



DREAM: Drafting with Refined Target Features and Entropy-Adaptive Cross-Attention Fusion for Multimodal Speculative Decoding

Yunhai Hu¹ Tianhua Xia^{2*} Zining Liu^{4*} Rahul Raman¹ Xingyu Liu²
Bo Bao³ Eric Sather³ Vithursan Thangarasa³ Sai Qian Zhang^{1,2}

¹Courant Institute of Mathematical Sciences, New York University

²Tandon School of Engineering, New York University

³Cerebras Systems Inc.

⁴University of Pennsylvania

{yh5961, tx856, rr4549, xl5444, sai.zhang}@nyu.edu

zliu0@seas.upenn.edu

{bo.bao, eric.sather, vithu}@cerebras.net

Abstract

Speculative decoding (SD) has emerged as a powerful method for accelerating autoregressive generation in large language models (LLMs), yet its integration into vision-language models (VLMs) remains underexplored. We introduce *DREAM*, a novel speculative decoding framework tailored for VLMs that combines three key innovations: (1) a cross-attention-based mechanism to inject intermediate features from the target model into the draft model for improved alignment, (2) adaptive intermediate feature selection based on attention entropy to guide efficient draft model training, and (3) visual token compression to reduce draft model latency. *DREAM* enables efficient, accurate, and parallel multimodal decoding with significant throughput improvement. Experiments across a diverse set of recent popular VLMs, including LLaVA, Pixtral, SmolVLM and Gemma3, demonstrate up to **3.6× speedup** over conventional decoding and significantly outperform prior SD baselines in both inference throughput and speculative draft acceptance length across a broad range of multimodal benchmarks. The code is publicly available at: <https://github.com/SAI-Lab-NYU/DREAM.git>.

1 Introduction

Large language models (LLMs) have shown impressive performance across diverse tasks, but their inference speed remains limited due to the standard autoregressive process, which includes both prefiling and decoding stages. To overcome this limitation, speculative decoding (SD) [48, 5, 19] accelerates the autoregressive process by splitting it into a low-cost drafting stage and a parallel verification stage, enabling multiple drafted tokens to be validated in a single pass through the target LLM. This approach accelerates the decoding stage while preserving the output quality of the target model through a mechanism of acceptance and rejection.

While these techniques have been extensively developed to accelerate inference in text-only LLMs [25, 24, 26, 4, 2, 55, 61, 40, 6, 50, 15], there has been limited work integrating SD into multimodal LMs (MLLM) [20, 43], especially vision-language models (VLM) [10]. VLMs differ from their text-only counterparts by requiring seamless integration of visual and textual information. This integration process typically occurs in two stages: first, extracting meaningful representations from images, and second, applying language reasoning capabilities to generate appropriate responses. For example, in

* Authors contributed equally; the order of authorship was assigned randomly.

LLaVA [29], a vision encoder transforms images using 24 layers of self-attention and feed-forward networks. These features are then projected into the text space and fused with text embeddings within the language model’s backbone. The effectiveness of this process heavily depends on the model’s ability to maintain coherent representations across modalities, which presents unique challenges for acceleration techniques.

In this paper, we propose *Drafting with Refined Target Features and Entropy-Adaptive Cross-Attention Fusion for Multimodal Speculative Decoding* (DREAM). Specifically, DREAM employs a specialized cross-attention mechanism that enhances the interaction between visual and textual features, ensuring that key information in the target model is adequately captured even in the draft generation stage. Another key innovation of DREAM lies in its selective use of intermediate-layer representations, which encapsulate the most informative features from both modalities to effectively supervise the draft model with high accuracy. Finally, DREAM introduces a visual input compression scheme for the draft model, guided by the intermediate features from the target model, which substantially reduces processing latency without compromising accuracy.

We evaluate DREAM on a diverse set of popular VLMs, including LLaVA-v1.6-Vicuna-7B/13B [29], SmolVLM-2B [38], Pixtral-12B [1], and Gemma3-12B [3], across multiple multimodal tasks. Our extensive experiments demonstrate that DREAM substantially outperforms well-established SD methods, while consistently achieving high acceptance rates across various multimodal applications, such as table structure recognition, interactive segmentation, animal keypoint detection, and chart question answering. Our main contributions are summarized as follows:

- DREAM incorporates a cross-attention mechanism to leverage intermediate outputs from the target model, facilitating more effective knowledge transfer to the draft model and leading to significant performance gains.
- During DREAM training phase, intermediate features from the middle layers of the target model are dynamically selected based on their entropy and used to supervise the draft model. This enhances the draft model’s predictive accuracy and increases token acceptance lengths.
- We investigate the relative importance of visual input and show that visual input in the draft model can be effectively compressed using a scheme guided by intermediate target features, reducing draft processing time and overall speculative decoding latency.
- Evaluation results show that these three strategies enable DREAM to achieve up to a $3.6\times$ reduction in latency compared to conventional decoding methods across a range of VLMs and tasks, outperforming existing speculative decoding approaches.

2 Background and Related Work

2.1 Speculative Decoding

Speculative decoding (SD) [48] has proven effective in mitigating the sequential bottleneck in language model inference. It operates in two stages: a lightweight draft model rapidly generates a sequence of candidate tokens, which are then verified in parallel by a more accurate target model. Specifically, the draft model generates tokens $d_1, d_2, \dots, d_\gamma$, where γ is a predefined number of tokens. These candidates are verified concurrently by the target model but accepted sequentially. Specifically, a draft token d_i is accepted with probability $\min(1, P_{\text{target}}(d_i)/P_{\text{draft}}(d_i))$, where P_{draft} and P_{target} are the probabilities assigned by the draft and target models, respectively. Otherwise, the token is rejected. This acceptance rule is compatible with various sampling temperatures. If a rejection occurs at d_i , all subsequent

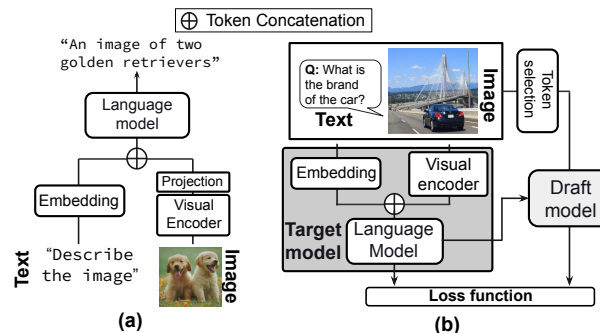


Figure 1: (a) Standard VLM. (b) DREAM overview.

Otherwise, the token is rejected. This acceptance rule is compatible with various sampling temperatures. If a rejection occurs at d_i , all subsequent

draft tokens from d_i onward are discarded, and the target model's token t_i is used by draft model to continue generation. If all tokens are accepted, the draft model proceeds to generate next batch.

Building on this foundation, researchers have proposed a range of techniques to enhance the efficiency of the draft-verify process. For drafting, various approaches have been developed to eliminate or improve draft models. Medusa [4] introduces lightweight decoding heads on the target model itself, eliminating the need for a separate drafter. Self-speculative techniques include layer skipping [14, 61, 8, 28, 32, 56], which accelerates draft generation by selectively processing fewer transformer layers. Other strategies leverage token distillation [64], N-gram prediction [42, 49, 34], and retrieval-augmented drafting [57, 13, 54] to improve draft quality while minimizing computational overhead. Tree-based verification methods [40, 52, 6, 47, 4] support parallel exploration of multiple completion paths, substantially improving throughput over traditional linear verification. EAGLE [25] introduces feature-based uncertainty estimation for tree construction, while EAGLE-2 [24] further improved this with dynamic context-aware trees. Numerical optimization-inspired approaches include Jacobi iterations [45] and Lookahead decoding [9], which reformulate autoregressive generation as parallel optimization problems. For production deployment, various frameworks enhance system-level efficiency. TRIFORCE [50] employs hierarchical speculative decoding combined with a sparse KV cache to support ultra-long sequences exceeding 100,000 tokens. Parallel scheduling approaches [41, 33] enable draft generation to run concurrently with target verification, while heterogeneous compute solutions like Dovetail [62] optimally distribute models across CPU/GPU resources.

While the aforementioned techniques focus on SD for language-only models, there has been limited exploration of SD in MLLMs. In speech synthesis, VADUSA [20] applies SD to accelerate inference in text-to-speech systems, simultaneously improving synthesis quality. Building on the principles of SD, the authors of [43] propose a multi-token prediction mechanism that significantly boosts inference efficiency for speech generation. In the VLM context, [10] applies speculative decoding to the LLaVA-7B model, demonstrating up to $2.37\times$ speedup by utilizing a lightweight, language-only draft model under memory constraints. IbED [18] proposes an *In-batch Ensemble Drafting* strategy that employs ensemble techniques at the batch level without introducing additional model parameters. In contrast, DREAM introduces a novel cross-attention mechanism and adaptive intermediate feature selection to enhance draft model training, resulting in substantial latency reductions.

2.2 Vision Language Model and Computation Profiling

Vision-Language Models (VLMs) are designed to jointly process visual and textual inputs, allowing machines to interpret and generate content that combines both modalities. As illustrated in Figure 1 (a), a typical VLM consists of a visual encoder and a language model. The input image is first processed by the visual encoder to generate visual tokens, which are then concatenated with the textual tokens. These combined tokens are passed to the language model, which generates the final textual outputs. Several VLMs [23, 22] have been developed. Recent works like LLaVA [31], InstructBLIP [7] and Pixtral [1] aim to enhance the zero-shot capabilities of VLMs by better aligning them with human preferences. While the large model sizes have resulted in significant performance improvements, their computational complexity and storage requirements limit their deployment on resource-constrained devices. Lightweight VLMs, such as TinyGPT-V [59] and TinyLLaVA [63], explore the potential of small-scale models and focus on developing efficient VLM architectures. The recent development of SmolVLM [38] introduces a family of compact VLMs with parameters ranging from 256M to 2B, achieving exceptional performance while maintaining smaller model sizes.

To quantify the computational cost introduced by visual inputs processing in VLMs, we profile the floating point operations (FLOPs) required by various models, including LLaVA-v1.6-Vicuna-7B [31], LLaVA-v1.6-Vicuna-7B, Pixtral-12B [1], SmolVLM-2B [38], and Gemma3-12B [11] over the ScienceQA dataset. We select a typical sample which contains a 480×300 image, a prompt with 166 tokens, and use the model to generate 500 tokens, representing an average case across the dataset. FLOPs are measured using PyTorch Profiler. Figure 2 compares the TFLOPs for each model when processing text-only (Txt) versus multimodal (Img+Txt) inputs, revealing an average $2.1\times$ increase

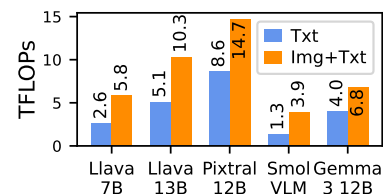


Figure 2: Computational cost of VLMs processing text only (Txt) and multi-modal (Img+Txt) inputs.

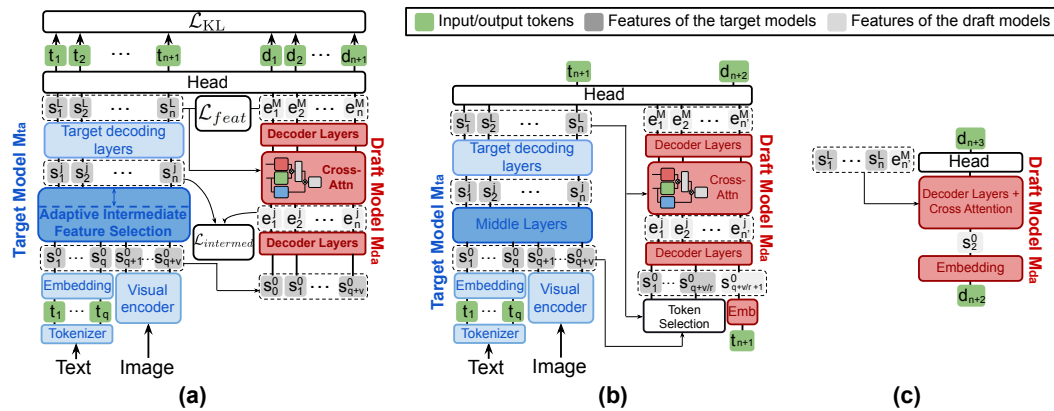


Figure 3: (a) illustrates the training paradigm of DREAM, while (b) and (c) depict its inference workflow. For simplicity, the tree decoding is not shown in (b) and (c).

in computation with the inclusion of visual data and highlighting the need for more efficient visual processing methods.

2.3 Intermediate Feature Distillation

Intermediate Feature Distillation (IFD) enhances traditional knowledge distillation by aligning student models not only with the teacher's final outputs but also with its intermediate representations. FitNets [44] introduced the idea of using intermediate *hints*, showing that projecting selected teacher layers through lightweight adapters can improve compact student models. Later methods refine this by identifying high-similarity layers [60], using attention-based weighting [27], or dynamically selecting task-relevant features, as in TED [27]. CoDIR [51] further introduces contrastive losses for tighter feature alignment. In multimodal contexts, OLA-VLM [16] and VLsI [17] adapt these ideas to distill visual embeddings into language representations. Recently, Skean et al. [46] proposed evaluating per-layer representations using information compression, geometric separability, and robustness, showing that mid-layer features often outperform final-layer ones across tasks, revealing a non-monotonic trade-off between information richness and task relevance.

3 Methodology

Figure 1 (b) provides an overview of DREAM, where the draft model receives the textual input alongside a subsampled visual input to accelerate output generation. Additionally, the intermediate features are adaptively selected to better guide the training of the draft model. Figure 3 presents the detailed training and inference scheme of DREAM. Specifically, let M_{ta} and M_{da} represent the target and draft models, respectively, with a total of L and M transformer blocks, respectively. let t_n and d_n denote the n -th token produced by M_{ta} and M_{da} . For M_{ta} , assume the textual prompt consists of q tokens and the visual input comprises v tokens, respectively. We use s_n^{j-1} and e_n^{j-1} to denote the n -th intermediate feature token at layer j for M_{ta} and M_{da} , respectively.

Figure 3 (b) highlights the DREAM inference process. Let r denote the subsampling factor, and q and v represent the number of prompt and visual tokens, respectively. As illustrated in Figure 3 (b), the target model processes all $q + v$ input feature vectors ($s_1^0, \dots, s_{q+v}^0$). In contrast, with our subsampling method applied, the draft model processes only $q + \lfloor v/r \rfloor$ tokens, substantially lowering the computational load and improving the decoding speed of the draft model, as motivated in Section 2.2. Next, we depict the cross-attention architecture of DREAM (Section 3.1), followed by the adaptive intermediate feature selection strategy for training the draft model (Section 3.2). Finally, we introduce the visual token compression method used to accelerate the draft model in Section 3.3.

3.1 Cross-attention Mechanism for Efficient Knowledge Injection

To more effectively transfer knowledge from the target model and enhance the performance of the draft model, we introduce a cross-attention mechanism to fuse features from the target model

to the draft model. This mechanism naturally aligns with the dual-model setup, using the draft's newly generated token embeddings as queries to efficiently retrieve cached target features. This lightweight attention layer integrates long-range, multimodal context into the decoding process. The attention weights act as soft gating, enabling adaptive selection of the refined target visual and textual cues. Crucially, unlike the EAGLE family of methods [25, 24], which concatenate draft token embeddings with precomputed target features and process them jointly, DREAM adopts a more structured fusion strategy. While EAGLE's concatenation approach is effective for purely textual tasks, it tends to weaken structured visual representations by treating visual and textual features as a simple concatenated sequence, potentially disrupting the spatial relationships learned by the vision encoder. In contrast, our cross-attention mechanism adaptively retrieves relevant visual and textual cues, preserving the integrity of multimodal information. This leads to more effective knowledge transfer and improved draft model performance, as demonstrated in our evaluation results in Section 4.2.

As depicted in Figure 3 (b), in the beginning decoding stage, both textual and visual prompt tokens $t_1, \dots, t_n$ are initially fed into the target model, which then begins generating the next token t_{n+1} , where $n = q + v$. The draft model M_{da} subsequently predicts the $(n + 2)$ -th token, denoted as d_{n+2} . To better help the draft token generation, the cached last-layer features from the target model M_{ta} , denoted as $S^L = (s_1^L, s_2^L, \dots, s_n^L)$, and the intermediate features from the draft model M_{da} , denoted as $E^j = (e_1^j, e_2^j, \dots, e_{n'}^j)$, where $n' = q + \lfloor \frac{v}{r} \rfloor + 1$, are integrated using a cross-attention mechanism. In this setup, E^j is used as the query, while S^L provides the keys and values. With z denoting the dimensionality of queries and keys, the cross-attention is then computed as follows:

$$Q = E^j W_Q, \quad K = S^L W_K, \quad V = S^L W_V, \quad F = \text{softmax}\left(\frac{QK^\top}{\sqrt{z}}\right) V \quad (1)$$

where W_Q, W_K, W_V denote the weight matrices in the draft model. The fused features F replace the original first-layer features and are propagated through the subsequent decoder block. For simplicity, we assume a single attention head in this illustration. After generating the first token d_{n+2} , the draft model leverages cross-attention over both the previously verified features from the target model and its own final-layer features. Specifically, we concatenate the target features $S = (s_1^L, s_2^L, \dots, s_n^L)$ with the latest draft feature $e_{n'}^M$ to form the key and value set $S = (s_1^L, s_2^L, \dots, s_n^L, e_{n'}^M)$ for generating the next draft token d_{n+3} , as described in Figure 3 (c).

Moreover, we adopt the top- k reordering logic and tree-masking strategy introduced in EAGLE-2 [24]. Specifically, we use the newly generated first-layer draft tag features and perform a tree-based width- k expansion to construct the candidate feature set $E = (e_{n'+1}^1, \dots, e_{n'+k}^1)$. The tree mask is then applied to filter out irrelevant tags, resulting in the final representations $E_{tree} = (e_{n'+1}^M, \dots, e_{n'+k}^M)$, which are used to simultaneously generate k new draft tokens. This approach enables efficient parallel decoding, and the same tree mask is also leveraged during verification to reduce computation overhead. More details are provided in the supplementary materials on tree.

3.2 Adaptive Intermediate Feature Selection for Draft Training

In this section, we detail the training strategy for the draft model. To help the draft model effectively learn the behavior of the target model M_{ta} , we utilize intermediate-layer features from the target model as supervision signals. For these features to effectively guide the training of the draft model M_{da} , they must satisfy two key criteria: they can provide key information and capture rich semantic content, and they should be essential, exhibiting low variability across tokens, to enable faster and more stable learning in the draft model. As discussed in Section 2.3, intermediate features with low token-level attention entropy are well-suited for distillation, as they focus on salient content and offer stable guidance. DREAM implements a simple yet effective mechanism to select such features from each layer of the target model to guide the efficient training of the draft model, which is shown in Figure 3 (a). For the l -th decoder block, its input tokens and output tokens are denoted as $S^{\ell-1} = (s_1^{\ell-1}, s_2^{\ell-1}, \dots, s_n^{\ell-1})$ and $S^\ell = (s_1^\ell, s_2^\ell, \dots, s_n^\ell)$, respectively. Let the attention matrix A_ℓ associated with l -th layer as $A_\ell = \text{softmax}\left(\frac{Q_\ell K_\ell^\top}{\sqrt{z}}\right)$, where $Q_\ell = S^{\ell-1} W_Q$ and $K_\ell = S^{\ell-1} W_K$, the average attention entropy (AE) is calculated as:

$$\text{AE}(\ell) = -\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n A_{\ell,i,j} \log A_{\ell,i,j} \quad (2)$$

where $A_{\ell,i,j}$ denotes the (i,j) -th element of A_ℓ . In practice with multiple heads, $AE(\ell)$ is also averaged over all attention heads. During each decoding step n , we dynamically select the layer ℓ^* with the lowest average entropy, defined as $\ell^* = \arg \min_{\ell \in L} [AE(\ell)]$. We then distill the information from $s_i^{\ell^*}$ into the initial decoder block of the draft model using a smooth ℓ_1 loss, guiding the draft model to align with the most informative intermediate representation of the target model. This adaptive feature distillation strategy leads to improved performance by minimizing the ℓ_1 loss between the feature vector of the m -th layer of the draft model $E^m = (e_1^m, \dots, e_n^m)$ and $S^{\ell^*} = (s_1^{\ell^*}, \dots, s_n^{\ell^*})$, namely:

$$\mathcal{L}_{intermed} = \text{smoothL1}(E^m, S^{\ell^*}), \quad (3)$$

where $\text{smoothL1}(x, y)$ equals $\frac{1}{2}(x - y)^2$ if $|x - y| < 1$, and $|x - y| - \frac{1}{2}$ otherwise [12].

3.2.1 Loss Functions

As shown in Figure 3 (a), the loss function of DREAM comprises three components. First, we encourage the output features of the draft model to closely match those of the target model by minimizing the difference between the final-layer features $E^M = (e_1^M, \dots, e_n^M)$ from the draft model and $S^L = (s_1^L, \dots, s_n^L)$ from the target model, using a smooth L1 loss: $\mathcal{L}_{feat} = \text{smoothL1}(E^L, S^L)$. This will improve the acceptance rate of the draft tokens. Second, following the adaptive feature selection strategy described in Section 3.2, we minimize the difference between an early-layer feature E^m and the selected intermediate target feature S^{ℓ^*} . For DREAM, we set $m = 1$, yielding the loss: $\mathcal{L}_{intermed} = \text{smoothL1}(E^m, S^{\ell^*})$. Finally, to ensure the token outputs from the draft model M_{da} align with those from the target model M_{ta} , we apply a KL divergence loss between their softmax outputs: $\mathcal{L}_{KL} = \text{KL}(\text{softmax}(D), \text{softmax}(T))$, where $D = (d_1, \dots, d_n)$ and $T = (t_1, \dots, t_n)$ are the predicted token logits from the draft and target models, respectively. Finally, the overall loss function can be described as:

$$\mathcal{L}_{final} = \lambda_{feat} \mathcal{L}_{feat} + \lambda_{intermed} \mathcal{L}_{intermed} + \lambda_{KL} \mathcal{L}_{KL}, \quad (4)$$

where λ_{feat} , $\lambda_{intermed}$ and λ_{KL} denotes the relative importance between the loss functions.

3.3 Visual Token Compression

Given the high computational cost of visual tokens, as discussed in Section 2.2, DREAM adopts subsampling strategy to reduce this overhead. A naive solution would be to uniformly subsample the visual tokens. However, this approach risks discarding crucial visual information necessary for accurate draft model predictions, while retaining redundant regions that could be aggressively compressed or removed. To address this, DREAM incorporates a simple yet effective token selection mechanism, as illustrated in the visual token selection block of Figure 3 (b). Specifically, it computes importance scores for the final-layer features $(s_1^L, \dots, s_{q+v}^L)$ by summing each token's attention weights across all other tokens, as derived from the attention matrix. These scores reflect the relative importance of each token in the last layer with respect to the final model accuracy.

Among the attention scores, we isolate those corresponding to the visual input and sort this subset by magnitude. We then record the indices of the top scores based on the subsampling ratio r . These indices are used to subsample the visual tokens $(s_{q+1}^1, \dots, s_{q+v}^1)$ with the top high scores, where they are considered critical to the final output. This strategy significantly reduces the number of tokens while retaining the most important visual information. Although visual token subsampling may cause a slight decrease in the draft model's accuracy, our evaluation in Section 4 shows that it can effectively reduce speculative decoding latency.

4 Empirical Results

We conduct experiments on five VLMs representing a range of parameter scales, including LLaVA-v1.6-Vicuna (7B, 13B) [30], Pixtral (12B) [1], SmolVLM (2B) [38], and Gemma3 (12B) [53]. DREAM is evaluated across eight diverse benchmarks: MMT-Bench [58], SEED-Bench-2 [21], ScienceQA [37], OCRBench [35], ChartQA [39], and MathVista [36]. All evaluations are performed under two softmax temperature settings: Temp = 0 and Temp = 1. We report two key metrics: (1) **Speedup ratio** over standard autoregressive generation, defined as t_{AR}/t_{method} , where t_{AR} is

the average wall-clock time per token for standard decoding, and t_{method} is the corresponding time for each evaluated method. A larger speedup directly corresponds to lower end-to-end latency in real-world use. (2) **Average token acceptance length** τ , representing the number of consecutive draft tokens accepted by the verification model. A larger τ implies fewer verification steps and higher effective decoding throughput. We adapt six recent SD baselines, originally developed for text-only LLMs, for use with VLMs, including SPD [10], Kangaroo [28], Medusa [4], Hydra [2], and EAGLE 1 and 2 [25, 24].

We freeze the target VLMs and train only the draft model using the LLaVA mix665k dataset, with 55,000 training samples and 1,000 training samples from each corresponding evaluation benchmark dataset. For benchmarks without predefined train-test splits (MMT-Bench, SEED-Bench-2, Math-Vista, and OCRBench), we applied stratified random sampling using `train_test_split` (with `random_state=42` for reproducibility) to select 1,000 training samples and up to 3,000 test samples per dataset. For ChartQA and ScienceQA, which already provide official splits, we randomly sampled 1,000 training examples from their respective training sets. All selected training samples were then processed through the target model to generate ground-truth responses in conversational form.

Training is performed for 68,000 iterations using AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.95$), a learning rate of 3×10^{-5} , and gradient clipping set to 0.5. Each draft model consists of an initial decoder block, an intermediate cross-attention block, and a final decoder block. The weights for the loss terms $\mathcal{L}_{\text{feat}}$, $\mathcal{L}_{\text{intermed}}$, and $\mathcal{L}_{\text{KL}}$ are set to 0.2, 0.2, and 1.0. The parameter sizes are 0.65B for LLaVA-v1.6-Vicuna-7B, 0.98B for LLaVA-v1.6-Vicuna-13B, 0.9B for Pixtral-12B, 0.28B for SmolVLM-2B, and 0.9B for Gemma3-12B. Training is conducted on two NVIDIA A100 80 GB GPUs with batch size set to 4. Our training procedure adopts a two-stage pipeline: (1) offline calibration for feature selection, and (2) draft model training. During the calibration stage, we compute the attention entropy for each sample once to identify the optimal intermediate layer ℓ^* , and cache these selections for subsequent training.

During the DREAM evaluation, we use speculative sampling with a batch size of 1, following prior work. The tree structure is configured with $k = 4$ child nodes, a depth of 6, and a maximum draft length of 32 tokens. Note that the average accepted token length (τ) reported in Table 1 may exceed a depth of 6, as the target model generates an additional token once all drafts are verified, resulting in up to 7 tokens per round [19]. 75% of the visual tokens are retained. These values were chosen based on preliminary experiments to balance aggressive speculation against verification overhead. A full sensitivity analysis and ablation study on these hyperparameters can be found in supplementary materials. All models are evaluated on a single NVIDIA A100 80GB GPU.

All models are based on their official Hugging Face implementations. Baseline methods, including Kangaroo [28], Medusa [4], EAGLE [25] and EAGLE-2 [24], are executed using their publicly released code and default configurations, with minimal modifications to support VLM inputs. During inference, all models are run with KV caching enabled to ensure efficient autoregressive decoding. To ensure a fair comparison, all methods are trained and evaluated using the same dataset, number of training epochs, learning rate, batch size, and hardware environment. We fix random seeds across all runs to reduce performance variance and report the average over three runs.

4.1 Evaluation Results

Table 1 presents the speedup ratios and average acceptance lengths τ for DREAM compared to baseline methods. Across all tasks and target models, DREAM consistently achieves the highest speedup and longest acceptance lengths. Notably, DREAM delivers a $1.5\times$ to $3.6\times$ speedup over standard autoregressive decoding with the target model, and achieves a 20% to 40% improvement over EAGLE-2. Particularly, we observe that larger models gain greater benefits from DREAM. At $T = 0$, DREAM achieves an average speedup of $3.06\times$ on LLaVA-v1.6-Vicuna-7B and $2.65\times$ on Pixtral-12B, whereas the speedups are lower for smaller models, with $2.27\times$ on SmolVLM-2B and $2.23\times$ on LLaVA-v1.6-Vicuna-7B. This is because larger models experience more severe decoding bottlenecks, allowing the draft model to more effectively substitute the costly decoding process of them. Furthermore, models like LLaVA and Pixtral embed visual features directly into the language decoder, offering clearer multimodal cues. This allows DREAM to achieve higher acceptance lengths, for example, $\tau = 5.51$ on LLaVA-v1.6-Vicuna-7B. In contrast, models such as Gemma3-12B, which handle cross-modal information through more complex processing pathways, reach lower acceptance rates, with $\tau = 2.36$.

Table 1: Evaluation of SD methods through speedup ratio (S) and average accepted token length (τ).

		MMT		SEED		ScienceQA		OCRBench		ChartQA		MathVista		Average	
Models	Methods	S	τ	S	τ	S	τ	S	τ	S	τ	S	τ	S	τ
Temperature = 0															
LLaVA-v1.6 Vicuna-7B	SPD [10]	1.10	1.88	0.81	1.17	1.08	1.87	0.89	1.25	0.91	1.24	1.06	1.76	0.97	1.53
	Kangaroo [28]	1.32	2.11	1.33	2.12	1.31	2.09	1.17	1.89	1.18	1.98	1.15	1.86	1.24	2.01
	Medusa [4]	1.58	2.88	1.59	3.01	1.44	2.77	1.22	2.33	1.25	2.41	1.22	2.34	1.38	2.62
	Hydra [2]	1.78	3.86	1.72	3.88	1.68	3.79	1.41	3.21	1.35	3.11	1.42	3.25	1.56	3.52
	EAGLE [25]	2.10	5.04	2.09	5.01	1.98	4.88	1.72	4.13	1.56	3.98	1.78	4.25	1.87	4.55
	EAGLE-2 [24]	2.31	5.48	2.31	5.61	2.15	5.22	1.92	4.88	1.77	4.22	1.87	4.67	2.05	5.01
	DREAM	2.52	6.40	2.48	6.20	2.33	5.82	2.05	4.88	1.89	4.44	2.11	5.32	2.23	5.51
LLaVA-v1.6 Vicuna-13B	SPD	1.07	1.78	1.06	1.79	1.09	1.88	0.86	1.12	0.89	1.25	0.87	1.22	1.00	1.58
	Kangaroo	1.43	1.77	1.51	1.87	1.22	1.55	1.21	1.54	1.27	1.61	1.53	2.01	1.36	1.72
	Medusa	1.99	2.67	1.96	2.76	1.93	2.77	1.40	2.92	1.51	2.82	1.51	2.62	1.72	2.76
	Hydra	2.12	2.87	2.08	2.99	2.21	3.12	1.49	3.07	1.65	3.03	1.66	2.87	1.87	2.99
	EAGLE	2.45	3.56	2.19	3.24	2.63	3.98	1.65	3.31	1.85	3.27	1.8	3.09	2.10	3.41
	EAGLE-2	2.89	4.05	3.18	4.33	3.09	4.97	2.20	4.12	2.41	4.15	2.39	3.76	2.69	4.23
	DREAM	3.68	5.58	3.51	5.34	3.36	5.29	2.69	4.64	2.59	4.20	2.53	4.18	3.06	4.87
Pixtral-12B	SPD	1.08	1.51	1.03	1.47	1.05	1.49	1.05	1.49	1.04	1.43	1.04	1.46	1.05	1.47
	Kangaroo	1.26	1.54	1.09	1.39	1.14	1.51	1.16	1.52	1.12	1.47	1.13	1.49	1.15	1.49
	Medusa	1.37	1.81	1.37	1.81	1.35	1.87	1.24	1.69	1.22	1.68	1.16	1.47	1.28	1.72
	Hydra	1.58	2.24	1.47	2.04	1.53	2.06	1.38	1.81	1.34	1.79	1.36	1.78	1.44	1.95
	EAGLE	2.38	3.47	1.97	2.53	2.31	3.64	1.69	2.73	1.78	2.84	1.64	2.47	1.96	2.95
	EAGLE-2	2.81	3.95	2.31	3.07	2.64	4.03	2.12	3.25	2.14	3.17	1.81	2.73	2.31	3.37
	DREAM	2.93	4.52	2.61	3.67	2.98	4.33	2.38	3.55	2.35	3.49	2.36	3.42	2.65	3.78
SmolVLM-2B	SPD	1.02	1.33	1.04	1.41	1.06	1.43	1.06	1.42	1.07	1.46	1.02	1.34	1.04	1.40
	Kangaroo	1.28	1.48	1.08	1.18	1.03	1.17	1.06	1.22	1.04	1.14	1.08	1.23	1.10	1.24
	Medusa	2.12	2.71	1.51	2.00	1.72	2.22	1.20	1.61	1.15	1.55	1.35	1.75	1.51	1.97
	Hydra	2.33	3.07	1.62	2.08	1.98	2.66	1.32	1.74	1.22	1.58	1.51	1.98	1.66	2.19
	EAGLE	2.57	3.42	1.85	2.56	2.16	2.76	1.42	1.88	1.34	1.77	1.65	2.22	1.83	2.44
	EAGLE-2	2.96	3.89	2.12	2.93	2.39	3.21	1.65	2.11	1.51	2.13	1.81	2.63	2.07	2.82
	DREAM	3.05	3.97	2.24	3.18	2.85	3.62	1.85	2.56	1.62	2.33	2.01	2.88	2.27	3.09
Gemma3-12B	Kangaroo	1.37	1.66	1.47	1.79	1.52	1.57	3.17	2.28	2.28	1.85	1.18	1.64	1.83	1.80
	EAGLE	1.73	1.98	1.69	2.52	1.72	1.97	4.26	2.42	3.40	1.99	1.42	1.89	2.37	2.13
	EAGLE-2	2.92	1.99	1.74	2.79	1.92	1.98	4.68	2.57	3.48	2.23	1.52	1.91	2.71	2.25
	DREAM	2.99	2.13	3.53	2.84	2.60	2.05	4.81	2.58	3.68	2.56	1.98	1.99	3.27	2.36
Temperature = 1															
LLaVA-v1.6 Vicuna-7B	SPD	0.83	1.19	0.81	1.15	0.85	1.18	0.75	1.06	0.72	1.08	0.92	1.48	0.81	1.19
	Kangaroo	1.20	1.97	1.26	2.03	1.23	2.01	1.09	1.80	1.11	1.89	1.07	1.77	1.16	1.91
	EAGLE-2	2.19	5.37	2.20	5.48	2.04	5.12	1.82	4.77	1.68	4.13	1.76	4.56	1.95	4.91
	DREAM	2.39	6.29	2.35	6.07	2.25	5.68	1.99	4.88	1.84	4.41	2.02	5.23	2.14	5.43
LLaVA-v1.6 Vicuna-13B	SPD	0.88	1.22	0.84	1.25	0.84	1.32	0.79	1.18	0.81	1.14	0.88	1.24	0.84	1.22
	Kangaroo	1.23	1.57	1.17	1.53	1.07	1.44	1.01	1.24	1.07	1.34	1.21	1.67	1.13	1.46
	EAGLE-2	2.35	3.75	3.02	4.30	3.03	4.67	2.03	3.87	2.18	3.83	2.18	3.41	2.46	3.97
	DREAM	3.34	5.38	3.32	5.06	3.20	5.98	2.22	3.89	2.43	4.04	2.29	4.03	2.80	4.73
Pixtral-12B	SPD	0.81	1.15	0.79	1.11	0.80	1.12	0.80	1.13	0.75	1.07	0.77	1.09	0.79	1.11
	Kangaroo	1.18	1.41	1.08	1.35	1.03	1.36	1.19	1.48	1.14	1.45	1.09	1.41	1.12	1.41
	EAGLE-2	2.76	3.81	2.24	3.01	2.76	3.87	2.23	3.24	2.03	3.09	1.79	2.69	2.30	3.28
	DREAM	2.90	4.02	2.47	3.57	2.93	3.94	2.29	3.46	2.21	3.21	2.16	3.27	2.49	3.58
SmolVLM-2B	SPD	1.07	1.47	1.01	1.33	1.07	1.46	0.97	1.26	1.06	1.44	0.85	1.20	1.00	1.36
	Kangaroo	1.37	1.59	1.12	1.24	1.22	1.41	1.12	1.29	1.18	1.36	1.28	1.42	1.22	1.39
	EAGLE-2	2.62	3.60	1.92	2.67	2.24	3.11	1.41	1.77	1.60	2.18	1.77	2.49	1.93	2.64
	DREAM	2.88	3.66	2.25	3.33	2.91	3.74	1.54	2.12	1.77	2.51	1.97	2.70	2.22	3.01
Gemma3-12B	Kangaroo	1.83	1.66	1.23	2.61	1.56	2.29	3.34	2.27	2.23	1.86	1.16	1.65	1.89	2.06
	EAGLE	2.23	1.96	1.60	2.52	2.16	1.97	3.74	2.65	3.30	2.03	1.59	1.86	2.44	2.16
	EAGLE-2	2.73	1.94	2.13	2.79	2.21	2.07	4.67	2.47	3.35	2.23	1.65	1.89	2.79	2.23
	DREAM	2.88	2.07	3.49	2.84	2.39	2.12	4.79	2.56	3.61	2.43	1.96	1.91	3.19	2.32

Second, the task-level analysis reveals that long-form QA tasks such as MMT-Bench and ScienceQA benefit most from speculative decoding. These tasks require generating structured answers grounded in high-level visual semantics, which DREAM captures efficiently. For example, on ScienceQA with LLaVA-v1.6-Vicuna-7B, DREAM achieves a speedup of $3.36\times$ with $\tau = 5.32$, outperforming Kangaroo ($1.22\times$) and EAGLE-2 ($3.09\times$). In contrast, OCR-style datasets such as MathVista pose greater challenges due to their reliance on fine-grained character-level recognition, which current draft models struggle to replicate. As a result, all methods show lower τ and speedup on this benchmark.

Finally, temperature settings critically impact performance. At low temperature ($T = 0$), deterministic decoding leads to higher token alignment and longer acceptance spans. For instance, DREAM achieves $\tau = 4.52$ on MMT-Bench with Pixtral-12B. When sampling is enabled ($T = 1$), perfor-

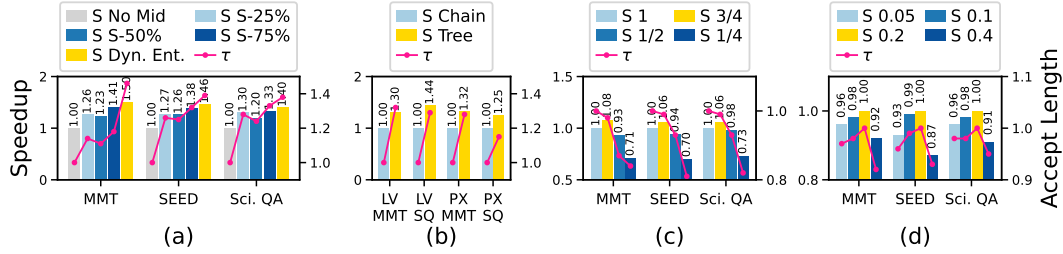


Figure 4: Normalized speedup S and normalized accepted token length τ across (a) intermediate feature selection strategies. (b) chain-based and tree-based decoding. (c) visual token compression ratios, where 1 and 3/4 denote 100% and 75% of the visual tokens are retained, respectively. (d) loss weight settings, where the number is the value for λ_{feat} and $\lambda_{intermed}$. λ_{KL} is fixed to 1.

mance moderately degrades due to increased token variance, yet DREAM’s tree-based architecture maintains robust performance. On MMT-Bench with Pixtral-12B, DREAM still delivers a speedup of $2.90\times$ and $\tau = 4.02$, outperforming all baselines under the same temperature.

4.2 Ablation Study

Impact of Draft Model Architecture

In this section, we evaluate the impact of draft model architecture by testing DREAM on LLaVA-v1.6-Vicuna-7B across multiple datasets, with VTC disabled for all. We design six variants of the DREAM draft model to assess architectural contributions. In the first three baseline configurations, we remove one component at a time: the initial decoder block, denoted as *w/o Initial*, the cross-attention block (*w/o CA*), or the final decoder block (*w/o Final*). Additionally, we include a variant with an extra cross-attention block to evaluate the effect of deeper feature fusion, denoted as *2 CA*. We further evaluate EAGLE-2 by varying its draft model depth, using one, three, and four decoder blocks, denoted as *E2-1B* (default setting in EAGLE-2), *E2-3B*, and *E2-4B*, respectively. Table 2 summarizes the results. Notably, the removal of the cross-attention block causes the most substantial performance drop, showing its essential role in the draft model architecture. Adding an extra cross-attention block increases the accepted sequence length, indicating improved draft quality. However, the larger draft model also incurs additional latency, which offsets the overall speedup gains. Similarly, increasing the number of blocks in the EAGLE-2 draft model improves the average acceptance length, but speedup diminishes once the depth reaches four blocks. Overall, DREAM outperforms EAGLE-2 in both speedup and acceptance length, highlighting the importance of the cross-attention block in DREAM.

Table 2: Comparison over different model architectures. Numbers are normalized to DREAM results.

	MMT-Bench		SEED-Bench		ScienceQA	
Baseline	Speedup	τ	Speedup	τ	Speedup	τ
DREAM	1	1	1	1	1	1
w/o Initial	0.60	0.59	0.60	0.58	0.64	0.61
w/o CA	0.48	0.44	0.46	0.41	0.41	0.32
w/o Final	0.73	0.63	0.71	0.65	0.67	0.71
2 CA	0.99	1.09	1.02	1.13	0.79	1.10
E2-1B	0.82	0.71	0.81	0.77	0.80	0.75
E2-3B	0.88	0.86	0.89	0.88	0.91	0.88
E2-4B	0.63	0.92	0.63	0.92	0.68	0.93

Impact of Intermediate Feature Selection In this section, we evaluate DREAM’s intermediate feature selection strategy for draft model training, referred to as *Dynamic Entropy* (Dyn. Ent.), against four baseline methods on the LLaVA-v1.6-Vicuna-7B model across the MMT-Bench, SEED-Bench, and ScienceQA datasets. The first baseline, *No Mid Tuning* (No Mid), trains the draft model without using any intermediate features. The second baseline, *Static-25%* (S-25%), uses intermediate features from the 25% depth of the target model (the 10th layer in LLaVA-v1.6-Vicuna-7B) for training. Similarly, *Static-50%* (S-50%) and *Static-75%* (S-75%) use features from the 50% and 75% depths, corresponding to the 20th and 30th layers, respectively. Figure 4 (a) presents the results. The draft model trained without intermediate features achieves limited speedup, as fewer tokens are accepted. *Static-25%* and *Static-50%* yield comparable speedup, while *Static-75%* performs best among the static approaches, suggesting that deeper intermediate layers provide more informative guidance. *Dynamic Entropy* applied by DREAM outperforms all baselines with the highest speedup.

Impact of Tree Structure As shown by prior work, tree-structured drafts improve the accepted sequence length compared to chain-structured drafts [25, 24, 6]. Figure 4 (b) shows the performance of DREAM with both chain- and tree-structured drafts on the LLaVA-v1.6-Vicuna-7B (LV) and Pixtral-12B (PX) models across the MMT-Bench (MMT) and ScienceQA (SQ) datasets. Tree decoding provides an average of $1.32\times$ speedup on top of our proposed methods described in Section 3. It is important to note that all the baseline algorithms we compared except Kangaroo utilize tree-structured decoding. Even without the help of tree decoding, compared with Kangaroo, DREAM achieves an average of 56% and 75% speedup on LLaVA and Pixtral, respectively.

Impact of Visual Token Compression Rate To improve decoding efficiency, DREAM subsamples and retains only a fraction of visual tokens from the ViT encoder, as detailed in Section 3.3. In Figure 4 (c), we evaluate the effect of different visual token compression ratios, where a fraction of 1 indicates no compression, and $3/4$ means 75% of the tokens are kept. Experiments on the LLaVA-v1.6-Vicuna-7B model across MMT-Bench, SEED-Bench, and ScienceQA show that retaining $3/4$ of the tokens yields 7% speedup with only a minor reduction in acceptance length. However, when only $1/2$ and $1/4$ of the tokens are retained, the reduced visual input leads to noticeable information loss, lowering acceptance length and diminishing the speedup.

Specifically, we examine how the visual token compression (VTC) rate affects both inference speed and draft token accuracy in DREAM. We test four compression levels (100%, 75%, 50%, and 25%) and evaluate their effects using two key metrics: the speedup ratio (S) and the average accepted token length (τ) across three benchmarks: MMT-Bench, SEED-Bench, and ScienceQA.

As presented in Table 3, the results show that a 75% VTC rate generally offers the best balance between speed and accuracy. For LLaVA-13B and Pixtral-12B, this setting achieves the highest or second-highest speedups ($3.68\times$ and $2.93\times$ on MMT-Bench, respectively) while keeping τ within about 2% of the full-token baseline. In comparison, reducing the visual tokens to 50% or 25% yields only minor speed improvements but causes τ to drop by 10–25%, indicating that excessive visual compression leads to more draft token rejections due to reduced contextual information.

Table 3: Effect of Visual Token Compression Rate on Speedup and Token Acceptance

Models	VTC Rate	MMT-Bench		SEED-Bench		ScienceQA	
		S	τ	S	τ	S	τ
LLaVA-v1.6 Vicuna-13B	100%	3.53	5.67	3.32	5.40	3.18	5.33
	75%	3.68	5.58	3.51	5.34	3.36	5.29
	50%	3.19	4.96	3.13	5.04	3.11	4.98
	25%	2.44	4.78	2.31	4.38	2.31	4.36
Pixtral-12B	100%	2.88	4.48	2.55	3.87	2.91	4.52
	75%	2.93	4.52	2.61	3.67	2.98	4.03
	50%	1.51	2.68	1.58	2.83	1.53	2.77
	25%	1.24	2.18	1.51	2.35	1.33	2.64

Impact of Lambda Settings The weights λ_{feat} , $\lambda_{intermed}$, and λ_{KL} balance different supervisory signals. Since $\mathcal{L}_{feat}$ and $\mathcal{L}_{intermed}$ are smooth L1 losses of similar scale and play comparable roles in guiding the model by promoting feature alignment, we set $\lambda_{feat} = \lambda_{intermed}$ to simplify tuning. The KL loss is typically smaller in magnitude and is fixed at $\lambda_{KL} = 1$ to maintain consistent influence. As shown in Figure 4 (d), on LLaVA-v1.6-Vicuna-7B, increasing λ_{feat} from 0.05 to 0.2 improves speedup and average accepted token length. However, at 0.4, both metrics drop, indicating that excessive feature supervision may harm generalization.

5 Conclusion and Limitation

We present DREAM, a speculative decoding framework optimized for VLMs. By integrating visual token compression, cross-attention feature fusion, and adaptive intermediate distillation, DREAM achieves up to $3.6\times$ speedup over standard decoding while maintaining high accuracy, offering a scalable and efficient solution for fast multimodal inference.

Although DREAM shows strong performance, its evaluation is limited to NVIDIA GPUs. As future work, it is important to assess its effectiveness across a broader range of hardware platforms, datasets, and VLMs. Moreover, although DREAM improves the efficiency of VLM, enabling faster and more accessible deployment in real-world applications such as assistive technologies and interactive agents. However, care should be taken to prevent misuse in generating harmful multimodal content.

References

- [1] Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault, Saurabh Garg, Theophile Gervet, et al. Pixtral 12b. *arXiv preprint arXiv:2410.07073*, 2024.
- [2] Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. *arXiv preprint arXiv:2402.05109*, 2024.
- [3] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.
- [4] 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. *arXiv preprint arXiv:2401.10774*, 2024.
- [5] 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.
- [6] Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *arXiv preprint arXiv:2402.12374*, 2024.
- [7] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: towards general-purpose vision-language models with instruction tuning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.
- [8] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024.
- [9] Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of llm inference using lookahead decoding. *arXiv preprint arXiv:2402.02057*, 2024.
- [10] Mukul Gagrani, Raghavv Goel, Wonseok Jeon, Junyoung Park, Mingu Lee, and Christopher Lott. On speculative decoding for multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8289, 2024.
- [11] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesh Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Pettrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Pöder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler

- Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025.
- [12] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
 - [13] Zhenyu He, Zexuan Zhong, Tianle Cai, Jason D Lee, and Di He. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*, 2023.
 - [14] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Hasan Genc, Kurt Keutzer, Amir Gholami, and Sophia Shao. Speed: Speculative pipelined execution for efficient decoding. *arXiv preprint arXiv:2310.12072*, 2023.
 - [15] Yunhai Hu, Zining Liu, Zhenyuan Dong, Tianfan Peng, Bradley McDanel, and Sai Qian Zhang. Speculative decoding and beyond: An in-depth survey of techniques. *arXiv preprint arXiv:2502.19732*, 2025.
 - [16] Jitesh Jain, Zhengyuan Yang, Humphrey Shi, Jianfeng Gao, and Jianwei Yang. Ola-vlm: Elevating visual perception in multimodal llms with auxiliary embedding distillation. *arXiv preprint arXiv:2412.09585*, 2024.
 - [17] Byung-Kwan Lee, Ryo Hachiuma, Yu-Chiang Frank Wang, Yong Man Ro, and Yueh-Hua Wu. Vlsi: Verbalized layers-to-interactions from large to small vision language models. *arXiv preprint arXiv:2412.01822*, 2024.
 - [18] Minjae Lee, Wonjun Kang, Minghao Yan, Christian Classen, Hyung Il Koo, and Kangwook Lee. In-batch ensemble drafting: Toward fast and robust speculative decoding for multimodal language models.
 - [19] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.
 - [20] Bohan Li, Hankun Wang, Situo Zhang, Yiwei Guo, and Kai Yu. Fast and high-quality auto-regressive speech synthesis via speculative decoding. *arXiv preprint arXiv:2410.21951*, 2024.
 - [21] Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. Seed-bench-2: Benchmarking multimodal large language models. *arXiv preprint arXiv:2311.17092*, 2023.
 - [22] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.
 - [23] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022.
 - [24] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-2: Faster inference of language models with dynamic draft trees. *arXiv preprint arXiv:2406.16858*, 2024.
 - [25] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.
 - [26] Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle-3: Scaling up inference acceleration of large language models via training-time test. *arXiv preprint arXiv:2503.01840*, 2025.
 - [27] Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Less is more: task-aware layer-wise distillation for language model compression. In *Proceedings of the 40th International Conference on Machine Learning*, pages 20852–20867, 2023.
 - [28] Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding via double early exiting. *arXiv preprint arXiv:2404.18911*, 2024.
 - [29] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023.

- [30] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024.
- [31] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.
- [32] Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. Speculative decoding via early-exiting for faster llm inference with thompson sampling control mechanism. *arXiv preprint arXiv:2406.03853*, 2024.
- [33] Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, and Winston Hu. Parallel speculative decoding with adaptive draft length. *arXiv preprint arXiv:2408.11850*, 2024.
- [34] Xukun Liu, Bowen Lei, Ruqi Zhang, and Dongkuan Xu. Adaptive draft-verification for efficient large language model decoding. *arXiv preprint arXiv:2407.12021*, 2024.
- [35] Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12), December 2024.
- [36] Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations (ICLR)*, 2024.
- [37] Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *The 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.
- [38] Andrés Marafioti, Orr Zohar, Miquel Farré, Merve Noyan, Elie Bakouch, Pedro Cuenca, Cyril Zakka, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Vaibhav Srivastav, Joshua Lochner, Hugo Larcher, Mathieu Morlon, Lewis Tunstall, Leandro von Werra, and Thomas Wolf. Smolvlm: Redefining small and efficient multimodal models. *arXiv preprint arXiv:2504.05299*, 2025.
- [39] Ahmed Masry, Do Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, Dublin, Ireland, May 2022. Association for Computational Linguistics.
- [40] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781*, 2023.
- [41] Giovanni Monea, Armand Joulin, and Edouard Grave. Pass: Parallel speculative sampling. *arXiv preprint arXiv:2311.13581*, 2023.
- [42] Jie Ou, Yueming Chen, and Wenhong Tian. Lossless acceleration of large language model via adaptive n-gram parallel decoding. *arXiv preprint arXiv:2404.08698*, 2024.
- [43] Desh Raj, Gil Keren, Junteng Jia, Jay Mahadeokar, and Ozlem Kalinli. Faster speech-llama inference with multi-token prediction. *arXiv preprint arXiv:2409.08148*, 2024.
- [44] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets, 2015.
- [45] Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. *arXiv preprint arXiv:2305.10427*, 2023.
- [46] Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models, 2025.
- [47] Benjamin Spector and Chris Re. Accelerating llm inference with staged speculative decoding. *arXiv preprint arXiv:2308.04623*, 2023.
- [48] Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.
- [49] Lawrence Stewart, Matthew Trager, Sujun Kumar Gonugondla, and Stefano Soatto. The n-grammys: Accelerating autoregressive inference with learning-free batched speculation. *arXiv preprint arXiv:2411.03786*, 2024.

- [50] Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *arXiv preprint arXiv:2404.11912*, 2024.
- [51] Siqi Sun, Zhe Gan, Yuwei Fang, Yu Cheng, Shuohang Wang, and Jingjing Liu. Contrastive distillation on intermediate representations for language model compression. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 498–508, 2020.
- [52] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport. *Advances in Neural Information Processing Systems*, 36, 2024.
- [53] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.
- [54] Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, et al. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*, 2024.
- [55] Heming Xia, Tao Ge, Peiyi Wang, Si-Qing Chen, Furu Wei, and Zhifang Sui. Speculative decoding: Exploiting speculative execution for accelerating seq2seq generation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3909–3925, 2023.
- [56] Heming Xia, Yongqi Li, Jun Zhang, Cunxiao Du, and Wenjie Li. Swift: On-the-fly self-speculative decoding for llm inference acceleration. *arXiv preprint arXiv:2410.06916*, 2024.
- [57] Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. Inference with reference: Lossless acceleration of large language models. *arXiv preprint arXiv:2304.04487*, 2023.
- [58] Kaining Ying, Fanqing Meng, Jin Wang, Zhiqian Li, Han Lin, Yue Yang, Hao Zhang, Wenbo Zhang, Yuqi Lin, Shuo Liu, Jiayi Lei, Quanfeng Lu, Runjian Chen, Peng Xu, Renrui Zhang, Haozhe Zhang, Peng Gao, Yali Wang, Yu Qiao, Ping Luo, Kaipeng Zhang, and Wenqi Shao. Mmt-bench: A comprehensive multimodal benchmark for evaluating large vision-language models towards multitask agi, 2024.
- [59] Zhengqing Yuan, Zhaoxu Li, Weiran Huang, Yanfang Ye, and Lichao Sun. Tinygpt-v: Efficient multimodal large language model via small backbones. *arXiv preprint arXiv:2312.16862*, 2023.
- [60] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.
- [61] Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. Draft & verify: Lossless large language model acceleration via self-speculative decoding. *arXiv preprint arXiv:2309.08168*, 2023.
- [62] Libo Zhang, Zhaoning Zhang, Baizhou Xu, Songzhu Mei, and Dongsheng Li. Dovetail: A cpu/gpu heterogeneous speculative decoding for llm inference. *arXiv preprint arXiv:2412.18934*, 2024.
- [63] Baichuan Zhou, Ying Hu, Xi Weng, Junlong Jia, Jie Luo, Xien Liu, Ji Wu, and Lei Huang. Tinyllava: A framework of small-scale large multimodal models. *arXiv preprint arXiv:2402.14289*, 2024.
- [64] 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. *arXiv preprint arXiv:2310.08461*, 2023.

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 accurately capture the paper's main contributions and scope. They clearly outline the proposed DREAM framework, including cross-attention fusion, entropy-based feature selection, and visual token compression, all of which are thoroughly developed and validated in the paper.

Guidelines:

- The answer NA 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 NA 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: Yes, the limitation is discussed in the conclusion section.

Guidelines:

- The answer NA 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: [No]

Justification: This paper does not involve theoretical results.

Guidelines:

- The answer NA 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: Yes, the paper provides sufficient information to reproduce the main experimental results to the extent that supports its core claims and conclusions. It details the model architectures, training procedures, datasets used (e.g., LLaVA mix665k), loss functions, hyperparameters, evaluation benchmarks, and metrics. We will release the code upon acceptance of the work.

Guidelines:

- The answer NA 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: We have provided the code with the supplementary materials.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.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://nips.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, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: Yes, the paper specifies the key training and testing details necessary to understand the results, as depicted in the beginning of Section 4.

Guidelines:

- The answer NA 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: [\[Yes\]](#)

Justification: Yes, the paper specifies the error bars and their calculation details in the appendix, or as supplemental material.

Guidelines:

- The answer NA 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: We describe the hardware resources information in the beginning of Section 4.

Guidelines:

- The answer NA 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 paper conforms with the NeurIPS Code of Ethics, providing transparency in methodology and acknowledging limitations.

Guidelines:

- The answer NA 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: [\[Yes\]](#)

Justification: We discuss it in the conclusion section of the paper.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We do not introduce any new models or datasets in this work.

Guidelines:

- The answer NA 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: Yes, the paper properly credits the original creators of all external assets used, including models, datasets, and baseline methods. Citations are provided for works such as LLaVA, Pixtral, and various speculative decoding methods.

Guidelines:

- The answer NA 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: [NA]

Justification: The paper does not release new assets.

Guidelines:

- The answer NA 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: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA 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: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA 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 rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA 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 (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.