DyMU: Dynamic Merging and Virtual Unmerging for Efficient Variable-Length VLMs

Zhenhailong Wang[1*], **Senthil Purushwalkam**[2*], **Caiming Xiong**[2],
Silvio Savarese[2], **Heng Ji**[1], **Ran Xu**[2]
[1]University of Illinois Urbana-Champaign [2]Salesforce Research
{wangz3,hengji}@illinois.edu, ran.xu@salesforce.com
[*]Equal contribution

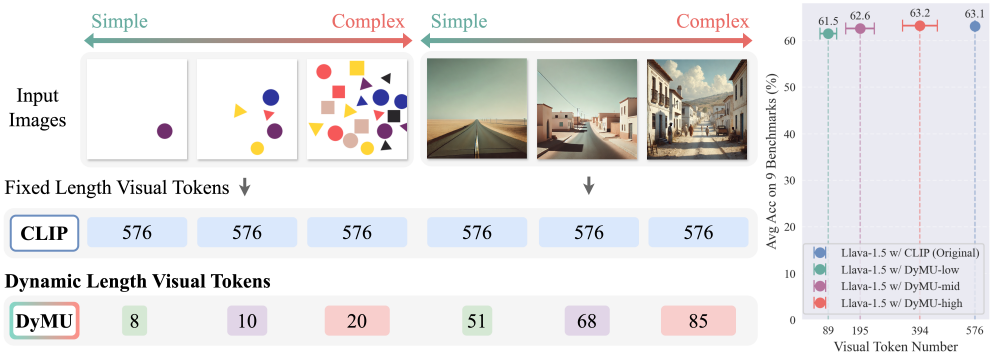

Figure 1: Dynamic Merging and Virtual Unmerging (DyMU) adaptively reduces visual token lengths based on *image complexity*, as shown on the left where simpler images are represented using fewer tokens. In contrast, existing representations (like CLIP [34]) always use the same number of tokens regardless of image content. Note that this limitation also exists in dynamic resolution encoders [18, 38], where token length depends solely on image dimensions rather than complexity. DyMU applied to recent VLMs (right) maintains competitive performance across different token compression levels. This training-free approach preserves key semantic information, offering a more efficient plug-and-play alternative to VLMs with fixed-length visual tokens.

Abstract

We present DyMU, an efficient, training-free framework that dynamically reduces the computational burden of vision-language models (VLMs) while maintaining high task performance. Our approach comprises two key components. First, Dynamic Token Merging (DToMe) reduces the number of visual token embeddings by merging similar tokens based on image complexity, addressing the inherent inefficiency of fixed-length outputs in vision transformers. Second, Virtual Token Unmerging (VTU) simulates the expected token sequence for large language models (LLMs) by efficiently reconstructing the attention dynamics of a full sequence, thus preserving the downstream performance without additional fine-tuning. Unlike previous approaches, our method dynamically determines token length based on the *image content*—not just resolution—and operates completely training-free, making it readily applicable to most state-of-the-art VLM architectures. Extensive experiments on image and video understanding tasks, demonstrate that DyMU can reduce the average visual token count by 32%-85% while achieving comparable performance to full-length models, across diverse VLM architectures. Furthermore, qualitative analyses show that the adaptive token reduction from DToMe aligns well with human perception and enables users to better control computational costs through flexible integration with additional vision tools and models.

1 Introduction

Recent large vision-language models (VLMs) follow a common approach: a visual encoder extracts features from images or videos and projects them into the same embedding space as textual features.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

These visual embeddings are then processed by LLMs alongside textual query features, enabling complex reasoning tasks while directly benefiting from advancements in LLM capabilities. As expected, the quality of the final predictions from the LLM relies heavily on the richness of the visual features and the amount of semantic detail captured by the encoder. Consequently, research has focused on improving visual encoders to extract increasingly fine-grained features, leading to architectures that can capture intricate details. However, this level of detail comes at a cost—the computational burden during training and inference.

To process high-resolution images while preserving fine-grained details, modern visual encoders generate a large number of tokenized representations. Furthermore, state-of-the-art VLMs like LLaVA-OneVision [18] and Qwen-2.5-VL [2] use vision transformers (ViTs) that scale the number of tokens with the resolution of the image or number of frames in videos. For example, the visual encoder in LLaVA-OneVision would produce 9477 tokens for an image of 1280×960 resolution. In contrast, the number of tokens in the textual queries for vision tasks is relatively low. On common benchmarks that represent real world use cases, textual queries often consist of just a few tokens, e.g., $\sim$24 on MME [11]. This stark contrast highlights that the computational burden of processing vision-language tasks arises primarily from the visual tokens.

We first identify a critical limitation in current visual encoders (including those with dynamic resolution): the number of tokens generated for an image does not depend on the **content** of the image. As shown in Figure 1, existing visual representations, such as CLIP [34], produce the same embedding size for a blank image with a small circle and for a complex scene depicting buildings, vehicles, and people. To address the redundancy of visual tokens, previous work has shown initial success with token pruning, merging, and distillation. However, these approaches remain limited in one or more of the following key aspects: (1) They rely on pre-defined compression rates regardless of the image complexity [3, 21, 35, 5, 20, 14, 48]; (2) They require full model retraining, which is often infeasible for recent mainstream open-weight-only VLMs [21, 14, 20, 43, 48]; (3) They depend on extra textual conditions, rather than implementing a natively dynamic-length visual encoder [49, 43].

In this work, we propose **Dynamic Merging and Virtual Unmerging (DyMU)**, which comprises two key methods for modifying existing pre-trained Vision-Language Models (VLMs). First, we introduce **Dynamic Token Merging (DToMe)** (§ 3.1), which allows the visual encoder to generate variable-length token sequences based on the *complexity*—not just resolution—of the image. The core idea is to perform *batch-level*, rather than instance-level as in ToMe[3], bipartite merging to determine layer-wise thresholds for dynamic token reduction.

Second, we propose a novel **Virtual Token Unmerging (VTU)** algorithm (§ 3.2), which allows the LLM decoder to process shorter, dynamically reduced visual token sequences while efficiently approximating the full-length representation. This method leverages the tracked positions of redundant tokens to reconstruct the full attention matrix without actually performing full attention. VTU enables the direct integration of DyMU into LLM backbones, offering a favorable trade-off between performance and efficiency.

Importantly, both of these modifications *do not require additional fine-tuning of the pre-trained VLM*. The threshold finding for DToMe only needs to be done *once*, offline, using any sufficiently diverse image-only datasets. Furthermore, DToMe is compatible with any Vision Transformer (ViT)-based visual encoder, and VTU can be applied to any LLM that utilizes Rotary Position Embedding (RoPE) [37].

We show that VLMs modified with our methods can maintain comparable performance of the full model while reducing the average token count by 32%-85%. Through comprehensive quantitative analysis (§ 4.1), we verify that our method works effectively across different VLM architectures, visual encoders, and threshold-finding datasets. We show additional insights on why variable length reduction is favorable in terms of both performance and efficiency. Our approach also offers users greater control over token costs compared to existing AnyRes models [18, 38, 2], which incur a fixed token count per image based solely on resolution. In § 4.2, we demonstrate example applications on how the number of visual tokens can be further reduced by combining DyMU with various task-specific tools such as background removal, object detection, etc.

2 Related Work

Efficient Vision-Language Models Recent efforts in large vision-language models (VLMs) have primarily focused on reducing computational overhead during the pre-filling and VLM decoding phases. That is, given a full sequence of visual tokens from a visual encoder, such as CLIP, these approaches perform token pruning and merging [35, 5, 40, 49, 15, 23, 43], distillation [44], or resampling [21, 14, 20, 48] to improve efficiency in either the projectors or the VLM decoder blocks.

	Component Improved	Dynamic Length	No Addn. Modules	Training Free	Granularity Control	Extra Cond.
LLaMA-VID [21]	Projector	✗	✗	✗	✗	None
Fast-V [5]	Decoder	✗	✓	✓	✓	None
SparseVLM [49]	Decoder	✓	✓	✓	✓	Text
MQT-LLaVA [14]	Projector	✗	✗	✗	✗	None
LLaVA-Prumerge [35]	Projector	✗	✓	✗	✓	None
TokenPacker [20]	Projector	✗	✗	✗	✓	None
ATP-LLaVA [43]	Decoder	✓	✗	✗	✓	Text
LLaVA-mini [48]	Projector	✗	✗	✗	✗	None
DyMU	**Encoder** & Decoder	✓	✓	✓	✓	None

However, we identify several key limitations: (1) Most existing methods predefine a fixed compression ratio for any input image regardless of its content. While [49] and [43] propose adaptive token pruning frameworks that enable variable-length compression, they require either additional textual conditions or retraining the backbone LLM with extra modules. Such training can be costly or infeasible as mainstream VLMs rarely open-source their full training recipe and data. (2) All existing methods retain a frozen, fixed-length visual encoder, overlooking the potential for further efficiency improvements within the visual encoder itself. In this work, we aim to explore a simple training-free algorithm for building natively variable-length visual encoders, which can be directly applied to cutting-edge VLM architectures including AnyRes models and RoPE embeddings.

Efficient Vision Transformers We also draw inspiration from a separate line of research [29, 3, 39, 45, 26] aimed at improving the efficiency of Vision Transformers (ViTs) themselves, which is still the main go-to architecture for visual encoders [34, 47, 33]. In particular, ToMe [3] merges a predefined number of tokens within each ViT block using bipartite soft matching. However, the effectiveness of such methods in coordination with LLM backbones remains largely unexplored. Our experiments in §4 show that naively applying ToMe to visual encoders in pretrained VLMs results in a significant drop in performance. To address this issue, we further propose the Virtual Token Unmerging algorithm to boost the performance of VLMs without training with the modified encoders that output reduced token numbers.

3 Method

In this section, we present the main technical details of our proposed method. In § 3.1, we first describe how to convert a fixed-length visual encoder into a natively dynamic-length encoder using Dynamic Token Merging (DToMe). In § 3.2, we then introduce Virtual Token Unmerging (VTU), a method that achieves a better trade-off between performance and efficiency when integrating the dynamic-length encoder into a language model backbone. The combination of these two techniques is referred to as DyMU—an entirely training-free approach compatible with any VLM that uses a ViT-based visual encoder and a RoPE-based language model. An overview of the method is illustrated in **Figure 2**.

3.1 Dynamic Token Merging (DToMe)

Our approach draws inspiration from ToMe[3], a prior work which reduces the number of output tokens of a ViT-based visual encoder to a *predefined fixed number*. However, predefining the reduction ratio can still lead to a misalignment between the information of an image and the number of tokens needed for representing it. Here we propose DToMe, an extension of ToMe that adaptively merges similar tokens in ViT layers, ensuring the output token count aligns with image complexity. DToMe merges tokens based on a similarity threshold while maintaining a record of merged tokens to ensure their influence is properly propagated through subsequent transformer layers. To find the thresholds, we propose a **inference-only batch-level bipartite merging** algorithm which leverages the natural variance of image complexity in randomly sampled images.

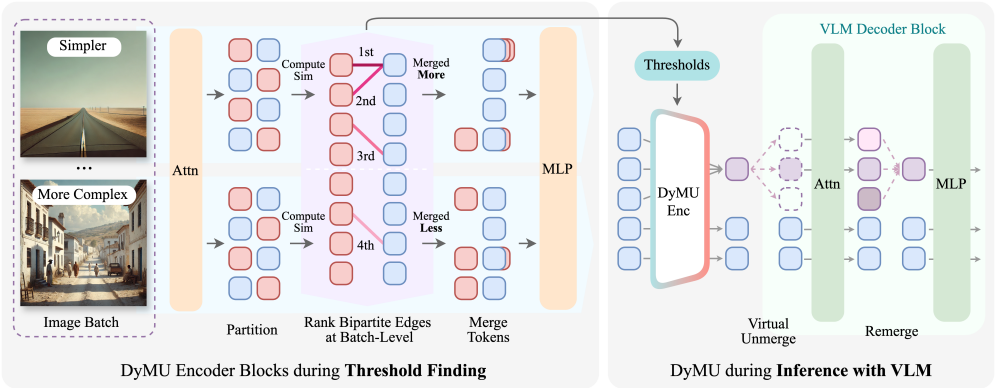

Figure 2: Method Overview. DyMU, is composed of two key ideas: Dynamic Token Merging (DToMe) and Virtual Token Unmerging (VTU). DToMe first determines per-layer thresholds (**left**) by feeding a large batch of images into the vision transformer and computing bipartite token similarities. Unlike [3], we rank these edges across the *entire batch* rather than within each instance. This naturally leads to more merges in simpler images (with more redundancy). During inference, DToMe merges tokens on a per-image basis using these pre-computed thresholds. We then introduce VTU (**right**), an efficient algorithm for reconstructing the full attention matrix without actually performing full attention. VTU enables a better trade-off between performance and efficiency, facilitating direct integration of dynamic-length embeddings into VLM backbones. The overall process is entirely training-free and allows more flexible control of token budget (Figure 6).

Identifying Redundant Tokens Let us represent the output of the self-attention layer in the ViT layer i as $x_i \in \mathbb{R}^{N_i \times D}$, where N_i is the sequence length[*] and D is the embedding dimension. Similarly, let the keys computed in the self-attention layer be represented by $k_i \in \mathbb{R}^{N_i \times D_k}$. In each transformer block, we apply an additional DToMe operator to x_i. Following [3], we use a bipartite soft matching strategy to identify which tokens need to be merged. First, we divide the N_i tokens into two sets (say $\mathbb{A}$ and $\mathbb{B}$) by assigning alternating tokens in sequence to them. We then compute a bipartite assignment between the two sets of tokens by assigning source token $t \in \mathbb{A}$ to target token $t' = \arg\max_{n \in \mathbb{B}}(k_i[t]^T\ k_i[n])$ (token with the most similar key). This gives us $|\mathbb{A}|$ edges with scores $S_i[t] = (k_i[t]^T\ k_i[t'])$ for $t \in \mathbb{A}$. We then apply a threshold τ_i to retain edges $t \to t'$ where $S_i[t] > \tau_i$. Unlike [3], this thresholding operation leads to a *variable number* of retained edges depending on the amount of redundancy demonstrated in the key embeddings k_i. The key question now is: how do we determine the thresholds without training?

Finding Redundancy Thresholds with Batch-level Bipartite Merging In order to determine the layer-wise thresholds τ_i, we rely on statistics from a large dataset of images.[†] First, we choose a hyper-parameter r_i for each layer i which represents the number of edges we expect to merge in a layer ***on average across images of all complexities***. This hyper-parameter serves as a granularity control for the user.[‡] The final output would then be expected to have an *average* of $N - \sum_i r_i$ tokens, where N is the input full sequence length. Using a dataset of images, we collect large batches of size B which are used to perform forward computation through the layers of the ViT sequentially. For each layer, we compute the B bipartite matching token edge score maps $S^{(b)}[t]$ where $b \in \{1, 2, \ldots, B\}$ as previously described. We then find the threshold τ_i as:

$$\tau_i = \max\left\{ \tau \mid \sum_{b=1}^{B} \sum_{t \in \mathbb{A}^{(b)}} \mathbb{I}\left(S^{(b)}[t] > \tau\right) = B * r_i \right\} \tag{1}$$

In words, this finds the largest threshold such that $B * r_i$ tokens are merged across the batch of images. It is important to note that the number of tokens merged in each image will not necessarily be equal to r_i but the *average* number of tokens merged per image will be r_i. Intuitively, since the ranking of edges is over the entire batch, *simpler images that have more redundant tokens will be merged more*. This process is done sequentially for each layer while only passing the remaining tokens to the next layer to obtain thresholds for every layer. We then average the layer-wise thresholds across all

[*]For standard ViT without any merging, N_i is constant across layers

[†]We show in Appendix E that DToMe is robust to different threshold-finding datasets.

[‡]We include additional analysis of different r_i scheduling in Appendix D

batches to ensure that they reflect the statistics across a diverse set of images. See Figure 2 (left) for an illustration of the proposed batch-level threshold finding.

Tracking and Merging Tokens We now describe how the merged tokens are computed. For each token in the sequence, $x_i[t]$, we also track the set of positions of the tokens that have already been merged into it. $\mathbf{P}_i[t] \subset \{1, 2, \ldots, N\}$. For each of the edges between chosen redundant tokens $t \to t'$, we compute merged token embeddings and the corresponding position sets as:

$$x_i\left[t'\right] \leftarrow \frac{x_i[t] \cdot |\mathbf{P}_i[t]| + x_i[t'] \cdot |\mathbf{P}_i[t']|}{|\mathbf{P}_i[t]| + |\mathbf{P}_i[t']|} \tag{2}$$

$$\mathbf{P}_i\left[t'\right] \leftarrow \mathbf{P}_i\left[t'\right] \cup \mathbf{P}_i[t]; \qquad \mathbf{P}_i[t] \leftarrow \varnothing \tag{3}$$

Intuitively, the representation of the target token t' is updated to the average of $x_i[t']$ and $x_i[t]$, weighted by their corresponding merged position set sizes, $\mathbf{P}_i[t']$ and $\mathbf{P}_i[t]$. The source token t is then dropped, thereby reducing the token count in the next layer.

Size Weighted Self-attention To ensure that the self-attention layers weigh each token based on the number of tokens that were previously merged into it, we also adopt the idea of size-weighted self-attention from [3] where the attention is computed as:

$$\mathbf{A} = \texttt{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} + \log\begin{bmatrix} |\mathbf{P}_i[t_1]| \\ \vdots \\ |\mathbf{P}_i[t_{N_i}]| \end{bmatrix}\right) \tag{4}$$

3.2 Virtual Token Unmerging (VTU)

The language model (LLM) in a pre-trained VLM is trained to operate on a fixed number of embeddings for each image[§]. When DToMe is applied to a visual encoder, this disrupts the optimized VLM and leads to a significant drop in performance (see § 4). In this section, we present an approach to circumvent this issue while still benefiting from processing fewer number of visual embeddings.

Consider the general case of a sequence of N embeddings $e \in \mathbb{R}^{N \times D}$ of which only $N_{\text{un}} \ll N$ rows are unique. Let $e_{\text{un}} \in \mathbb{R}^{N_{\text{un}} \times D}$ be the unique embeddings and $M \in \{0, 1\}^{N \times N_{\text{un}}}$ be a mapping such that $e = M\ e_{\text{un}}$. Here M is a sparse matrix with one-hot rows. We now ask the question: *for various operators f in an LLM, can we approximate $f(e)$ using some efficient function of e_{un} and M?*

Sequence-independent Operators For any operator f that processes each sequence location independently, we can express $f(e)$ as $f(e) = M\ f(e_{\text{un}})$ by definition. This means that we only need to apply f to the unique embeddings e_{un}, significantly reducing computational cost while preserving the original outputs. Many key components of modern LLMs fall into this category, including Linear layers, Activation functions (ReLU, GeLU, etc.), and Layer Normalization (along the embedding dimension D). The overall complexity of the MLP layers is reduced from $O(ND^2)$ to $O(N_{\text{un}}D^2)$, resulting in a linear speedup with $N_{\text{un}} \ll N$.

Virtual Token Unmerging for Self-Attention Blocks A common layer in recent LLMs is the Self-Attention operation with Rotary Position Embedding (RoPE). Unlike sequence-independent operators, self-attention considers pairwise interactions between embeddings and assigns a unique position to each of the N locations in e. Consequently, directly applying $f(e_{\text{un}})$ fails to capture the structure of e, generally leading to significant discrepancies in the output.

To address this, we provide a theoretical derivation of an efficient method to compute $f(e)$ while preserving the benefits of token reduction. The key insight is to reconstruct the self-attention matrix without explicitly expanding the token sequence. We leverage the linearity of the RoPE transformation to efficiently simulate the appropriate repetitions and the positions of the unique embeddings, significantly reducing computational overhead while maintaining consistency with the full sequence computation.

[§]AnyRes [18] leads to multiple fixed length embeddings.

Let $Q = W_q e$, $K = W_k e$ and $V = W_v e$ be the full query, key and value matrices. Similarly, $Q_{\text{un}}, K_{\text{un}}$ and V_{un} are the unique queries, keys and values satisfying the mapping M defined above. The RoPE Self-Attention similarity matrix is computed as $A = \texttt{RoPE}(Q) \cdot \texttt{RoPE}(K)^T$.

For simplicity, let us consider the case where $D = 2$, so that we can write $Q = [Q_1, Q_2]$ where $Q_1, Q_2 \in \mathbb{R}^N$. We will follow a similar notation for all queries, keys and values. This allows us to express each query and key as a complex number *i.e.* $Q[n] = Q_1[n] + iQ_2[n]$. Let $\boldsymbol{\theta} \in [0, 2\pi)^N$ be the rotation angle associated with each position for RoPE. For positions $n, m \in 1, 2, \ldots N$, the RoPE-based similarity [37] is defined as:

$$A[m,n] = \text{Re}\big(\, e^{i\boldsymbol{\theta}[\boldsymbol{m}]} Q[m] \;\; \overline{e^{i\boldsymbol{\theta}[\boldsymbol{n}]} K[n]} \,\big) = \text{Re}\big(\, Q[m]\overline{K[n]} \;\; e^{i(\boldsymbol{\theta}[m] - \boldsymbol{\theta}[n])} \,\big) \tag{5}$$

where $\overline{x}, \text{Re}(x)$ denote the complex conjugate and the real part of x respectively. This can be expanded as:

$$\begin{aligned} A[m,n] = (Q_1[m]K_1[n] + Q_2[m]K_2[n])\cos(\boldsymbol{\theta}[m] - \boldsymbol{\theta}[n]) \\ + (Q_1[m]K_2[n] - Q_2[m]K_1[n])\sin(\boldsymbol{\theta}[m] - \boldsymbol{\theta}[n]) \end{aligned} \tag{6}$$

We also have the trigonometric identities:

$$\begin{aligned} \cos(\boldsymbol{\theta}[m] - \boldsymbol{\theta}[n]) &= \cos(\boldsymbol{\theta}[m])\cos(\boldsymbol{\theta}[n]) + \sin(\boldsymbol{\theta}[m])\sin(\boldsymbol{\theta}[n]) \\ \sin(\boldsymbol{\theta}[m] - \boldsymbol{\theta}[n]) &= \sin(\boldsymbol{\theta}[m])\cos(\boldsymbol{\theta}[n]) - \cos(\boldsymbol{\theta}[m])\sin(\boldsymbol{\theta}[n]) \end{aligned} \tag{7}$$

Let $C = \texttt{diag}(\cos(\boldsymbol{\theta})), S = \texttt{diag}(\sin(\boldsymbol{\theta}))$. Using Eq 6 & 7, the matrix form for self-attention similarities is:

$$A = CQK^\top C + SQK^\top S + S(Q \times K^\top)C - C(Q \times K^\top)S \tag{8}$$

where $QK^T = Q_1K_1^T + Q_2K_2^T$, $Q \times K^\top = Q_1K_2^\top - Q_2K_1^\top$. This formulation can be applied to queries and keys of any dimension D by repeating this for the $(D/2)$ complex numbers obtained by dividing the representation into two parts. In practice, a different $\boldsymbol{\theta}$ is used for each of the $(D/2)$ components.

Using this formulation and the mapping M, we can rewrite the attention matrix in terms of the unique queries and keys as:

$$A = CMQ_{\text{un}}K_{\text{un}}^\top M^\top C + SMQ_{\text{un}}K_{\text{un}}^\top M^\top S + SM(Q_{\text{un}} \times K_{\text{un}}^\top)M^\top C - CM(Q_{\text{un}} \times K_{\text{un}}^\top)M^\top S \tag{9}$$

Observe CM, M^TC, SM, M^TS are highly sparse, each with at most N non-zero entries. These matrices can also be pre-computed and reused across all self-attention layers. Computing $Q_{\text{un}}K_{\text{un}}^\top$ and $Q_{\text{un}} \times K_{\text{un}}^\top$ incurs an $O(N_{\text{un}}^2)$ cost whereas the each of the other matrix multiplications in Eq 9 can be efficiently computed using sparse matrix operations in $O(NN_{\text{un}})$. We can then use the attention matrix to compute the final output of the layer as: $f(e) = \texttt{smax}(\frac{A}{\sqrt{D}})V = [\texttt{smax}(\frac{A}{\sqrt{D}})M]V_{\text{un}}$

Unfortunately, the output $f(e) \in \mathbb{R}^{N \times D}$ will not necessarily exhibit the same redundancy as e. This in turn means that the future self-attention layers cannot benefit from the efficiency of virtual token unmerging. In order to remedy this, before passing the output to the future layers, we re-introduce the redundancy by averaging the embeddings in the positions that were originally equal. We denote this *re-merged* output by $f'(e_{\text{un}}, M)$ which can be written as:

$$f'(e_{\text{un}}, M) = (M^\top M)^{-1}M^T f(e) = (M^\top M)^{-1}M^T \texttt{smax}(\frac{A}{\sqrt{D}})V \tag{10}$$

While the above averaging operation breaks the exactness of the future operations, we observe empirically (see § 4) that this re-merging of tokens, that are known to be redundant, causes minimal drop in performance.

Overall Efficiency The computation of attention matrix A incurs a cost of $O(N_{\text{un}}^2 D + NN_{\text{un}}D)$ (due to the $D/2$ components). Followed by the softmax and sparse matrix multiplications in Eq 10 which incur a cost of $O(N^2 + N_{\text{un}}^2 D)$. Therefore, the overall complexity for RoPE Self-Attention with Virtual Token Unmerging is $O(N_{\text{un}}ND)$. For comparison, the full RoPE Self-Attention on a

Table 1: Comparison of million floating-point operations per second (MFLOPs) of the attention blocks on full sequence and DyMU sequence w/ and w/o VTU. N refers to full sequence length, N_{un} refers to unique sequence length after merging. The statistics are computed with batch size 1, head number 32, and head dimension 128. We use the fvcore package for counting FLOPs.

Methods	Avg N_{un}/N	MFLOPs
Full Attention	576 / 576	1359.0
DyMU-low w/o VTU	89 / 576	32.4
DyMU-mid w/o VTU	195 / 576	155.75
DyMU-high w/o VTU	394 / 576	635.85
DyMU-low w/ VTU	89 / 576	64.9
DyMU-mid w/ VTU	195 / 576	311.5
DyMU-high w/ VTU	394 / 576	1272.0

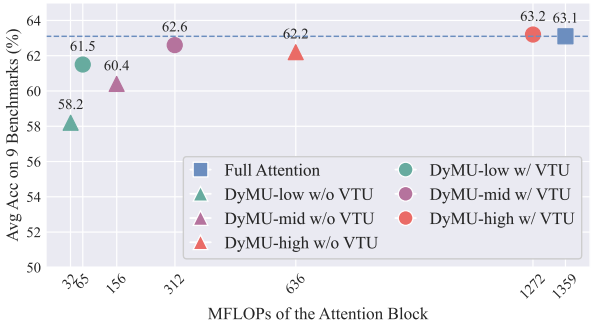

Figure 3: Accuracy vs. FLOPs. Without VTU, DyMU achieves greater reduction in FLOPs, but also resulting in a more significant drop in performance. Adding VTU achieves a better trade-off between performance and efficiency, particularly at *lower token counts*—for example, DyMU-mid preserves 99.2% of the original performance while using 4.3× fewer FLOPs.

Table 2: Comparison with state-of-the-art methods for improving efficiency on LLaVA 1.5 [25]. DyMU-low achieves 97.5% of the original full-length LLaVA baseline's performance while using only ∼15% of the tokens. Importantly, DyMU is entirely training-free and generally outperforms previous fixed-length, training-free methods such as [3, 5, 49], while also enabling variable-length outputs based on image complexity.

Methods	# Visual Tokens	Compression in Encoder	GQA	MMB	MME (prcp, all)	POPE	SQA^I	$SEED^I$	VQA^T	MMVet	$LLaVA^W$	Avg
LLaVA-1.5-7B	576	-	62.0	64.6	1506,1862	86.9	69.4	66.2	58.3	30.7	63.5	63.1
Fixed Length Compression & Training-Required												
MQT-LLaVA [14]	256	No	61.6	64.3	1435, -	84.4	67.6	-	-	29.8	64.6	-
Prumerge [35]	32	No	-	60.9	1350, -	76.3	68.5	-	56.0	-	-	-
Prumerge++ [35]	144	No	-	64.9	1462, -	84.0	68.3	-	57.1	-	-	-
LLaMA-VID [21]	2	No	55.5	-	- , -	83.1	68.8	-	49.0	-	-	-
VoCo-LLaMA [44]	1	No	57.0	58.8	1323, -	81.4	65.4	-	-	-	-	-
TokenPacker [20]	36	No	59.6	62.8	- , -	86.2	-	-	-	29.6	-	-
LLaVA-Mini [48]	1	No	60.9	65.6	1466, -	84.4	70.4	-	57.0	36.6	68.9	-
Dynamic Length Compression & Training-Required												
DiffRate [6]	∼57	Yes	57.9	-	1341, -	-	66.4	-	30.6	-	-	-
Fixed Length Compression & Training-Free												
Prumerge-no-ft [35]	32	No	-	-	1250, -	76.2	68.0	-	54.0	-	-	-
FastV [5]	128	No	49.6	56.1	- , 1490	53.4	64.4	-	50.6	26.3	-	-
PDrop [40]	128	No	56.6	61.4	- , 1713	82.3	69.2	-	55.9	30.8	-	-
SparseVLM [49]	128	No	57.2	62.3	- , 1721	85.0	67.8	-	55.8	29.0	-	-
PiToMe [6]	∼57	Yes	59.9	-	1448, -	-	69.0	-	43.0	-	-	-
VisionZip [42]	128	No	57.6	62.0	-, 1762	83.2	68.9	-	56.8	32.6	64.8	-
ToMe [3]	94	Yes	57.3	59.7	1357, 1673	86.8	68.9	60.5	53.2	25.6	61.0	59.2
ToMe [3]	209	Yes	59.2	62.4	1418, 1734	87.4	69.2	63.5	54.9	30.9	62.9	61.4
ToMe [3]	393	Yes	59.5	64.1	1454, 1769	86.7	68.4	65.1	55.8	30.8	66.0	62.2
Variable Length Compression & Training-Free												
DyMU-low	$89_{\pm 27}$	Yes	60.8	62.1	1438, 1787	86.3	69.3	65.0	53.1	30.0	62.9	61.5
DyMU-mid	$195_{\pm 47}$	Yes	61.7	62.8	1483, 1862	86.6	69.2	65.9	55.1	30.9	65.1	62.6
DyMU-high	$394_{\pm 57}$	Yes	61.9	64.3	1498, 1846	86.8	69.9	66.1	58.0	31.5	64.5	63.2

sequence length of N would be an $O(N^2D)$ operation. Therefore, in theory, efficiency improves approximately linearly with the number of redundant tokens in terms of FLOPs. Table 1 shows the FLOPs comparison for the attention block under different levels of token reduction. Figure 3 further illustrates the trade-off between performance and efficiency. We demonstrate that VTU significantly improves performance while still maintaining lower FLOPs than full attention, especially under more aggressive reduction regimes. In practice, we find that the wall-clock time difference for the attention blocks and the end-to-end inference is less significant due to PyTorch's highly optimized dense matrix-multiplication implementations. We provide an in-depth discussion of this gap in §5.

4 Experiments

We present a comprehensive analysis demonstrating the practical benefits and efficacy of utilizing DyMU with various visual encoders and VLMs. Implementation details on the model variants and threshold-finding datasets can be found in **Appendix A**.

Table 3: DyMU demonstrates similar efficacy on a different visual encoder, SigLIP [47]. We obtain the baseline by following the same training recipe as LLaVA-1.5 [25]. DyMU-SigLIP-low achieves 96.1% of the baseline performance while using ∼15% visual tokens.

Methods	# Visual Tokens	GQA	MMB	MME (prcp, all)	POPE	SQA^I	SEED^I	VQA^T	MMVet	LLaVA^W	Avg
LLaVA-1.5-w-SigLIP	576	62.7	65.1	1471, 1770	85.7	68.2	66.7	57.6	30.2	59.8	62.1
ToMe [3]	114	59.3	61.4	1380, 1717	85.1	66.9	61.8	52.1	26.1	57.9	59.1
DyMU-SigLIP-low	$90_{\pm 26}$	61.3	62.5	1398, 1695	84.9	66.7	64.4	51.8	26.7	58.6	59.7
DyMU-SigLIP-mid	$176_{\pm 43}$	62.2	63.9	1442, 1744	85.0	67.4	65.2	54.5	26.7	59.5	60.7
DyMU-SigLIP-high	$318_{\pm 57}$	62.4	65.0	1449, 1765	86.0	67.6	66.0	56.8	29.4	58.3	61.6

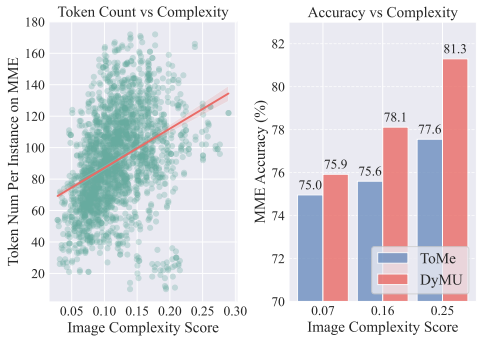

Figure 4: Image Complexity vs Token Count and Accuracy. The scatter plot (**left**) demonstrates a strong correlation between DyMU's token count and image complexity score. On the **right**, MME accuracy at varying complexity levels is compared between ToMe (fixed-length) and DyMU (dynamic-length), highlighting the benefit of assigning additional tokens to complex images.

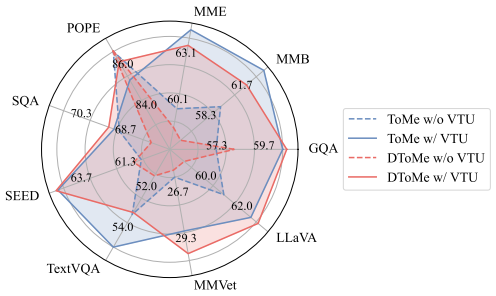

Figure 5: Importance of Virtual Token Unmerging (VTU). We ablate the performance of LLaVA 1.5 with two token reduction methods applied to the visual encoder—ToMe (fixed-length) and DToMe (dynamic-length). We observe that applying VTU significantly improves performance on 8 out of 9 benchmarks, demonstrating robustness to varied token reduction methods.

4.1 Quantitative Evaluation

Comparing Visual Token Reduction Methods for VLMs To evaluate the efficacy of our approach, we compare it against several existing methods that focus on reducing the number of tokens in VLMs. To the best of our knowledge, our proposed method is the first to (1) natively produce variable-length visual tokens and (2) require neither further fine-tuning of the VLM nor additional textual conditions. In Table 2, we present a quantitative evaluation of all methods applied to a pre-trained LLaVA 1.5 [25] architecture on standard VLM benchmarks, including GQA [16], MMBench [27], MME [11], POPE [22], ScienceQA [28], SEED-IMG [19], TextVQA [36], MMVet [46], LLaVA-Bench [25]. DyMU achieves average performances of 97.5%, 99.2%, and 100.2%, relative to the original pretrained model, while reducing the token number by 84.5%, 66.1%, and 31.6%, respectively. DyMU also outperforms previous training-free methods while enabling varied length output per instance. When decreasing the number of tokens, the largest performance drop occurs in TextVQA, which aligns with our expectation—understanding visual text is highly sensitive to the spatial locations of visual tokens, which are often disrupted by token merging. We leave addressing this issue to future work.

Compatibility with Different LLMs and Visual Encoders DyMU can be seamlessly integrated into multiple variants of VLMs featuring different LLMs, visual encoders, and pretraining strategies. In Tables 2 and 3, we demonstrate that DyMU effectively maintains baseline performance when applied both CLIP [34] to SigLIP [47] representations within the LLaVA 1.5 framework, using a Vicuna-7B [7] LLM. Furthermore, in Table 4 we evaluate DyMU on LLaVA-OneVision [18], a recent Any-Resolution (AnyRes) model with SigLIP-so400M [1] as visual encoder and Qwen2 [41] as LLM backbone. AnyRes enables processing images of arbitrary resolutions by

Table 4: DyMU shows consistent effectiveness on an AnyRes VLM, LLaVA-OneVision [18]. We additionally show performance on two comprehensive video understanding benchmarks.

Methods	% Visual Tokens	Image Benchmarks MMB	MME	SEED	MathVista	Video Benchmarks VidMME	MMBVid
LLaVA-ov-7B	100%	79.3	75.8	75.6	58.0	61.3	1.18
ToMe [3]	14.4%	71.2	63.1	68.3	46.6	57.6	1.08
DyMU-ov-low	∼14.4%	73.6	68.0	72.9	47.4	59.3	1.08
DyMU-ov-mid	∼25.1%	76.0	70.3	73.7	51.7	60.1	1.12
DyMU-ov-high	∼46.5%	77.8	73.6	74.2	54.4	60.1	1.16

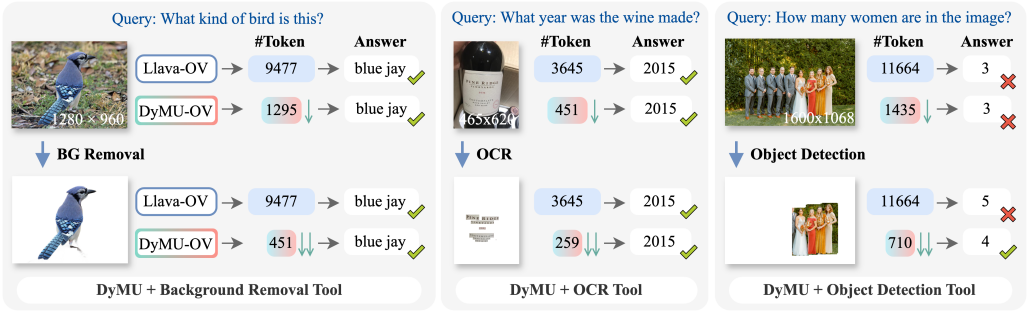

Figure 6: Controllable Visual Token Length. DyMU enables more flexible control over computational cost. In these examples, we combine DyMU with additional task-specific tools—background removal, OCR, or object detection—to focus only on the relevant regions. As a result, token count is substantially reduced without degrading performance, showcasing the flexibility of DyMU to adapt to different tasks.

segmenting them into smaller regions and encoding each individually. Our results show that DyMU remains compatible with this complex operation on both image and video tasks.

Image Complexity vs Number of Tokens In Figure 4 (left), we show how the number of tokens varies with image complexity. We quantify image complexity $C(I)$ by computing the JPEG compression ratio, i.e., $C(I) = \frac{S_{JPEG}(I)}{H \times W}$, where S_{JPEG} is the size (in bytes) of the image I after JPEG encoding, and H, W are the original height and width. For this experiment, we use CLIP-L/14-336 with DToMe *-low* to encode images in the MME benchmark. We observe a strong correlation between the number of output tokens and image complexity, indicating that DToMe effectively preserves essential details in complex images while reducing redundancy in simpler ones. We include more qualitative visualizations in Figure 8.

Fixed vs Dynamic Token Reduction In Figure 4 (right), we categorize images into three bins based on their complexity scores, and compare the performance of ToMe (fixed-length token reduction) and DToMe on the MME benchmark. A key drawback of fixed token reduction is its inability to adapt to image complexity, leading to over-compression for complex images and under-compression for simpler ones. While our method outperforms ToMe across all complexity levels, we observe the most significant gains on complex images, where ToMe struggles due to an insufficient number of tokens.

Importance of Virtual Token Unmerging VTU efficiently reconstructs the representation of a full visual token sequence from a reduced set of visual tokens. To demonstrate its impact, we compare LLaVA 1.5 variants with and without VTU. In the latter, the LLM does not undergo any modifications and directly receives fewer tokens. In Figure 5, we evaluate this effect on two token reduction methods: ToMe [3], which produces fixed-length sequences, and DToMe (ours). Across both cases, we observe that applying VTU significantly improves performance on 8 out of 9 benchmarks, demonstrating its effectiveness in preserving model capabilities despite token reduction.

4.2 Qualitative Analysis

Visualizing Variable Visual Token Length DToMe facilitates producing variable number of token embeddings for images based on complexity of the content. In Appendix Figure 8, we visualize the number of visual tokens for various images from nine benchmarks. For each benchmark, we present three images corresponding to the minimum, median, and maximum token numbers output by DyMU-low. We observe a strong correlation, both within and across different benchmarks, between image complexity and the number of tokens retained by DyMU. In Figure 7, we further visualize the token merging behavior at the patch level, showing that DyMU dynamically merges more tokens in regions of low complexity.

Controllable Visual Token Length DToMe offers a key advantage over fixed token reduction methods: cost controllability. By dynamically adjusting the number of visual tokens based on image complexity, users gain more control over the computational cost incurred per image. This flexibility allows flexible combination of task-specific tools with DyMU to further boost efficiency while maintaining performance. For instance, in Figure 6, we show example applications of combining DyMU with background removal [4], OCR [10], and object detection [31] models, to extract focused regions and further reduce token count. Unlike existing VLMs, which impose a fixed token budget

Figure 7: Visualization of merged patch tokens by DyMU. Patches with the same color are merged into a single token, and the number inside each patch indicates the final merged token index. We find that the merging adapts to the complexity of the regions: simpler regions, such as the sky or a white background, tend to be merged more, while complex regions preserve a larger number of unique tokens.

per image regardless of content, our method enables adaptive token allocation, ensuring that simpler regions consume fewer resources while more complex regions retain the necessary level of detail.

5 Improvement Gap Between FLOPs and Inference Time

In Table 1 and Figure 3, we show that DyMU achieves substantial FLOP reductions while preserving model performance. In Table 5, we further examine inference time metrics and observe less pronounced improvements. In this section, we provide a deeper analysis of why there exists a non-trivial gap between FLOP reduction and actual inference time gains, particularly for algorithms like DyMU that fundamentally alter attention computation. Unlike FLOPs, which serve as a hardware-agnostic measure of algorithmic efficiency, wall-clock inference time can fluctuate due to hardware conditions, batching strategies, and shared server workloads. Moreover, implementation details often dominate runtime behavior—low-level kernel optimizations and computation scheduling can introduce substantial discrepancies between theoretical and practical efficiency. For example, we compare two implementations for computing an N$\times$N attention matrix from two (N, D) vectors. In Version 1, a single `torch.matmul` operation produces the full matrix, whereas Version 2 splits the vectors into N/4 chunks and performs matrix multiplications on the corresponding submatrices before combining the results. Both methods yield the same total FLOPs (339.74 MFLOPs), but Version 2 runs significantly slower (2.311 ms vs. 1.374 ms) because multiple small matrix multiplications underutilize optimized GPU kernels. Similarly, our VTU operation decomposes large multiplications into smaller components, achieving theoretical FLOP efficiency but incurring longer wall-clock times. This example highlights that translating theoretical compute savings into practical speedups often requires non-trivial engineering efforts, such as developing new kernels, efforts comparable to those seen in works like FlashAttention [8].

6 Conclusions and Limitations

In this work, we introduced DyMU, the first training-free framework that dynamically reduces visual token counts in VLMs based on per-image complexity. DyMU can be directly integrated into mainstream VLM architectures that combine ViT-based visual encoders with RoPE-based LLM backbones. Despite its effectiveness, DyMU still faces limitations on information-dense, token-sensitive tasks, such as TextVQA [36], OCR [32], and DocVQA [30], where preserving every fine-grained visual cue is crucial for maintaining accuracy. Addressing this issue may require future work on complexity-aware token merging mechanisms capable of selectively retaining semantically critical regions in information-dense scenes. Additionally, extending DyMU to reduce temporal redundancy in videos represents a promising future direction.

Acknowledgments

We would like to express our gratitude to the anonymous reviewers for their insightful comments and suggestions. We would also like to thank our colleagues at Salesforce Research for their valuable internal discussions and feedback. This research is based upon work supported by U.S. DARPA ECOLE Program No. #HR00112390060. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright.

References

[1] Ibrahim M Alabdulmohsin, Xiaohua Zhai, Alexander Kolesnikov, and Lucas Beyer. Getting vit in shape: Scaling laws for compute-optimal model design. Advances in Neural Information Processing Systems, 36:16406–16425, 2023.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. arXiv preprint arXiv:2502.13923, 2025.

[3] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. arXiv preprint arXiv:2210.09461, 2022.

[4] Bria AI. RMBG-1.4: Background Removal Model. `https://huggingface.co/briaai/RMBG-1.4`, 2024.

[5] 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. In European Conference on Computer Vision, pages 19–35. Springer, 2024.

[6] Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Fei Chao, Rongrong Ji, Yu Qiao, and Ping Luo. Diffrate: Differentiable compression rate for efficient vision transformers. In Proceedings of the IEEE/CVF international conference on computer vision, pages 17164–17174, 2023.

[7] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality, March 2023.

[8] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in neural information processing systems, 35:16344–16359, 2022.

[9] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. arXiv preprint arXiv:2409.17146, 2024.

[10] Felix Dittrich. Onnxtr: Optical character recognition made seamless & accessible to anyone, powered by onnx. `https://github.com/felixdittrich92/OnnxTR`, 2024.

[11] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, and Rongrong Ji. MME: A Comprehensive Evaluation Benchmark for Multimodal Large Language Models. arXiv prepring arXiv:2306.13394, 2023.

[12] 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 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[13] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3608–3617, 2018.

[14] Wenbo Hu, Zi-Yi Dou, Liunian Harold Li, Amita Kamath, Nanyun Peng, and Kai-Wei Chang. Matryoshka query transformer for large vision-language models. arXiv preprint arXiv:2405.19315, 2024.

[15] Wenxuan Huang, Zijie Zhai, Yunhang Shen, Shaoshen Cao, Fei Zhao, Xiangfeng Xu, Zheyu Ye, and Shaohui Lin. Dynamic-llava: Efficient multimodal large language models via dynamic vision-language context sparsification. arXiv preprint arXiv:2412.00876, 2024.

[16] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 6700–6709, 2019.

[17] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. Int. J. Comput. Vis., 123(1):32–73, 2017.

[18] 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, 2024.

[19] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. SEED-Bench: Benchmarking Multimodal LLMs with Generative Comprehension. arXiv prepring arXiv:2307.16125, 2023.

[20] Wentong Li, Yuqian Yuan, Jian Liu, Dongqi Tang, Song Wang, Jie Qin, Jianke Zhu, and Lei Zhang. Tokenpacker: Efficient visual projector for multimodal llm. arXiv preprint arXiv:2407.02392, 2024.

[21] Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. In European Conference on Computer Vision, pages 323–340. Springer, 2024.

[22] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. arXiv preprint arXiv:2305.10355, 2023.

[23] Xiaoyu Liang, Chaofeng Guan, Jiaying Lu, Huiyao Chen, Huan Wang, and Haoji Hu. Dynamic token reduction during generation for vision language models. arXiv preprint arXiv:2501.14204, 2025.

[24] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In Computer Vision - ECCV 2014 - 13th European Conference, 2014.

[25] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning. arXiv prepring arXiv:2310.03744, 2023.

[26] Xiangcheng Liu, Tianyi Wu, and Guodong Guo. Adaptive sparse vit: Towards learnable adaptive token pruning by fully exploiting self-attention. arXiv preprint arXiv:2209.13802, 2022.

[27] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is Your Multi-modal Model an All-around Player? arXiv prepring arXiv:2307.06281, 2023.

[28] 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.

[29] Dmitrii Marin, Jen-Hao Rick Chang, Anurag Ranjan, Anish Prabhu, Mohammad Rastegari, and Oncel Tuzel. Token pooling in vision transformers. arXiv preprint arXiv:2110.03860, 2021.

[30] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. Docvqa: A dataset for vqa on document images. In Proceedings of the IEEE/CVF winter conference on applications of computer vision, pages 2200–2209, 2021.

[31] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In European conference on computer vision, pages 728–755. Springer, 2022.

[32] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In ICDAR, 2019.

[33] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision. arXiv prepring arXiv:2304.07193, 2023.

[34] 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 Proceedings of the 38th International Conference on Machine Learning (ICML), 2021.

[35] 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.

[36] Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8317–8326, 2019.

[37] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 568:127063, 2024.

[38] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint arXiv:2409.12191, 2024.

[39] Xinjian Wu, Fanhu Zeng, Xiudong Wang, and Xinghao Chen. Ppt: Token pruning and pooling for efficient vision transformers. arXiv preprint arXiv:2310.01812, 2023.

[40] Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. arXiv preprint arXiv:2410.17247, 2024.

[41] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024.

[42] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. In Proceedings of the Computer Vision and Pattern Recognition Conference, pages 19792–19802, 2025.

[43] Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. Atp-llava: Adaptive token pruning for large vision language models. arXiv preprint arXiv:2412.00447, 2024.

[44] Xubing Ye, Yukang Gan, Xiaoke Huang, Yixiao Ge, Ying Shan, and Yansong Tang. Voco-llama: Towards vision compression with large language models. arXiv preprint arXiv:2406.12275, 2024.

[45] Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 10809–10818, 2022.

[46] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. MM-Vet: Evaluating Large Multimodal Models for Integrated Capabilities. arXiv prepring arXiv:2308.02490, 2023.

[47] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In Proceedings of the IEEE/CVF international conference on computer vision, pages 11975–11986, 2023.

[48] Shaolei Zhang, Qingkai Fang, Zhe Yang, and Yang Feng. Llava-mini: Efficient image and video large multimodal models with one vision token. arXiv preprint arXiv:2501.03895, 2025.

[49] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. arXiv preprint arXiv:2410.04417, 2024.

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 claims are clearly presented in the abstract and intro, and well supported by empirical results and analysis in the section 4.

   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: The limitations are transparently discussed in Section 4.1 and Section 5.

   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: [Yes]

Justification: We provide a detailed step-by-step derivation in Section 3.2 for one of our key contribution, the virtual token unmerging (VTU).

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: We provide implementation details in Appendix A, and also provide the code as part of the supplementary.

   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: All datasets used in this work are publicly available. We also provide code as part of supplemental material.

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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We include all details in the Section 4 and Appendix A.

   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: We provide detailed information on how we evaluation is done and how the scores are computed in Appendix A.

   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 provide this information in Appendix A.

   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 conform with the NeurIPS Code of Ethics.

   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: [NA]

    Justification: The paper is foundational research and do not directly tied to a specific application.

    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: This paper does not have such risks.

    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: All datasets, code and models are properly cited.

    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: [Yes]

    Justification: We provide readme for how to use the code and models.

    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: This paper does not involve crowdsourcing and 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: This paper does not involve crowdsourcing and 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.

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.

A Implementation Details

Dynamic Token Merging For DToMe, we find layer-wise thresholds using a diverse dataset of 250k images sampled from the SFT instruction tuning data of LLaVA 1.5 [25] comprising of images from MS-COCO [24], VisualGenome [17], OCR-VQA [32], TextVQA [36] and GQA [16]. We also ablate the choice of image datasets in §E. In general, a sufficiently diverse image set suffices, and performance remains robust to dataset changes. Importantly, we only use the images to estimate the thresholds (in inference mode) and do not use the associated annotations or text in any way.

DyMU variants For each visual encoder in the experiments, including CLIP [34][¶] and SigLIP [47, 1][‖**], we find thresholds for three variants of the encoder by choosing different *average* number of tokens to drop (r_i) in each layer. We represent these variants by •*-low,*•*-mid,*•*-high* corresponding to the expected average number of tokens. We also explore different VLMs including fixed-resolution models, e.g., LLaVA 1.5 [25] and any-resolution models, e.g., LLaVA-OneVision [18].

Evaluation Details For results on LLaVA-1.5 (as in Tables 2 and 3) we leverage the official evaluation code from LLaVA-1.5. The results on MME and LLaVA-Bench for DyMU are averaged across three runs, as we observe a higher variance on these two tasks. We omit VQAv2 [12] and VizWiz [13] as the evaluation server was unavailable at the time of writing. For results on LLaVA-OneVision (as in Table 4), we leverage VLMEvalKit[††] for getting the evaluation results. For ToMe [3], we implemented the code for connecting ToMe to different VLMs under the same framework as DyMU. All results for ToMe are evaluated with the same setting as DyMU to ensure fair comparison. For other prior work, the results are copied from their original papers.

B Additional Analysis on Wall-Clock Inference Time

Table 5: Additional efficiency analysis comparing FLOPs and wall-clock inference times.

Methods	N_{un}/N	Attn MFLOPs	Attn Inference Time	End-to-End Inference Time
Full Attention	576 / 576	1359.0	9.17	131
DyMU-low w/o VTU	89 / 576	32.4	1.26	115
DyMU-mid w/o VTU	195 / 576	155.75	1.29	121
DyMU-high w/o VTU	394 / 576	635.85	2.92	132
DyMU-low w/ VTU	89 / 576	64.9	7.20	131
DyMU-mid w/ VTU	195 / 576	311.5	7.49	123
DyMU-high w/ VTU	394 / 576	1272.0	7.60	132

C Visualization of Variable Token Length

In Figure 8, we present a comprehensive visualization of example images along with their encoded visual token counts. We use DyMU-low (based on CLIP-L/14-336) as the encoder, where the full token length is 576. Three images are shown for each benchmark, corresponding to the minimum, median, and maximum number of tokens, respectively. A clear correlation can be observed between semantic richness and token count. We also note variations in the token range across different benchmarks. For instance, ScienceQA [28], which primarily contains figures and charts, tends to have fewer tokens than benchmarks featuring complex real-world scenes.

D Impact of Token Merging Schedule

We conduct an additional ablation study on one of the hyperparameters in DToMe, the merging schedule, during threshold finding. As detailed in Section 3, we set a target reduction number, r_i , for each layer. By default, r_i is set to a constant value across all layers. Alternatively, we can vary r_i across layers to encourage merging more or fewer tokens at different depths.

[¶]CLIP version: openai/clip-vit-large-patch14-336

[‖]SigLIP with LLaVA-1.5: timm/ViT-B-16-SigLIP-384

[**]SIgLIP version with LLaVA-OV: google/siglip-so400m-patch14-384

[††]`https://github.com/open-compass/VLMEvalKit`

Figure 8: DToMe Token Count Across Benchmarks. For each dataset, we show three examples processed by our method—those yielding the fewest tokens, the median number of tokens, and the most tokens. Observe that visually simple or nearly blank images consistently require fewer tokens, while more detailed, semantically complex or cluttered images produce more tokens. This demonstrates how DToMe effectively adapts to image complexity across diverse benchmarks, allocating fewer tokens to simpler content and preserving more tokens for complex scenes.

Table 6: Ablation study on merging schedules in DToMe. We compare three strategies: constant, linear (more merging in early layers), and reverse linear (more merging in later layers). Results show that merging fewer tokens in early layers yields better performance, while the constant schedule provides a balanced trade-off between performance and token count.

Schedule	# Visual Tokens	GQA	MMB	MME$^{(prcp,all)}$	POPE	SQA^{I}	SEED^{I}	VQA^{T}	MMVet	LLaVA^{W}	Avg
Constant	$195_{\pm 47}$	61.7	62.8	1483, 1862	86.6	69.2	65.9	55.1	30.9	65.1	62.6
Linear	$163_{\pm 43}$	61.3	62.3	1437, 1767	86.2	69.4	65.3	52.1	28.8	58.6	60.8
Reverse Linear	$213_{\pm 49}$	61.8	63.8	1491, 1863	86.7	69.3	66.0	57.5	31.8	65.3	63.2

In Table 6, we present an ablation study on two alternative scheduling strategies: (1) *linear*, which merges *more* tokens in *earlier* layers and *fewer* tokens in *later* layers, and (2) *reverse linear*, which follows the opposite trend. The results indicate that merging fewer tokens in earlier layers tends to yield better performance, while the constant schedule provides a balanced trade-off between performance and token count. This observation echoes the findings in the ToMe paper [3], where a constant schedule was found to be nearly optimal.

E Impact of Dataset for Threshold Finding

The DToMe thresholds are computed using images from the LLaVA instruction tuning dataset. Here, we investigate the sensitivity of DToMe to the threshold estimation dataset. In Figure 9, we evaluate DYMU-LLaVA 1.5 with DToMe thresholds estimated on the Pixmo-Cap [9] image-captioning dataset.

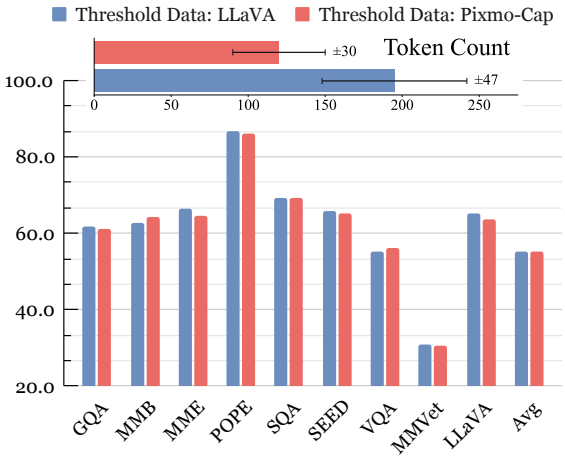

Figure 9: Comparing thresholds using LLaVA Instruct Data vs Pixmo-Cap. Although both methods use the same per-layer merging hyperparameter (r_i), the Pixmo-based thresholds lead to fewer tokens (**top**)—likely due to domain differences. However, performance across a range of benchmarks shows minimal drop (**bottom**), indicating the robustness of DToMe on different image datasets.

Table 7: Impact of Virtual Token Unmerging. Full results for Figure 5.

Method	# Visual Tokens	GQA	MMB	MME$^{(prcp,all)}$	POPE	SQA^I	SEED^I	VQA^T	MMVet	LLaVA^W	Avg
ToMe [3]	94	57.3	59.7	1357, 1673	86.8	68.9	60.5	53.2	25.6	61.0	59.2
+ VTU	94	60.6	63.7	1464, 1815	85.4	69.1	64.9	54.8	28.7	62.5	61.6
DyMU-low	$89_{\pm 27}$	60.8	62.1	1438, 1787	86.3	69.3	65.0	53.1	30.0	62.9	61.5
w/o VTU	$89_{\pm 27}$	58.2	56.0	1346, 1639	86.9	67.7	60.9	51.3	25.2	58.8	58.2

We observe a minimal performance change across all the benchmarks, highlighting the robustness of our method to dataset variation. Interestingly, we observe that the thresholds estimated using the Pixmo-Cap dataset lead to fewer tokens during inference on the benchmarks. We hypothesize that this is due to the domain shift between the Pixmo-Cap images and a more diverse LLaVA-instruct dataset which covers diverse real-world use cases.

F Full Results for Figure 5

We present the complete results of the ablation experiments on the effect of our proposed Virtual Token Unmerging, as shown in Figure 5. The results are provided in Table 7.

G Full Results for Figure 9

We present the complete results of the ablation experiments on threshold-finding datasets, as shown in Figure 9. The results are provided in Table 8.

Table 8: Impact of dataset for threshold finding. Full results for Figure 9.

Model	Thresh Finding Dataset	# Visual Tokens	GQA	MMB	MME$^{(prcp,all)}$	POPE	SQA^{I}	SEED^{I}	VQA^{T}	MMVet	LLaVA^{W}	Avg
DyMU-mid	**Llava**	$195_{\pm 47}$	61.7	62.8	1483, 1862	86.6	69.2	65.9	55.1	30.9	65.1	62.6
DyMU-mid	**Pixmo**	$120_{\pm 30}$	61.1	64.4	1474, 1808	86.0	69.4	65.3	56.2	30.5	63.7	62.4