DarterSD: Dynamic Token-Weighted Distillation for High-Efficient Speculative Decoding

Zhaoning Zhang*, **Baizhou Xu***, **Libo Zhang**†, **Yi Chen, Zhiliang Tian, Zhen Huang, Dongsheng Li**

PDL Lab
College of Computer Science and Technology
National University of Defense Technology, Changsha, China

{zhangzhaoning, xubaizhou23, zhanglibo, chenyi_, tianzhiliang, huangzhen, dsli}@nudt.edu.cn

Abstract

Speculative decoding accelerates autoregressive LLM generation with a lightweight draft model, yet existing distillation methods treat all historical tokens equally despite their divergent contributions to next-token prediction. We discover that attention in draft models is extremely concentrated: the top-3 historical tokens capture 80% of attention weight, while over 90% of tokens contribute negligibly. This uniform training paradigm forces the draft model to learn from noisy, low-importance tokens, diluting the learning signal from informative positions and limiting the alignment between draft and target distributions. To address this, we propose DarterSD, a dynamic token-weighted distillation framework for efficient speculative decoding. DarterSD employs a learnable sigmoid gate in self-attention to compute per-token importance scores, and designs a multi-objective distillation loss comprising distribution alignment, logit matching, and gate regularization to focus training on high-informative tokens. We further introduce a lightweight Boolean-mask denoising strategy at inference to suppress contextual noise, without modifying the target model's verification pipeline. Experiments show DarterSD delivers a lossless speedup up to 5.52×, outperforming EAGLE3 (4.67×), and boosts the average accepted token length by 12.9%.

1 Introduction

Large Language Models (LLMs) have achieved remarkable success across diverse tasks, yet their autoregressive decoding generate one token per forward pass remains a fundamental efficiency bottleneck. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) mitigates this by employing a lightweight draft model to propose candidate tokens that a target model verifies in parallel, substantially reducing inference latency while preserving output fidelity.

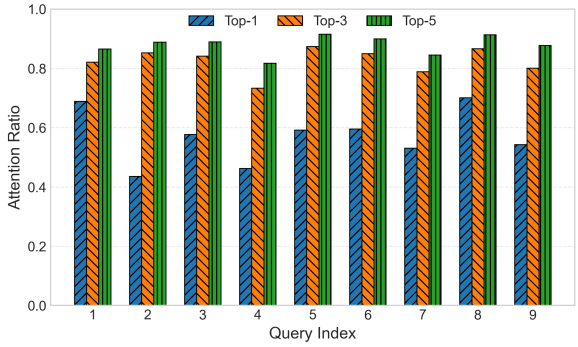


Figure 1: Top-k attention concentration at the first generation step. Top-k denotes the cumulative attention mass assigned to the k most attended historical tokens.

Recent work has advanced speculative decoding along several axes: draft architecture (feature-level prediction and dynamic draft trees, multi-head drafting) (Li et al., 2024b, 2025; Cai et al., 2024), draft-free approaches (self-speculative decoding) (Zhang et al., 2024; Elhoushi et al., 2024; Xia et al., 2025), speculation strategy (adaptive speculation length) (Huang et al., 2025; Liu et al., 2025), and training efficiency (fast draft-model training) (Zafrir et al., 2025). However, none of these methods explicitly models the varying importance of individual historical tokens. While H2O (Zhang et al., 2023), StreamingLLM (Xiao et al., 2024), Rhea (Hong et al., 2025), etc. identifie attention sparsity in the target model and proposes KV-cache eviction, we systematically quantify this phenomenon in the draft model of a speculative decoding pipeline and to exploit it for both training and inference optimization.These findings suggest that different historical tokens or context segments contribute unevenly to the generation process. Inspired by these observations, our analysis reveals striking attention concentration in draft-model self-attention: as shown in Figure 1, the top-3 historical

*These authors contributed equally to this work.
†Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 15773–15788
October 24-29, 2026 ©2026 Association for Computational Linguistics

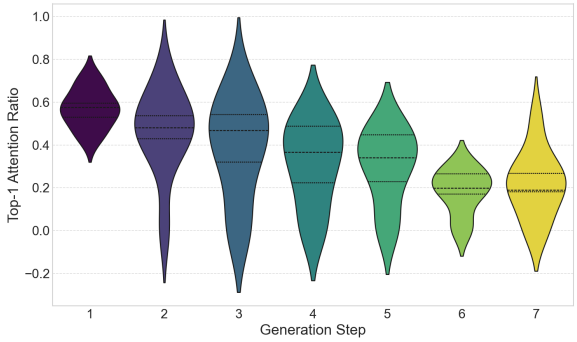


Figure 2: Distribution of Top-1 attention ratios across decoding steps.

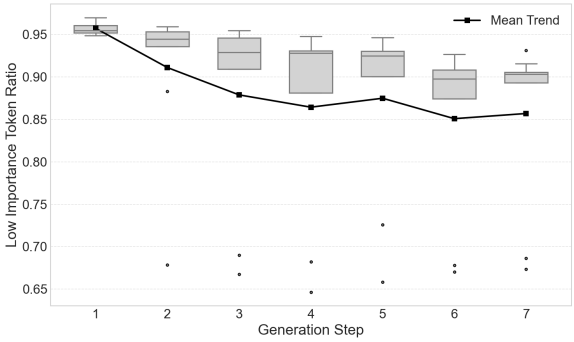


Figure 3: Proportion of low-importance tokens across decoding steps.

tokens account for 80% of the attention mass at a representative generation step, while Figure 3 shows that across decoding steps, over 90% of historical tokens receive attention weights below 0.003, confirming that the vast majority of tokens contribute negligibly to draft generation. This indicates that draft generation is dominated by a small subset of informative tokens, while the majority—receiving negligible attention—activateion as contextual noise that degrades candidate generation quality.

This token-importance imbalance primarily raises a critical challenge for draft model distillation: treating all tokens equally during training prevents the lightweight draft model from focusing its limited capacity on the most informative tokens. As a result, the learning signal of high-value positions is diluted, and the draft model fails to align well with the target model's output distribution, reducing the average accepted length. Besides, during inference, retaining all cached tokens without distinction allows low-importance historical tokens to introduce contextual noise into draft generation, which further limits acceleration performance.

We propose **DarterSD**, a dynamic token-weighted distillation framework built upon a unified token importance scoring mechanism. Centered on training-side distillation optimization, our framework enhances the distribution alignment between the draft model and the target model. During training, we embed a learnable sigmoid gating unit into the self-attention module of the draft model to generate per-token importance scores. Based on these scores, we construct a multi-objective distillation loss comprising distribution alignment, logit matching, and sparsity-inducing gate regularization, paired with adaptive loss weight averaging to balance all optimization objectives. This design directs the limited modeling capacity of the lightweight draft model toward high-informative tokens. During inference, these pre-computed scores can be reused to derive a Boolean mask via an adaptive threshold, which mitigates interference from low-importance tokens in attention computation. The original output distribution is fully preserved as we leave the target model's verification pipeline intact.

We evaluate DarterSD on Vicuna-13B-v1.3 (Chiang et al., 2023), Llama-3.1-8B-Instruct (Llama Team, 2024) and DeepSeek-R1-Distill-LLaMA-8B (DeepSeek-AI, 2025), across benchmarks including MT-Bench and GSM8K. For example, on Vicuna-13B-v1.3, DarterSD achieves up to speedup of $5.52\times$, improving over EAGLE3's $4.67\times$. Meanwhile, the average accepted length increases from 6.74 to 7.61, corresponding to a relative improvement of 12.9%.

Our main contributions are summarized as follows:

- We conduct systematic analysis of attention concentration in the draft model of speculative decoding, revealing that over 90% of historical tokens receive negligible attention weight, and quantifying its impact on both distillation quality and inference efficiency.

- We propose a token-weighted distillation method with learnable sigmoid gated attention for draft models. It focuses distillation on informative tokens during training, and reuses learned token-importance scores to construct an adaptive Boolean mask for inference-time contextual denoising.

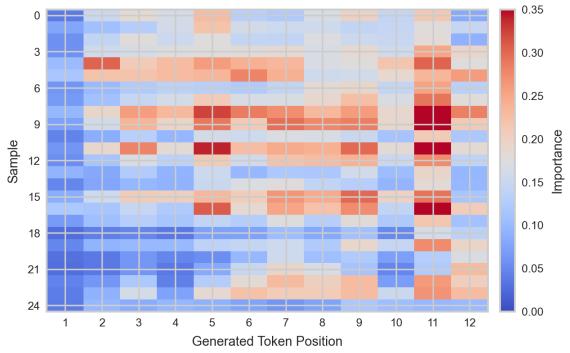


Figure 4: Heatmap of token-importance scores produced by the gated attention mechanism on MT-bench. Rows denote samples, columns denote generated token positions, and warmer colors indicate higher learned importance scores.

2 Motivation

This section provides the empirical motivation for DarterSD. We first analyze the self-attention distribution of the draft model and show that draft generation relies on a highly selective subset of historical tokens. We then verify whether such token-importance differences can be explicitly captured by a lightweight learnable gating mechanism. These two observations motivate the design of token-importance-aware training and inference in DarterSD.

2.1 Self-Attention Distribution Analysis

To examine whether draft generation depends uniformly on all historical tokens, we record the attention weights of the draft model during multiple forward steps before target-model verification. We define the Top-k attention concentration as the ratio between the sum of the largest k attention weights and the total attention mass:

$$C_q(k) = \frac{\sum_{j=1}^{k} \hat{a}_{q,j}}{\sum_{j=1}^{L} a_{q,j}}, \tag{1}$$

where $a_{q,j}$ denotes the normalized attention weight from query position q to key position j, $a_q = [a_{q,1}, a_{q,2}, \ldots, a_{q,L}]$ is the full attention vector, and $\hat{a}_{q,j}$ is the j-th element after sorting a_q in descending order. A larger value of $C_q(k)$ indicates that the attention distribution is more concentrated on the top-k historical tokens.

Figure 2 shows that the Top-1 attention ratio can be very high in several decoding steps, indicating that a single historical token may dominate the attention distribution. Figure 3 further shows that the median proportion of low-importance tokens is above 0.90 across decoding steps. These results demonstrate a clear token-importance imbalance: draft generation does not uniformly depend on all historical tokens, but instead selectively focuses on a small number of informative positions. This observation is consistent with prior studies (Liu et al., 2023; Zhang et al., 2023; Yuan et al., 2025) on attention sparsity and contextual sparsity in Transformer inference.

2.2 Learnable Token Importance Verification

The above analysis reveals the existence of token-importance imbalance. We further examine whether such differences can be explicitly captured by a lightweight learnable mechanism. We use the gated attention module in DarterSD to record token-importance scores during forward passes on MT-bench. As shown in Figure 4, the learned gate scores are not uniformly distributed: tokens with clearer semantic meanings tend to receive higher scores, while function words and low-information subword tokens receive lower scores. This verifies that token-importance differences are not only observable from attention distributions, but can also be explicitly modeled by a learnable gating mechanism. A visualization of the learned gate scores is provided in Appendix C.

3 Method

This section presents the proposed DarterSD framework. DarterSD introduces token-importance awareness into speculative decoding from two aspects: dynamic token-weighted distillation during training and Boolean-mask-based dynamic denoising during inference.

3.1 Overall Framework

We study speculative decoding with a lightweight draft model M_{draft} and a target model M_{target}. Given a context $x_{1:t}$, the draft model proposes candidate tokens, which are then verified by the target model in parallel. The target-model verification procedure is kept unchanged, so the final output distribution remains consistent with the target model.

Figure 5 compares the draft-model structures of EAGLE3 and DarterSD. Compared with EAGLE3, DarterSD introduces a gated attention module to estimate token-level importance and applies importance-aware denoising during draft generation. This design allows the draft model to suppress

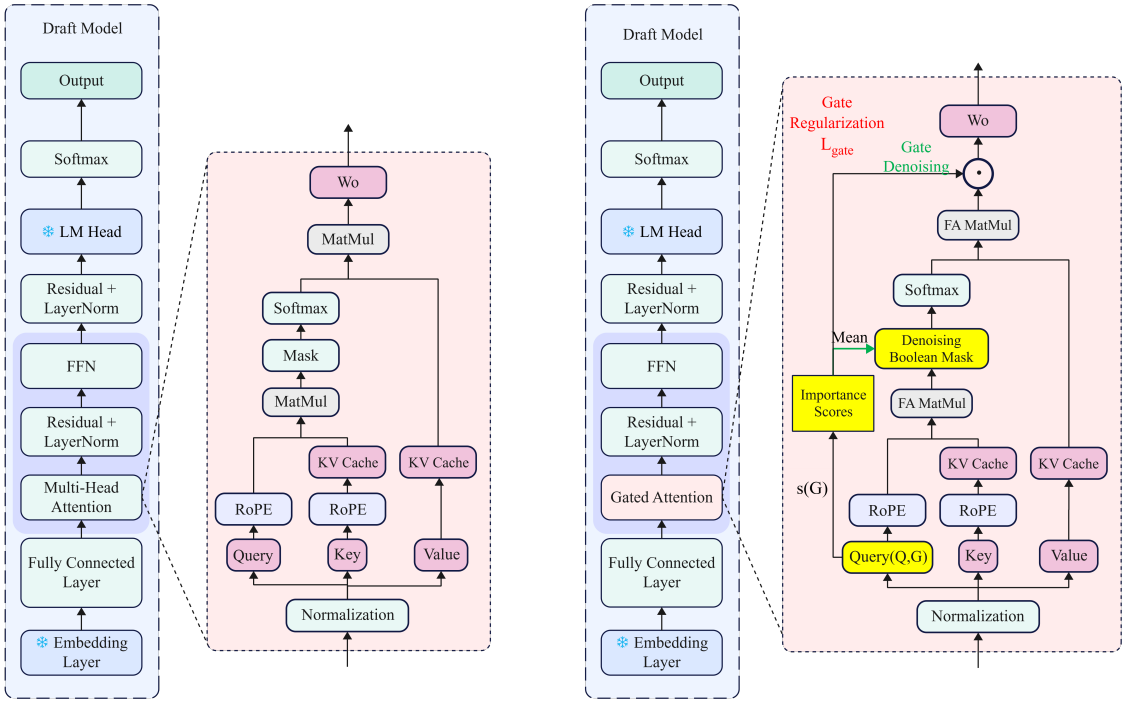


Figure 5: Comparison between EAGLE3 and DarterSD. Compared with EAGLE3, DarterSD introduces gated attention for token-importance estimation and applies importance-aware denoising during draft generation, while keeping target-model verification unchanged.

low-importance contextual tokens while keeping the target-model verification procedure unchanged.

3.2 Dynamic Token-Weighted Distillation

Dynamic token-weighted distillation aims to improve the alignment between the draft model and the target model by modeling the importance differences among tokens. This module consists of three components: a gated attention layer, a multi-objective distillation loss, and Dynamic Weight Averaging (DWA). These components work together to encourage the draft model to better learn informative token features while aligning its output distribution with that of the target model.

3.2.1 Gated Attention Layer

The gated attention layer is designed to estimate fine-grained token importance. Given fused hidden states $\mathbf{X} \in \mathbb{R}^{B\times L\times D}$, we expand the query projection to produce both query and gate representations:

$$[\mathbf{Q}, \mathbf{G}] = \text{split}(\text{Linear}_q(\mathbf{X})). \quad (2)$$

Here, $\mathbf{Q}$ is used for standard attention computation, while $\mathbf{G}$ is used to generate token-importance scores.

Let $\mathbf{O}_{\text{attn}}$ denote the standard multi-head attention output. We apply a sigmoid function to the gate representation and modulate the attention output by element-wise multiplication:

$$\mathbf{O}_{\text{out}} = \mathbf{O}_{\text{attn}} \odot \sigma(\mathbf{G}). \quad (3)$$

The gate scores $\sigma(\mathbf{G})$ are dynamically computed from the input context and can be interpreted as token-, head-, and dimension-level importance estimates. This design allows the draft model to suppress low-importance features while preserving informative contextual signals.

3.2.2 Dynamic Weight Averaging Based Multi-Objective Distillation

The draft model should approximate the target model well enough to produce candidates that are likely to be accepted. However, because the draft model is much smaller or shallower than the target model, uniform distillation over all contextual

features may dilute important signals. DarterSD therefore combines target-model alignment with gate-based token-importance modeling.

Let $\mathbf{p}_i^t$ and $\mathbf{p}_i^d$ denote the probability distributions produced by the target and draft models at position i, respectively. Let $\mathbf{z}_i^t$ and $\mathbf{z}_i^d$ denote their corresponding logits. We use three complementary objectives.

First, the KL divergence loss aligns the output distribution of the draft model with that of the target model:

$$\mathcal{L}_{\mathrm{KL}} = \frac{1}{N} \sum_{i=1}^{N} D_{\mathrm{KL}} \left(\mathbf{p}_i^t \,\|\, \mathbf{p}_i^d \right), \tag{4}$$

where $N = B \times L$ is the number of training tokens in a batch.

Second, the logits alignment loss directly matches the unnormalized output representations:

$$\mathcal{L}_{\mathrm{logits}} = \frac{1}{N} \sum_{i=1}^{N} \left\| \mathbf{z}_i^t - \mathbf{z}_i^d \right\|_2^2. \tag{5}$$

While the KL loss aligns normalized probability distributions, the logits loss preserves fine-grained preference information before softmax normalization. These two losses jointly encourage the draft model to approximate the target model at both distribution and representation levels.

Third, the gate regularization loss encourages the gated attention layer to form a sparse and discriminative importance pattern:

$$\mathcal{L}_{\mathrm{gate}} = \frac{1}{N_g} \sum_{j=1}^{N_g} \left| \sigma(g_j) \right|, \tag{6}$$

where N_g is the total number of gate elements and g_j is an element of the gate representation. This regularization does not aim to remove tokens blindly. Instead, it encourages the model to avoid uniformly high gate scores, so that informative contextual features can be distinguished from low-importance features.

The three objectives have different optimization dynamics. Distribution and logits alignment may converge at different speeds from gate regularization. To avoid manually tuning fixed loss weights, we adopt Dynamic Weight Averaging (DWA). Let $\mathcal{L}_k(e)$ denote the average value of the k-th loss at epoch e, where $k \in \{\mathrm{KL}, \mathrm{logits}, \mathrm{gate}\}$. The relative training rate of each objective is computed as

$$r_k(e) = \frac{\mathcal{L}_k(e)}{\mathcal{L}_k(e-1) + \epsilon}, \tag{7}$$

where ϵ is a small constant for numerical stability. The dynamic weight for each loss is

$$w_k(e) = m \cdot \frac{\exp\left(r_k(e)/T\right)}{\sum_{j=1}^{m} \exp\left(r_j(e)/T\right)}, \tag{8}$$

where $m = 3$ is the number of objectives and T is a temperature parameter. The final training loss is

$$\begin{aligned} \mathcal{L}_{\mathrm{total}} = {} & w_{\mathrm{KL}}(e)\mathcal{L}_{\mathrm{KL}} + w_{\mathrm{logits}}(e)\mathcal{L}_{\mathrm{logits}} \\ & + w_{\mathrm{gate}}(e)\mathcal{L}_{\mathrm{gate}}. \end{aligned} \tag{9}$$

Through this objective, DarterSD learns a draft model that is aligned with the target model while also acquiring an explicit notion of token importance.

3.3 Boolean-Mask-Based Dynamic Denoising

During speculative decoding, attention masks are used to constrain token dependencies and control the attention range. Standard self-attention usually relies on a fixed causal mask and treats all historical tokens equally. Although tree-based, phrase-based, and cascade-style speculative decoding methods (Li et al., 2024b; Cai et al., 2024; Miao et al., 2024; Zhao et al., 2024; Chen et al., 2024) introduce structured masks or longer drafts for parallel verification, they still lack an explicit distinction between high-importance and low-importance historical tokens. As many low-importance tokens contribute little to the current prediction, retaining all historical tokens without distinction may introduce contextual noise and degrade draft quality.

To further complement the training-side token-weighted distillation optimization and mitigate the contextual interference from trivial tokens during inference, DarterSD introduces a Boolean-mask-based dynamic denoising strategy. The strategy uses token-importance scores learned by the gated attention layer to construct a dynamic Boolean mask, which suppresses low-importance historical tokens in the attention computation of the draft model. Unlike methods that physically reorganize, page, or compress the KV cache (Kwon et al., 2023; Liu et al., 2024), Boolean masking only modifies the attention visibility pattern and does not change the tensor layout of the cache, thereby avoiding additional tensor reorganization and memory fragmentation.

In speculative decoding, the inference process of the draft model can be divided into two stages. The first stage is global history encoding, where the draft model encodes the prompt and the previously

generated context and stores the global historical features. The second stage is tree-based candidate generation, where the draft model expands candidate tokens based on the global historical features. Since the tree-based generation stage is usually shallow and its local dependencies are strong, DarterSD applies dynamic denoising only to the global history encoding stage. This design reduces historical contextual noise without breaking the local dependency structure of the draft tree.

Specifically, for each historical token j, we first aggregate its gate vector $\sigma(\mathbf{g}_j) \in (0,1)^{n_h \times d}$ into a scalar importance score:

$$s_j = \mathrm{Mean}(\sigma(\mathbf{g}_j)). \tag{10}$$

Given an importance threshold θ, we construct a dynamic Boolean mask. For query position i and historical key position j, the mask is defined as

$$\mathbf{M}^{\mathrm{keep}}_{i,j} = \begin{cases} 1, & s_j \geq \theta, \\ 0, & s_j < \theta. \end{cases} \tag{11}$$

A value of 1 indicates that the historical token is kept visible in draft attention, while 0 indicates that it is masked out.

The dynamic Boolean mask is then combined with the original causal mask to constrain the attention range of the draft model. If the importance score of a historical token is lower than the threshold, the corresponding attention position is masked out, reducing its interference with draft generation.

In the overall pipeline, the draft model first encodes the global contextual history and calculates token importance scores to construct the Boolean keep mask. During candidate token generation, this mask works jointly with the causal mask and tree mask, after which the target model performs parallel verification over all candidate outputs. Paired with the training-side token-importance distillation, our inference denoising mechanism further helps the lightweight draft model concentrate on informative historical tokens and reduce redundant contextual interference. Unlike approaches that physically prune the KV cache, our method realizes denoising via attention mask adjustment and incurs no extra overhead from tensor reorganization. As the proposed strategy only acts on the candidate generation phase of the draft model while keeping the verification pipeline of the target model intact, it fully preserves the original output distribution and generation fidelity of speculative decoding.

4 Experiments

4.1 Experimental Setup

Models and Implementation. We evaluate DarterSD on three target models: Vicuna-13B-v1.3 (Chiang et al., 2023), Llama-3.1-8B-Instruct (Llama Team, 2024), and DeepSeek-R1-Distill-LLaMA-8B (DeepSeek-AI, 2025). All models are loaded and evaluated using PyTorch (Paszke et al., 2019) and the Hugging Face Transformers library (Wolf et al., 2020). During training, we use one H800 GPU and optimize the draft model with AdamW (Loshchilov and Hutter, 2017). The learning rate is set to 5×10^{-5}, the weight decay is 0.01, the batch size is 8, and the model is trained for 30 epochs. During inference, we use four RTX 3090 GPUs. All methods are evaluated under the same decoding configuration, including greedy decoding, a tree depth of 8, 10 branches per layer, a maximum of 60 tokens in the draft tree, and an active set capacity of $K_{\max} = 1024$.

Datasets. We train the draft model using ShareGPT, UltraChat (Ding et al., 2023), and OpenThoughts-114k-math (Open-R1 Team, 2025). We use the full ShareGPT dataset and randomly sample 10,000 instances from UltraChat and 10,000 instances from OpenThoughts-114k-math for gated draft-model fine-tuning. For evaluation, we use GSM8K (Cobbe et al., 2021), CNN/DM (Nallapati et al., 2016), HumanEval (Chen et al., 2021), Alpaca (Taori et al., 2023), and MT-bench (Zheng et al., 2023), covering mathematical reasoning, summarization, code generation, instruction following, and multi-turn dialogue. For each benchmark, we randomly sample 80 examples to measure inference latency and draft-token acceptance. All methods are evaluated on the same sampled instances.

Evaluation Metrics. We use two main metrics to evaluate speculative decoding performance.

Speedup (sp) measures the ratio between the latency of target-only autoregressive decoding and that of speculative decoding when generating the same output sequence:

$$sp = \frac{T_{\mathrm{target}}}{T_{\mathrm{spec}}}, \tag{12}$$

where T_{target} denotes the generation time of the target model under standard autoregressive decoding,

Table 1: Main results across benchmarks. V 13B denotes Vicuna-13B-v1.3, L31 8B denotes Llama-3.1-8B-Instruct, and DSL 8B denotes DeepSeek-R1-Distill-LLaMA-8B. (SP denotes speedup and AAL denotes average accepted length.)

Model	Method	GSM8K		CNN/DM		HumanEval		Alpaca		MT-bench		Average	
		SP	AAL	SP	AAL	SP	AAL	SP	AAL	SP	AAL	SP	AAL
V 13B	EAGLE2	2.75×	4.58	1.97×	3.48	4.25×	5.76	2.69×	4.34	3.56×	5.04	3.04×	4.64
	HASS	3.88×	5.38	2.98×	4.65	4.41×	6.09	3.75×	5.28	3.68×	5.26	3.74×	5.33
	EAGLE3	4.60×	6.29	3.87×	6.52	5.80×	7.36	4.44×	6.93	4.65×	6.60	4.67×	6.74
	DarterSD	**5.99×**	**7.09**	**4.85×**	**7.32**	**6.42×**	**8.59**	**5.13×**	**7.69**	**5.19×**	**7.35**	**5.52×**	**7.61**
L31 8B	EAGLE2	2.75×	4.58	1.97×	3.48	2.79×	4.71	2.69×	4.34	2.51×	4.32	2.54×	4.29
	HASS	3.69×	5.11	2.76×	4.35	4.00×	5.70	3.60×	5.41	3.36×	4.80	3.48×	5.07
	EAGLE3	4.09×	6.18	3.39×	5.32	4.66×	6.42	4.21×	6.56	4.04×	6.02	4.08×	6.10
	DarterSD	**4.41×**	**6.77**	**3.61×**	**5.62**	**6.23×**	**7.35**	**4.68×**	**7.81**	**4.28×**	**6.66**	**4.64×**	**6.84**
DSL 8B	EAGLE3	4.01×	6.71	3.32×	4.92	4.09×	6.24	3.45×	5.32	3.05×	5.47	3.58×	5.73
	DarterSD	**4.85×**	**7.56**	**3.53×**	**5.10**	**4.45×**	**6.77**	**3.72×**	**5.59**	**3.98×**	**6.01**	**4.06×**	**6.21**

Table 2: Ablation results on Llama-3.1-8B-Instruct. Here, SP denotes speedup and AAL denotes average accepted length.

Method	GSM8K		CNN/DM		HumanEval		Alpaca		MT-bench		Average	
	SP	AAL	SP	AAL	SP	AAL	SP	AAL	SP	AAL	SP	AAL
DarterSD	**4.41×**	**6.77**	**3.61×**	**5.62**	**6.23×**	**7.35**	**4.68×**	**7.81**	**4.28×**	**6.66**	**4.64×**	**6.84**
DarterSD w/o FlashAttn	4.31×	6.77	3.29×	5.62	5.85×	7.35	4.57×	7.81	4.24×	6.66	4.49×	6.84
DarterSD w/o Denoising	4.23×	6.57	3.07×	5.51	6.19×	6.99	4.26×	7.54	4.17×	6.38	4.35×	6.60
DarterSD Gate only	3.92×	6.57	2.90×	5.51	5.60×	6.99	3.54×	7.54	4.13×	6.38	4.05×	6.60
DarterSD Base distill	3.93×	5.93	2.96×	4.58	4.76×	6.47	2.83×	6.62	3.56×	5.39	3.61×	5.80

and T_{spec} denotes the end-to-end generation time of speculative decoding.

Average Accepted Length (AAL) denotes the average number of draft tokens accepted by the target model in each speculative decoding round. A higher AAL indicates that the draft model produces candidates that better match the target model, thereby reducing the number of target-model decoding steps.

Baselines. We compare DarterSD with three representative speculative decoding baselines: EAGLE2 (Li et al., 2024b), HASS (Zhang et al., 2025a), and EAGLE3 (Li et al., 2025). EAGLE2 uses a context-aware dynamic draft tree for candidate generation. HASS reduces the training-inference mismatch by exposing the draft model to its own generated contextual features and introducing a Top-K distillation objective. EAGLE3 is a strong baseline that removes explicit feature prediction, adopts multi-layer feature fusion, and supports dynamic tree decoding during verification. Further details of these baselines are provided in Appendix G.

4.2 Main Results

Table 1 reports the speedup and average accepted length (AAL) of different speculative decoding methods on Vicuna-13B-v1.3, Llama-3.1-8B-Instruct, and DeepSeek-R1-Distill-LLaMA-8B.

From the average results, DarterSD consistently outperforms the strongest baseline EAGLE3. On Vicuna-13B-v1.3, DarterSD improves the average speedup from $4.67\times$ to $5.52\times$ and increases AAL from 6.74 to 7.61. On Llama-3.1-8B-Instruct and DeepSeek-R1-Distill-LLaMA-8B, DarterSD also improves the average speedup from $4.08\times$ to $4.64\times$ and from $3.58\times$ to $4.06\times$, respectively.

The improvement is more pronounced on structured tasks such as GSM8K and HumanEval, where draft generation benefits from focusing on informative historical tokens. On more open-ended tasks such as CNN/DM, the absolute speedup is relatively lower, likely due to the more diverse output space, but DarterSD still outperforms EAGLE3. These results suggest that token-importance-aware distillation and inference-time denoising improve draft quality across different target models and tasks. Additional model-level and benchmark-level visualizations are provided in Appendix C.

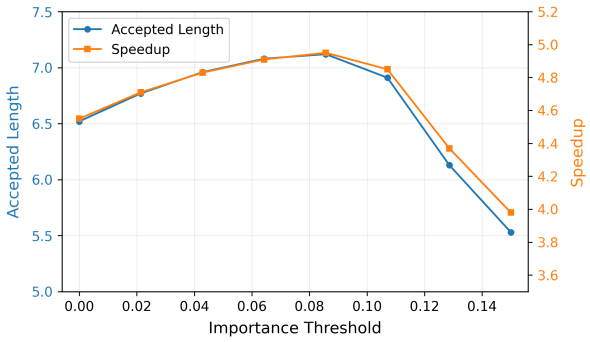


Figure 6: Effect of the importance threshold θ on average accepted length and end-to-end speedup on GSM8K with Vicuna-13B-v1.3.

4.3 Ablation Study

Component-wise Ablation. We conduct comprehensive ablation experiments on Llama-3.1-8B-Instruct to validate the effectiveness of each core component in DarterSD, as shown in Table 2. We remove key modules sequentially and directly summarize the functional conclusions of each component.

- **DarterSD w/o FlashAttn**: Removing FlashAttention only reduces the average speedup from 4.64× to 4.49×, while the average accepted length (AAL) remains unchanged. Conclusion: FlashAttention only optimizes computational efficiency and memory access, and does not affect the quality of draft candidate tokens.
- **DarterSD w/o Denoising**: Removing the Boolean-mask dynamic denoising module leads to clear drops in both speedup (4.35×) and AAL (6.60). Conclusion: The inference-time dynamic denoising strategy serves as an effective auxiliary optimization module of DarterSD. It can filter redundant contextual noise and further polish draft token quality, bringing steady performance gains on the basis of training-side distillation optimization.
- **DarterSD Gate only**: Removing both FlashAttention and dynamic denoising results in a further performance decline. Conclusion: The gated attention mechanism alone can improve draft-target alignment, but is insufficient to achieve full acceleration benefits without inference denoising.
- **DarterSD Base distill**: Removing gate regularization and Dynamic Weight Averaging (DWA) causes the largest performance drop, with speedup decreasing to 3.61× and AAL dropping to 5.80. Conclusion: The gated attention and DWA-based multi-objective distillation are indispensable core designs for DarterSD, which determine the draft model's ability to learn informative token features and align with the target model.

The ablation also shows that gated distillation already yields substantial gains even without inference-time masking; Boolean-mask denoising brings further complementary improvement.

Overall, the superior performance of DarterSD stems from the dominant training-side optimization, which includes gated attention token importance modeling and DWA-based multi-objective distillation, assisted by inference-side denoising refinement. The three designs work synergistically, where the distillation framework lays the foundation for high draft-target alignment, and denoising further eliminates inference-time contextual noise to boost final performance. We provide further discussion on training dynamics in Appendix E.

Threshold Sensitivity Analysis. Since DDS uses the importance threshold θ to control which historical tokens are retained in draft-model attention, we further examine its sensitivity. As shown in Figure 6, a moderate threshold improves both AAL and end-to-end speedup, suggesting that filtering out low-importance historical tokens can reduce contextual noise and improve draft quality. However, when the threshold becomes too large, performance decreases, indicating that excessive masking may also remove useful contextual information. These results show that DDS is effective when it maintains a proper balance between denoising and information preservation.

5 Conclusion

We proposed DarterSD, a dynamic token-weighted distillation framework with auxiliary inference denoising for high-efficiency speculative decoding. DarterSD augments the draft model's self-attention with a learnable sigmoid gate and adopts DWA-based multi-objective distillation to enhance the draft-target distribution alignment by focusing training on informative tokens. At inference time, the learned token importance scores are further binarized into an adaptive Boolean mask to prune low-importance tokens and reduce redundant contextual noise, which further improves the

generation quality and acceleration effect on the basis of well-trained draft model, without altering the target model's verification pipeline. Experiments on multiple models and benchmarks show that DarterSD improves both speedup and average accepted length over strong baselines, and ablation studies confirm the contribution of gated attention, adaptive loss-weight averaging, and inference-time denoising. We believe token-importance-aware draft modeling will become a general paradigm for efficient speculative decoding.

Limitations

The Boolean-mask denoising strategy relies on an importance threshold to determine which tokens are retained or suppressed. Although sensitivity analysis shows that moderate suppression consistently improves performance, the optimal threshold may vary across model architectures, task types, and decoding configurations. Future work could explore fully adaptive threshold selection or task-aware denoising strategies.

Acknowledgments

This work is sponsored in part by the National Natural Science Foundation of China (Grant No. 62421002) and the Fundamental and In terdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (Grant No. JYB2025XDXM202).

References

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, pages 5209–5235.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2025. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *Preprint*, arXiv:2302.01318.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Pondé 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, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Ziyi Chen, Xiaocong Yang, Jiacheng Lin, Chenkai Sun, Kevin Chen-Chuan Chang, and Jie Huang. 2024. Cascade speculative drafting for even faster LLM inference. In *Advances in Neural Information Processing Systems*, volume 37.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing GPT-4 with 90%* ChatGPT quality. https://lmsys.org/blog/2023-03-30-vicuna/.

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. 2021. Training verifiers to solve math word problems. *Preprint*, arXiv:2110.14168.

DeepSeek-AI. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *Preprint*, arXiv:2305.14233.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 12622–12642, Bangkok, Thailand. Association for Computational Linguistics.

Wanyang Hong, Zhaoning Zhang, Yi Chen, Libo Zhang, Baihui Liu, Linbo Qiao, Zhiliang Tian, and Dongsheng Li. 2025. Rhea: Role-aware heuristic episodic attention for conversational LLMs. *Preprint*, arXiv:2512.06869.

Kaixuan Huang, Xudong Guo, and Mengdi Wang. 2025. SpecDec++: Boosting speculative decoding via adaptive candidate lengths. In *Second Conference on Language Modeling*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pages 19274–19286.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024a. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems*, volume 37, pages 22947–22970.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024c. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, pages 28935–28948.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2025. EAGLE-3: Scaling up inference acceleration of large language models via training-time test. In *Advances in Neural Information Processing Systems*, volume 38.

Jiesong Liu, Brian Park, and Xipeng Shen. 2025. A drop-in solution for on-the-fly adaptation of speculative decoding in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9778–9794, Vienna, Austria. Association for Computational Linguistics.

Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. 2024. CacheGen: KV cache compression and streaming for fast large language model serving. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 38–56.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. 2023. Deja vu: Contextual sparsity for efficient LLMs at inference time. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 22137–22176. PMLR.

Llama Team. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2024. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, pages 932–949.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290. ACL.

Open-R1 Team. 2025. OpenThoughts-114k-math. https://huggingface.co/datasets/open-r1/OpenThoughts-114k-math.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. QUEST: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 47901–47911. PMLR.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. https://crfm.stanford.edu/2023/03/13/alpaca.html.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. 2025. SWIFT: On-the-fly self-speculative decoding for LLM inference acceleration. In *International Conference on Learning Representations*.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth*

International Conference on Learning Representations.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097, Vienna, Austria. Association for Computational Linguistics.

Ofir Zafrir, Igor Margulis, Dorin Shteyman, Shira Guskin, and Guy Boudoukh. 2025. FastDraft: How to train your draft. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22488–22505, Vienna, Austria. Association for Computational Linguistics.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024. Draft & verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 11263–11282.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. 2025a. Learning harmonized representations for speculative sampling. In *International Conference on Learning Representations*.

Libo Zhang, Zhaoning Zhang, Wangyang Hong, and Dongsheng Li. 2026. Sparrow: Text-anchored window attention with visual-semantic glimpsing for speculative decoding in video llms. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pages 9928–9942. Association for Computational Linguistics.

Libo Zhang, Zhaoning Zhang, Xubaizhou, Rui Li, Zhiliang Tian, Songzhu Mei, and Dongsheng Li. 2025b. Dovetail: A CPU/GPU heterogeneous speculative decoding for LLM inference. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pages 17382–17395. Association for Computational Linguistics.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710.

Weilin Zhao, Yuxiang Huang, Xu Han, Wang Xu, Chaojun Xiao, Xinrong Zhang, Yewei Fang, Kaihuo Zhang, Zhiyuan Liu, and Maosong Sun. 2024. Ouroboros: Generating longer drafts phrase by phrase for faster speculative decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13378–13393, Miami, Florida, USA. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

A Related Work

Speculative decoding and draft-model training. Speculative decoding (Leviathan et al., 2023; Chen et al., 2023; Zhang et al., 2025b, 2026) accelerates autoregressive generation by using a lightweight draft model to propose candidate tokens and a larger target model to verify them in parallel. Recent methods mainly improve draft quality or draft-target alignment. EAGLE (Li et al., 2024c) performs autoregression over second-to-top-layer features to reduce feature uncertainty, EAGLE-2 (Li et al., 2024b) introduces context-aware dynamic draft trees, and EAGLE-3 (Li et al., 2025) further uses multi-layer feature fusion and training-time test simulation to better match inference dynamics. HASS (Zhang et al., 2025a) addresses the mismatch between training and decoding through harmonized objective distillation and context alignment, while FastDraft (Zafrir et al., 2025) studies efficient training and alignment of compact draft models. These methods improve speculative decoding from the perspective of feature prediction, context alignment, or draft-model training. In contrast, DarterSD focuses on the overlooked token-importance imbalance inside draft-model attention and uses learned importance scores for both training-time distillation and inference-time denoising.l

Candidate construction and verification. Another line of work improves speculative decoding by designing stronger candidate structures or verification strategies. Medusa (Cai et al., 2024) predicts multiple future tokens with parallel decoding heads, SpecInfer (Miao et al., 2024) organizes candidates into token trees for efficient serving, and Ouroboross (Zhao et al., 2024) generates longer phrase-level draft. Adaptive and self-speculative methods further reduce decoding cost by adjusting candidate length or reusing partial layers, such as SpecDec++, Draft & Verify, LayerSkip, and SWIFT (Huang et al., 2025; Zhang et al., 2024; Elhoushi et al., 2024; Xia et al., 2025). These methods mainly optimize how draft candidates are generated or verified. DarterSD is complementary to them: it keeps target-model verification unchanged and improves draft generation by suppressing low-importance historical tokens inside the draft model.

Token importance and attention sparsity. The motivation of DarterSD is related to recent observations that Transformer inference exhibits non-uniform token contribution and structured sparsity. DejaVu (Liu et al., 2023) reveals contextual sparsity in attention heads and MLP parameters during inference, while H2O (Zhang et al., 2023) identifies heavy-hitter tokens that contribute disproportionately to attention and KV-cache utility. Native Sparse Attention studies (Yuan et al., 2025) trainable sparse attention for efficient long-context modeling. KV-cache methods such as Quest, SnapKV, and PyramidKV also exploit token or layer-wise importance to select, compress, or allocate cached states (Tang et al., 2024; Li et al., 2024a; Cai et al., 2025). Unlike these methods, which mainly target target-model inference or KV-cache compression, DarterSD applies token-importance modeling to the speculative draft model. It constructs Boolean attention masks from learned gate scores rather than physically pruning or reorganizing the KV cache, thereby reducing contextual noise while preserving the target-model verification process.

B Dataset Details

Table B.1 summarizes the evaluation datasets used in our experiments and complements the dataset description in Section 4.1. These datasets cover mathematical reasoning, text summarization, code generation, instruction following, and multi-turn dialogue evaluation. For each benchmark, we randomly sample 80 examples for measuring inference latency and draft-token acceptance.

C Additional Result Visualizations

This section provides additional visualizations of the main experimental results and token-importance modeling behavior. The corresponding numerical results are reported in Table 1 in the main text.

Figure C.1 provides a model-level overview of the main results. It shows that DarterSD consistently improves both average speedup and average accepted length across different target models, indicating that the proposed token-importance-aware mechanism is not tied to a single model architecture.

Figures C.2 and C.3 further break down the results by benchmark. The speedup gains are more pronounced on structured tasks such as GSM8K and HumanEval, while DarterSD still maintains consistent advantages on more open-ended tasks. The accepted-length results show a similar trend, suggesting that the speedup improvements mainly

Table B.1: Core information of the evaluation datasets.

Dataset	Scale	Task	Evaluation Focus
GSM8K	8.5K grade-school math problems	Mathematical reasoning	Numerical computation and logical reasoning ability
CNN/DM	Approx. 300K news-summary pairs	Text summarization	Document summarization and information compression ability
HumanEval	164 handwritten Python programming problems	Code generation	Programming ability and code correctness verification
Alpaca	52K instruction-following dialogue examples	Instruction following	Instruction understanding and response generation ability
MT-bench	80 multi-turn dialogue scenarios	Multi-turn dialogue	Multi-turn interaction ability and dialogue quality

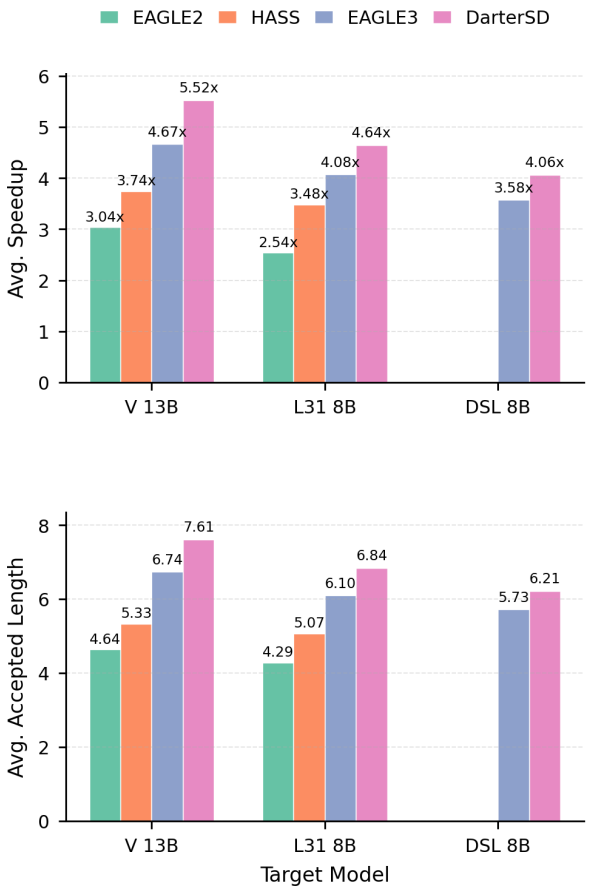


Figure C.1: Average speedup and average accepted length across target models.

come from higher-quality draft candidates.

Figure C.4 complements Figure 4 in the main text by providing the complete visualization of learned gate scores. The upper panel shows that the learned importance scores vary across samples and generation positions, while the lower panel illustrates token-level importance scores for a representative prompt. These patterns support the claim that token-importance differences can be explicitly captured by the gated attention mechanism.

D Threshold Sensitivity

This section provides additional interpretation of the threshold sensitivity results shown in Figure 6 in the main text. The importance threshold θ controls which historical tokens are retained in draft-model attention. A smaller threshold keeps more contextual tokens, while a larger threshold filters out more low-importance tokens.

When $\theta = 0$, all historical tokens are retained in the attention computation of the draft model. This setting preserves complete contextual information, but low-importance tokens are also kept, which may introduce additional noise into draft generation. As θ gradually increases from 0 to around 0.08, both average accepted length and speedup improve, suggesting that moderately suppressing low-importance tokens helps improve draft-token quality.

However, when θ continues to increase beyond 0.10, both metrics start to decrease. When θ reaches 0.15, the performance drops clearly compared with the best setting. This indicates that an overly large threshold may lead to excessive masking, where the model not only filters out noisy tokens but also removes useful contextual information. Overall, the results reveal a trade-off between denoising and information preservation, and show that an appropriate threshold is important for achieving better speculative decoding performance.

E Training Dynamics

We further analyze the training stability of dynamic token-weighted distillation on Llama-3.1-8B-Instruct. Specifically, we record the changes in different loss terms, dynamic loss weights, validation metrics, and validation losses at different draft-tree depths.

Figure E.1 is used to examine whether the multi-

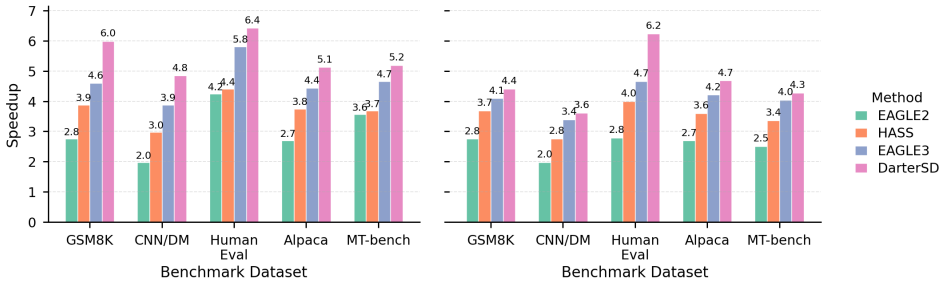


Figure C.2: Per-benchmark inference speedup comparison on Vicuna-13B-v1.3 and Llama-3.1-8B-Instruct.

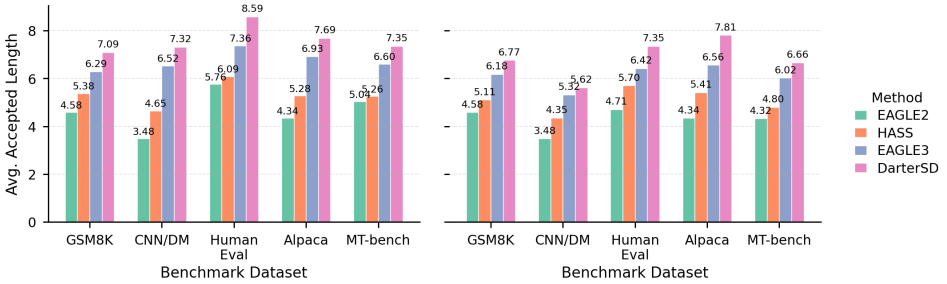


Figure C.3: Per-benchmark average accepted length comparison on Vicuna-13B-v1.3 and Llama-3.1-8B-Instruct.

objective distillation process remains stable when the logits alignment loss, KL divergence loss, and gate regularization loss are optimized jointly.

As shown in Figure E.1, the logits alignment loss decreases rapidly in the early stage, while the KL divergence loss and the gate regularization loss decrease more gradually. This indicates that different training objectives have different convergence rates. On the validation set, the total loss decreases steadily as training proceeds, and the average generation accuracy gradually improves. These trends suggest that the draft model progressively approaches the output distribution of the target model without showing obvious training instability.

Dynamic Weight Averaging (DWA) helps alleviate the optimization imbalance among different objectives. At the beginning of training, the weights of different losses remain close to their initial values. As training proceeds, the weight of the rapidly converging logits alignment loss is reduced, while the weight of the more slowly converging gate regularization loss is increased. In the middle and later stages of training, the convergence rates of different losses become closer, and the dynamic weights gradually return to a more balanced state. This behavior shows that DWA can adaptively adjust the optimization focus according to the convergence speed of each objective.

The validation losses at different draft-tree depths also exhibit similar decreasing trends. Here, the depth denotes the number of consecutive autoregressive draft steps performed by the draft model based on the context verified by the target model. The losses at different depths decrease smoothly, suggesting that the draft model can not only align effectively on shallow candidate tokens but also maintain stable prediction ability along deeper draft generation paths.

F Training and Inference Algorithms

Algorithm F.1 summarizes the training procedure of DarterSD. The draft model is trained with multi-objective distillation, where Dynamic Weight Averaging (DWA) balances the KL divergence loss, logits alignment loss, and gate regularization loss. Algorithm F.2 presents the inference procedure with Boolean-mask-based dynamic denoising. The target-model verification process remains unchanged during inference.

Algorithm F.1 makes explicit how the draft model is optimized. The key point is that the draft model is not only trained to match the target distribution, but is also regularized to learn discriminative token-importance patterns through the gate scores. DWA dynamically adjusts the relative weights of the three objectives, reducing the need for manual loss-weight tuning.

Algorithm F.2 describes how the learned token-importance scores are used at inference time. The

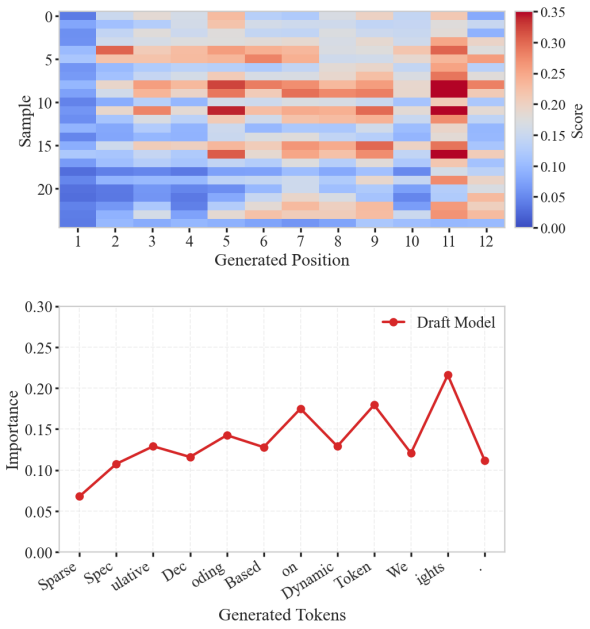


Figure C.4: Visualization of token-importance scores produced by the gated attention mechanism on MT-bench. The heatmap shows importance distributions across samples and generation positions, while the curve illustrates token-level importance scores for a representative prompt.

draft model first computes gate-based importance scores over the global history and converts them into a Boolean keep mask. This mask is combined with the causal mask and tree mask to suppress low-importance historical tokens during candidate generation. The target model then verifies the candidates in parallel, so the final target-model verification procedure remains unchanged.

G Baseline Details

This section provides additional details about the speculative decoding baselines used in our experiments. These descriptions clarify what each baseline changes in the draft-generation process and how it differs from the proposed DarterSD framework.

EAGLE2. EAGLE2 follows the EAGLE framework (Li et al., 2024c) and constructs the input to the draft model by concatenating the current hidden-state feature with the embedding of the token generated in the previous step. It further introduces a context-aware dynamic draft tree, where the tree structure is adjusted according to the confidence scores produced by the draft model. Candidate sequences are then verified in parallel through tree attention.

Algorithm F.1 Training Procedure of DarterSD

Require: M_{target}, $\mathcal{D}$, $\mathcal{D}_{\text{val}}$, E, K, learning rate η
Ensure: Trained draft model M_{draft}

Initialize M_{draft} and the gating projections
Initialize loss histories for $\mathcal{L}_{\text{logits}}$, $\mathcal{L}_{\text{KL}}$, and $\mathcal{L}_{\text{gate}}$
for $e = 1$ to E **do**
  Update loss weights with DWA based on previous loss histories
  for each batch in $\mathcal{D}$ **do**
    Extract and fuse hidden states from the first three layers of M_{target}
    for $k = 0$ to $K - 1$ **do**
      Compute queries and gate scores $[\mathbf{Q}, \mathbf{G}]$
      Compute gated attention output $\mathbf{O}_{\text{attn}} \odot \sigma(\mathbf{G})$
      Obtain draft logits through FFN and LM head
      Compute $\mathcal{L}^k_{\text{logits}}$, $\mathcal{L}^k_{\text{KL}}$, and $\mathcal{L}^k_{\text{gate}}$
      Compute $\mathcal{L}^k_{\text{total}} = w_1\mathcal{L}^k_{\text{KL}} + w_2\mathcal{L}^k_{\text{logits}} + w_3\mathcal{L}^k_{\text{gate}}$
    end for
    Backpropagate and update trainable parameters
  end for
  Evaluate on $\mathcal{D}_{\text{val}}$ and save loss histories
end for
return M_{draft}

HASS. HASS aims to reduce the discrepancy between training and decoding contexts. During training, the draft model is exposed to contextual features generated by itself, which may contain prediction errors, rather than only relying on the correct contextual features produced by the target model. It also introduces a Top-K distillation loss to improve the prediction quality of tokens that are more important for decoding.

EAGLE3. EAGLE3 is a strong speculative decoding baseline. It removes the explicit feature prediction constraint and adopts multi-layer feature fusion to enhance the representation ability of the draft model. It also simulates multi-step draft generation during training, making the training process closer to inference. During verification, EAGLE3 is compatible with the dynamic tree decoding mechanism of EAGLE2.

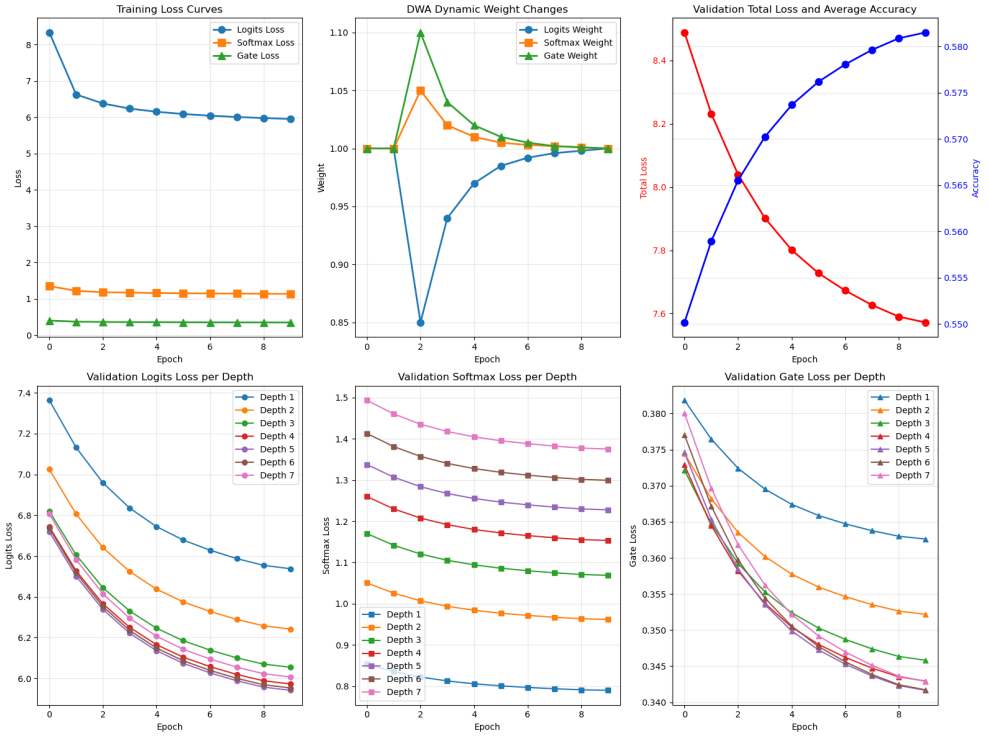


Figure E.1: Training dynamics of DarterSD on Llama-3.1-8B-Instruct. The figure shows training losses, dynamic loss weights, validation metrics, and validation losses at different draft-tree depths.

Algorithm F.2 Inference Procedure of DarterSD with Boolean-Mask-Based Dynamic Denoising

Require: Target model M_{target}, draft model M_{draft}, prompt x, importance threshold θ, tree depth K, branching factor b, maximum length $L_{\max}$
Ensure: Generated sequence y

$\mathbf{x} \leftarrow \text{Tokenize}(x)$
$\mathbf{H} \leftarrow \text{FuseHidden}(M_{\text{target}}(\mathbf{x}))$
$\mathbf{s} \leftarrow \text{ComputeGateScore}(M_{\text{draft}}, \mathbf{H})$
$\mathbf{M}^{\text{keep}} \leftarrow (\mathbf{s} \geq \theta)$
$\mathbf{M}^{\text{causal}} \leftarrow \text{CausalMask}(|\mathbf{x}|)$
$\mathbf{M}^{\text{global}} \leftarrow \mathbf{M}^{\text{causal}} \wedge \text{Expand}(\mathbf{M}^{\text{keep}})$
$y \leftarrow \mathbf{x}$
while $|y| < L_{\max}$ **do**
  $\mathbf{M}^{\text{tree}} \leftarrow \text{TreeMask}(K, b)$
  $\mathbf{M}^{\text{attn}} \leftarrow \mathbf{M}^{\text{global}} \wedge \mathbf{M}^{\text{tree}}$
  $\hat{\mathbf{x}}, \hat{\mathbf{z}} \leftarrow M_{\text{draft}}.\text{Generate}(\mathbf{H}, \mathbf{M}^{\text{attn}}, K, b)$
  $\mathbf{x}_{\text{acc}} \leftarrow M_{\text{target}}.\text{Verify}(\hat{\mathbf{x}}, \hat{\mathbf{z}})$
  $y \leftarrow \text{Concat}(y, \mathbf{x}_{\text{acc}})$
  if $|\mathbf{x}_{\text{acc}}| > 0$ **then**
    $\mathbf{H}_{\text{new}} \leftarrow \text{FuseHidden}(M_{\text{target}}(\mathbf{x}_{\text{acc}}))$
    $\mathbf{H} \leftarrow \text{Concat}(\mathbf{H}, \mathbf{H}_{\text{new}})$
    $\mathbf{s}_{\text{new}} \leftarrow \text{ComputeGateScore}(M_{\text{draft}}, \mathbf{H}_{\text{new}})$
    $\mathbf{s} \leftarrow \text{Concat}(\mathbf{s}, \mathbf{s}_{\text{new}})$
    Recompute $\mathbf{M}^{\text{keep}}$, $\mathbf{M}^{\text{causal}}$, and $\mathbf{M}^{\text{global}}$
  end if
end while
return y