# DyMU: Dynamic Merging and Virtual Unmerging for Efficient 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
[*]Equal Contribution

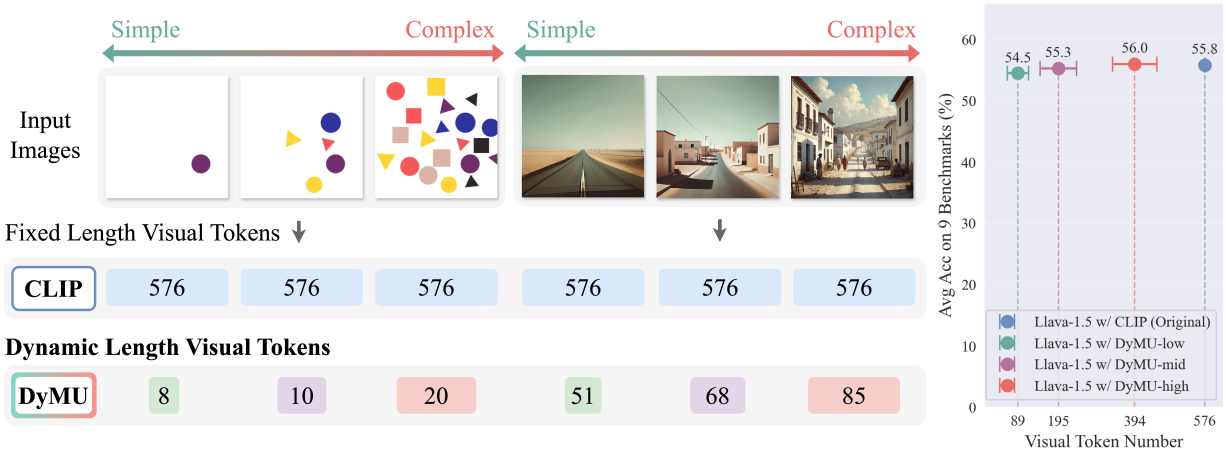


**Figure 1. Dynamic Merging and Virtual Unmerging** (DyMU) adaptively reduces visual token lengths based on image complexity, as shown on the left where simpler images are represented using fewer tokens. In contrast, existing representations (like CLIP) always use the same number of tokens regardless of image content. 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 adapts token compression to the content of the image 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, including the recently popularized AnyRes-based visual encoders. Furthermore, through qualitative analyses we demonstrate that DToMe effectively adapts token reduction based on image complexity, and unlike existing systems, provides users more control over computational costs. Project page:* https://mikewangwzhl.github.io/dymu.

## 1. Introduction

Recent large vision-language models (VLMs) have demonstrated breakthroughs in computer vision tasks such as image captioning [6], open-vocabulary object detection [13], visual-question answering [12] and OCR [31] by leveraging the reasoning capabilities of large language models (LLMs) to enhance visual understanding. Most state-of-the-art 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. These visual embeddings are then processed by LLMs alongside textual query features, enabling complex understanding and 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 [19] and Qwen-2.5VL [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., ∼24 on MME [11]. This stark contrast highlights that the computational burden of processing vision tasks generally arises primarily from the large number of visual tokens.

We first make an interesting observation: in current visual encoders, the number of tokens generated for an image does not depend on the content of the image. In Figure 1, we illustrate this with some examples — a CLIP [33] representation leads to the same embedding size on a blank image with a small circle and on a complex scene depicting buildings, vehicles and people. In contrast, textual tokens are more closely tied to the amount of content conveyed — more words are required to describe more information. An average sentence length in English is around 15–20 words [10], meaning that *regardless of the content of the image*, the language model in LLaVA-OneVision [19] has to process the equivalent of 400-500 sentences for each high-resolution image.

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 (Sec 3.1), which allows the *visual encoder* to generate variable-length token sequences based on the complexity of the image. Second, we present Virtual Token Unmerging (Sec 3.2), enabling the *LLM decoder* to process shorter dynamic visual token sequences while efficiently approximating the full-length sequence. Crucially, we demonstrate that both of these modifications *do not require additional fine-tuning of the pre-trained VLM*. Furthermore, Dynamic Token Merging is compatible with any Vision Transformer (ViT)-based visual encoder, and Virtual Token Unmerging can be applied to any LLM that utilizes Rotary Position Embedding (RoPE) [36].

We show that VLMs modified with our methods can maintain the performance of the full model while reducing the average token count by 32%-85% (see Sec 4.2). In addition to improving efficiency, our approach offers users greater control over token costs compared to existing systems (e.g., GPT-4o), which incur a fixed token count per image based solely on resolution. In Sec 4.3, we demonstrate example applications on how the number of visual tokens can be further reduced by combining DyMU with pre-processing tools such as background removal, object detection, etc. Through comprehensive quantitative experiments (Sec 4.2), we verify that our method works effectively across different VLM architectures, with varying pre-training strategies, visual encoders, and training datasets.

## 2. Related Work

| | Component Improved | Dynamic Length | No Addn. Modules | Training Free | Granularity Control | Extra Cond. |
|---|---|---|---|---|---|---|
| LLaMA-VID [22] | Projector | ❌ | ❌ | ❌ | ❌ | None |
| Fast-V [22] | Decoder | ❌ | ✅ | ✅ | ✅ | None |
| SparseVLM [22] | Decoder | ❌ | ✅ | ✅ | ✅ | Text |
| MQT-LLaVA [14] | Projector | ❌ | ❌ | ❌ | ❌ | None |
| LLaVA-Prumerge [34] | Projector | ❌ | ✅ | ❌ | ✅ | None |
| TokenPacker [34] | Projector | ❌ | ❌ | ❌ | ✅ | None |
| ATP-LLaVA [34] | Decoder | ✅ | ❌ | ❌ | ✅ | Text |
| LLaVA-mini [45] | Projector | ❌ | ❌ | ❌ | ❌ | None |
| **DyMU** | Encoder & Decoder | ✅ | ✅ | ✅ | ✅ | None |

**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 [5, 15, 23, 34, 38, 40, 46], distillation [41], or resampling [14, 20, 22, 45] to improve efficiency in either the projectors or the VLM decoder blocks. However, we identify several key limitations: (1) Most existing methods, including all training-free approaches [5, 38, 46] predefine a fixed compression ratio for any input image regardless of its complexity. While [40] proposed an adaptive token pruning framework that enables variable-length compression, it requires retraining the backbone VLM with additional 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 variable length visual token compression, which can be directly applied to cutting-edge VLM architectures including Any-Resolution models and RoPE embeddings.

**Efficient Vision Transformers** We also draw inspiration from a separate line of research [3, 26, 29, 37, 42] aimed at improving the efficiency of Vision Transformers (ViTs)

themselves, which is still the main go-to architecture for visual encoders [32, 33, 44]. 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 VLM 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 an efficient "virtual 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 proposed method. In Section 3.1, we present our proposed Dynamic Token Merging (DToMe) — a training-free method to dynamically reduce the number of output tokens by visual encoders based on the complexity of the image content. In Section 3.2, we introduce Virtual Token Unmerging (VTU) — an approach to process the reduced visual tokens through the language model while efficiently simulating the standard number of visual tokens. This method utilizes the tracked positions of the redundant tokens to recreate a full attention matrix of the original length.

The combination of both methods is referred to as DyMU, short for Dynamic Merging and Virtual Unmerging. We illustrate the core idea in Figure 2. DyMU can be applied to any VLM that uses transformer-based visual encoders and RoPE-based transformer language models. The proposed modifications to the architecture do not introduce any additional learnable parameters and most importantly, do not require any additional fine-tuning of the VLM.

### 3.1. Dynamic Token Merging (DToMe)

Most recent large vision-language models (VLMs) use vision transformers (ViTs) like CLIP [33] or SigLIP [44] to encode images into a sequence of visual tokens. For a fixed resolution input image, the ViT architecture always outputs the same number of token embeddings, leading to low efficiency in VLMs. Our approach draws inspiration from ToMe[3], a prior work which reduces the number of output tokens 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 Dynamic Token Merging (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.

**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 $x_i$. Drawing inspiration from [3], we follow 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 them. We then compute a bipartite assignment between the two sets of tokens by assigning token $t \in \mathbb{A}$ to $t_B = \arg\max_{n \in \mathbb{B}}(k_i[t]^T \; k_i[n])$ (token with the most similar key). This gives us $|\mathbb{A}|$ edges with scores $S_i[t] = (k_i[t]^T \; k_i[t_B])$ for $t \in \mathbb{A}$. We then apply a threshold $\tau_i$ to retain edges $t \to t_B$ 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 $k_i$. We describe our approach for computing the thresholds below.

**Tracking and Merging Tokens** 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. $\mathbf{P}_i[t] \subset \{1, 2, \ldots, N_1\}$. For each of the edges between chosen redundant tokens $t \to t_B$, we compute merged token embeddings and the corresponding position sets as:

$$x_i[t_B] \leftarrow \frac{x_i[t] \cdot |\mathbf{P}_i[t]| + x_i[t_B] \cdot |\mathbf{P}_i[t_B]|}{|\mathbf{P}_i[t]| + |\mathbf{P}_i[t_B]|} \tag{1}$$

$$\mathbf{P}_i[t_B] \leftarrow \mathbf{P}_i[t_B] \cup \mathbf{P}_i[t] \tag{2}$$

$$\mathbf{P}_i[t] \leftarrow \varnothing \tag{3}$$

Intuitively, the representation of token $t_B$ is updated to the average of $x_i[t_B]$ and $x_i[t]$, weighted by their corresponding merged position set sizes, $\mathbb{P}_i[t_B]$ and $\mathbf{P}_i[t]$. The token $t$ is then dropped since it has been merged with $t_B$, thereby reducing the token count in the next layer.

**Finding Redundancy Thresholds** The layer-wise thresholds $\tau_i$ play a crucial role in determining how many tokens are merged. In order to determine the thresholds, we rely of 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***. The final output would then be expected to have an *average* of $N - \sum r_i$ tokens. 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 standard ViT without any merging, $N_i$ is constant across layers

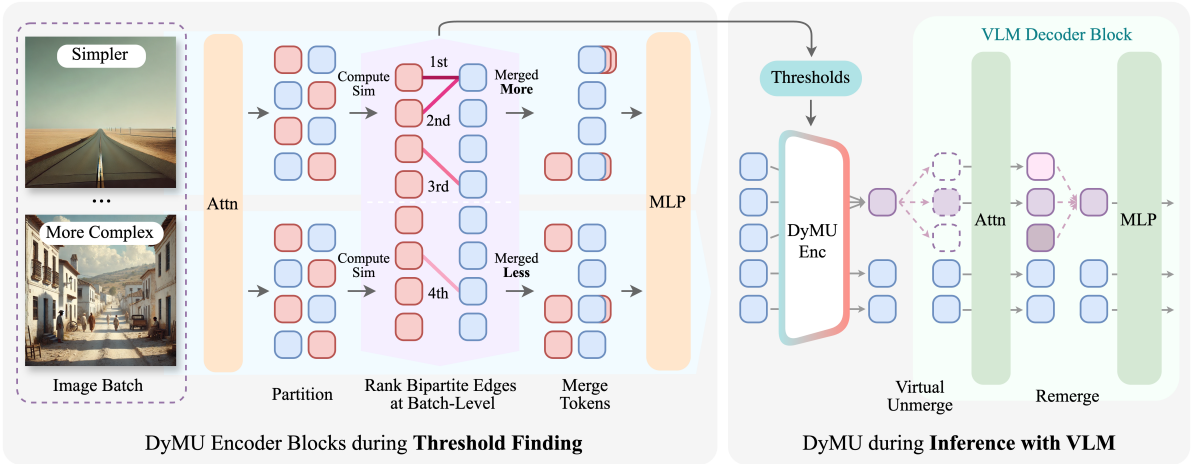


**Figure 2. Method Overview.** DyMU, is composed of two key ideas: Dynamic Token Merging (DToMe) and Virtual Token Unmerging (VTU). DToMe first determines per-layer thresholds (**left**) by feeding a large batch of images into the vision transformer and computing bipartite token similarities. We rank these edges across the *entire batch* and choose the top-$Br$ ($r$ = desired *average* number of tokens, batch size $B$). This leads to more edges from simpler images (with more redundancy) being chosen, while complex images remain less merged. During inference, DToMe merges tokens on a per-image basis using these pre-computed thresholds. We then apply VTU (**right**) in the self-attention layers of the pretrained VLM to efficiently expand the attention matrices to the standard token count—ensuring the model's original weights and outputs remain compatible—before re-merging the tokens for the next layer. The overall process is training-free and utilizes crucial image information by allocating the token budget more effectively for both simple and complex images.

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 $\tau_i$ as:

$$\tau_i = \max \left\{ \tau \mid \sum_{b=1}^{B} \sum_{t \in \mathbb{A}^{(b)}} \mathbb{I}\left(S^{(b)}[t] > \tau\right) = B * r_i \right\} \quad (4)$$

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 several 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.

**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 adopt the idea of size-weighted self-attention from ToME[3] where the attention is computed as:

$$\mathbf{A} = \texttt{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}} + \log \begin{bmatrix} |\mathbb{P}_i[1]| \\ \vdots \\ |\mathbb{P}_i[N_i]| \end{bmatrix}\right) \quad (5)$$

## 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 Dynamic Token Merging is applied to a visual encoder, this disrupts the optimized VLM and leads to a significant drop in performance (see Sec 4). In this section, we present an approach to circumvent this issue while still benefiting from processing fewer number of visual embeddings. Our proposed approach, Virtual Token Unmerging (VTU), can be easily applied to any mainstream LLM that uses a RoPE [36]-based transformer architecture.

Consider the general case of a sequence of $N$ embeddings $e \in \mathbb{R}^{N \times D}$ of which only $N_{\text{un}} << N$ 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 $e_{\text{un}}$ 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 $e_{\text{un}}$, 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

[†]AnyRes[19] leads to multiple fixed length embeddings

[‡]Note that due to the sparsity of $M$, the time complexity of multiplying $MD, DM, M^TD, DM^T$ are all $O(NK)$ if $D$ is a dense matrix with dimensions $N \times K$ or $K \times M$.

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 $N_{\text{un}} << N$.

**Virtual Unmerging for Self-Attention with RoPE**

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.

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 $V_{\text{un}}$ are the unique queries, keys and values satisfying the mapping $M$ defined above. The RoPE Self-Attention similarity matrix is computed as $A = \texttt{RoPE}(Q)\texttt{RoPE}(K)^T$.

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.* $Q[n] = Q_1[n] + \mathbf{i}Q_2[n]$. 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 [36] is defined as:

$$A[m,n] = \text{Re}\left( e^{i\boldsymbol{\theta}[\boldsymbol{m}]}Q[m] \ \overline{e^{i\boldsymbol{\theta}[\boldsymbol{n}]}K[n]} \right) \tag{6}$$

$$= \text{Re}\left( Q[m]\overline{K[n]} \ e^{i(\boldsymbol{\theta}[m]-\boldsymbol{\theta}[n])} \right) \tag{7}$$

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

$$\begin{aligned} A[m,n] = &(Q_1[m]K_1[n] + Q_2[m]K_2[n])\cos(\boldsymbol{\theta}[m] - \boldsymbol{\theta}[n]) \\ &+ (Q_1[m]K_2[n] - Q_2[m]K_1[n])\sin(\boldsymbol{\theta}[m] - \boldsymbol{\theta}[n]) \end{aligned} \tag{8}$$

We also have the trigonometric identities:

$$\begin{aligned} \cos(\theta[m]\text{–}\theta[n]) &= \cos(\theta[m])\cos(\theta[n]) + \sin(\theta[m])\sin(\theta[n]) \\ \sin(\theta[m]\text{–}\theta[n]) &= \sin(\theta[m])\cos(\theta[n]) - \cos(\theta[m])\sin(\theta[n]) \end{aligned} \tag{9}$$

Let $C = \texttt{diag}(\cos(\boldsymbol{\theta}))$, $S = \texttt{diag}(\sin(\boldsymbol{\theta}))$. Using Eq 8 & 9, the matrix form for self-attention similarities is:

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

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.

| **Methods** | $\mathbf{N_{un}/N}$ | **MFLOPs** |
|---|---|---|
| Full Attention | 576 / 576 | 1359.0 |
| VTU Attention-low | 94 / 576 | 72.4 |
| VTU Attention-mid | 209 / 576 | 357.8 |
| VTU Attention-high | 393 / 576 | 1265.0 |

**Table 1.** Comparison of million floating-point operations per second (MFLOPs) between original attention and Virtual Token Unmerging (VTU) attention. $N$ 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.

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:

$$\begin{aligned} A = &CMQ_{\text{un}}K_{\text{un}}^\top M^\top C + SMQ_{\text{un}}K_{\text{un}}^\top M^\top S \\ &+ SM(Q_{\text{un}} \times K_{\text{un}}^\top)M^\top C - CM(Q_{\text{un}} \times K_{\text{un}}^\top)M^\top S \end{aligned} \tag{10}$$

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 10 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 reintroduce the redundancy by averaging the embeddings in the positions that were originally equal. We denote this *remerged* output by $f'(e_{\text{un}}, M)$ which can be written as:

$$\begin{aligned} f'(e_{\text{un}}, M) &= (M^\top M)^{-1}M^T f(e) \\ &= (M^\top M)^{-1}M^T\texttt{smax}(\frac{A}{\sqrt{D}})V \end{aligned} \tag{11}$$

While the above averaging operation breaks the exactness of the future operations, we observe empirically (see Section 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 $O(N_{un}^2D + NN_{un}D)$ (due to the $D/2$ components). Followed by the softmax and sparse matrix multiplications in Eq 11 which incur a cost of $O(N^2 + N_{\text{un}}^2D)$. 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 sequence length of $N$ would be an $O(N^2D)$ operation. Therefore, in

| 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 | 55.8 |
| *Fixed Length Compression & Training-Required* | | | | | | | | | | | | |
| MQT-LLaVA [14] | 256 | No | 61.6 | 64.3 | 1435, - | 84.4 | 67.6 | - | - | 29.8 | 64.6 | |
| Prumerge [34] | 32 | No | - | 60.9 | 1350, - | 76.3 | 68.5 | - | 56.0 | - | - | |
| Prumerge++ [34] | 144 | No | - | 64.9 | 1462, - | 84.0 | 68.3 | - | 57.1 | - | - | |
| LLaMA-VID [22] | 2 | No | 55.5 | - | - , - | 83.1 | 68.8 | - | 49.0 | - | - | |
| VoCo-LLaMA [20] | 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 [45] | 1 | No | 60.9 | 65.6 | 1466, - | 84.4 | 70.4 | - | 57.0 | 36.6 | 68.9 | |
| *Fixed Length Compression & Training-Free* | | | | | | | | | | | | |
| Prumerge-no-ft [34] | 32 | No | - | - | 1250, - | 76.2 | 68.0 | - | 54.0 | - | - | |
| FastV [38] | 128 | No | 49.6 | 56.1 | - , 1490 | 53.4 | 64.4 | - | 50.6 | 26.3 | - | |
| PDrop [38] | 128 | No | 56.6 | 61.4 | - , 1713 | 82.3 | 69.2 | - | 55.9 | 30.8 | - | |
| SparseVLM [46] | 128 | No | 57.2 | 62.3 | - , 1721 | 85.0 | 67.8 | - | 55.8 | 29.0 | - | |
| ToMe [3] | 94 | **Yes** | 57.3 | 59.7 | 1357, 1673 | 86.8 | 68.9 | 60.5 | 53.2 | 25.6 | 61.0 | 52.6 |
| ToMe [3] | 209 | **Yes** | 59.2 | 62.4 | 1418, 1734 | 87.4 | 69.2 | 63.5 | 54.9 | 30.9 | 62.9 | 54.6 |
| ToMe [3] | 393 | **Yes** | 59.5 | 64.1 | 1454, 1769 | 86.7 | 68.4 | 65.1 | 55.8 | 30.8 | 66.0 | 55.2 |
| ***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 | 54.5 |
| **DyMU-mid** | $195_{\pm 47}$ | **Yes** | 61.7 | 62.8 | 1483, 1862 | 86.6 | 69.2 | 65.9 | 55.1 | 30.9 | 65.1 | 55.3 |
| **DyMU-high** | $394_{\pm 57}$ | **Yes** | 61.9 | 64.3 | 1498, 1846 | 86.8 | 69.9 | 66.1 | 58.0 | 31.5 | 64.5 | 56.0 |

**Table 2.** Comparison with state-of-the-art methods for improving efficiency on LLaVA 1.5 [25]. DyMU-low achieves 97.7% 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, 46], while also enabling variable-length outputs.

| 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 | 55.2 |
| ToMe [3] | 114 | 59.3 | 61.4 | 1380, 1717 | 85.1 | 66.9 | 61.8 | 52.1 | 26.1 | 57.9 | 52.4 |
| **DyMU-SigLIP-low** | $90_{\pm 26}$ | 61.3 | 62.5 | 1398, 1695 | 84.9 | 66.7 | 64.4 | 51.8 | 26.7 | 58.6 | 53.1 |
| **DyMU-SigLIP-mid** | $176_{\pm 43}$ | 62.2 | 63.9 | 1442, 1744 | 85.0 | 67.4 | 65.2 | 54.5 | 26.7 | 59.5 | 53.9 |
| **DyMU-SigLIP-high** | $318_{\pm 57}$ | 62.4 | 65.0 | 1449, 1765 | 86.0 | 67.6 | 66.0 | 56.8 | 29.4 | 58.3 | 54.7 |

**Table 3.** DyMU demonstrates similar efficacy on a different visual encoder, SigLIP [44]. We obtain the baseline by following the same training recipe as LLaVA-1.5[25]. DyMU-SigLIP-low achieves 96.2% of the baseline performance while using ∼15% visual tokens.

theory, efficiency improves at least linearly with the number of redundant tokens in terms of FLOPs. Table 1 shows the FLOPs comparison for the attention block. In practice, we find that the wall-clock time difference is marginal due to PyTorch's highly optimized attention and dense matrix-multiplication implementations.

## 4. Experiments

In this section, we present all the details of our implementation of the proposed method. We also present a comprehensive analysis demonstrating the practical benefits and efficacy of utilizing DyMU with various VLMs, visual encoders and LLM architectures.

### 4.1. 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 [31], TextVQA [35] and GQA [16]. We also ablate the choice of image datasets in §4.2. 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 [33][§] and SigLIP [1, 44][¶‖], 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 VLM backbones including fixed-resolution models, e.g., LLaVA 1.5 [25] and any-resolution models, e.g., LLaVA-OneVision [19].

### 4.2. Quantitative Evaluation

**Comparing Visual Token Merging Methods for VLMs** In order to evaluate efficacy of our approach, we compare against several existing methods that focus on reducing the number of tokens for VLMs. To the best of our knowl-

[§]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

| Methods | % Visual Tokens | Image Benchmarks | | | | Video Benchmarks | |
|---|---|---|---|---|---|---|---|
| | | MMB | MME | SEED | MathVista | 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 |

**Table 4.** DyMU shows consistent effectiveness on an AnyRes VLM backbone, LLaVA-OneVision [19]. We additionally show performance on two comprehensive video understanding benchmarks, where DyMU-ov-low achieves ∼96.5% of the baseline's performance with only ∼14% tokens.

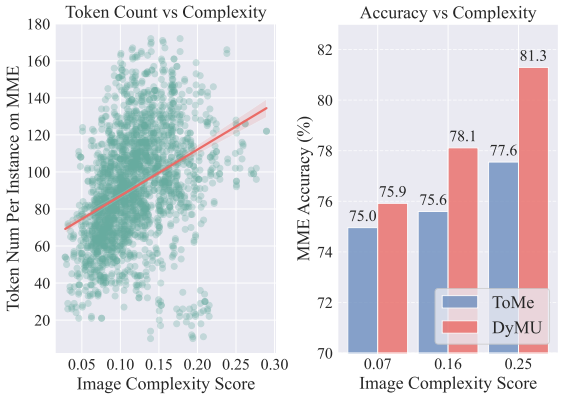


**Figure 3. Image Complexity vs Token Count and Accuracy** The scatter plot (**left**) demonstrates a strong correlation between DyMU's token count and image complexity score—more complex images naturally receive more tokens. On the **right**, MME accuracy at varying complexity levels is compared between ToMe (fixed-length) and DyMU (dynamic-length), highlighting the benefit of assigning additional tokens to complex images.

edge, our proposed approach is the first to 1) enable varied number of visual tokens and 2) not require further finetuning of the VLM. Nevertheless, we compare to methods that are designed to reduce the number of tokens by a fixed length. 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 [21], ScienceQA [28], SEED-IMG [18], TextVQA [35], MMVet [43], LLaVA-Bench [25]. DyMU achieves average performances of 97.7%, 99.1%, and 100.4%, 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 token number, the largest drop happens in TextVQA, which fits our expectation as understanding visual text is highly sensitive to the spatial location of visual tokens, on which the token merging tend to break.

**Compatibility with Different LLMs and Visual Encoders** DyMU can be seamlessly integrated into multiple variants of VLMs featuring different LLMs, visual

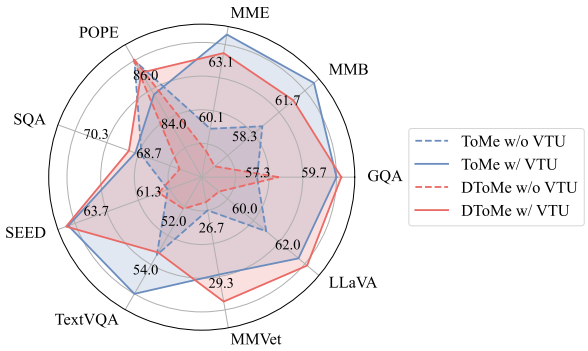


**Figure 4. Importance of Virtual Token Unmerging (VTU).** We ablate the performance of LLaVA 1.5 with two token reduction methods applied to the visual encoder—ToMe (fixed-length) and DToMe (variable-length). We observe that applying VTU significantly improves performance on 8 out of 9 benchmarks, demonstrating robustness to varied token reduction methods.

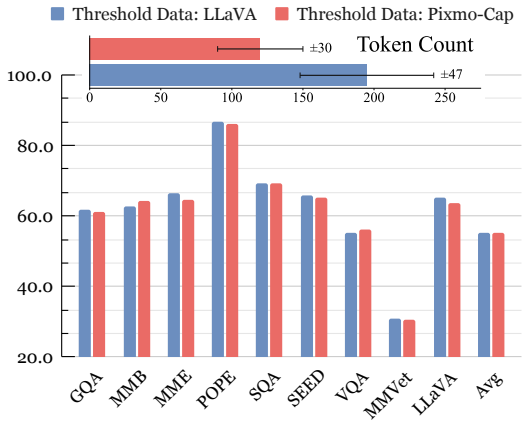


**Figure 5. Comparing thresholds using LLaVA Instruct Data vs Pixmo-Cap**. Although both methods use the same per-layer merging hyperparameter ($r_i$ ), the Pixmo-based thresholds lead to fewer tokens (**top**)—likely due to domain differences. However, performance across a range of benchmarks shows minimal drop (**bottom**), indicating the robustness of our threshold estimation.

encoders, and pretraining strategies. In Tables 2 and 3, we demonstrate that DyMU effectively maintains baseline performance when applied both CLIP [33] to SigLIP [44] representations within the LLaVA 1.5 framework, using a Vicuna-7B [7] LLM.

Furthermore, in Table 4 we evaluate DyMU on LLaVA-OneVision [19], a recent Any-Resolution (AnyRes) model with SigLIP-so400M [1] as visual encoder and Qwen2 [39] as LLM backbone. AnyRes enables processing images of arbitrary resolutions by segmenting them into smaller regions and encoding each individually. Our results show that DyMU remains compatible with this complex operation, preserving performance while dynamically reducing token counts. Additionally, we extend our evaluation to video benchmarks using LLaVA-OneVision. By applying DyMU to the visual encoder, we achieve a variable reduction in feature representations per frame while maintaining

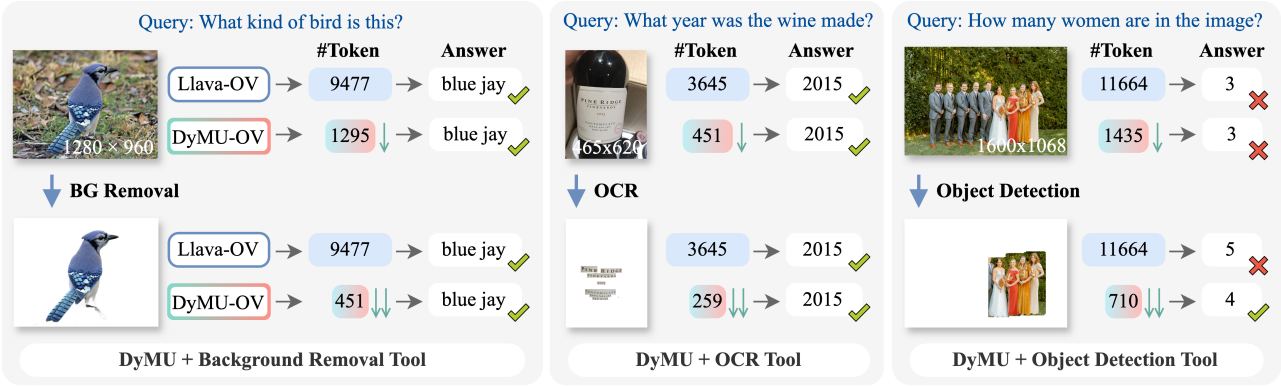


**Figure 6. Controllable Visual Token Length.** By dynamically allocating tokens based on image complexity, DyMU enables direct control over computational cost. In these examples, we combine DyMU with additional vision tools—background removal, OCR, or object detection—to focus only on the relevant regions. As a result, token count is substantially reduced without degrading performance, showcasing the flexibility of DyMU to adapt token usage according to the task's requirements.

strong performance across benchmarks.

**Image Complexity vs Number of Tokens** In Figure 3 (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 Appendix A.

**Fixed vs Dynamic Token Reduction** In Figure 3 (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 4, 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.

**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 5, we evaluate DyMU-LLaVA 1.5 with DToMe thresholds estimated on the Pixmo-Cap [8] image-captioning dataset. 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.

### 4.3. 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 7, 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.

**Controllable Visual Token Length** Dynamic Token Merging 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 direct control over the computational cost incurred per image. This flexibility allows flexible combination of visual reasoning tools with DyMU to further boost efficiency while maintaining performance. For instance, in Figure 6, we show example applications of combining DyMU with

additional tools, i.e., background removal [4], OCR [9], and object detection [30] models, to extract focused regions and further reduce token count. Unlike existing VLMs, which impose a fixed token budget 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. Conclusions and Future Work

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 plugged into all mainstream VLMs that comprise ViT-based visual encoders and RoPE-based LLM backbones. Future work includes improving DyMU's ability to preserve VLM performance on spatially sensitive tasks such as TextVQA [35] and spatial reasoning. Additionally, exploring the extension of DyMU to reduce temporal redundancy in videos is another promising direction.

## 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. 6, 7

[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. 2

[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. 2, 3, 4, 6, 7, 8, 12, 14

[4] Bria AI. RMBG-1.4: Background Removal Model. https://huggingface.co/briaai/RMBG-1.4, 2024. 9

[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. 2, 6

[6] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO Captions: Data Collection and Evaluation Server. CoRR, abs/1504.00325, 2015. 1

[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, 2023. 7

[8] 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. 8

[9] Felix Dittrich. Onnxtr: Optical character recognition made seamless & accessible to anyone, powered by onnx. https://github.com/felixdittrich92/OnnxTR, 2024. 9

[10] W. N. Francis and H. Kucera. Brown corpus manual. Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979. 2

[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. 2, 7

[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. 1

[13] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019. 1

[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. 2, 6

[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. 2

[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. 6, 7

[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. 6

[18] 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. 7

[19] 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. 2, 4, 6, 7

[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. 2, 6

[21] 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. 7

[22] 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. 2, 6

[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. 2

[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. 6

[25] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning. arXiv prepring arXiv:2310.03744, 2023. 6, 7

[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. 2

[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. 7

[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. 7, 12

[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. 2

[30] 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. 9

[31] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In ICDAR, 2019. 1, 6

[32] 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. 3

[33] 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. 2, 3, 6, 7

[34] 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. 2, 6

[35] 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. 6, 7, 9

[36] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. Neurocomputing, 568:127063, 2024. 2, 4, 5

[37] Xinjian Wu, Fanhu Zeng, Xiudong Wang, and Xinghao Chen. Ppt: Token pruning and pooling for efficient vision transformers. arXiv preprint arXiv:2310.01812, 2023. 2

[38] 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. 2, 6

[39] 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. 7

[40] 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. 2

[41] 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. 2

[42] 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. 2

[43] 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. 7

[44] 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. 3, 6, 7

[45] 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. 2, 6

[46] 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. 2, 6

## A. Visualization of Variable Token Length

In Figure 7, 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.

## B. 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.

In Table 5, 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.

## C. Full Results for Figure 4

We present the complete results of the ablation experiments on the effect of our proposed Virtual Token Unmerging, as shown in Figure 4. The results are provided in Table 6.

## D. Full Results for Figure 5

We present the complete results of the ablation experiments on threshold-finding datasets, as shown in Figure 5. The results are provided in Table 7.


**Figure 7. 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.

| 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 | 55.3 |
| Linear | $163_{\pm 43}$ | 61.3 | 62.3 | 1437, 1767 | 86.2 | 69.4 | 65.3 | 52.1 | 28.8 | 58.6 | 53.8 |
| Reverse Linear | $213_{\pm 49}$ | 61.8 | 63.8 | 1491, 1863 | 86.7 | 69.3 | 66.0 | 57.5 | 31.8 | 65.3 | 55.9 |

**Table 5.** 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.

| 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 | 52.6 |
| **+ VTU** | 94 | 60.6 | 63.7 | 1464, 1815 | 85.4 | 69.1 | 64.9 | 54.8 | 28.7 | 62.5 | 54.5 |
| DyMU-low | $89_{\pm 27}$ | 60.8 | 62.1 | 1438, 1787 | 86.3 | 69.3 | 65.0 | 53.1 | 30.0 | 62.9 | 54.5 |
| **w/o VTU** | $89_{\pm 27}$ | 58.2 | 56.0 | 1346, 1639 | 86.9 | 67.7 | 60.9 | 51.3 | 25.2 | 58.8 | 51.7 |

**Table 6.** Impact of Virtual Token Unmerging. Full results for Figure 4.

| 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 | 55.3 |
| **DyMU-mid** | **Pixmo** | $120_{\pm 30}$ | 61.1 | 64.4 | 1474, 1808 | 86.0 | 69.4 | 65.3 | 56.2 | 30.5 | 63.7 | 55.2 |

**Table 7.** Impact of dataset for threshold finding. Full results for Figure 5.