Verification-Aware Training for Speculative Decoding

Geonmo Gu[1,3], **Byeongho Heo**[1], **HeeJae Jun**[2], **Yoohoon Kang**[2],
Sangmin Lee[3], **Sangdoo Yun**[1], **Dongyoon Han**[1]
[1]NAVER AI Lab [2]NAVER AI Search Platform [3]Korea University

Abstract

Speculative decoding accelerates large language model inference by using a draft model to generate candidate tokens, which are verified by the target model in a single forward pass. Verification proceeds sequentially and discards every position from the first rejection onward, yet existing draft training relies on token-level imitation of the target with a fixed per-position weighting that reflects neither property. We introduce Verification-Aware Training (VAT), a plug-in framework that simulates verification at every training step and turns the resulting accept and reject patterns into supervision. VAT consists of two components: (i) a verification head, a lightweight jointly trained binary classifier that supervises the draft model on whether each position survives sequential verification; (ii) verification-adaptive weighting, which replaces the fixed weighting schedule by keeping full weight up to each sample's first rejection point and re-anchoring the decay to start there. VAT modifies only the training objective, so it can be layered on top of existing methods without changing the draft architecture, the target model, or the inference procedure. Applied to EAGLE-3 and DFlash on Qwen3-4B, Qwen3-8B, and LLaMA-3.1-8B, VAT improves average acceptance length by up to 11.4% and wall-clock speedup by up to 8.7%, with consistent gains across math, code, and chat benchmarks. Code is available at https://github.com/naver-ai/VAT.

1 Introduction

Large Language Models (LLMs) [8, 1, 35, 16, 29, 39] generate tokens autoregressively, requiring a full forward pass of the model for every generated token. As models scale to hundreds of billions of parameters and reasoning techniques such as chain-of-thought [36] further lengthen outputs, inference latency has become a critical bottleneck in practical deployment. Speculative decoding [23, 10] alleviates this bottleneck by splitting generation into two stages, drafting and verification. A lightweight draft model first proposes several candidate tokens at low cost, and the target model then verifies all candidates in a single forward pass. Verification proceeds sequentially from the first draft position under an acceptance rule that preserves the target distribution, and once a token is rejected, all subsequent candidates are discarded, and a new drafting round begins from the last accepted position. Because the accepted output follows the target distribution, speculative decoding achieves *lossless* acceleration and has become one of the most widely adopted inference acceleration techniques.

Recent advances have substantially pushed the frontier of speculative decoding. EAGLE [25] and its extensions [26, 27] reuse hidden states of the target model to improve draft quality, and EAGLE-3 further improves draft training by exposing the draft model to multi-step draft-generated contexts. More recently, DFlash [11] replaces autoregressive drafting with a block diffusion model [4] that generates all draft tokens in parallel. While differing in their drafting architectures, both directions train the draft model under the same paradigm of imitating the target model's outputs.

Despite these advances, the training of draft models remains misaligned with the verification process that ultimately determines speedup. Two properties of verification are central to this misalignment.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

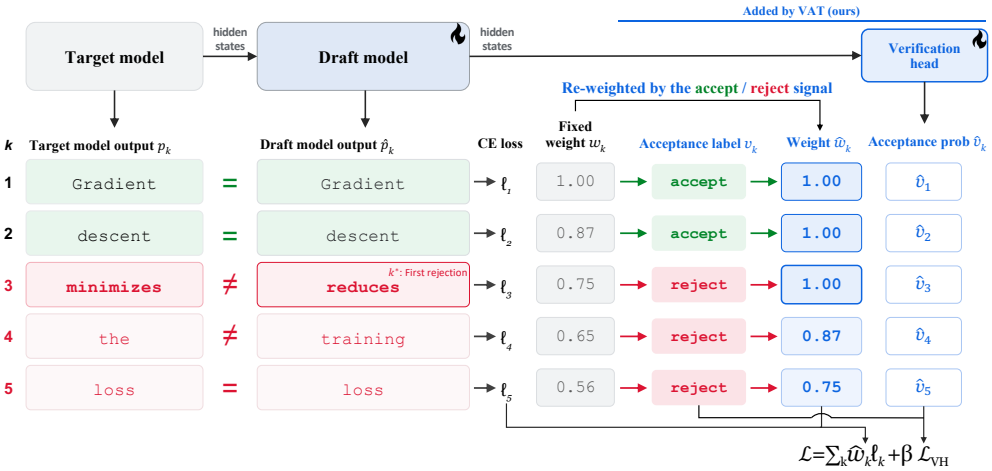


Figure 1: **Overview of Verification-Aware Training.** At each training step, VAT simulates target verification by comparing the target and draft predictions at every position k, yielding acceptance labels v_k. In this example, the first mismatch occurs at $k^* = 3$, so all subsequent positions are labeled `reject`. The labels drive two components: (i) verification-adaptive weighting, which replaces the fixed schedule w_k with $\hat{w}_k$ by keeping full weight up to k^* and re-anchoring the base decay to start at k^*, and (ii) a lightweight verification head on top of the draft's hidden states, trained to predict per-position acceptance $\hat{v}_k$. The final objective combines the reweighted cross-entropy losses with the verification head loss $\mathcal{L}_{\text{VH}}$.

First, the speedup of speculative decoding is governed by how many draft tokens pass the target model's verification, namely the acceptance length. Second, verification proceeds sequentially, and once a rejection occurs at any position, all subsequent tokens are discarded regardless of their individual quality, so each draft token contributes to the acceptance length only when all of its preceding positions are accepted. Existing training objectives, however, reflect neither of these properties. The draft model is trained simply to imitate the target model's outputs as per-position labels, receiving no signal about whether its tokens would actually survive verification. Although per-position loss weighting has been adopted to emphasize earlier draft positions, the schedule is fixed in advance and shared across all samples, and thus cannot adapt to where the first rejection occurs in each sample. These observations suggest two principles for verification-aligned draft training. First, training should go beyond imitating target outputs and incorporate verification signals from the target model. Second, the per-position training signal should adapt to each sample's verification outcome, *i.e.*, the position of its first rejection.

In this paper, we propose Verification-Aware Training (VAT), a simple yet effective plug-in framework that aligns draft training with the target's verification, as shown in Fig. 1. VAT simulates the target's verification at every training step and realizes the two principles above through two coordinated components. For the first principle, we propose a Verification Head, a lightweight classifier that predicts whether each draft token will be accepted by the target. During training, we simulate verification by comparing the draft and target predictions at each position, and train the head against the resulting accept/reject labels. Since the head is built on top of the draft's hidden states, this auxiliary objective backpropagates through the draft and shapes its representations toward features that determine agreement with the target. This auxiliary signal complements the standard token-level prediction by reflecting verification signals.

For the second principle, we introduce Verification-Adaptive Weighting, a per-position weighting scheme that adapts to each sample's verification pattern. As positions beyond the first rejection are discarded at inference, they should receive a reduced, though not eliminated, learning signal. To this end, we identify the first rejection point from the simulated verification and progressively decay the weights of subsequent positions, while keeping full weight on the positions that precede it. Since the rejection point varies across samples, the resulting schedule is sample-adaptive, unlike the fixed schedules of existing methods, and concentrates learning on the positions that determine acceptance length.

We evaluate VAT on two state-of-the-art methods spanning both drafting paradigms, the autoregressive EAGLE-3 and the diffusion-based DFlash, with three target models, Qwen3-4B, Qwen3-8B, and

LLaMA-3.1-8B. Across math, code, and chat benchmarks [14, 28, 12, 5, 22, 41, 34], VAT consistently improves both average acceptance length and wall-clock speedup on every combination of baseline and target model, improving acceptance length by up to 11.4% and speedup by up to 8.7%. For instance, on Qwen3-4B, VAT raises the speedup of EAGLE-3 from $4.07\times$ to $4.39\times$ (**+7.9%**) and that of DFlash from $4.54\times$ to $4.81\times$ (**+5.9%**), with corresponding gains in average acceptance length from 6.28 to 6.78 (**+8.0%**) and from 5.73 to 6.08 (**+6.1%**). These consistent gains across target models and drafting paradigms indicate that verification-aware training yields benefits independent of the underlying drafting mechanism.

2 Related Work

Speculative Decoding. Speculative decoding [33, 23, 10, 18] accelerates LLM inference by drafting candidate tokens with a lightweight model and verifying them against the target distribution in a single forward pass. Early formulations [23, 10] rely on a separately trained draft model from the same family as the target, which entails substantial engineering effort to obtain and align a compatible drafter [25, 9]. To eliminate the dependence on a separate model, Medusa [9] attaches parallel decoding heads on top of the target's hidden states, and Hydra [3] adds sequential dependence among these heads. Building on this direction, EAGLE [25] replaces parallel heads with a lightweight autoregressive drafter that operates at the feature level, with EAGLE-2 [26] adding a dynamic draft tree and EAGLE-3 [27] introducing multi-layer feature fusion and a training-time test that exposes the drafter to its own rollouts. A more recent line departs from autoregressive drafting altogether and adopts diffusion-based generation [13, 24, 32], producing all draft tokens in a single forward pass. DFlash [11] represents the current frontier of this paradigm, training a block diffusion drafter conditioned on hidden states injected from the target. Across both autoregressive and diffusion-based directions, draft model design has been the principal axis of innovation, while the training objective has remained largely standardized to token-level cross-entropy with sample-agnostic positional weighting.

Training Objectives for Draft Models. Beyond architectural design, a smaller line of work studies the training objective for the draft model. Knowledge distillation between draft and target distributions has been examined as a means to improve drafting quality [42, 17, 30, 31]. To address the train-test mismatch introduced by multi-step drafting, HASS [40] and the training-time test of EAGLE-3 [27] expose the drafter to its own rollouts during training, although both retain a uniform per-position loss. More directly relevant to the verification step, several recent works incorporate accept/reject information into the drafter, either as an inference-time signal for adaptive draft length [21, 6, 15] or as a training-time signal that down-weights or masks the loss after rejection [7, 20], whereas VAT retains a decayed learning signal beyond the first rejection while concentrating learning on the prefix that determines acceptance length. Concurrent to our work, PARD-2 [2] and D-PACE [37] also replace the fixed positional weighting with adaptive per-position weights, each within its own parallel drafting framework. Specifically, PARD-2 reweights each position by the target's cumulative confidence over the preceding prefix, while D-PACE derives per-position weights from a differentiable surrogate of expected acceptance length based on the draft's own confidences. In contrast, VAT conditions the weighting on the observed first-rejection position rather than a confidence-based proxy, couples it with a verification head supervising the cumulative acceptance outcome, and applies to both the autoregressive EAGLE-3 and the diffusion-based DFlash.

3 Preliminaries

Drafting and Verification. Speculative decoding [23, 10] accelerates LLM inference by alternating between *drafting* and *verification*. Given a prefix, a draft model $\mathcal{M}_d$ takes hidden states extracted from the target model $\mathcal{M}_t$ at the prefix positions as conditioning input and generates K candidate tokens, which $\mathcal{M}_t$ evaluates in a single forward pass. Verification proceeds sequentially from the first candidate under an acceptance rule that preserves the target distribution [23]. Once the first rejection occurs, all subsequent candidates are discarded regardless of their individual quality, and a new drafting cycle begins from the last accepted position. The expected number of tokens accepted per verification cycle, referred to as the average acceptance length τ, is the key performance metric of a speculative decoding system. Since each verification cycle incurs a fixed cost, a larger τ produces more tokens per cycle and results in higher speedup.

Draft Model Training. The draft model is trained to predict K future tokens at each step, with each position supervised against the target model. Two recent state-of-the-art methods, the autoregressive EAGLE-3 [27] and the diffusion-based DFlash [11], differ in their drafting mechanisms but share a common form of training objective, which can be written as:

$$\mathcal{L}_{\text{draft}} = \sum_{k=1}^{K} w_k \, \ell_k, \tag{1}$$

where ℓ_k is the cross-entropy loss between the draft model's prediction and the target model's output at position k, and w_k is a per-position loss weight. For ℓ_k, EAGLE-3 uses the target's output distribution as a soft label, while DFlash uses the target's sampled token as a hard label. For w_k, both methods adopt a predetermined position-dependent schedule, with EAGLE-3 using $w_k = 0.8^{k-1}$ and DFlash using $w_k = \exp(-(k-1)/\gamma)$, giving larger weight to earlier positions under the intuition that earlier tokens are more likely to be accepted.

Neither element of this objective reflects the verification process described above. Regarding ℓ_k, the draft model is trained simply to imitate the target model's outputs as per-position labels, receiving no signal about whether each token would actually survive sequential verification, where the contribution of position k is conditioned on the acceptance of all preceding positions. Regarding w_k, the schedule is fixed in advance and shared across all samples, so it cannot adapt to where the first rejection occurs in each sample and continues to discount positions that would contribute fully whenever the rejection occurs late. In the next section, we introduce VAT, which addresses the former by supervising simulated verification outcomes through a verification head and the latter by anchoring the per-position weights to each sample's first rejection point.

4 Method: Verification-Aware Training

This section introduces Verification-Aware Training (VAT), a simple method that can be integrated with any speculative decoding method. First, §4.1 introduces the concept of verification-aware training. We present a lightweight head to the draft model and jointly train it with it (§4.2). Finally, §4.3 and §4.4 describe our weighting and loss designs based on Eq. (1). VAT simulates target verification at training time and uses the resulting accept/reject patterns as additional supervision.

4.1 Verification at Training Time

During training, the draft and target models produce distributions $\hat{p}_k$ and p_k at every draft position k, which allows VAT to simulate the verification step of speculative decoding at every training step. Following speculative sampling [23, 10], a token x drafted at position k is accepted by the target with probability $\min(1, p_k(x)/\hat{p}_k(x))$, and we define the per-position acceptance indicator as

$$m_k = \mathbb{1}[\text{the drafted token at position } k \text{ is accepted}]\,. \tag{2}$$

Following the sequential acceptance described in §3, the first rejection point is

$$k^* = \min\{k : m_k = 0\}, \tag{3}$$

with the convention $k^* = K + 1$ when all positions are accepted. The acceptance label at position k is then

$$v_k = \mathbb{1}[k < k^*] = \prod_{j \le k} m_j, \tag{4}$$

which is one only if every preceding position is accepted. Unlike m_k, which is determined at each position independently, the label v_k reflects the actual outcome of sequential verification at inference, where a single rejection invalidates all subsequent tokens. VAT turns these simulated outcomes into two forms of supervision. A verification head trains the draft model to predict v_k directly (§4.2), and a verification-adaptive weighting anchors the per-position loss weights at k^* (§4.3).

4.2 Verification Head

The first component of VAT is a verification head, a lightweight binary classifier attached on top of the draft model's last hidden states and trained jointly with the draft model. The head serves purely

as a training-time auxiliary objective, leaving the drafting and verification procedures at inference unchanged without any additional cost.

Why Is the Verification Head Necessary? The training objective in Eq. (1) pushes the draft model to match the target's predictive distribution at each position, but the signal that actually determines the acceptance length is the acceptance label v_k, which depends on the outcomes of all preceding positions. Under the training objective, the draft model receives no gradient that distinguishes two qualitatively different cases at position k: (1) a position whose preceding tokens have all been accepted, so that a correct prediction extends the accepted prefix, and (2) a position whose prefix has already been invalidated by an earlier rejection, so that its prediction cannot affect the acceptance length. The draft model is thus trained through local next-token supervision alone, without any signal about whether its predictions would survive sequential verification. The verification head bridges this gap by supervising the draft model directly on v_k.

Design and Training. The head is a single dense layer that maps each position's hidden state to a predicted acceptance probability $\hat{v}_k$, and is trained with binary cross-entropy against the acceptance labels:

$$\mathcal{L}_{\text{VH}} = -\frac{1}{K}\sum_{k=1}^{K}\left[v_k \log \hat{v}_k + (1 - v_k)\log(1 - \hat{v}_k)\right]. \tag{5}$$

Since v_k assigns zero to every position beyond the first rejection, this supervision matches the conditions under which acceptance is decided at inference. Because the head is trained jointly with the draft model, gradients from $\mathcal{L}_{\text{VH}}$ flow back through the draft model and shape its hidden states toward features that determine agreement with the target, an aspect that the training objective in Eq. (1) leaves implicit. Although the head is not used at inference in our main experiments, its predicted $\hat{v}_k$ can additionally support early-exit drafting, which we explore in §5.3.

4.3 Verification-Adaptive Weighting

The second component of VAT revisits the per-position weight w_k in Eq. (1). To allocate the learning signal in line with each sample's verification outcome, we replace the predetermined schedule with an instance-adaptive schedule conditioned on k^*:

$$\hat{w}_k = \begin{cases} 1 & \text{if } k < k^*, \\ w_{k-k^*+1} & \text{if } k \geq k^*, \end{cases} \tag{6}$$

where w_k denotes the base method's predetermined schedule (*e.g.*, $w_k = 0.8^{k-1}$ for EAGLE-3 and $w_k = \exp(-(k-1)/\gamma)$ for DFlash). Positions before the first rejection receive full weight, since these tokens contribute to the acceptance length on this sample. Beyond k^*, the same decay curve is reused but shifted to start at k^* instead of $k = 1$, so the decay applies only to positions that follow the first rejection. As $w_1 = 1$ for both base schedules, k^* itself also receives full weight.

This design is motivated by aligning the decay anchor with each sample's verification pattern. Verification discards every position after the first rejection, so position k's contribution to acceptance length is conditional on all earlier positions being accepted. For a given sample, this contribution remains full up to k^* and only drops beyond it, meaning the natural anchor for the decay is k^* rather than $k = 1$. Because k^* varies across samples, this anchor is sample-specific. The predetermined schedule instead decays from $k = 1$ regardless of k^*, which discounts positions that still contribute fully and misaligns the decay curve from where each sample's contribution actually breaks down. The first rejection position itself also receives full weight, since it is the nearest correctable failure: verification fails exactly there, and an improved prediction at this position directly extends the accepted prefix. We validate this design against alternative weighting schemes in Appendix A.

4.4 Training Objective

We use both soft-label and hard-label cross-entropy losses as the per-position training objectives with the dynamic per-position weights $\hat{w}_k$ in Eq. (6). This follows the standard objective in knowledge distillation [19], combining the dense signal of the soft label with direct supervision of the hard label. Empirically, this combination yields longer acceptance length than either loss alone as shown in

Table 2. The full objective combines the reweighted draft loss with the verification head loss,

$$\mathcal{L} = \sum_{k=1}^{K} \hat{w}_k \left(\ell_k^{\text{soft}} + \ell_k^{\text{hard}} \right) + \beta \, \mathcal{L}_{\text{VH}}, \tag{7}$$

where ℓ_k^{soft} is the cross-entropy against the target's output distribution, ℓ_k^{hard} is the cross-entropy against the target's sampled token, and $\beta > 0$ balances the two terms. In this work, we set β as 1.0 for all experiments. VAT thus modifies only the training-side objective, leaving the draft architecture, the target model, and the inference procedure unchanged, so it can be layered on top of existing speculative decoding methods.

5 Experiments

5.1 Experimental Setup

We evaluate VAT on top of EAGLE-3 [27] and DFlash [11] using three target models: Qwen3 (4B, 8B) [39], and LLaMA-3.1-Instruct-8B [16]. For each baseline, we follow the training hyperparameters reported in the original papers [27, 11]. For example, we set $\gamma = 7$ for the DFlash weighting schedule. We evaluate on three task categories: math (GSM8K [14], MATH-500 [28], AIME25), code (HumanEval [12], MBPP [5], LiveCodeBench (LCB) [22]), and chat (MT-Bench [41], Alpaca [34]). Following prior works [27, 11], we construct a training set by pairing Perfectblend [38] user prompts with responses generated by the target model for better target alignment, and train each draft model for 3 epochs on this dataset. The responses are generated with greedy decoding, and verification during training is simulated under the same decoding scheme. Results with a training set generated at temperature $= 1$ are provided in §5.3.5. For fair comparison, both the baselines and their VAT counterparts are trained on this same corpus. For evaluation, we report the average acceptance length τ and the decoding speedup over the autoregressive baseline. All training and evaluation runs use NVIDIA A100 80GB GPUs with bf16 precision. We measure speedup and acceptance length τ with Hugging Face's Transformers library using the public evaluation codes[1].

5.2 Main Results

Table 1 reports decoding speedup and average acceptance length on Qwen3-4B, Qwen3-8B, and LLaMA-3.1-8B across math, code, and chat benchmarks. VAT improves both metrics on top of EAGLE-3 and DFlash across all three target models. The gains are most apparent in the average acceptance length τ, which directly reflects how many tokens the draft contributes per verification cycle. Averaged over the eight benchmarks, EAGLE-3 + VAT improves τ by 8.0% on Qwen3-4B, 5.7% on Qwen3-8B, and 2.5% on LLaMA-3.1-8B, and DFlash + VAT improves τ by 6.1%, 11.4%, and 3.8% on the same three models. Wall-clock speedup follows the same trend, with average gains of 7.9%, 5.0%, and 3.8% over EAGLE-3, and 5.9%, 8.7%, and 3.4% over DFlash. A similar pattern holds at decoding temperature $= 1$, with consistent improvements in both τ and speedup across all three target models for both baselines. The gains hold across three target models, two model families, and both drafting paradigms, indicating that VAT is not tied to a specific target or speculative decoding method.

5.3 Empirical Analysis

All ablation studies in this section are conducted on top of DFlash with Qwen3-4B, and evaluated with the temperature set to 0 unless otherwise stated.

5.3.1 Effect of Components

Table 2 ablates the verification head, verification-adaptive weighting, and the joint use of soft and hard labels in Eq. (7) on top of DFlash with Qwen3-4B. Each factor improves the average acceptance length on its own, raising τ from 5.73 to 5.87 with the verification head, 5.91 with verification-adaptive weighting, and 5.82 with the soft + hard labels. Since DFlash originally trains with hard labels alone, the gain from adding the soft label suggests that aligning the draft with the target's output distribution,

[1] EAGLE-3: https://github.com/SafeAILab/EAGLE, DFlash: https://github.com/z-lab/dflash

Table 1: **Decoding speedup over the autoregressive baseline and average acceptance length (τ)**, evaluated with up to 2048 generated tokens. Temperature $= 0$ and $= 1$ denote the decoding temperature used at evaluation. All results are measured on A100 GPUs using the HuggingFace Transformers library. Boldface indicates the best result in each setting.

Model	Method	MATH						CODE						CHAT				*Avg.*	
		GSM8K		MATH-500		AIME25		HumanEval		MBPP		LCB		MT-Bench		Alpaca			
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Temperature $= 0$																			
Q3-4B	EAGLE-3	4.94×	7.62	5.00×	7.55	4.46×	6.79	4.33×	6.67	3.99×	6.18	3.75×	5.91	3.35×	5.20	2.74×	4.28	4.07×	6.28
	+ VAT	**5.31×**	**8.25**	**5.23×**	**7.91**	**4.61×**	**7.01**	**4.83×**	**7.37**	**4.57×**	**7.12**	**4.11×**	**6.47**	**3.52×**	**5.41**	**2.95×**	**4.67**	**4.39×**	**6.78**
	DFlash	6.69×	8.28	5.93×	7.46	5.05×	6.21	4.58×	5.57	4.30×	5.23	4.67×	5.86	2.82×	4.20	2.27×	3.04	4.54×	5.73
	+ VAT	**7.01×**	**8.66**	**6.31×**	**7.91**	**5.19×**	**6.44**	**4.89×**	**5.97**	**4.67×**	**5.71**	**4.92×**	**6.19**	**3.03×**	**4.53**	**2.43×**	**3.26**	**4.81×**	**6.08**
Q3-8B	EAGLE-3	4.95×	7.54	5.07×	7.52	4.42×	6.69	4.30×	6.47	3.90×	5.95	3.66×	5.66	3.31×	5.04	2.67×	4.12	4.04×	6.12
	+ VAT	**4.97×**	**7.68**	**5.14×**	**7.73**	**4.66×**	**6.98**	**4.62×**	**6.95**	**4.08×**	**6.25**	**4.02×**	**6.26**	**3.47×**	**5.23**	**2.94×**	**4.68**	**4.24×**	**6.47**
	DFlash	6.21×	7.45	5.96×	7.28	4.97×	6.01	4.67×	5.59	4.10×	4.86	4.43×	5.51	3.03×	4.24	2.41×	3.15	4.47×	5.51
	+ VAT	**7.03×**	**8.71**	**6.43×**	**8.14**	**5.16×**	**6.40**	**5.06×**	**6.22**	**4.62×**	**5.69**	**4.88×**	**6.27**	**3.14×**	**4.46**	**2.55×**	**3.26**	**4.86×**	**6.14**
L3.1-8B	EAGLE-3	4.39×	6.48	4.53×	6.53	4.44×	6.33	4.45×	6.51	4.52×	6.50	3.88×	5.86	3.56×	5.18	3.55×	5.28	4.17×	6.08
	+ VAT	**4.51×**	**6.54**	**4.69×**	**6.65**	**4.59×**	**6.45**	**4.59×**	**6.59**	**4.69×**	**6.61**	**4.02×**	**6.01**	**3.79×**	**5.42**	**3.79×**	**5.57**	**4.33×**	**6.23**
	DFlash	4.75×	6.30	4.78×	6.41	4.45×	6.06	4.84×	6.37	4.63×	6.08	4.33×	5.77	2.75×	4.33	2.13×	3.26	4.08×	5.57
	+ VAT	**4.97×**	**6.66**	**4.99×**	**6.76**	**4.56×**	**6.19**	**4.96×**	**6.56**	**4.75×**	**6.27**	**4.43×**	**5.94**	**2.79×**	**4.48**	**2.27×**	**3.35**	**4.22×**	**5.78**
Temperature $= 1$																			
Q3-4B	EAGLE-3	4.57×	7.15	4.48×	6.95	3.69×	5.68	4.05×	6.27	3.69×	5.76	3.60×	5.71	3.05×	4.75	2.48×	3.99	3.70×	5.78
	+ VAT	**4.95×**	**7.85**	**4.59×**	**7.20**	**3.85×**	**5.96**	**4.48×**	**6.99**	**4.31×**	**6.78**	**3.96×**	**6.41**	**3.17×**	**5.00**	**2.78×**	**4.58**	**4.01×**	**6.35**
	DFlash	5.71×	7.20	4.82×	6.33	3.39×	4.29	4.19×	5.19	3.95×	4.88	4.15×	5.28	2.62×	3.87	2.17×	2.89	3.88×	4.99
	+ VAT	**6.17×**	**7.72**	**5.11×**	**6.58**	**3.50×**	**4.45**	**4.48×**	**5.47**	**4.37×**	**5.37**	**4.47×**	**5.64**	**2.80×**	**4.11**	**2.30×**	**3.06**	**4.15×**	**5.30**
Q3-8B	EAGLE-3	4.53×	6.99	4.44×	6.80	3.47×	5.32	4.00×	6.14	3.67×	5.64	3.40×	5.35	2.97×	4.59	2.41×	3.84	3.61×	5.58
	+ VAT	**4.67×**	**7.36**	**4.55×**	**7.05**	**3.74×**	**5.76**	**4.17×**	**6.51**	**3.84×**	**6.03**	**3.68×**	**5.88**	**3.15×**	**4.90**	**2.54×**	**4.09**	**3.79×**	**5.95**
	DFlash	5.10×	6.37	4.64×	5.93	3.39×	4.32	3.98×	4.91	3.63×	4.44	4.05×	5.12	**2.72×**	**3.87**	2.27×	3.05	3.72×	4.75
	+ VAT	**5.84×**	**7.40**	**5.02×**	**6.49**	**3.56×**	**4.54**	**4.18×**	**5.15**	**4.05×**	**4.98**	**4.60×**	**5.92**	2.63×	3.76	**2.35×**	**3.15**	**4.03×**	**5.17**
L3.1-8B	EAGLE-3	3.74×	5.60	3.21×	4.71	2.00×	2.93	4.00×	5.97	3.97×	5.91	3.45×	5.32	2.68×	4.00	3.10×	4.69	3.27×	4.89
	+ VAT	**3.80×**	**5.79**	**3.33×**	**4.98**	**2.49×**	**3.74**	**4.03×**	**6.10**	**4.05×**	**6.09**	**3.58×**	**5.58**	**2.84×**	**4.32**	**3.21×**	**4.98**	**3.42×**	**5.20**
	DFlash	**3.32×**	**4.87**	2.15×	**3.48**	1.23×	1.73	3.65×	4.94	3.54×	4.71	3.03×	4.16	1.78×	2.87	1.83×	2.66	2.57×	3.68
	+ VAT	3.30×	4.64	**2.20×**	**3.48**	**1.38×**	**1.97**	**3.73×**	**4.96**	**3.61×**	**4.84**	**3.15×**	**4.26**	**1.85×**	**3.01**	**1.86×**	**2.69**	**2.64×**	**3.73**

Table 2: **Factor analysis**. We ablate the verification head, verification-adaptive weight, and the use of both soft and hard labels in Eq. (7), where unchecked denotes training with hard labels only. Results are averaged over math (GSM8K, MATH-500, AIME25), code (HumanEval, MBPP, LCB), and chat (MT-Bench, Alpaca). The best result in each column is in bold.

Verification head	Verification-adaptive weight	Soft + hard labels	Math		Code		Chat		*Avg.*	
			Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
			5.89×	7.32	4.52×	5.55	2.55×	3.62	4.54×	5.73
✓			5.99×	7.53	4.57×	5.64	2.63×	3.73	4.62×	5.87
	✓		5.97×	7.46	4.67×	5.82	2.64×	3.72	4.65×	5.91
		✓	5.93×	7.40	4.54×	5.63	2.63×	3.74	4.58×	5.82
	✓	✓	6.02×	7.55	4.69×	5.90	2.69×	3.87	4.69×	6.05
✓		✓	6.09×	7.63	4.66×	5.77	2.71×	3.88	4.72×	5.99
✓	✓		6.12×	7.61	4.75×	5.84	2.70×	3.79	4.76×	5.99
✓	✓	✓	**6.17×**	**7.67**	**4.83×**	**5.96**	**2.73×**	**3.90**	**4.81×**	**6.08**

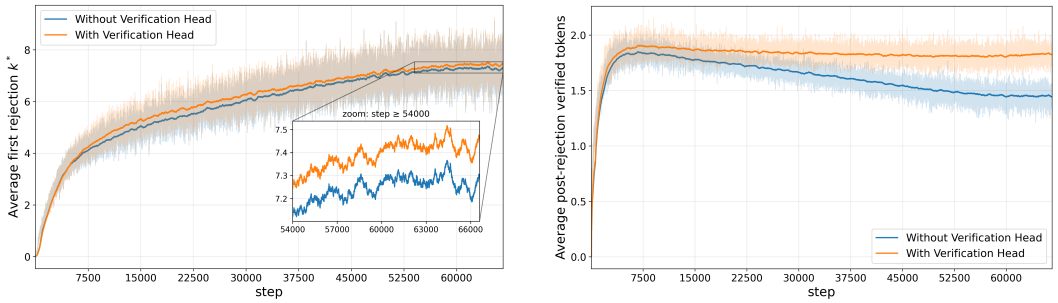
beyond matching its generated token, further helps the draft agree with the target. Combining any two factors compounds these gains, and the verification head with verification-adaptive weighting achieves the highest speedup among the pairs ($4.76\times$), recovering most of the total gain with hard labels alone. Combining all three yields the best average τ (6.08) and the best speedup ($4.81\times$) and attains the best results in all three task categories. Explicit supervision through the verification head and per-position reweighting anchored at each sample's first rejection address different aspects of the verification process, and the gains compound when applied together.

5.3.2 Ablation on Verification-adaptive Weighting

Table 3 decomposes our weighting design into two factors: the choice of base weight w_k and whether the weighting is made verification-adaptive. The results show two consistent patterns. First, uniform weighting and the EAGLE-3-style predefined decay 0.8^{k-1} produce nearly identical performance

Table 3: **Ablation on Verification-adaptive weighting.** We compare different per-token weighting strategies w_k. Results are averaged over math (GSM8K, MATH-500, AIME25), code (HumanEval, MBPP, LCB), and chat (MT-Bench, Alpaca). The best result in each column is in bold.

Schedule	Base w_k	Math		Code		Chat		*Avg.*	
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
Uniform	1.0	5.81×	7.30	4.45×	5.52	2.52×	3.62	4.48×	5.72
Predefined weight	0.8^{k-1} [27]	5.81×	7.28	4.51×	5.56	2.57×	3.65	4.51×	5.73
	$\exp(-(k-1)/\gamma)$ [11]	6.09×	7.63	4.66×	5.77	2.71×	3.88	4.72×	5.99
Verification-adaptive weight	0.8^{k-1} [27]	6.04×	**7.70**	4.73×	5.94	2.68×	**3.90**	4.71×	**6.09**
	$\exp(-(k-1)/\gamma)$ [11]	**6.17**×	7.67	**4.83**×	**5.96**	**2.73**×	**3.90**	**4.81**×	6.08



(a) First rejection position

(b) Post first-rejection matched tokens

Figure 2: **Training dynamics with and without the verification head.** We track two quantities under simulated verification, averaged over the batch at each training step. (a) The first rejection position k^*. With the verification head, k^* shifts to later positions, meaning that more consecutive draft tokens pass verification. (b) The number of post-first-rejection tokens that still match the target. The baseline gradually declines as training proceeds, while the verification-head variant remains stable, despite its post first-rejection region beginning at deeper positions on average. Curves are smoothed with a trailing moving average over a 200-step window, and the shaded thin line shows the raw per-step values.

(Avg. τ of 5.72 and 5.73), while the DFlash-style predefined decay $\exp(-(k-1)/\gamma)$ alone improves τ to 5.99. Second, verification-adaptive weighting achieves the best results regardless of the base weight, reaching $\tau = 6.09$ with the EAGLE-3-style base and $\tau = 6.08$ with the DFlash-style base, with a negligible difference between the two, indicating that the gain is driven primarily by the adaptation mechanism rather than by the functional form of the underlying decay. Unlike a predefined schedule that applies the same coefficient to every sample, verification-adaptive weighting concentrates the loss on positions that contribute to each sample's acceptance length. Since the rejection point varies across samples, this sample-adaptive behavior cannot be approximated by any sample-agnostic schedule.

5.3.3 Training Dynamics of Verification Head

To examine how the verification head shapes draft behavior during training, we track two quantities across training steps in Figure 2. The first rejection consistently occurs at later positions when the verification head is attached, as shown in Figure 2(a). Since the head supervises whether each position survives sequential verification, its gradient flows back through the draft and steers the hidden states toward features that determine acceptance. More consecutive draft tokens therefore pass verification before the first mismatch, which directly translates into longer verified prefixes at inference time, since speedup is governed by acceptance length.

A less obvious pattern emerges among post-rejection tokens, as shown in Figure 2(b). Both variants reach a similar peak in match count early in training, after which the baseline gradually declines while the verification-head variant remains stable, even though its post-rejection tokens begin at deeper positions on average (Figure 2(a)), where the context is harder to predict. Since the head does not alter the cross-entropy term that drives token-level predictions but instead encourages hidden states that are informative about the verification outcome, the two signals are not in conflict, and the

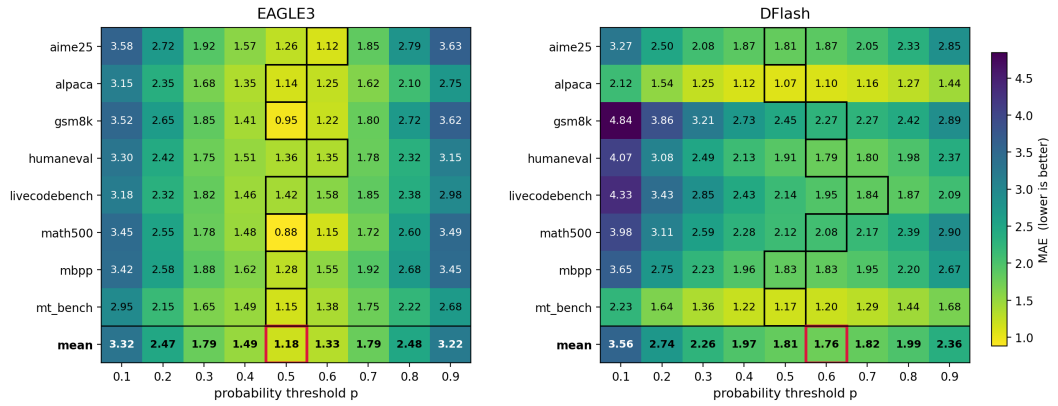


Figure 3: **First-rejection prediction error of the verification head.** Mean absolute error (MAE) between the first-rejection position obtained from target-model verification, k^*, and the position predicted by the verification head, as the threshold p varies from 0.1 to 0.9 on EAGLE-3 (left) and DFlash (right). Per-token acceptance probabilities are thresholded at p to locate the first predicted rejection. The bottom row reports the mean across eight benchmarks. Black outlines mark per-benchmark minima, and red outlines mark the threshold that minimizes the overall mean (p=0.5 for EAGLE-3, p=0.6 for DFlash).

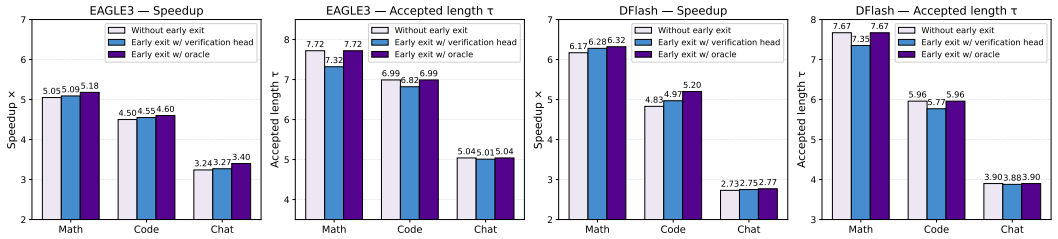


Figure 4: **Early-exit drafting guided by the verification head.** Speedup and mean acceptance length τ on Math, Code, and Chat for Qwen3-4B with VAT applied to EAGLE-3 and DFlash. "Early exit w/ verification head" truncates the draft at the predicted first-rejection position from the head's per-token probabilities, sending only the predicted-accept prefix to the target model. "Early exit w/ oracle" uses the true first-rejection position and serves as an upper bound. Early exit with the verification head improves speedup over drafting without early exit, with a small drop in τ.

induced representations appear to support better generalization across positions, including those past the first rejection.

5.3.4 Verification Head at Inference

The verification head is trained as an auxiliary objective and is not required at inference, but it can serve as a cheap proxy for the target's verification outcome to reduce inference cost. Figure 3 sweeps the decision threshold p and reports the mean absolute error (MAE) between the head's predicted first-rejection position and the one obtained from the target. The mean MAE is minimized at p=0.5 for EAGLE-3 and p=0.6 for DFlash, with average errors of only 1.18 and 1.76 tokens, and we adopt these thresholds at inference. For DFlash, which drafts all 16 tokens in a single parallel forward pass, the head sends only the predicted-accept prefix to the target. For EAGLE-3, which drafts autoregressively, the head can additionally terminate drafting once a rejection is predicted, saving both drafting and verification compute.

Figure 4 compares early exit with the verification head against drafting without early exit and an oracle variant that uses the true first rejection point k^*. Early exit with the verification head recovers most of the oracle speedup, and the gains are larger on DFlash (*e.g.*, 4.83× → 4.97× on Code, oracle 5.20×) than on EAGLE-3, where autoregressive drafting already dominates the compute and verification savings have less headroom. The small drop in τ (*e.g.*, 7.67 → 7.35 on DFlash Math) reflects occasional false rejection predictions by the head. Overall speedup improves, since the compute saved by early exit outweighs the tokens lost to these false rejections.

Table 4: **Correlation between the two simulated verification rules.** Pearson correlation between the accepted lengths simulated under greedy (T=0) and sampling (T=1) verification on the same training batches. Each column denotes the verification rule used for training.

Training progress	Rule used for training	
	Greedy	Sampling
0–25%	0.93	0.93
25–50%	0.93	0.93
50–75%	0.92	0.92
75–100%	0.92	0.92

Table 5: **Effect of the simulated verification rule and corpus temperature.** DFlash + VAT on Qwen3-4B trained under each combination of corpus generation temperature and verification rule, evaluated at both T=0 and T=1. Results are reported as speedup / τ, averaged over all benchmarks.

Training corpus	Simulated verification	Evaluation	
		T=0	T=1
T=0	Greedy	4.81× / 6.08	4.15× / 5.30
T=0	Sampling	4.80× / 6.06	4.14× / 5.29
T=1	Greedy	4.81× / 6.10	4.15× / 5.31
T=1	Sampling	4.83× / 6.11	4.17× / 5.32

5.3.5 Effect of the Simulated Verification Rule and Corpus Temperature

All preceding experiments generate the training corpus with greedy decoding and simulate verification under the same decoding scheme (§5.1), where the acceptance rule in §4.1 reduces to top-1 agreement between $\hat{p}_k$ and p_k. This section examines whether VAT is sensitive to these choices by varying both the verification rule used during training and the temperature used to generate the corpus.

Table 4 first compares the two verification rules at the level of the training signal itself. Throughout training, the accepted lengths simulated under the two rules maintain a correlation of at least 0.92 regardless of which rule is used for training, indicating that both rules provide highly consistent information about where each draft block is first rejected. Table 5 then compares the downstream performance across all four combinations of corpus temperature and verification rule. The two rules differ by at most 0.02 in both speedup and τ regardless of the corpus and evaluation temperature, and regenerating the corpus at T=1 yields nearly identical results. Consistent with the signal-level agreement above, VAT is insensitive to both the verification rule simulated during training and the corpus temperature.

6 Conclusion

We presented Verification-Aware Training (VAT), a plug-in framework that aligns draft model training with the target model's verification process. VAT simulates verification at every training step and turns the resulting accept and reject patterns into supervision through two components. The verification head treats the simulated verification outcome as an explicit prediction target. It supervises the draft model on whether each position survives sequential verification. Verification-adaptive weighting keeps full weight up to each sample's first rejection point and decays the weights only after it. Both components modify only the training objective. VAT can therefore be layered on top of existing speculative decoding methods without changing the draft architecture, the target model, or the inference procedure. Although the verification head is not required at inference, its predictions can also guide early-exit drafting and yield additional speedup. When applied to EAGLE-3 and DFlash, VAT consistently improves both the average acceptance length and the wall-clock speedup across math, code, and chat benchmarks. Our current evaluation is limited to models with up to 8B parameters. Exploring whether verification-aware training scales to significantly larger models is a promising direction for future work.

Acknowledgments and Disclosure of Funding

We sincerely thank our colleagues at NAVER AI Lab for their valuable support and insightful feedback. This work was supported by NAVER Corporation through research funding and computing resources, and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (IITP-2026-RS-2026-25616664, AI Star Fellowship Support Program).

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

[2] 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. *arXiv preprint arXiv:2605.08632*, 2026. 3

[3] Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. In *Conference on Language Modeling (COLM)*, 2024. 3

[4] Marianne Arriola, Aaron Gokaslan, Justin T Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *The Thirteenth International Conference on Learning Representations*, 2025. 1

[5] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. 3, 6

[6] Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas K Kohler. Judge decoding: Faster speculative sampling requires going beyond model alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[7] Shrenik Bhansali and Larry Heck. Draft, verify, and improve: Toward training-aware speculative decoding. *arXiv preprint arXiv:2510.05421*, 2025. 3

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

[9] 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 *International Conference on Machine Learning (ICML)*, 2024. 3

[10] 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. 1, 3, 4

[11] Jian Chen, Yesheng Liang, and Zhijian Liu. DFlash: Block diffusion for flash speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. 1, 3, 4, 6, 8

[12] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 3, 6

[13] Jacob K. Christopher, Brian R. Bartoldson, Tal Ben-Nun, Michael Cardei, Bhavya Kailkhura, and Ferdinando Fioretto. Speculative diffusion decoding: Accelerating language generation through diffusion. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2025. 3

[14] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 3, 6

[15] Roman Garipov, Fedor Velikonivtsev, Ivan Ermakov, Ruslan Svirschevski, Vage Egiazarian, and Max Ryabinin. AutoJudge: Judge decoding without manual annotation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 3

[16] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 1, 6

[17] Lujun Gui, Bin Xiao, Lei Su, and Weipeng Chen. Boosting lossless speculative decoding via feature sampling and partial alignment distillation. *arXiv preprint arXiv:2408.15562*, 2024. 3

[18] Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D. 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 (NAACL)*, 2024. 3

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 5

[20] Shijing Hu, Jingyang Li, Xingyu Xie, Zhihui Lu, Kim-Chuan Toh, and Pan Zhou. GRIFFIN: Effective token alignment for faster speculative decoding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 3, 14

[21] Kaixuan Huang, Xudong Guo, and Mengdi Wang. SpecDec++: Boosting speculative decoding via adaptive candidate lengths. In *Conference on Language Modeling (COLM)*, 2025. 3

[22] 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. In *The Thirteenth International Conference on Learning Representations*, 2025. 3, 6

[23] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023. 1, 3, 4

[24] Guanghao Li, Zhihui Fu, Min Fang, Qibin Zhao, Ming Tang, Chun Yuan, and Jun Wang. DiffuSpec: Unlocking diffusion language models for speculative decoding. *arXiv preprint arXiv:2510.02358*, 2025. 3

[25] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Forty-first International Conference on Machine Learning*, 2024. 1, 3

[26] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 7421–7432, 2024. 1, 3

[27] 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. 1, 3, 4, 6, 8

[28] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024. 3, 6

[29] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. 1

[30] Xiaoxuan Liu, Lanxiang Hu, Peter Bailis, Alvin Cheung, Zhijie Deng, Ion Stoica, and Hao Zhang. Online speculative decoding. In *International Conference on Machine Learning (ICML)*, 2024. 3

[31] Alexander Samarin, Sergei Krutikov, Anton Shevtsov, Sergei Skvortsov, Filipp Fisin, and Alexander Golubev. LK losses: Direct acceptance rate optimization for speculative decoding. In *Forty-third International Conference on Machine Learning*, 2026. 3

[32] Jameson Sandler, Jacob K. Christopher, Thomas Hartvigsen, and Ferdinando Fioretto. SpecDiff-2: Scaling diffusion drafter alignment for faster speculative decoding. *arXiv preprint arXiv:2511.00606*, 2025. 3

[33] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 3

[34] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm.stanford.edu/2023/03/13/alpaca.html*, 3(6):7, 2023. 3, 6

[35] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

[36] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022. 1

[37] Tianyu Wu, Yu Yao, Zhenting Qi, Han Zheng, Zhuohan Wang, Haoran Ma, Lawrence Liao, Himabindu Lakkaraju, Ju Li, and Yilun Du. D-pace: Dynamic position-aware cross-entropy for parallel speculative drafting. *arXiv preprint arXiv:2605.18810*, 2026. 3, 14

[38] Tengyu Xu, Eryk Helenowski, Karthik Abinav Sankararaman, Di Jin, Kaiyan Peng, Eric Han, Shaoliang Nie, Chen Zhu, Hejia Zhang, Wenxuan Zhou, et al. The perfect blend: Redefining rlhf with mixture of judges. *arXiv preprint arXiv:2409.20370*, 2024. 6

[39] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 1, 6

[40] Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. Learning harmonized representations for speculative sampling. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[41] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023. 3, 6

[42] 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. 3

A Ablation on Weighting Schemes

This section validates the design of verification-adaptive weighting in Eq. (6) against alternative treatments of positions from the first rejection point k^* onward, as well as against two schemes that reweight all positions without conditioning on k^*, on top of DFlash with Qwen3-4B. All variants replace only the per-position weights $\hat{w}_k$ while keeping the rest of the VAT recipe, and use a 1-epoch training budget.

Table A: **Ablation on weighting schemes.** We compare verification-adaptive weighting in Eq. (6) against alternative treatments of positions from the first rejection point k^* onward, and against two schemes that reweight all positions without conditioning on k^*, on top of DFlash with Qwen3-4B under a 1-epoch training budget. Unless noted otherwise, $\hat{w}_k = 1$ for $k < k^*$. Results are averaged over math (GSM8K, MATH-500, AIME25), code (HumanEval, MBPP, LCB), and chat (MT-Bench, Alpaca). The best result in each column is in bold.

Weighting scheme	Weight assignment	Math		Code		Chat		*Avg.*	
		Speedup	τ	Speedup	τ	Speedup	τ	Speedup	τ
DFlash baseline	$\exp(-(k-1)/\gamma)$	5.51×	7.06	4.25×	5.37	2.45×	3.53	4.27×	5.54
Prefix-only	0 from k^*	3.10×	3.94	2.31×	2.89	1.66×	2.21	2.44×	3.11
Hard cutoff	1 at k^*, then 0	5.74×	7.30	4.41×	5.58	2.55×	3.68	4.44×	5.75
Unshifted decay	w_k at all k	5.79×	7.29	4.43×	5.60	2.52×	3.68	4.46×	5.75
Marginal contribution	soft, exact derivative	5.59×	7.21	4.39×	5.59	2.57×	3.75	4.38×	5.74
GRIFFIN-style masking [20]	0 on top-m mismatch	5.54×	7.15	4.32×	5.50	2.49×	3.61	4.32×	5.64
D-PACE-style weights [37]	confidence-based, all k	5.77×	7.41	4.53×	5.77	**2.62**×	**3.80**	4.52×	5.89
Verification-adaptive weight (Eq. (6))	w_{k-k^*+1} from k^*	**6.05**×	**7.67**	**4.57**×	**5.92**	2.53×	3.73	**4.61**×	**6.03**

Table A reports the results, where we refer to Eq. (6) as ours and cite average speedup and τ as a pair. Prefix-only assigns zero weight from k^* onward and collapses below the baseline, since a sample rejected at the first draft position contributes no draft loss, and Figure 2(a) shows k^* is small in early training when the gradient is needed most. Hard cutoff differs only by keeping full weight at k^* and recovers most of the gap ($4.44\times$ / 5.75 against $4.61\times$ / 6.03 for ours), and the residual margin indicates that positions beyond the first rejection carry useful learning signal once decayed rather than eliminated. Re-anchoring accounts for a comparable margin, as unshifted decay applies the base schedule from $k = 1$ and reaches $4.46\times$ / 5.75.

Two schemes derived from the draft's own confidences land closer: the marginal-contribution scheme weights each position by the exact derivative of the expected accepted length, and the probability that verification reaches each position, which appears as a factor in this derivative, likewise discounts k^* whenever the preceding prefix is uncertain ($4.38\times$ / 5.74), while the confidence-based weights of the concurrent D-PACE [37] reach $4.52\times$ / 5.89 within our framework. Masking on a per-position criterion, as in GRIFFIN [20], which zeroes the loss where the drafted token falls outside the target's top-m (m=3, its default), gives $4.32\times$ / 5.64. Across all six alternatives, whether or not they condition on the first rejection, anchoring the decay at the observed k^* gives the highest average speedup and τ.

B Training Overhead of VAT

Table B: **Training overhead of VAT.** Per-step training time and peak GPU memory with Qwen3-4B, measured over 100 steps on a single A100 80GB GPU.

Method	Time (s/step)	Peak memory (GB)
EAGLE-3	0.511	18.5
+ VAT	0.517	18.6
DFlash	1.044	23.8
+ VAT	1.108	31.5

VAT modifies only the training objective, and this section quantifies its training-time cost. We measure the per-step training time and peak GPU memory of EAGLE-3 and DFlash with and without VAT on Qwen3-4B, averaged over 100 steps on a single NVIDIA A100 80GB GPU. No verification labels or

target distributions are precomputed or cached. All quantities are computed online at each step with the frozen target model.

Table B reports the results. VAT adds 1.2% per-step time to EAGLE-3 and 6.1% to DFlash. The gap between the two baselines comes from where the target distributions required for simulating verification are obtained. EAGLE-3 already computes the target LM head distribution for its soft labels, so VAT reuses it and the remaining cost is only the lightweight verification head. DFlash never applies the target LM head in its original training, so simulating verification adds one LM head pass per step, which accounts for both the larger time increase and the higher peak memory. In both cases, the overhead applies only to training, and the inference procedure remains unchanged.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the two contributions of VAT (a verification head and verification-adaptive weighting) and the scope of evaluation. These claims are supported by the experimental results in Section 5 on EAGLE-3 and DFlash with three target models.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitation of our work is discussed at the end of Section 6 (Conclusion). Specifically, our empirical evaluation is restricted to target models up to 8B parameters, and we note that verifying the scalability of verification-aware training on significantly larger models remains an open direction for future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include formal theoretical results, theorems, or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 fully specifies the method, including the verification label (Eq. 4), the verification head (Eq. 5), the adaptive weighting (Eq. 6), and the full training objective (Eq. 7). Section 5.1 specifies the target models, baselines, and training data construction.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available at https://github.com/naver-ai/VAT.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training and evaluation details are provided in Section 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We do not report error bars. Instead, consistency is shown across two baselines, three target models, two temperatures, and eight benchmarks.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 5.1 reports NVIDIA A100 80GB GPUs with bf16 precision. Appendix B reports the training overhead of VAT.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: VAT is a lossless inference acceleration method that does not change the target model's output distribution.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release pretrained generative models or scraped datasets that pose misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All target models, baselines, and benchmarks are properly cited and used under their original licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new datasets are introduced.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human subjects research.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are the subject of the acceleration method, not a non-standard component of the core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.