IN-BATCH ENSEMBLE DRAFTING: TOWARD FAST AND ROBUST SPECULATIVE DECODING FOR MULTIMODAL LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal Large Language Models (MLLMs) have emerged as powerful tools for processing modalities beyond text by combining a visual encoder with Large Language Models (LLMs) to incorporate visual context. This integration, however, leads to higher computational costs during LLM inference, specifically in the *Prefill* and *Decoding* stages. Existing MLLM acceleration methods primarily focus on reducing the cost of long prefills caused by visual context, but this approach has limitations: (1) From a latency perspective, it mainly benefits the prefill stage, offering minimal improvements for decoding. (2) It does not guarantee output distributions that are identical to those of the original MLLM. To ensure identical output distribution while mitigating decoding latency, we focus on speculative decoding (SD)—an acceleration technique that uses a smaller draft model verified by a larger model. Despite its importance for LLM acceleration, SD's application to MLLMs remains largely unexplored, even though decoding constitutes a significant portion of MLLM inference latency. We investigate various drafting techniques—multimodal, text-only, image-pooling, and caption-based—for multimodal scenarios and analyze their integration with MLLMs. Building on these insights, we propose *In-batch Ensemble Drafting (IbED)*, which combines probability distributions from multiple drafting methods via batch inference during the SD draft phase. This approach requires no additional model parameters, incurs minimal overhead, and significantly increases the likelihood of draft tokens passing verification, thereby enhancing performance and robustness across diverse input scenarios.

1 INTRODUCTION

Large Language models are rapidly advancing, and in particular, Multimodal Large Language Models (MLLMs) that can process various modalities beyond text are gaining significant attention (OpenAI, 2023; Anthropic, 2024; Gemini Team Google: Anil et al., 2023). MLLMs share the characteristics of LLMs (Brown et al., 2020; Ouyang et al., 2022; Touvron et al., 2023), which include: (1) The *Prefill Stage*, involving parallel processing of the provided input context. (2) The *Decoding Stage*, where generation is performed through an autoregressive decoding method based on the processed context. Specifically, decoding n tokens requires a total of n serial runs of the model. In addition, MLLMs require an extra process before the decoding stage: (3) The *Vision Encoder Stage*, where image inputs are converted into visual context tokens by embedding patches through a visual encoder (Radford et al., 2021). Typically, each image yields several hundred visual context tokens.

As a result, the computational cost of inference with MLLMs has significantly increased. To mitigate this cost, various methodologies have been proposed to accelerate MLLMs by focusing on reducing the number of visual tokens. These approaches include dynamically retaining only the most important visual tokens based on attention sparsity (Shang et al., 2024), layer-wise pruning of less significant visual tokens to enhance efficiency (Chen et al., 2024b; Lin et al., 2024), and reducing redundant key-value caches through consolidation and compression strategies (Liu et al., 2024b; Wan et al., 2024). Despite effectively minimizing performance degradation from the original model, these approaches have fundamental limitations: (1) From a latency perspective, reducing prefill length mainly benefits the prefill stage while offering negligible advantages for the decode

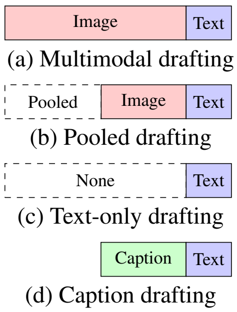


Figure 1: Prompt candidates for drafting.

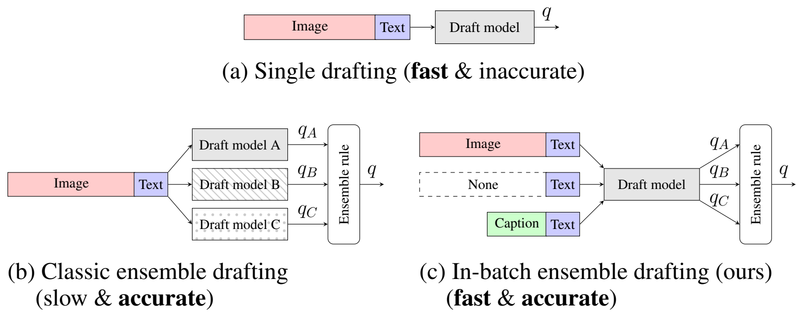


Figure 2: Framework of single drafting, classic ensemble drafting, and in-batch ensemble drafting (ours). Each figure visualizes single-token generation per timestep.

stage; and (2) they inherently rely on approximation, which does not guarantee an identical output to that of the original MLLM.

Recently, Speculative Decoding (SD) (Leviathan et al., 2023; Chen et al., 2023) has been rapidly emerging in the field of LLMs. It accelerates language models while preserving the output distribution generated by the model, offering a quality-neutral advantage. Specifically, SD methods split the decoding process into two distinct stages: (1) a *Draft Phase*, where a small "draft" model sequentially creates low-cost tokens; and (2) a *Verification Phase*, where a large "target" model reviews these draft tokens in parallel. The efficiency comes from the insight that combining autoregressive decoding with a small model for drafting, followed by parallel verification with a large model, reduces costs by avoiding the iterative process, compared to using the large model alone for autoregressive decoding. In the LLM field, various attempts have been made to enhance acceleration through SD, such as performing knowledge distillation on the draft model (Zhou et al., 2024), generating multiple draft candidates to find a better draft (Sun et al., 2024b), or altering the verification phase based on a tree structure (Miao et al., 2023b).

However, to the best of our knowledge, research on Speculative Decoding for MLLMs has been far less explored, with only one study (Gagrani et al., 2024) available. This paper is significant as the first attempt to apply SD to MLLMs, demonstrating that a draft model with multimodality processing capabilities can surprisingly accelerate the target MLLM even when it does not use the image input. However, the paper did not factorize and analyze the time cost associated with choosing each drafting method, and it also has limitations in that it is impossible to know in advance which drafting method to choose when none shows consistent superiority, leaving these questions for future work.

In this study, we present a comprehensive analysis aimed at elucidating the fundamental principles of Multimodal Large Language Model (MLLM) Speculative Decoding across diverse input scenarios. Based on extensive benchmarking, we primarily focus on a comparative analysis between multimodal drafting and text-only drafting approaches.

Secondly, we explore several key questions that arise during the drafting stage when applying SD to MLLMs: Can a very small model effectively handle multimodality, which often results in long context lengths during the *Prefill Stage*? Is it necessary for such a small draft model to process the lengthy context derived from image inputs? And can effective drafting still occur if this long image context is compressed or replaced by much shorter text?

Lastly, we propose *In-batch Ensemble Drafting (IbED)*. Based on our observation that different drafting approaches available in the SD for MLLM settings have unique advantages (Figure 5), this method combines the probability distributions from these approaches by batch inference to decode each draft token during the *Draft Phase* of SD (Figure 2). Unlike conventional ensembles, it requires no additional model parameters, resulting in negligible cost. This approach significantly improves the likelihood of draft tokens passing target model verification and enhances performance across tasks and datasets, making it more robust. Furthermore, it can be effectively integrated with existing MLLM acceleration techniques that focus on the *Prefill stage* and SD methods optimized for the *verification phase*.

Our method demonstrates a 2-10% performance improvement compared to multimodal drafting for single-image and two-image scenarios. Moreover, in cases involving five images where multimodal drafting's performance significantly deteriorates, our method maintains stable performance, even surpassing that of text-only approaches.

In summary, the main contributions of our work are:

- We conduct an extensive benchmark of Multimodal Large Language Model for Speculative Decoding, focusing on a comparative analysis between multimodal drafting and text-only drafting approaches across diverse input scenarios.
- We investigate various drafting methods available for MLLM acceleration by testing the necessity of image input during drafting, compressing, or replacing the long context from images with other modalities. We open-source our custom-trained draft MLLM, evaluated on various tasks, along with its recipe.
- We introduce *In-batch Ensemble Drafting (IbED)*, which combines various drafting methods with negligible cost, achieving greater speed-ups and robust performance across diverse scenarios.

2 Related Work

2.1 Multimodal Large Language Models

MLLMs Frontier proprietary MLLMs (OpenAI, 2023; Anthropic, 2024; Gemini Team Google: Anil et al., 2023) demonstrate state-of-the-art performance across multimodalities beyond just text. Meanwhile, open-source models like the LLaVA series (Liu et al., 2023; 2024a; Li et al., 2024b;a) and LLaMA 3.2 (Dubey et al., 2024) are also rapidly advancing. While various methods exist for embedding image inputs (Yin et al., 2024; Jin et al., 2024), one of the most prominent approaches, LLaVA, employs an off-the-shelf vision encoder (Radford et al., 2021; Zhai et al., 2023) and a trainable projector to convert its output into the visual tokens of an LLM.

Inference Acceleration for MLLMs To address the inefficiency of handling visual tokens from images, several approaches have been proposed based on a common finding: only a sparse subset of the hundreds of visual tokens is important, allowing for reduced computational cost with minimal information loss. Shang et al. (2024); Chen et al. (2024b); Lin et al. (2024) dynamically prune significant visual tokens based on attention sparsity. Further focusing on reducing redundant key-value caches, (Liu et al., 2024b; Wan et al., 2024) retain key-value vectors by merging or discarding less critical caches during output generation. However, from a latency perspective, these approaches primarily benefit the prefill stage while providing negligible advantages for the decode stage.

2.2 Speculative Decoding

Speculative Decoding for LLMs Although forefront LLMs demonstrate revolutionary performance (Brown et al., 2020; OpenAI, 2023; Anthropic, 2024), deploying these large models is computationally intensive, posing significant challenges to serving efficiency. To improve the inference process for large language models, various approaches have been proposed, ranging from algorithmic innovations to system optimizations (Miao et al., 2023a; Khoshnoodi et al., 2024).

Recently, Speculative Decoding (Leviathan et al., 2023; Chen et al., 2023) has gained significant attention for accelerating inference using a small draft model while preserving the model's output distribution. To improve the drafting stage in SD, various efforts have been made, including generating multiple draft candidates to select the best one (Sun et al., 2024b; Yang et al., 2024), and finetuning the draft model with knowledge distillation (Zhou et al., 2024). On the other hand, some research focuses on modifying the verification phase using a tree structure (Miao et al., 2023b). Additionally, several studies address cases with exceptionally long prefill lengths (e.g., 100k), which significantly affect decoding efficiency (Sun et al., 2024a; Chen et al., 2024a). These studies systematically varied prefill length and batch size, finding that with prefill lengths of 1k to 10k tokens and low batch sizes, decoding speed is generally unaffected, which is typical of real-world multimodal input scenarios.

Speculative Decoding for MLLMs Most relevant to our work, Gagrani et al. (2024) conducted the first study on speculative decoding for MLLMs, advocating the use of a draft model for text-only drafting (i.e., without multimodal input). However, the paper does not provide extensive analysis between multimodal drafting and text-only drafting approaches across diverse input scenarios, and it lacks clarity on the source of speedup for text-only drafting—is it due to lower per-token latency or a higher likelihood of passing the target model's verification? Additionally, it is unclear which drafting method to choose when none consistently performs best, limiting the effective use of multiple drafting strategies. Lastly, we cannot reproduce or verify these issues or explore other possible draftings, as the training recipe and model checkpoints are not publicly available.

3 PRELIMINARIES

3.1 THEORETICAL LATENCY OF TRANSFORMERS

Compute bounded vs Memory Bounded The latency bottlenecks in transformer models can be categorized into two primary constraints: compute-boundedness and memory-boundedness. Compute-bound operations are limited by processing speed, typically during matrix calculations and attention mechanisms. Memory-bound scenarios arise when available memory becomes a limiting factor, often due to large model sizes or long input sequences. Arithmetic intensity, the ratio of computational operations to memory operations, bridges these concepts and influences overall efficiency. High arithmetic intensity operations tend to be compute-bound, while low intensity operations are often memory-bound. In transformers, this balance varies depending on the generation phase (i.e., prefill or decode), model architecture, hardware specifications, and other factors.

Prefilling Since prefilling requires parallel computations for a large number of tokens, it is compute-bound, leading to significant increases in latency as the prefill length grows. In the case of MLLMs, the proportion of visual tokens within the prefill length is significantly large. Therefore, addressing the redundancy of visual tokens is essential for cost-efficient prefilling.

Decoding Because only one token is processed at each step during decoding, the process is *memory-bound*. The memory access cost is divided between the model weights and the key-value cache. Except for long contexts, model weights dominate this cost. Consequently, decoding latency remains nearly constant regardless of context length. Similarly, parallel decoding with a small number of tokens—as in the verification stage of speculative decoding—or slightly increasing the batch size from 1 has minimal impact on latency.

3.2 SPECULATIVE DECODING

We briefly outline how SD works, using mathematical notations following (Leviathan et al., 2023; Zhou et al., 2024).

Overview Let M_p be the larger "target" model, whose inference we aim to accelerate, and let M_q be the smaller "draft" model for the same task. For a given prefix $x_{<t}$ and $n = 0, \dots, \gamma - 1$, following steps are repeated until either an end-of-sequence token is accepted or the maximum sequence length is reached.:

(1) A *Draft phase*, where M_q sequentially generates γ draft tokens from $q(\cdot|x_{<t+n})$.

(2) A *Verification phase*, where M_p reviews these draft tokens in parallel, comparing them to $p(x_{t+n}|x_{<t+n})$.

(3) For sampling, each token x_{t+n} is sequentially accepted with probability $\min\left(1, \frac{p(x_{t+n}|x_{<t+n})}{q(x_{t+n}|x_{<t+n})}\right)$. If any token is rejected before the end of the block, subsequent tokens are discarded, and the rejected token is resampled from the adjusted distribution $\text{norm}(\max(0, p(x) - q(x)))$.

Given input, *block efficiency* $\tau_{p,q}(\gamma)$ is defined as the expected number of accepted tokens per block. For a fixed γ, the maximum block efficiency is $\gamma+1$, which occurs when all draft tokens are accepted and an additional token is sampled by the target model.

Wall-clock Time Improvement Following Chen et al. (2024a), for a given sequence length S, we use the notation $T_p(S, 1)$ and $T_q(S, 1)$ to indicate the required time for M_p and M_q, respectively, to decode a single token. Similarly, $T_V(S, \gamma)$ represents the required time for M_p to verify γ tokens in parallel. If we ignore the latency of the prefilling stage, we can see the wall-clock time improvement as:

$$\text{Token rate (target)} = \frac{1}{T_p}, \quad \text{Token rate (SD)} = \frac{\tau_{p,q}(\gamma)}{\gamma \cdot T_q + T_V(\gamma)}, \tag{1}$$

$$\text{Speed up} = \frac{\text{Token rate (SD)}}{\text{Token rate (target)}} = \frac{\tau_{p,q}(\gamma)}{\gamma \cdot \frac{T_q}{T_p} + \frac{T_V(\gamma)}{T_p}} \approx \frac{\tau_{p,q}(\gamma)}{\gamma \cdot \frac{T_q}{T_p} + 1}, \tag{2}$$

Note that the decoding stage is memory bound and $\frac{T_V(\gamma)}{T_p}$ converges to 1 if we assume a single batch scenario (Chen et al., 2024a; Fu, 2024). For a given M_p, the choice of M_q determines both the block efficiency τ and the *draft-to-target latency ratio* $\frac{T_p}{T_q}$. Notably, this ratio remains consistent, even as the long context varies from under 1K to 3K stemming from the image modality. In our setting, we empirically demonstrate this consistency in Appendix G.2. To improve the throughput of speculative decoding, one should focus on improving block efficiency $\tau_{p,q}$.

4 Analysis of Speculative Decoding for MLLMs

In this section, we systematically study speculative decoding for MLLMs, evaluating the performance of multimodal and text-only drafting across various benchmark datasets.

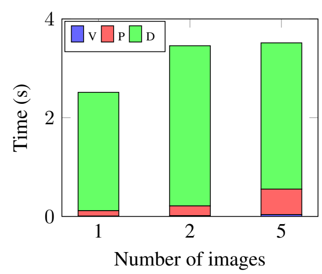


Figure 3: Time analysis of the target model's inference process. Each bar corresponds to prefill lengths of 600, 1200, and 3000 tokens respectively.

4.1 Experiment Settings

Models: Target and Draft Models We employ LLaVA-1.5 7B (Liu et al., 2024a) as the target model to accelerate. For our draft model, we perform visual instruction tuning on LLaMA 68M—used as the draft model in SpecInfer (Miao et al., 2023b)—by following the training approach of the target model. We design our draft model with practical use in mind, aiming to accelerate the target MLLM through speculative decoding while considering deployment costs. We evaluated the model on language and vision-language tasks and observed that the trained model has the ability to perceive multimodality [1] (see Appendix H).

Note on the Draft Model To effectively accelerate the target MLLM using speculative decoding, the relative speed of the draft model to the target model—represented by $\frac{T_p}{T_q}$ in Equation (1)—is crucial[2] (the ratio $\frac{T_p}{T_q}$ remains approximately constant across moderate context lengths, as described in Section 3.1).

Benchmark Datasets and Tasks Selecting benchmark datasets is crucial for evaluating performance; however, a benchmark for MLLM speculative decoding has not yet been established. Therefore, we carefully reviewed existing multimodal datasets for single-image and multi-image settings (with 2 and 5 images) and curated a set of benchmark datasets specifically for MLLM speculative decoding. Details of the benchmark datasets are provided in Appendix B. We construct a question-answering task for all datasets using prompts that guide the model to describe the answer and reasoning, allowing it to interpret the question and image descriptively (see Appendix C for prompt details). This setup aligns with typical MLLM use cases like ChatGPT.

[1] We also release the trained checkpoint of this model.

[2] One might consider LLaVA-Next-Interleave 0.5B—the smallest carefully trained off-the-shelf MLLM—as a draft model, its latency ratios $\frac{T_p}{T_q}$ in Equation (1) to the 7B and 70B models in the same series exceed 0.5 and 0.1 respectively (QwenTeam), making it unsuitable for achieving speed-up. The speed depends not only on parameter count but also on the depth-width trade-off of the architecture (Yan et al., 2024).

Draftings: Multimodal and Text-only The multimodal drafting process is the same as the general MLLM generation process. After concatenating text embeddings and image embeddings, which are obtained by passing images through a vision encoder and projector, the prefill process is performed in the language model. Then, tokens are decoded up to a predefined chunk length γ. In our setting, each image is converted into an embedding of length 576 and $\gamma = 5$. In contrast, text-only drafting, whose potential was first recognized in (Gagrani et al., 2024), eliminates the image input and relies solely on textual data as input for the draft model. Its generation process then follows that of a standard LLM. All drafting is performed using greedy decoding with a batch size of 1. The maximum number of newly generated tokens is fixed at 128. See Appendix C for details on the prompt used for each drafting.

4.2 Time Analysis for Generation Process of MLLM

By adopting the perspective of LLM acceleration, we divide the generation process of MLLMs into *vision token processing*, *prefill*, and *decoding stages* to identify bottlenecks.

Target Model: Generation for the Whole Sequence We visualize the factorized generation time in Figure 3. The time taken in the vision encoder and prefill stages is proportional to the number of images. Since each image is converted into several hundred context tokens and processed through the prefill stage, images have a greater impact than text tokens. Decoding time is more variable than the previous two stages, as the number of decoded tokens depends on the input scenario and the model's learned distribution. We selected the TextVQA, Spot, and PororoSV datasets to represent datasets containing 1, 2, and 5 images, respectively. Though maximum number of newly generated tokens is fixed, the resulting number of decoded tokens for each dataset in average is 91.89, 116.52, and 88.17, respectively. In conclusion, the latency induced by the decoding phase exceeds the combined latency of the other two stages.

Draft Model: Chunk-wise Generation by Drafting The timing trends for the draft model in the vision encoder and prefill stages align with those of the target model. Multimodal drafting, which involves processing through a vision encoder, transforms a single image into several hundred tokens, thereby incurring a higher prefill cost compared to text-only drafting, which operates with a shorter text context. However, the absolute scale of this cost is very small and can be overshadowed by the target model's prefill time. As shown in Figure 9, the per-step latency for decoding tokens remains consistent up to a context length of 3K (equivalent to around five input images), indicating no difference in token rate due to the longer context (Section 3.1). Consequently, the ratio $\frac{T_q}{T_p}$ in Equation (1), induced by the draft model, remains unchanged regardless of the long context from the image modality. Since this factor remains the same regardless of the drafting method, the speed-up in the decoding phase primarily depends on block efficiency γ. The following discussions will focus on speed-up in terms of block efficiency.

4.3 Block Efficiency and Speed-up by Drafting

Table 1a shows the block efficiency results of multimodal drafting and text-only drafting on various benchmark datasets. Multimodal drafting provide relatively higher block efficiency (speed-up) then text-only drafting when there are one or two images in the input. However, when we broaden the input scenario to cases with five images, the tendencies of the drafting methods are completely reversed, and the performance drop of text-only drafting for multi-image cases is much less than that of multimodal drafting. Figure 4 illustrates how multimodal and text-only drafting differ in the tokens they generate when the image and text prompt are fixed. For example, multimodal drafting, which references the image, can generate 'Zane', whereas text-only drafting cannot.

4.4 Summary: Draftings

The relative performance of multimodal versus text-only drafting methods varies depending on the input scenario, with no consistent winner. While multimodal drafting often provides a higher speed-up, it is less robust compared to text-only drafting. Therefore, it's difficult to know in advance which method is better before execution, and even if known, it is difficult to address with a single drafting.

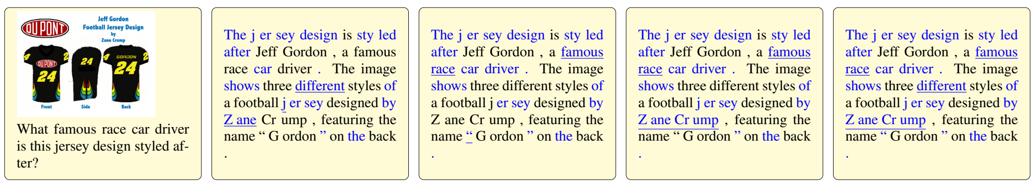
Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
LLaVA 1.5	7B / 68M	multimodal	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		text-only	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05

(a) Block efficiency results of multimodal drafting and text-only drafting.

Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
LLaVA 1.5	7B / 68M	multimodal	1.71	1.61	1.72	1.82	1.78	1.67	0.91	0.88	1.71	1.72	0.89
		text-only	1.69	1.55	1.68	1.78	1.73	1.70	1.56	1.56	1.68	1.71	1.56

(b) Speed up results of multimodal drafting and text-only drafting.

Table 1: Speculative decoding results by multimodal drafting and text-only drafting.



(a) Instruction (b) Multimodal (c) Text-only (d) Caption (e) Ensemble

Figure 4: Qualitative samples from the TextVQA dataset by various drafting methods: multimodal, text-only, caption, and in-batch ensemble. Blue tokens denote acceptance by the target model. The image caption obtained by the lightweight image captioning model is "A football jersey design by Zane Crump is shown."

5 EXPLORING DRAFTING METHODS FOR MLLMS

In this section, we address several questions that arise during the drafting stage when applying SD to MLLMs, particularly about the necessity of images for drafting and the potential to substitute the image modality, considering that the draft model is imperfect and its performance can vary significantly across input scenarios.

5.1 HOW NECESSARY IS THE IMAGE MODALITY FOR DRAFTING?

What if we include image information in the draft model but compress its context? According to previous studies (Shang et al., 2024; Chen et al., 2024b), although image tokens are more numerous than text tokens, their importance is relatively sparse, receiving meaningful attention only in certain layers. Therefore, we can compress these image tokens, using this approach as a simple proxy for previous work aimed at reducing image prefill tokens.

Multimodal Drafting with Image-pooling To compress image information, we performed average pooling, preserving the 2D spatial structure of the image just before it is transformed into the text representation space by the projector. Image prefill tokens are then created by passing the pooled data through the projector. The notation pool (n) indicates the number of visual tokens remaining after pooling from the original 576 tokens per image. Since this compression is parameter-free, the cost is negligible. Additionally, we conducted experiments during the instruction fine-tuning stage (one of the two stages of training a pretrained LM into an MLLM), where we trained the model while pooling the images at the same compression rates.

Experimental Results From the perspective of block efficiency, for both the single image dataset and the multi-image dataset with $n = 2$, the results after pooling were slightly worse than those without pooling. However, they still outperformed the text-only approach for the single image dataset. This indicates that even the pooled visual tokens exhibit a certain level of image awareness.

However, multimodal drafting with pooling demonstrated significantly better performance than multimodal drafting without pooling on a multi-image dataset with $n = 5$. Reducing the tokens from

Target / Draft			$n = 1$				$n = 2$		$n = 5$		$n = 1$	$n = 2$	$n = 5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
LLaVA 1.5	7B / 68M	multimodal	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		pool (144)	2.23	2.08	2.26	2.36	2.23	2.22	2.07	2.09	2.23	2.23	2.08
		pool (36)	2.17	2.01	2.21	2.32	2.20	2.23	2.05	2.06	2.18	2.21	2.05
		pool (9)	2.20	2.03	2.21	2.34	2.25	2.24	2.06	2.08	2.20	2.25	2.07
		pool (1)	2.23	2.03	2.23	2.37	2.25	2.26	2.06	2.07	2.21	2.25	2.06
		text-only	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05
		caption	2.28	2.08	2.24	2.41	2.31	2.29	2.08	2.10	2.25	2.30	2.09

Table 2: Block efficiency results of pooled multimodal drafting and caption drafting.

576 to just 144 significantly decreases the number of tokens, making multimodal drafting—which has limited capacity to process a large number of images—more robust. In the case of multimodal drafting with a model fine-tuned through pooling, the trend was maintained while performance improved (see Table 2). To see the full results, refer Appendix D.

5.2 Can We Replace Image Modality with Another one for Drafting?

Even without the image modality (i.e., text-only drafting), we observed substantial speed-ups (block efficiency), along with more robust performance compared to multimodal drafting. Therefore, is the image modality truly necessary for drafting? In this section, we investigate how injecting image information into a text-only draft model *without* providing image input can enhance block efficiency.

Caption Drafting One of the most straightforward ways to map images to the text modality is through captions. In our experimental setup, we employ a lightweight image captioning model to generate captions for each image, using these captions as input for the text-only draft model instead of the images themselves. We used BLIP (Li et al., 2022; 2023) and Florence (Xiao et al., 2024) as lightweight image captioning models. The captioning model only needs to perform inference once during the prefill, with latency shorter than the prefill time of the target model. Further details of the image captioning models are provided in Appendix E.

Experimental Results As shown in Table 2, caption-based drafting showed improvements over text-only drafting from the perspective of block efficiency. Fig. 4 shows that caption drafting outperforms text-only and multimodal drafting in image comprehension, as the lightweight captioning model extracts specific details like "Zane Crump." Furthermore, we conducted a detailed investigation into which tokens each drafting method successfully decoded (i.e., passed the target model's verification) and which tokens it failed to decode. As shown in Figure 5, no single drafting method encompassed all the tokens correctly predicted by the others. Full experimental results of caption drafting are provided in Appendix E.

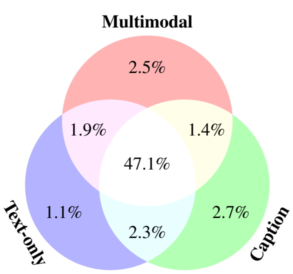


Figure 5: Venn diagram of the accepted rates for each drafting method on the ChartQA dataset.

6 In-batch Ensemble Drafting

Algorithm 1 In-batch Ensemble Drafting (IbED)

Input: Generated sequence $x_{1:t}$ until current step t
Parameter: Prompt list $[c_1, ..., c_n]$ ▷ multimodal, text-only, caption, ...
Output: Next predicted token x_{t+1}

procedure IBED($x_{1:t}$; $[c_1, ..., c_n]$)
 $q_1, q_2, ..., q_n =$ BATCHINFERENCE($[c_1 + x_{1:t}, c_2 + x_{1:t}, ..., c_n + x_{1:t}]$)
 q = AVERAGE($q_1, q_2, ..., q_n$)
 $x_{t+1} =$ SAMPLE(q)
 return x_{t+1}
end procedure

To summarize our conclusions so far: (1) the draft model is not perfect, and even with the same model, different drafting methods can be applied depending on the input scenario; (2) each approach shows distinct advantages in achieving 'robust speed-up,' as demonstrated through experiments across various scenarios using representative drafting methods—multimodal, text-only, caption, and pooled. The main issue, however, is that these pros and cons are not easily predictable without extensive testing across multiple scenarios.

6.1 Why "In-batch" Ensemble?

Unlike typical ensemble learning, which requires multiple models with different parameters, in-batch ensemble drafting works differently. The model parameters are shared across all drafting methods, and each drafting outputs differently based on the varying context by batch inference. Increasing the batch size for a small draft model is nearly cost-free. Since the Transformer model's decoding stage is memory-bound, for moderate context lengths and small batch sizes (4 or fewer), the latency of multi-batch inference converges to that of single-batch inference (Fu, 2024). To empirically demonstrate this in our setting, we measured the per-step latency for token decoding with different batch sizes, as shown in Figure 9. The latency gap between batch size 1 and larger sizes is less than 0.1ms, resulting in a negligible computational cost. Based on Eq. (1), if we assume $T_q/T_p = 0.05$ and $\tau_{q,p}(\gamma) = 2.5$, the difference in speed-up between using the draft model with a batch size of 2 versus 1 is mcuh less than 1%.

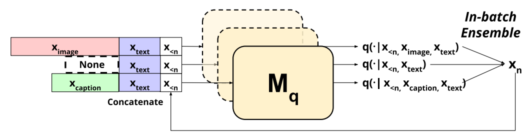


Figure 6: Framework of In-batch Ensemble Drafting (IbED). Given an input scenario, all draftings share the parameters of the draft model M_q, and the resulting distributions are ensembled to sample the next token in the draft candidate. For details, see Algorithm 1.

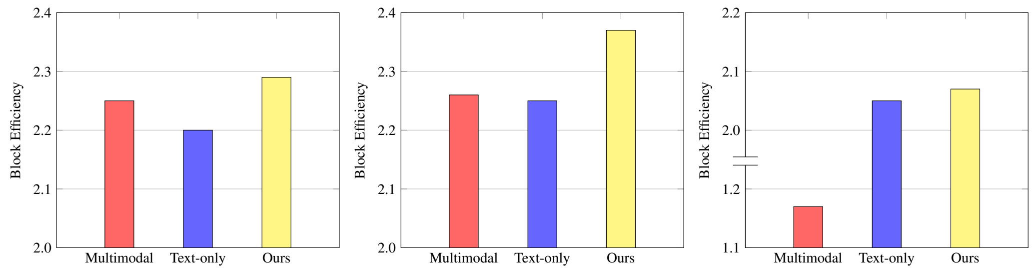
6.2 How to Proceed with In-batch Ensemble

As discussed earlier, we use four types of drafting for ensemble learning: multimodal drafting (M), text-only drafting (T), caption drafting (C), and pooled multimodal drafting (P). For each decoding timestep, we apply a simple weighted averaging ensemble method, and then sample a token from the averaged distribution to continue drafting. We use equal weight ratios for all ensemble drafting methods to demonstrate effectiveness without hyperparameter tuning: 1:1 for MT and MC, 1:1:1 for MTC, and 1:1:1:1 for MTCP. Full experimental results with different weight settings are provided in Appendix E.

Experimental Results Table 3 and Fig. 7 illustrate the block efficiency results of ensemble drafting. In comparison to single drafting, ensemble drafting demonstrates superior block efficiency across most datasets, exhibiting not only improved average performance but also consistent enhancement across all datasets. Notably, when $n = 5$, ensemble drafting achieves performance comparable to or surpassing text-only methods, despite the inclusion of less effective multimodal drafting techniques. This outcome demonstrates the robustness of ensemble drafting, which is particularly significant given that the ensemble was constructed using equal weight ratios. Full experimental results with different weight settings are provided in Appendix E.

Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
LLaVA 1.5	7B / 68M	M	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		T	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05
		C	2.28	2.08	2.24	2.41	2.31	2.29	2.08	2.10	2.25	2.30	2.09
		P	2.23	2.08	2.26	2.36	2.23	2.22	2.07	2.09	2.23	2.23	2.08
		MT	2.26	2.13	2.27	2.39	2.40	2.31	1.94	1.91	2.26	2.35	1.92
		MC	2.30	2.17	2.29	2.42	2.39	2.32	1.99	1.93	2.29	2.35	1.96
		MTC	2.29	2.15	2.28	2.41	2.41	2.30	2.08	2.06	2.28	2.35	2.07
		MTCP	2.29	2.17	2.29	2.42	2.41	2.33	1.99	1.93	2.29	2.37	1.96

Table 3: Block efficiency results of ensemble drafting.



(a) Single image datasets (b) Multi image datasets ($n = 2$) (c) Multi image datasets ($n = 5$)

Figure 7: Performance comparison of speculative decoding on various datsets. Our method achieves the best and most robust block efficiency results compared to multimodal and text-only drafting.

7 Limitations and Future Works

Integration with Acceleration Methods While our work focuses on a single draft candidate and single verification scheme to understand the fundamentals of multimodal speculative decoding, other approaches use multiple draft candidates (Yang et al., 2024; Cai et al., 2024) and multi-verification schemes with tree attention (Miao et al., 2023b). Our method is easily compatible with token tree verification and could benefit from such integrations.

Extending to additional Modalities Most MLLMs focus on text and image modalities, but recent efforts are expanding to include other types, such as audio Fu et al. (2024). A lightweight Automatic Speech Recognition (ASR) model could convert audio to text for integration into text-only drafting, particularly since audio data often involves long context and high computational costs. This approach could also support ensemble drafting, potentially improving performance and robustness.

8 Conclusion

This paper provides a comprehensive analysis of MLLM speculative decoding, exploring and integrating drafting techniques for speculative decoding in multimodal scenarios, with a focus on the often-overlooked decoding stage of MLLM inference. We introduce *In-batch Ensemble Drafting (IbED)*, which combines probability distributions from multiple drafting methods by batch inference during speculative decoding, requiring no additional model parameters and adding negligible overhead. This approach significantly improves block efficiency and robustness across diverse inputs. Our work demonstrates that efficient acceleration of MLLMs is achievable without compromising output fidelity, paving the way for practical and widespread applications of MLLMs.

References

Anthropic. The Claude 3 Model Family: Opus, Sonnet, Haiku. 2024. URL `https://api.semanticscholar.org/CorpusID:268232499`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

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.

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

Jian Chen, Vashisth Tiwari, Ranajoy Sadhukhan, Zhuoming Chen, Jinyuan Shi, Ian En-Hsu Yen, and Beidi Chen. Magicdec: Breaking the latency-throughput tradeoff for long context generation with speculative decoding. *arXiv preprint arXiv:2408.11049*, 2024a.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. *arXiv preprint arXiv:2403.06764*, 2024b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Chaoyou Fu, Haojia Lin, Zuwei Long, Yunhang Shen, Meng Zhao, Yifan Zhang, Xiong Wang, Di Yin, Long Ma, Xiawu Zheng, et al. Vita: Towards open-source interactive omni multimodal llm. *arXiv preprint arXiv:2408.05211*, 2024.

Yao Fu. Challenges in deploying long-context transformers: A theoretical peak performance analysis. *arXiv preprint arXiv:2405.08944*, 2024.

Mukul Gagrani, Raghavv Goel, Wonseok Jeon, Junyoung Park, Mingu Lee, and Christopher Lott. On speculative decoding for multimodal large language models. *arXiv preprint arXiv:2404.08856*, 2024.

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

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6904–6913, 2017.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, et al. Hallusionbench: an advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14375–14385, 2024.

Ting-Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, et al. Visual storytelling. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pp. 1233–1239, 2016.

Harsh Jhamtani and Taylor Berg-Kirkpatrick. Learning to describe differences between pairs of similar images. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Yizhang Jin, Jian Li, Yexin Liu, Tianjun Gu, Kai Wu, Zhengkai Jiang, Muyang He, Bo Zhao, Xin Tan, Zhenye Gan, Yabiao Wang, Chengjie Wang, and Lizhuang Ma. Efficient multimodal large language models: A survey, 2024. URL https://arxiv.org/abs/2405.10739.

Mahsa Khoshnoodi, Vinija Jain, Mingye Gao, Malavika Srikanth, and Aman Chadha. A comprehensive survey of accelerated generation techniques in large language models, 2024. URL https://arxiv.org/abs/2405.13019.

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

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024a.

Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*, 2024b.

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*, pp. 12888–12900. PMLR, 2022.

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*, pp. 19730–19742. PMLR, 2023.

Yitong Li, Zhe Gan, Yelong Shen, Jingjing Liu, Yu Cheng, Yuexin Wu, Lawrence Carin, David Carlson, and Jianfeng Gao. Storygan: A sequential conditional gan for story visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6329–6338, 2019.

Zhihang Lin, Mingbao Lin, Luxi Lin, and Rongrong Ji. Boosting multimodal large language models with visual tokens withdrawal for rapid inference. *arXiv preprint arXiv:2405.05803*, 2024.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, 2023.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26296–26306, 2024a.

Zuyan Liu, Benlin Liu, Jiahui Wang, Yuhao Dong, Guangyi Chen, Yongming Rao, Ranjay Krishna, and Jiwen Lu. Efficient inference of vision instruction-following models with elastic cache. *arXiv preprint arXiv:2407.18121*, 2024b.

Ahmed Masry, Xuan Long Do, 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*, pp. 2263–2279, 2022.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems, 2023a. URL https://arxiv.org/abs/2312.15234.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 2023b.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=TG8KACxEON.

QwenTeam. Qwen speed benchmark. https://qwen.readthedocs.io/en/latest/benchmark/speed_benchmark.html. Accessed: YYYY-MM-DD.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8317–8326, 2019.

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*, 2024a.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. Spectr: Fast speculative decoding via optimal transport, 2024b. URL https://arxiv.org/abs/2310.15141.

Hao Tan, Franck Dernoncourt, Zhe Lin, Trung Bui, and Mohit Bansal. Expressing visual relationships via language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1873–1883, 2019.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. Look-m: Look-once optimization in kv cache for efficient multimodal long-context inference. *arXiv preprint arXiv:2406.18139*, 2024.

Bin Xiao, Haiping Wu, Weijian Xu, Xiyang Dai, Houdong Hu, Yumao Lu, Michael Zeng, Ce Liu, and Lu Yuan. Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4818–4829, 2024.

Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. *arXiv preprint arXiv:2402.01528*, 2024.

Sen Yang, Shujian Huang, Xinyu Dai, and Jiajun Chen. Multi-candidate speculative decoding. *arXiv preprint arXiv:2401.06706*, 2024.

Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models, 2024. URL https://arxiv.org/abs/2306.13549.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023. URL https://arxiv.org/abs/2303.15343.

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

Appendix

A TRAINING AND HYPERPARAMETERS

The process for creating LLaVA-ft was divided into two stages: pre-training and instruction fine-tuning (IFT). Pre-training focuses on training the projector while the parameters of the LLM and vision encoder are frozen. During the IFT stage, visual instruction tuning is used to teach the LLM to follow multimodal instructions. The vision encoder remains frozen throughout both stages. We trained the draft model using datasets curated by the original author of Llava (Liu et al., 2023). For more training details, see https://github.com/haotian-liu/LLaVA/tree/main.

Hyperparameter	Value
Training Epochs	1
Batch Size	256
Learning Rate (LR)	1e-3
LR Schedule Type	Cosine
Warm-up Ratio	0.03
Weight Decay	0.0

(a) Hyperparameters used for pretraining LLaVA-ft

Hyperparameter	Value
Training Epochs	1
Batch Size	128
Learning Rate (LR)	2e-5
LR Schedule Type	Cosine
Warm-up Ratio	0.03
Weight Decay	0.0

(b) Hyperparameters used for fine-tuning LLaVA-ft

Table 4: Training details and hyperparameters.

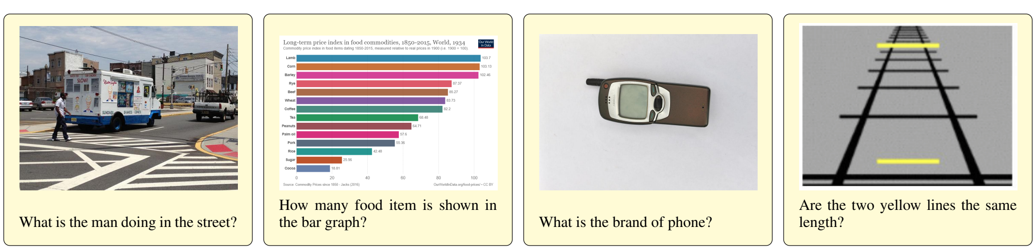
B BENCHMARK DATASETS

B.1 CURATION OF BENCHMARK DATASETS

Single-Image vs Multi-Image In the LLaVA-1.5 model, each image is represented by 576 visual tokens. Therefore, the proportion of visual tokens is significantly higher compared to text tokens, and as the number of images increases, this proportion becomes even larger. Hence, it is important to examine how the evaluation results vary based on the number of images. Consequently, we assess the performance of speculative decoding across a range of images, from single-image datasets to multi-image datasets.

Open-ended vs Closed-ended Although the prefilling stage is significantly more time-consuming than a single decoding step, speculative decoding has been developed primarily for the decoding stage rather than the prefill stage. Therefore, open-ended questions are better than closed-ended ones for generating sufficiently long outputs to evaluate the performance of speculative decoding.

B.2 DETAILS OF SINGLE IMAGE DATASETS



(a) VQAv2 (b) ChartQA (c) TextVQA (d) HallusionBench

Figure 8: Qualitatative samples of single image datasets.

VQAv2 (Goyal et al., 2017) A visual question answering dataset that is well-balanced due to the inclusion of pairs of images/prompts that are similar but result in different answers. The subset used for evaluation in our work contains 100 pairs of images and questions.

`https://huggingface.co/datasets/lmms-lab/VQAv2`

ChartQA (Masry et al., 2022) An image-text question answering dataset for testing visual comprehension of charts. The subset used for evaluation in our work contains 100 pairs of images and questions.

`https://huggingface.co/datasets/lmms-lab/ChartQA`

TextVQA (Singh et al., 2019) A visual question answering dataset that requires reading and reasoning about text within a provided image. The subset used for evaluation in our work contains 100 pairs of images and questions.

`https://huggingface.co/datasets/lmms-lab/textvqa`

HallusionBench (Guan et al., 2024) A dataset designed to measure the ability of large vision language models to reason despite hallucinations. The subset used for evaluation in our work contains 100 question and answer pairs.

`https://huggingface.co/datasets/lmms-lab/HallusionBench`

B.3 Details of Multi Image Datasets

Spot the Difference (Jhamtani & Berg-Kirkpatrick, 2018) A dataset of crowd-sourced descriptions of differences between a pair of images. The subset used for evaluation in our work contains 100 annotated image pairs collected using individual frames of security-footage data.

`https://huggingface.co/datasets/lmms-lab/LLaVA-NeXT-Interleave-Bench`

IEdit (Tan et al., 2019) A dataset to train models to describe the relationship between images via editing instructions. The subset used for evaluation in our work contains 100 image pairs of a source image and a target image, accompanied by instructions on how to transform the source image into the target.

`https://huggingface.co/datasets/lmms-lab/LLaVA-NeXT-Interleave-Bench`

Pororo-SV (Li et al., 2019) A dataset of stories each created by pairing 5 consecutive frames from the animated series *Pororo* with a text description. The subset used for evaluation in our work contains 100 stories.

`https://huggingface.co/datasets/lmms-lab/LLaVA-NeXT-Interleave-Bench`

VIST (Huang et al., 2016) A dataset of sequential images paired with three types of descriptions ranging from isolated factual descriptions to causal, narrative interpretations. The subset used for evaluation in our work contains 100 sequences of 3 images.

`https://huggingface.co/datasets/lmms-lab/LLaVA-NeXT-Interleave-Bench`

C System Prompts and Text-only Drafting

We use the following system prompts for their respective tasks. The <image> token is used to represent image data within a prompt. [QUESTION] and [CAPTION] are a placeholders denoting information unique to each sample of a dataset. For text-only drafting, the <image> token is replaced by the escape character \n. We experimented with several replacement methods: (1) tokenizing the <image> string into three tokens, and (2) retaining the special token <image> without replacing it with an image embedding. Method (2) resulted in very poor block efficiency, but method (1) showed comparable block efficiency. Our replacement approach is simple because it ensures that the prompt length remains consistent before and after replacement.

ChartQA *<s> USER: <image> For the following question, provide a detailed explanation of your reasoning leading to the answer. [QUESTION] ASSISTANT:*

TextVQA *<s> USER: <image> For the following question, provide a detailed explanation of your reasoning leading to the answer. [QUESTION] ASSISTANT:*

VQAv2 *<s> USER: <image> For the following question, provide a detailed explanation of your reasoning leading to the answer. [QUESTION] ASSISTANT:*

HallusionBench *<s> USER: <image> For the following question, provide a detailed explanation of your reasoning leading to the answer. [QUESTION] ASSISTANT:*

Spot The Difference *<s> USER: Explain the disparities between the first and second image. <image> <image> Difference: ASSISTANT:*

IEdit *<s> USER: Please provide instructions for editing the source image to match the target image. Source Image: <image> Target Image: <image> Instruction: ASSISTANT:*

PororoSV *<s> USER: Given the progression of the story with the first few images, can you write a fitting end considering the last image? <image> Caption #1: [CAPTION] <image> Caption #2: [CAPTION]. <image> Caption #3: [CAPTION] <image> Caption #4: [CAPTION] <image> Caption #5: ASSISTANT:*

VIST *<s> USER: With the narratives paired with the initial images, how would you conclude the story using the last picture? <image> Caption #1: [CAPTION] <image> Caption #2: [CAPTION]. <image> Caption #3: [CAPTION] <image> Caption #4: [CAPTION] <image> Caption #5: ASSISTANT:*

D Pooled Multimodal Drafting

While we conduct pooled multimodal drafting without further fine-tuning, we also investigate how the performance of speculative decoding changes when visual instruction tuning is performed using pooling.

Table 5 presents the block efficiency results for the finetuned draft model across various pooling methods. The results demonstrate that the block efficiency of the finetuned model is higher than that of the non-finetuned model.

Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
LLaVA 1.5	7B / 68M	multimodal	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		pool (144)	2.23	2.08	2.26	2.36	2.23	2.22	2.07	2.09	2.23	2.23	2.08
		pool (144, ft)	2.26	2.09	2.26	2.39	2.38	2.24	2.27	2.27	2.25	2.31	2.27
		pool (36)	2.17	2.01	2.21	2.32	2.20	2.23	2.05	2.06	2.18	2.21	2.05
		pool (36, ft)	2.22	2.06	2.25	2.38	2.36	2.26	2.19	2.23	2.23	2.31	2.21
		pool (9)	2.20	2.03	2.21	2.34	2.25	2.24	2.06	2.08	2.20	2.25	2.07
		pool (9, ft)	2.23	2.05	2.22	2.37	2.37	2.25	2.18	2.21	2.22	2.31	2.20
		pool (1)	2.23	2.03	2.23	2.37	2.25	2.26	2.06	2.07	2.21	2.25	2.06
		pool (1, ft)	2.23	2.06	2.21	2.37	2.39	2.27	2.21	2.22	2.22	2.33	2.21
		text-only	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05

Table 5: Block efficiency results for various pooling methods.

E Caption Drafting

In this section, we describe various types of lightweight image captioning models that can be used for caption drafting and report the performance of speculative decoding when each model is utilized.

E.1 Model Lists

BLIP (Li et al., 2022) A vision-language model trained on bootstrapped synthetic captions. It uses a visual transformer and the text encoder of BERT Devlin et al. (2019) to separately encode image and text.

https://huggingface.co/Salesforce/blip-image-captioning-base

BLIP-2 (Li et al., 2023) A vision-language model using a frozen off-the-shelf image encoder and LLM. A querying transformer trained using boostrapped data is included for cross-modal alignment.

https://huggingface.co/Salesforce/blip2-opt-2.7b

Florence-2 (Xiao et al., 2024) A vision-language model that is instruction-trained for a variety of tasks. Its architecture consists of a single sequence-to-sequence transformer and a vision encoder.

https://huggingface.co/microsoft/Florence-2-large-ft

E.2 ADDITIONAL EXPERIMENTAL RESULTS

The default caption model utilized in our study is Florence-2, which also supports the generation of detailed captions. However, the latency associated with generating detailed captions is longer compared to default captions. We report the results obtained using the detailed captions from Florence-2 and additionally evaluate the performance of other off-the-shelf image captioning models such as BLIP and BLIP-2.

Table 6 presents the block efficiency results for various image captioning models. Florence-2 (C) refers to our default setting, while Florence-2 (MDC) refers to the more detailed caption.

Table 7 presents the block efficinecy results of ensemble drafting by detailed captions. The block efficiency is higher in the ensemble result using detailed captions compared to the case with default captions. Table 8 presents the block efficiency results with detailed caption by various ensemble weights.

Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
LLaVA 1.5	7B / 68M	Multimodal	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		Text-only	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05
		Florence-2 (C)	2.28	2.08	2.24	2.41	2.31	2.29	2.08	2.10	2.25	2.30	2.09
		Florence-2 (MDC)	2.27	2.11	2.26	2.44	2.28	2.29	2.10	2.11	2.27	2.29	2.10
		BLIP	2.23	2.02	2.23	2.40	2.28	2.27	2.12	2.10	2.22	2.27	2.11
		BLIP-2	2.25	2.07	2.23	2.37	2.30	2.29	2.09	2.12	2.23	2.29	2.10

Table 6: Block efficiency results for various image captioning models.

Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
LLaVA 1.5	7B / 68M	M	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		T	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05
		C (C)	2.28	2.08	2.24	2.41	2.31	2.29	2.08	2.10	2.25	2.30	2.09
		C (MDC)	2.27	2.11	2.26	2.44	2.28	2.29	2.10	2.11	2.27	2.29	2.10
		MT	2.26	2.13	2.27	2.39	2.40	2.31	1.94	1.91	2.26	2.35	1.92
		MC	2.30	2.17	2.29	2.42	2.39	2.32	1.99	1.93	2.29	2.35	1.96
		MC (MDC)	2.31	2.17	2.30	2.46	2.38	2.33	1.99	1.96	2.31	2.35	1.98
		MTC (C)	2.29	2.15	2.28	2.41	2.41	2.30	2.08	2.06	2.28	2.35	2.07
		MTC (MDC)	2.29	2.15	2.29	2.44	2.40	2.33	2.09	2.08	2.29	2.37	2.08

Table 7: Block efficiency results of ensemble drafting with detailed captions.

F ENSEMBLE DRAFTING

In this section, we investigate how the performance of ensemble drafting varies as we adjust the ensemble weights. Specifically, given our prior assumption that multimodal drafting generally performs better, we conduct experiments by varying the weight of multimodal drafting from 1 to 4. The numbers in parentheses represent the weight assigned to multimodal drafting.

Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
		M	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		T	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05
		C (C)	2.28	2.08	2.24	2.41	2.31	2.29	2.08	2.10	2.25	2.30	2.09
		C (MDC)	2.27	2.11	2.26	2.44	2.28	2.29	2.10	2.11	2.27	2.29	2.10
		MC (C, 1)	2.30	2.17	2.29	2.42	2.39	2.32	1.99	1.93	2.29	2.35	1.96
		MC (C, 2)	2.30	2.17	2.30	2.41	2.39	2.31	1.80	1.71	2.29	2.35	1.75
		MC (C, 3)	2.29	2.16	2.29	2.40	2.38	2.29	1.66	1.56	2.29	2.33	1.61
		MC (C, 4)	2.28	2.16	2.29	2.40	2.37	2.28	1.56	1.46	2.28	2.33	1.51
		MC (MDC, 1)	2.31	2.17	2.30	2.46	2.38	2.33	1.99	1.96	2.31	2.35	1.98
LLaVA 1.5	7B / 68M	**MC (MDC, 2)**	2.30	2.17	2.30	2.44	2.37	2.31	1.83	1.73	2.30	2.34	1.78
		MC (MDC, 3)	2.28	2.16	2.30	2.43	2.37	2.29	1.68	1.58	2.29	2.33	1.63
		MC (MDC, 4)	2.27	2.16	2.29	2.43	2.36	2.27	1.57	1.48	2.29	2.31	1.52
		MTC (1)	2.29	2.15	2.28	2.41	2.41	2.30	2.08	2.06	2.28	2.35	2.07
		MTC (2)	2.29	2.17	2.29	2.42	2.41	2.30	1.99	1.96	2.29	2.35	1.98
		MTC (3)	2.28	2.17	2.29	2.41	2.39	2.29	1.90	1.83	2.29	2.34	1.86
		MTC (4)	2.28	2.16	2.29	2.40	2.39	2.28	1.80	1.71	2.28	2.33	1.75
		MTC (MDC, 1)	2.29	2.15	2.29	2.44	2.40	2.33	2.09	2.08	2.29	2.37	2.08
		MTC (MDC, 2)	2.29	2.16	2.30	2.43	2.41	2.34	2.01	1.95	2.29	2.38	1.98
		MTC (MDC, 3)	2.29	2.17	2.30	2.42	2.40	2.32	1.92	1.82	2.29	2.36	1.87
		MTC (MDC, 4)	2.28	2.17	2.30	2.42	2.39	2.30	1.81	1.71	2.29	2.34	1.76

Table 8: Block efficiency results with detailed captions for various ensemble weights.

Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
		M	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		T	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05
		C	2.28	2.08	2.24	2.41	2.31	2.29	2.08	2.10	2.25	2.30	2.09
		P	2.23	2.08	2.26	2.36	2.23	2.22	2.07	2.09	2.23	2.23	2.08
		MT (1)	2.26	2.13	2.27	2.39	2.40	2.31	1.94	1.91	2.26	2.35	1.92
		MT (2)	2.26	2.13	2.29	2.39	2.40	2.29	1.76	1.69	2.27	2.34	1.73
		MT (3)	2.26	2.13	2.28	2.40	2.38	2.28	1.63	1.54	2.27	2.33	1.58
		MT (4)	2.26	2.14	2.28	2.40	2.36	2.26	1.54	1.45	2.27	2.31	1.50
		MC (1)	2.30	2.17	2.29	2.42	2.39	2.32	1.99	1.93	2.29	2.35	1.96
LLaVA 1.5	7B / 68M	**MC (2)**	2.30	2.17	2.30	2.41	2.39	2.31	1.80	1.71	2.29	2.35	1.75
		MC (3)	2.29	2.16	2.29	2.40	2.38	2.29	1.66	1.56	2.29	2.33	1.61
		MC (4)	2.28	2.16	2.29	2.40	2.37	2.28	1.56	1.46	2.28	2.33	1.51
		MTC (1)	2.29	2.15	2.28	2.41	2.41	2.30	2.08	2.06	2.28	2.35	2.07
		MTC (2)	2.29	2.17	2.29	2.42	2.41	2.30	1.99	1.96	2.29	2.35	1.98
		MTC (3)	2.28	2.17	2.29	2.41	2.39	2.29	1.90	1.83	2.29	2.34	1.86
		MTC (4)	2.28	2.16	2.29	2.40	2.39	2.28	1.80	1.71	2.28	2.33	1.75
		MTCP (1)	2.29	2.17	2.29	2.42	2.41	2.33	1.99	1.93	2.29	2.37	1.96
		MTCP (2)	2.28	2.17	2.29	2.41	2.40	2.31	1.90	1.81	2.29	2.35	1.85
		MTCP (3)	2.28	2.16	2.29	2.40	2.40	2.31	1.79	1.70	2.28	2.35	1.75
		MTCP (4)	2.28	2.16	2.29	2.40	2.39	2.29	1.71	1.62	2.28	2.34	1.67

Table 9: Block efficiency results of ensemble drafting for various weights.

G Latency Analysis

G.1 Prefill vs Decode

This experiment shows how long it takes to perform prefill versus autoregressive decoding on our 68M VLM draft model. With sequence length set to 200 and batch size 1, we found that the latency of prefilling is slightly higher than autoregressive decoding, as shown in Table 10, and in Table 11 and Figure 9. This is because the model processes longer sequences during the prefill stage. For models this small, the autoregressive stage is neither bounded by memory nor computation and can leverage GPU cache to store parts of KV cache and therefore leads to lower autoregressive decoding latency compared to the prefilling stage.

Stage	**Time Taken (ms)**
Prefill	35.6
Decode	27.1

Table 10: This table shows the time taken for prefill stage and autoregressive decoding stage on our 68M VLM draft model.

G.2 PREFILL LENGTH

The experiment examines the time taken to perform the prefill operation for different sequence lengths, specifically at lengths of 200, 1200, and 2200 tokens. The results are summarized in Table 11, which shows the time taken in milliseconds (ms) for each sequence length. We perform this on an A100 GPU on our 68M VLM draft model. This table shows that at our draft model's size, prefill time does not vary with different prefill lengths since we are not computationally bound.

Prefill Length	Time Taken (ms)
200	35.6
1200	35.7
2200	35.7

Table 11: This table shows the time taken for prefill stage at different sequence lengths for our 68M VLM draft model.

G.3 BATCH SIZE

This experiment shows how decoding time changes as we increase batch size.

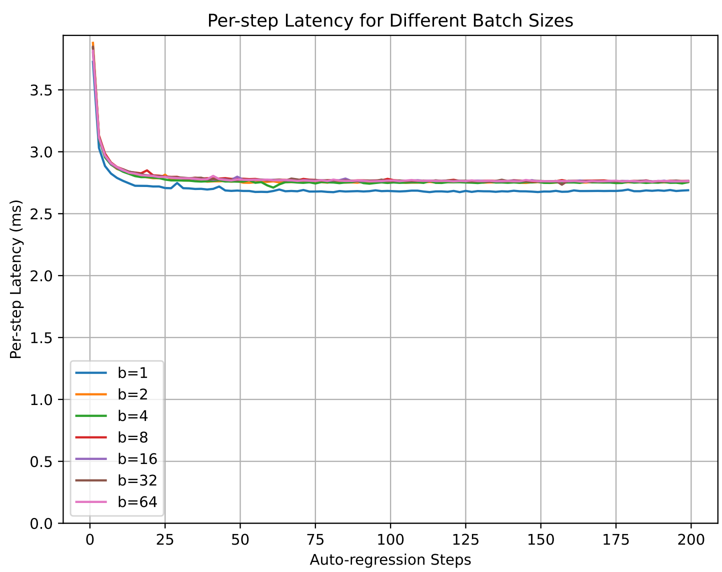


Figure 9: This figure shows the per-step autoregressive decoding latency for different batch sizes across varying auto-regression steps on our 68M VLM draft model.

ANALYSIS

Figure 9 illustrates the per-step latency for different batch sizes (from 1 to 64) as the number of auto-regression steps increases from 0 to 200. The x-axis represents the number of auto-regression steps, while the y-axis shows the per-step latency in milliseconds (ms) for our 68M multi-modal draft model.

Several key observations can be made from this figure: The per-step latency increases slightly with larger auto-regression steps. However, this increase is marginal, suggesting that the model maintains consistent performance across a wide range of sequence lengths. The plot shows that increasing batchsize does not affect per-step decoding latency as we are neither bounded by computation nor by memory bandwidth.

H Evaluation of Target and Draft Models on Multimodal Tasks

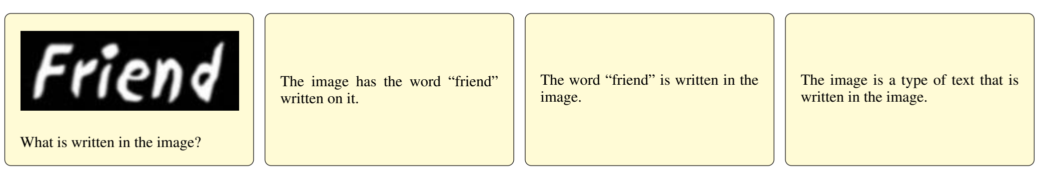
In this section, we present a comprehensive evaluation of both target and draft models on multimodal tasks to better understand the multimodal performance of the model itself. We evaluate LLaVA-1.5 7B, which serves as the target model in our experimental setting, and LLaVA-1.5 68M, which functions as the draft model. Additionally, to investigate the relationship between image-aware capability and language modeling proficiency, we fine-tune LLaVA-1.5 68M using varying numbers of visual tokens per image.

Table 12 shows the evaluation results on various MLLM tasks. For the 7B model, it shows significantly better performance for each task compared to the 68M model, but it can be confirmed that the 68M model also meets the minimum performance requirements. As the number of visual tokens increases, it can be observed that the performance of multimodal improves, whereas the performance of text-only slightly decreases. This suggests that the limited capacity of the 68M model is shared between image-aware capabilities and language modeling capabilities.

Fig. 10 presents qualitative evaluation samples from the OCRBench dataset, comparing the performance of LLaVA-1.5 7B and 68M models. Both LLaVA-1.5 7B and 68M models provided accurate responses, whereas the text-only LLaVA-1.5 68M model failed to answer correctly due to its lack of image-processing capabilities.

				ChartQA	OCRBench	TextCaps	
Model	Size	# visual tokens	Method	Accuracy	Accuracy	METEOR	ROUGE
LLaVA 1.5	7B	576 (default)	multimodal	0.20	0.207	0.249	0.48
LLaVA 1.5	68M	576 (default)	multimodal	0.09	0.048	0.133	0.254
		144 (finetuned)		0.08	0.039	0.125	0.251
		36 (finetuned)		0.02	0.025	0.106	0.176
		9 (finetuned)		0.00	0.009	0.116	0.192
		1 (finetuned)		0.00	0.002	0.066	0.136
LLaVA 1.5	68M	576 (default)	text-only	0.04	0.014	0.064	0.132
		144 (finetuned)		0.06	0.017	0.076	0.141
		36 (finetuned)		0.07	0.016	0.080	0.161
		9 (finetuned)		0.07	0.017	0.085	0.178
		1 (finetuned)		0.08	0.016	0.079	0.152

Table 12: Evaluation results on MLLM tasks.



(a) Instruction (b) Multimodal (7B) (c) Multimodal (68M) (d) Text-only (68M)

Figure 10: Qualitative evaluation samples from the OCRBench dataset by LLaVA-1.5 7B and 68M.

I Draft Model without Visual Instruction Tuning

In this section, we examine the performance of speculative decoding when using a pretrained LLaMA 68M model without visual instruction tuning as the draft model. Furthermore, we assess the performance of a model fine-tuned through visual instruction tuning using text only, without a visual encoder.

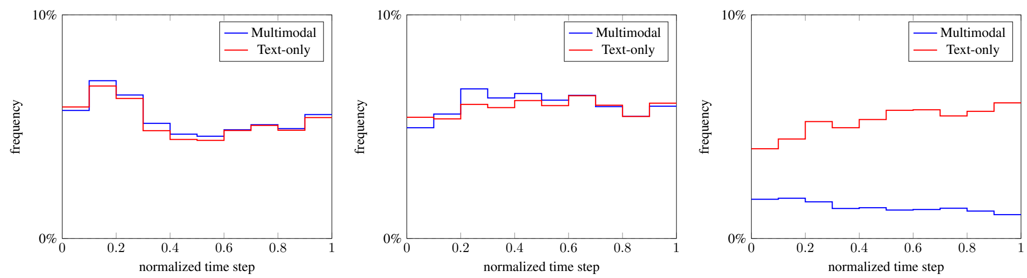
Table 13 shows the block efficiency results of pretrained and finetuned LLaMA 68M. In the case of the LLaMA 68M, it has not been fine-tuned with the dataset used for training the target model, its performance is inferior compared to the text-only LLaVA 68M.

Target / Draft			$n=1$				$n=2$		$n=5$		$n=1$	$n=2$	$n=5$
Model	Size	Method	ChartQA	TextVQA	VQAv2	Hallusion	Spot	IEdit	PororoSV	VIST	Avg.	Avg.	Avg.
LLaVA / LLaMA	7B / 68M	**pretrained**	2.06	1.75	1.83	2.23	1.95	2.06	1.76	1.72	1.97	2.00	1.74
		finetuned	2.21	2.03	2.24	2.37	2.27	2.27	2.02	2.05	2.21	2.27	2.04
LLaVA / LLaVA	7B / 68M	multimodal	2.24	2.12	2.26	2.39	2.34	2.19	1.19	1.16	2.25	2.26	1.17
		text-only	2.22	2.03	2.20	2.34	2.27	2.23	2.05	2.05	2.20	2.25	2.05

Table 13: Block efficiency results of pretrained and finetuned LLaMA 68M.

J Additional Analysis on Multimodal Drafting and Text-only Drafting

Figure 11 analyzes the frequency of correctly decoded tokens by dataset as decoding progresses. In the very early stages, text-only drafting tends to outperform multimodal drafting, as the text context alone is often sufficient (e.g., 'The jersey design' in Figure 4 can be easily inferred from the text prompt alone). However, when image information becomes necessary, multimodal drafting gains an advantage, until the middle-to-later stages, where the accumulated text context leads to similar performance for both methods.



(a) HallusionBench ($n=1$) (b) Spot-the-Diff ($n=2$) (c) VIST ($n=5$)

Figure 11: Histograms of accepted token count according to normalized time step on various datasets.