (%) |
|---|---|---|
| ALR + IRA, 1 epoch vs. erase | +5.33 | +5.24 |
| ALR + IRA, 3 epochs vs. erase | +8.54 | +8.64 |
| ALR, $R{=}6$ vs. $R{=}14$ | −0.37 | −0.31 |
| ALR, $R{=}4$ vs. $R{=}14$ | −0.93 | −0.81 |
| ALR, $R{=}2$ vs. $R{=}14$ | −2.68 | −2.41 |

Table 15: Relative MAT difference $\Delta$ on ALLaVA from erase outside the main data order, under sampling and with chain decoding, and from a slot-matched control. Sampling uses common random numbers for both arms. Blue shading marks ALR + IRA.

| Setting | Seeds per arm | $\Delta$ MAT (%) |
|---|---|---|
| *ALR, greedy decoding, other data orders* | | |
| Data-order replicate 1 | 1 | +3.99 |
| Data-order replicate 2 | 1 | +4.37 |
| Mean of the two orders | 1 per order | +4.18 |
| *Sampling at $T{=}1$ on the main data order* | | |
| ALR | 3 | +3.23 |
| ALR + IRA | 3 | **+7.72** |
| *Chain decoding of all 15 slots at $T{=}0$* | | |
| ALR + IRA | 3 | **+5.90** |
| *Against the slot-matched control at $T{=}0$* | | |
| ALR + IRA | 3 and 2 | **+3.92** |

supervised offsets and slot weights, yet its accepted length remains close to ALR. The contrast tests the joint change in context and continuation labels introduced by IRA.

**Choice of erase comparator.** Table 1 uses the soft survival weight of PARD-2 as the erase comparator. With the same KD loss and training conditions at $\gamma = 2$, the hard gate in Eq. (2) scores 0.62% lower, and removing the gate scores 4.85% lower. These checks make soft-gated distillation a strong reference for evaluating relabelling.

**Difficulty stratification.** Figure 4b–c stratifies drafting offsets in held-out captions by the acceptance length of the compared methods' common parent, a distillation drafter trained on the other corpus. These captions are disjoint from the main evaluation. At first-token failures, IRA adds about 0.39 accepted tokens at each offset on both corpora, exceeding the gain from relabelling alone.

**Regeneration comparison.** Table 4a first takes the geometric mean of MAT across COCO, TextVQA and DocVQA within each training replicate, then averages across replicates. The position profiles in Table 4b use one checkpoint of each method and group COCO verification rounds by their starting token offset.

**Position profile.** Table 4b shows that IRA's gain over ALR is larger in the later COCO caption bins for drafters trained on ALLaVA. A seed control shows no such rise. Scoring every drafter on one fixed text preserves this pattern: the mean gain of the last three bins exceeds the first by 3.47 percentage points, compared with 3.45 in the original scoring. For drafters trained on ShareGPT4V, IRA's gain is already large in the first bin, so we report this positional pattern for ALLaVA training.

**Regeneration gap.** Table 4b also shows that the truncated regeneration reference's advantage over ALR grows after the first 16 caption tokens and then levels off. The first-bin gap remains below its 3.83% comparison threshold. This pattern motivates exposing the drafter to recent target-generated context through IRA.

## H SPEED MEASUREMENT

Table 16 reports timing results for the trained drafters. The speedup of a domain is the ratio of autoregressive to drafter milliseconds per token in the same run, and the geometric mean over the three domains gives the speedup of a row. AR timing excludes the first target token, and drafter timing excludes the first draft construction. Both timers divide elapsed decoding time by the number of returned output tokens.

The additional vision-language timings in Table 1 use the same paired AR protocol, with concurrent evaluators on separate H100 GPUs. These measurements also include sampled decoding. The smaller-target ALR speedups average two timing runs, one per training seed.

Table 16: Decoding speedup over autoregressive decoding on one H100, by domain and as the geometric mean. Runs gives the timing runs averaged, with the standard deviation of their speedups in parentheses, and MAT carries its seed count $n$. Blue shading marks ALR + IRA.

| | | Speedup per domain (×) | | | | |
|---|---|---|---|---|---|---|
| Row | Speedup (×) | Caption | TextVQA | DocVQA | Runs | MAT ($n$) |
| *Trained on the fixed ALLaVA corpus* | | | | | | |
| DFlash, iso-data | 2.09 | 1.86 | 1.99 | 2.46 | 1 | 2.878 (3) |
| Erase (PARD-2 soft survival weight) | 2.62 | 2.28 | 2.55 | 3.10 | 3 (0.03) | 3.539 (3) |
| ALR | 2.68 | 2.22 | 2.77 | 3.13 | 1 | 3.696 (3) |
| ALR, rollout depth $R$=4, one epoch | 2.77 | 2.34 | 2.71 | 3.37 | 1 | 3.649 (3) |
| ALR + IRA, one epoch | 2.69 | 2.27 | 2.75 | 3.11 | 1 | 3.728 (3) |
| ALR + IRA | 2.82 | 2.46 | 2.95 | 3.08 | 4 (0.03) | 3.841 (3) |
| *Trained on target-regenerated ALLaVA responses* | | | | | | |
| Regeneration, truncated | 2.89 | 2.49 | 3.01 | 3.23 | 1 | 3.833 (2) |
| *Trained on the fixed ShareGPT4V corpus* | | | | | | |
| Erase, one epoch | 2.66 | 2.46 | 2.55 | 3.01 | 1 | 3.640 (2) |
| ALR + IRA | 2.77 | 2.72 | 2.67 | 2.93 | 1 | 3.825 (2) |

**Repeated timing.** Table 16 averages three paired timing runs for ALLaVA erase and four for ALR + IRA. ALR + IRA's averaged advantage over erase is 0.20×, with a 95% Welch interval of 0.13–0.26×. Repeating ALR + IRA at the start and end of the first session gave a 0.62% drift, below the prespecified 3% threshold. Autoregressive timing had a coefficient of variation of 3.4–5.6% across the three domains in that session, so we base close comparisons on accepted length rather than small speed differences.

## I H100 COMPARISON WITH EAGLE-3

Table 17 compares the ALLaVA drafters with the same Qwen3-VL-8B target on H100. EAGLE-3 (HF) uses the official dynamic-tree proposal (Li et al., 2025) in our Hugging Face adapter, with 60 packed nodes including the anchor, depth 7 and top-$K$ 10 fixed before evaluation. Both families use the common target runtime and, at $T$=1, target sampling with common random numbers at each output position. Each method uses one checkpoint and one evaluation of each domain, with 100 prompts, one excluded warmup and a 256-token output limit.

Table 17 uses the paired AR reference and the decoding window defined in Appendix H, excluding the first draft construction. This comparison uses a separate H100 timing session from the main results. Including the first draft construction gives EAGLE-3 speedups of 1.26× at $T$=0 and 1.23× at $T$=1. These H100 acceptance lengths are separate from the A6000 measurements in the main table.

## J LIMITATION

Our implementation trains feature-conditioned block drafters and requires access to the target's hidden states and token distributions. Its rollouts are greedy, while the resulting drafter supports both

Table 17: Observed H100 decoding speedup over AR and mean acceptance length $\tau$ for Qwen3-VL-8B drafters trained on ALLaVA. Values are geometric means across COCO, TextVQA and DocVQA. EAGLE-3 (HF) denotes our Hugging Face adapter.

| | $T$=0 | | $T$=1 | |
|---|---|---|---|---|
| Method | Speedup | $\tau$ | Speedup | $\tau$ |
| EAGLE-3 (HF) | 1.27× | 2.32 | 1.24× | 2.29 |
| DFlash | 2.09× | 2.87 | 2.04× | 2.78 |
| Erase | 2.65× | 3.54 | 2.49× | 3.38 |
| ALR | 2.68× | 3.69 | 2.55× | 3.47 |
| ALR + IRA | **2.85×** | **3.84** | **2.68×** | **3.63** |

greedy and sampled decoding. Constructing rollouts for the sampling policy used at inference is a natural extension.