ALGM: Adaptive Local-then-Global Token Merging for Efficient Semantic Segmentation with Plain Vision Transformers

Narges Norouzi, Svetlana Orlova, Daan de Geus, Gijs Dubbelman
Eindhoven University of Technology
{n.norouzi, s.orlova, d.c.d.geus, g.dubbelman}@tue.nl

Abstract

This work presents Adaptive Local-then-Global Merging (ALGM), a token reduction method for semantic segmentation networks that use plain Vision Transformers. ALGM merges tokens in two stages: (1) In the first network layer, it merges similar tokens within a small local window and (2) halfway through the network, it merges similar tokens across the entire image. This is motivated by an analysis in which we found that, in those situations, tokens with a high cosine similarity can likely be merged without a drop in segmentation quality. With extensive experiments across multiple datasets and network configurations, we show that ALGM not only significantly improves the throughput by up to 100%, but can also enhance the mean IoU by up to +1.1, thereby achieving a better trade-off between segmentation quality and efficiency than existing methods. Moreover, our approach is adaptive during inference, meaning that the same model can be used for optimal efficiency or accuracy, depending on the application. Code is available at https://tue-mps.github.io/ALGM.

1. Introduction

Vision Transformers (ViTs) have shown to be very effective for image segmentation tasks [9, 10, 19, 37, 38, 46, 47, 51, 52]. However, the computational complexity of the multi-head self-attention operation scales quadratically with the number of input pixels. This harms the computational efficiency, especially on the high-resolution images that are typically used for image segmentation. To alleviate this burden and improve the efficiency, several works have proposed methods to reduce the number of tokens that the ViT has to process. Most token reduction methods have been introduced for image classification [2, 3, 7, 8, 15–18, 20, 22, 24, 28, 30, 33, 36, 40, 42–44, 48–50, 54], but there is also an increasing amount of work that focuses on tasks like semantic segmentation [12, 21, 23, 25, 26, 29, 39]. In this work, we also focus on semantic segmentation, and aim to design a token reduction method that achieves a better balance between efficiency and segmentation quality than existing works.

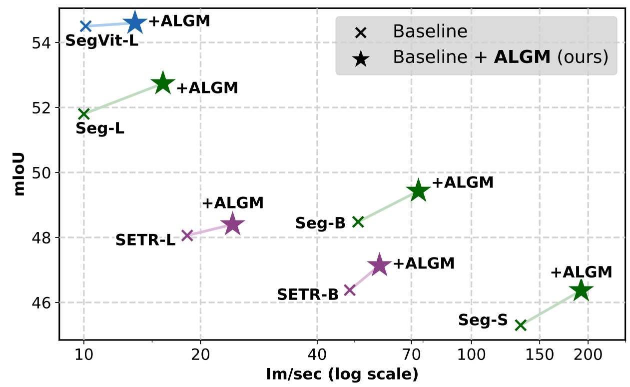


Figure 1. **Efficiency and segmentation quality for ALGM,** applied to Segmenter [38], SegViT [51], and SETR [52] on ADE20K. On average, ALGM improves the throughput of these baselines by 39%, while improving the mIoU by +0.7.

This objective is motivated by the limitations of existing works. First, *token pruning* methods [15, 30, 33, 48], which are popular for image classification, discard uninformative tokens. They are not directly applicable to semantic segmentation, as each token requires a prediction. To overcome this, *token pausing* or *halting* methods [12, 26, 39] retain discarded tokens and aggregate them with the other tokens at the end of the ViT. However, these methods observe a drop in segmentation quality, possibly because useful information contained in the halted tokens is not available in later network layers. Alternatively, *token sharing* and *merging* methods avoid discarding tokens, and represent multiple image patches or tokens with a smaller set of tokens [21, 23, 29]. This approach allows these methods to maintain the segmentation quality, but requires them to introduce additional computational overhead to identify tokens for sharing or merging, and they apply token reduction only once, limiting the efficiency gain. Furthermore, token merging methods that have been designed for image classification yield a notable decline in segmentation quality when applied to semantic segmentation [2, 23, 24].

Based on these existing works, we make two observations: **(a)** CTS [29] demonstrates that local token sharing in early network stages enhances efficiency without compromising segmentation quality, but it inefficiently requires a pre-processing network. Therefore, our first objective is to *merge redundant tokens early in the network without requiring pre-processing, and still maintain the segmentation quality*. **(b)** Token merging approaches like ToMe [2] show that gradually merging redundant tokens across the entire image (*i.e.*, globally) can greatly boost the efficiency, but at the cost of segmentation quality. Thus, our second objective is to also *apply global token merging to further improve efficiency, but without harming the segmentation quality.*

To achieve these objectives, we need to find an efficient method to identify tokens that can be merged without causing a drop in segmentation quality. Inspired by existing token merging methods [2, 28, 43], in Sec. 3.2, we explore if the cosine similarity is a suitable measure to identify mergeable tokens. Concretely, we compare the cosine similarities between tokens representing the same category – *i.e.*, *intra-class tokens*, which are potentially redundant and can be merged – and tokens representing different categories – *i.e.*, *inter-class tokens*, which should not be merged. We find that **(a)** already in the 1$^{\text{st}}$ network layer, the similarities between intra-class tokens in small local windows are much higher than for inter-class tokens, and **(b)** comparing tokens globally, intra-class token similarities become increasingly higher than inter-class similarities in later layers.

Based on these new findings, we present our Adaptive Local-then-Global Merging (ALGM) module that integrates two token merging phases. In the first network layer, ALGM adopts a local merging strategy. This is followed by a global merging mechanism in an intermediate layer, to also reduce global token redundancies. Moreover, rather than using a predetermined number of merged tokens, our approach dynamically decides the number of merged tokens based on the semantic complexity of the image content. Finally, we restore the original token resolution to make segmentation predictions. For details, see Sec. 3.3.

ALGM offers multiple advantages. **(a)** Unlike methods that use token pausing, redundant tokens remain active in the network, and continue to contribute in subsequent network layers via their merged representation. **(b)** ALGM avoids the need for preprocessing layers and the significant overhead associated with token sharing or merging methods. **(c)** Global merging is only applied when token similarity is sufficiently reflective of category similarity, reducing the chance of merging tokens that should remain separate. **(d)** Being a parameter-free approach, the ALGM module is naturally compatible with all plain ViT backbones, as well as any segmentation decoder, with or without re-training.

Through experiments outlined in Sec. 4, we demonstrate that ALGM consistently enhances the throughput by considerable margins when applied to a wide range of different segmentation methods (see Fig. 1). Moreover, we observe that, on top of this improved efficiency, ALGM also enhances the segmentation quality. From an investigation into the cause of this improvement, we find that it can be attributed to two factors: a better balance between frequent and infrequent categories in the self-attention operation, and the denoising of tokens. For more results, we refer to Sec. 5.

We summarize our main contributions as follows:

- A generally applicable token merging framework that integrates local and global merging, enhancing both the efficiency and segmentation quality of ViT-based semantic segmentation networks.
- An analysis of similarities between intra- and inter-class tokens, within local windows and across network layers.
- An exploration of the cause of the segmentation quality improvement obtained by ALGM.

2. Related work

Since the introduction of the Vision Transformer (ViT), a substantial amount of work has been dedicated to improving the efficiency of these ViTs. In this work, we focus on token reduction, which aims to decrease the number of tokens that are processed by the ViT, to improve efficiency.

Token reduction in general. The majority of token reduction methods have been introduced for ViTs that conduct image classification. Some methods use a token pruning strategy, where uninformative tokens are identified and simply discarded [15, 20, 24, 30, 33, 40, 48]. Uninformative tokens are identified by making intermediate predictions with auxiliary heads [20, 30, 33], or obtaining importance scores from attention weights [15, 24, 40]. Pruned tokens can be discarded completely [15, 30, 33, 48] or fused into one token to preserve information flow [20, 24]. While token pruning can notably enhance the throughput of transformers, discarding tokens is not possible for semantic segmentation as each token requires a prediction, and fused tokens representing multiple regions or categories cannot be trivially reconstructed to make a semantic segmentation prediction. Alternatively, token merging methods combine groups of tokens into a smaller set of representative tokens. Some works introduce a learned layer to map the original token set to a smaller one [6, 34, 36, 54], while most methods merge tokens if they have a high similarity score [2, 3, 5, 8, 17, 28, 43]. Other methods combine different merging, pruning or fusing approaches [3, 5, 8, 28, 44].

Token reduction for semantic segmentation. Some token merging methods for image classification can also be applied to semantic segmentation [2, 6, 17, 24, 28] by reconstructing merged tokens to their original positions to make a prediction. However, while these approaches improve efficiency, they consistently cause a drop in segmentation quality, as also shown by Liang *et al*. [23].

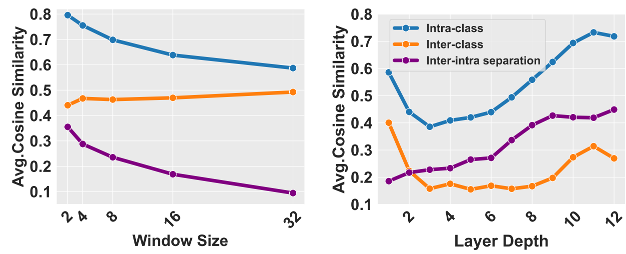
Other token reduction methods have been proposed specifically for the semantic segmentation task. Token pausing or halting approaches [12, 26, 39] identify tokens that produce high-confidence predictions in early network layers, and do not process them any further. Instead of discarding tokens like pruning methods, they retain tokens and reconstruct them later to make a final prediction. However, in subsequent layers, these 'paused' tokens do not participate in the self-attention operation anymore, meaning that potentially useful information is no longer available, which negatively affects the segmentation quality. Alternatively, ELViT [23] and AiluRus [21] introduce a non-parametric token clustering layer that merges redundant neighboring tokens in one network layer. These methods are able to reduce tokens while maintaining the segmentation quality, but their efficiency gain is limited. It is likely that this is because their clustering layer introduces computational overhead, and because they reduce tokens only once, not using the token redundancies potentially present in other layers. CTS [29] uses a CNN-based policy network to identify image patches that can share a token before the first transformer layer. This approach can also reduce tokens while maintaining segmentation quality, but its policy network introduces computational overhead, and it only merges tokens in local windows, ignoring global redundancies.

Inspired by the advantages and limitations of existing work, this work proposes a parameter-free token merging method for semantic segmentation that applies both local and global merging based on cosine similarities between tokens. Importantly, with minimal computational overhead, we identify where cosine similarities can be used to select tokens for merging while maintaining or even improving the segmentation quality.

3. Method

3.1. Preliminaries

A ViT-based segmentation architecture typically consists of two subnetworks: **(a)** An encoder $\mathcal{E} : I \xrightarrow{\mathcal{L}_1,\dots,\mathcal{L}_L} T_L$, which uses L distinct transformer layers to map the input image I to T_L, the set of tokens representing the image content at the final layer $\mathcal{L}_L$. The encoder $\mathcal{E}$ first splits the input image $I \in \mathbb{R}^{3\times H\times W}$ into $N = \frac{H\times W}{p^2}$ non-overlapping patches, determined by a patch size p. Following patch embedding and positional encoding steps, an initial set of token embeddings $T_0 = \{t_1, \dots, t_N\}$ is obtained. Each token t_i belongs to $\mathbb{R}^d$, with d denoting the feature dimension. These token embeddings are subsequently processed by transformer layers $\mathcal{L} = \{\mathcal{L}_1, ..., \mathcal{L}_L\}$, resulting in the output T_L. Each layer $\mathcal{L}_l$ in $\mathcal{L}$ integrates a multi-head self-attention (MHSA) block followed by a multi-layer perceptron (MLP) block which output $T'_l = \text{MHSA}(T_{l-1}) + T_{l-1}$ and $T_l = \text{MLP}(T'_l) + T'_l$ respectively. **(b)** A decoder $\mathcal{D} : T_L \rightarrow P$, which utilizes transformer or convolutional layers to process T_L and generate the per-pixel segmentation prediction P, where $P \in \mathbb{R}^{C\times H\times W}$ and C represents the number of classes. Our primary objective is to reduce the number of tokens in T, the total set of tokens, by identifying those tokens that can be merged without adversely affecting the segmentation quality of P.



(a) Local token similarities (b) Global token similarities

Figure 2. **Comparison of cosine similarity between intra-class and inter-class tokens.** On ADE20K training set using Segmenter + ViT-S [13, 38]. (a) Local similarities across 5 window sizes in the first layer. (b) Layer-wise analysis of global similarities.

3.2. Token similarity analysis

As highlighted in Sec. 1, considering the advantages and limitations of existing methods, our goal is to find a method to (a) apply early local token merging without requiring a pre-processing network and (b) also apply global token merging to further improve efficiency, without harming the segmentation quality. To achieve this, we need to find a method that can efficiently identify tokens suitable for merging while preserving the segmentation quality. CTS [29] is based on the hypothesis that tokens that represent the same semantic class can be merged without compromising segmentation quality, since they carry redundant information. On the other hand, several token merging methods for image classification [2, 28, 43] merge tokens with a high cosine similarity. They get promising efficiency gains, but sometimes at the cost of accuracy. This motivates us to examine if and when cosine similarity can be an effective metric to identify tokens that represent the same category and are thus suitable for both local and global merging.

To analyze this, we extract and compare the similarities between tokens from Segmenter with ViT-S [13, 38] trained on the ADE20K [53] training set. (1) We first analyze the local similarities within $k\times k$ windows in the first transformer layer. We calculate the cosine similarity between tokens representing different categories (*i.e.*, *inter-class tokens*), which should not be merged, and tokens representing the same category (*i.e.*, *intra-class tokens*), which can be merged. As illustrated in Fig. 2a, the smaller the window size k, the more accurately cosine similarity reflects that tokens depict the same category. Consequently, tokens

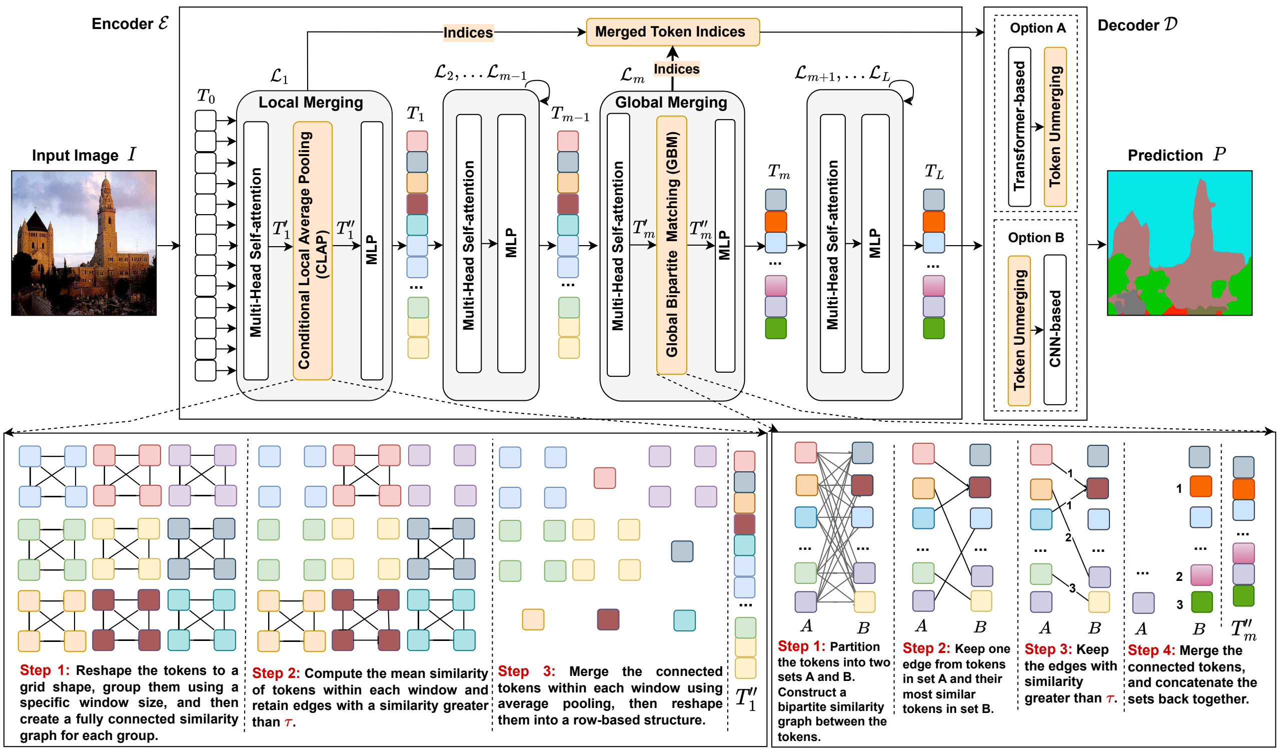


Figure 3. **ALGM** comprises two primary modules: **(1) Conditional Local Average Pooling (CLAP)** for local merging and **(2) Global Bipartite Matching (GBM)** for global merging. The top section illustrates the placement of these modules in the first and middle layers, while the bottom provides a detailed visualization of the individual modules.

with a high cosine similarity within small local windows in the first layer can likely be merged without a drop in segmentation quality. (2) We analyze the global similarities by calculating the cosine similarities for inter-class and intra-class tokens across the entire image for all transformer layers. As seen in Fig. 2b, the global similarities in early layers do not accurately represent category correspondence, so they should not be employed to identify tokens for merging. However, deeper in the network, cosine similarity becomes a better measure to identify tokens that can be merged globally without affecting segmentation quality.

3.3. Adaptive Local-then-Global Merging

From the analysis we know that (a) local token similarities in early layers and (b) global token similarities in intermediate layers are likely good indicators of the mergeability of tokens. To exploit this, we propose the Adaptive Local-then-Global Merging (ALGM) approach. As demonstrated in Fig. 3, the process begins with local merging in the first layer using the Conditional Local Average Pooling (CLAP) module. In an intermediate layer, we adopt the Global Bipartite Merging (GBM) module which is based on the BSM [2] algorithm for global merging. The procedure concludes with a token unmerging module to restore the original token resolution.

Local token merging. Given the insights from Sec. 3.2, we aim to merge tokens in the first layer if they have a high similarity with neighboring tokens in a small window. To implement this, we introduce the CLAP module, which is positioned in layer $\mathcal{L}_1$, between the MHSA and MLP blocks, as illustrated in Fig. 3. The CLAP module follows these steps: (1) It receives the token embeddings $T'_1 \in \mathbb{R}^{N \times d}$ from layer $\mathcal{L}_1$ and reshapes them into a grid $T'_{G1} \in \mathbb{R}^{\frac{H}{p} \times \frac{W}{p} \times d}$. It then defines a window of size $k \times k$ and groups the tokens within each window into separate sets $W = \{w_1, \ldots, w_s\}$, where $s = \frac{N}{k^2}$. Each $w \in \mathbb{R}^{k \times k \times d}$ is a set of token embeddings, represented as $w = \{t_1, t_2, \ldots, t_{k^2}\}$. (2) Subsequently, for each w in W, it computes the cosine similarity between all pairs of tokens t_i in w, and calculates the mean of these similarities to get μ_w. Then, as we hypothesize that the similarity between tokens represents the mergeability, CLAP merges only the tokens in windows w for which $\mu_w > \tau$, where τ is an automatically determined similarity threshold, which is elaborated later in this section. (3) Finally, the tokens inside selected windows w are merged into a single token by taking the average of these tokens. The indices of these tokens are also stored for later unmerging. Once completed, merged and non-merged tokens are concatenated to produce the output token embeddings $T''_1 = \{t_1, \ldots, t_{N'}\}$ where $N' \leq N$.

Global token merging. After local merging, the tokens are processed through standard transformer layers up to layer $\mathcal{L}_m$, where the GBM module is applied. Similar to the steps of the BSM [2] algorithm: (1) In the first step, the tokens in T'_m are split into two sets: $A = \{t_1, t_3, \dots, t_{N'-1}\}$ and $B = \{t_2, t_4, \dots, t_{N'}\}$. A fully-connected bipartite graph is then constructed between the tokens within these sets based on their cosine similarity. (2) Then, GBM only retains the edges that represent the highest similarity from a token in set A to any token in set B. This means that for a given token A_i , the edge to token B_j is only retained if B_j is the most similar token to A_i when compared to all other tokens in set B. (3) Next, unlike the original BSM which employs a constant number of merged tokens, GBM uses a similarity threshold τ. Thus, edges are only retained if their similarity exceeds τ. (4) Finally, the tokens with remaining edges are merged by taking their average, and their indices are stored for future unmerging. The two sets are then concatenated, yielding the output embeddings $T''_m = \{t_1, \dots, t_{N''}\}$ where $N'' \leq N'$.

Token unmerging. Upon completing global merging, the embeddings T''_m are processed through the remaining $L - m$ transformer layers, resulting in the final token embeddings T_L. These embeddings, along with the indices of merged tokens retained during the merging phases, are provided as inputs to the decoder $\mathcal{D}$. In this phase, we deploy an *unmerging* module that duplicates the embeddings of the merged tokens at the indices of the tokens from which they were merged. For transformer-based decoders, which are designed to handle tokens, the unmerging module is applied after the decoder. Conversely, for CNN-based decoders, which require spatially organized features, token unmerging is executed prior to the decoder.

Adaptive token merging. As the complexity of images varies, reducing a constant number of tokens can lead to a suboptimal efficiency or segmentation quality. Merging too many tokens in complex images can lead to insufficient representation of their complexity. Conversely, simpler images can benefit from a more reduced token count, enhancing efficiency. To tackle this challenge, we introduce an adaptive method that automatically determines a similarity threshold. Before training, we take the base segmentation model to which we want to apply ALGM, and then run inference on the training set. We then extract the tokens after the MHSA block in each layer $\mathcal{L}_l$, calculate the cosine similarities between all token pairs, and compute the mean μ_{sim} and standard deviation σ_{sim} of these similarities across the entire training set. Given these statistics, we then set the threshold $\tau = \mu_{\text{sim}} + \sigma_{\text{sim}}$ to merge token pairs with above-average similarities. Using this threshold, the number of remaining tokens N' and N'' after the CLAP and GBM modules will vary per image. During training, to facilitate batching of images and tokens, we take the maximum number of remaining tokens N' and N'' per batch, and apply this to all images in the batch.

4. Experimental setup

Datasets. We conduct our main experiments on ADE20K [53], which is widely recognized as a challenging scene parsing dataset. Additionally, we show ALGM's general applicability on the Pascal Context [31], Cityscapes [11], and COCO-Stuff-10K [4] datasets.

Implementation details. ALGM can be applied to any segmentation model that uses plain ViTs. We apply it to three popular networks: Segmenter [38], SETR [52], and SegViT [51] using four standard ViT backbones [13]: ViT-T/S/B/L. For a fair comparison, we train all networks using the original hyperparameters and official implementations. By default, we integrate our CLAP and GBM modules at the 1^{st} and 5^{th} layers for ViT-T/S/B models, and at the 1^{st} and 7^{th} layers for ViT-L, using the automatically generated threshold τ for both merging phases. For more details, see the supplementary material.

Evaluation metrics. To assess the segmentation quality, we use the standard mean Intersection-over-Union (mIoU) metric, and for computational efficiency we evaluate the throughput in terms of images per second (im/sec) and the number of floating point operations (FLOPs). For the throughput, we calculate the average im/sec on the validation set with a batch size of 32 on an Nvidia A100 GPU, after a 50-iteration warmup. To calculate the number of FLOPs, we use fvcore [35] and compute the average number of operations over all images in the validation set. We report the number of GFLOPs, *i.e.*, FLOPs $\times 10^9$.

5. Experimental results

5.1. Main results

To evaluate the effectiveness of ALGM, we apply it to several ViT-based segmentation networks and compare it with existing state-of-the-art token reduction methods. For each existing method, we report the version with the highest efficiency while maintaining the segmentation quality as much as possible. For a more comprehensive analysis across various settings, additional results are provided in the supplementary material. For our ALGM, we report two versions: (1) the default **ALGM**, which uses the automatic threshold during both training and inference, and (2) **ALGM***, which is the same trained model, but during inference it uses the smallest threshold τ for which the mIoU is higher than the baseline. In other words, ALGM* is tuned for optimal efficiency while maintaining the segmentation quality. See Sec. 5.4 and the supplementary material for more details on the use of different merging thresholds.

ADE20K. Tab. 1 presents the results of ALGM and existing token reduction methods for different segmentation models and ViT backbones on the ADE20K dataset [53]. We find that, across all settings, ALGM is able to considerably improve the throughput and number of GFLOPs with

Method	mIoU (%)↑	Im/sec ↑	GFLOPs ↓
Seg-T [38]	38.1	287	12.8
+ CTS [29]	38.2	309	9.8
+ ToMe [2]	37.9	346	9.2
+ ALGM (ours)	**38.9**	388	8.4
+ ALGM* (ours)	38.4	**427**	**7.3**
Seg-S [38]	45.3	134	38.6
+ CTS [29]	45.1	174	27.2
+ ToMe [2]	45.1	170	28.2
+ ALGM (ours)	**46.4**	192	26.3
+ ALGM* (ours)	45.5	**235**	**20.9**
Seg-B [38]	48.5	51	130
+ CTS [29]	48.7	73	91
+ ToMe [2]	48.5	68	97
+ ELViT‡ [23]	48.2	73	92
+ ALGM (ours)	**49.4**	73	91
+ ALGM* (ours)	48.5	**87**	**76**
Seg-L [38]	51.8	10	672
+ CTS [29]	51.8	16	446
+ ToMe [2]	51.6	14	505
+ ELViT [23]	51.4	12	539
+ ELViT‡ [23]	51.9	12	539
+ AiluRus‡ [21]	52.2	-	479
+ ALGM (ours)	**52.7**	16	438
+ ALGM* (ours)	51.9	**20**	**370**
SegViT-L [51]	54.5	10	638
+ ALGM (ours)	**54.6**	14	476
+ ALGM* (ours)	54.5	**15**	**461**
SETR-L [52]	48.1	18	363
+ ALGM (ours)	**48.4**	24	277
+ ALGM* (ours)	48.1	**29**	**227**

Table 1. **Main results on ADE20K.** ALGM applied to Segmenter (Seg) [38], SegViT [51], and SETR [52] across 4 ViT backbones. ALGM* is the same trained model as ALGM, but uses the threshold τ during inference that achieves the best efficiency while maintaining the mIoU w.r.t. the baseline. ‡Indicates a training-free method, applied directly to the baseline model.

respect to the base segmentation networks, and also achieve a substantial mIoU increase. This shows that our token merging approach does not only improve the efficiency, but also boosts the segmentation quality. In Sec. 5.4, we provide a more detailed analysis into the cause of this mIoU improvement. The ALGM* variant, which is optimized for efficiency, is able to improve the throughput even further (up to +100% for Seg-L), while consistently achieving an mIoU that is the same or slightly higher than the base networks. Moreover, ALGM and ALGM* consistently outperform all existing token reduction works, in terms of both the mIoU and the efficiency metrics. This shows that our method can find a better balance between segmentation quality and efficiency, which is the objective of this work.

Other datasets. When applying ALGM to COCO-Stuff, Cityscapes and Pascal-Context [4, 11, 31] in Tab. 2, we observe very similar results as for ADE20K. For all datasets, our default ALGM significantly improves the throughput with respect to the base segmentation networks, while also obtaining a better segmentation quality. Again, ALGM* can improve the throughput even further, obtaining throughput improvements of +90% on Cityscapes for Seg-S and +72% on COCO-Stuff for Seg-L without any drop in segmentation quality.

On the COCO-Stuff and Pascal-Context datasets, ALGM and ALGM* also consistently outperform all existing methods. For Cityscapes, ALGM outperforms ToMe [2], AiluRus [21], and ELViT [23], but it does not achieve the same efficiency improvements as CTS [29]. We observe that the visual homogeneity of the Cityscapes images causes tokens to be similar in the first transformer layer even if they do not belong to the same category, requiring a higher merging threshold and limiting the efficiency improvement. Overall, taking into account all datasets, we can conclude that ALGM is a robust and generally applicable method that consistently improves the efficiency of ViT-based segmentation models while also enhancing their accuracy.

Comparison with other works. In Tab. 3, we compare ALGM to DToP [39] and DoViT [26]. However, these works report mIoU and GFLOPs results without token reduction that differ from the results we obtain from the official code of SETR [52] and Segmenter [38], while DToP and DoViT do not release their code. Therefore, we can only compare the relative performance differences obtained due to token reduction. In Tab. 3, we observe that DToP can maintain the mIoU while reducing the GFLOPs by 25%, whereas ALGM* maintains the mIoU and reduces the GFLOPs by 30%. Compared to DoViT, ALGM considerably improves both the segmentation quality and efficiency. For further comparisons, see the supplementary material.

5.2. Application to state-of-the-art model

To demonstrate the effectiveness of ALGM on a large-scale state-of-the-art network, we apply it to the EVA backbone [14] with a ViT-Adapter + Mask2Former decoder [9, 10]. Here, we calculate the average throughput on 4 Nvidia A6000 GPUs due to memory requirements. The results in Tab. 4 show that without training, we can improve the throughput by 26% while keeping the mIoU constant. When we also train the model, we achieve the same efficiency gains but now also further improve the mIoU with +0.2. These results show the general effectiveness and compatibility of ALGM with large-scale pre-trained networks.

5.3. Ablations

CLAP module window size. In Tab. 5a, we evaluate the effect of using different window sizes for our CLAP module. We find that smaller window sizes yield higher mIoU scores, whereas larger window sizes result in a better efficiency. This is as expected, as we have found in Fig. 2a

Method	mIoU (%)↑	Im/sec↑	GFLOPs↓
Seg-S [38]	42.3	132	39
+ CTS [29]	42.2	164	28
+ ALGM (ours)	**43.1**	182	27
+ ALGM* (ours)	42.3	**219**	**22**
Seg-B [38]	43.6	50	130
+ CTS [29]	43.7	71	89
+ ALGM (ours)	**44.4**	69	96
+ ALGM* (ours)	43.8	**84**	**79**
Seg-L [38]	46.8	18	401
+ CTS [29]	46.7	27	272
+ ALGM (ours)	**47.4**	25	287
+ ALGM* (ours)	46.8	**31**	**241**

(a) **COCO-Stuff [4].**

Method	mIoU (%)↑	Im/sec↑	GFLOPs↓
Seg-S [38]	76.5	41	116
+ CTS [29]	76.5	**81**	**56**
+ ALGM (ours)	**76.9**	65	76
+ ALGM* (ours)	76.5	78	64
Seg-L [38]	79.1	6.2	1005
+ CTS [29]	**79.5**	**13.2**	**523**
+ ToMe [2]	78.7	8.1	822
+ ELViT [23]	78.7	7.3	857
+ ELViT‡ [23]	78.8	7.3	857
+ AiluRus‡ [21]	78.8	-	711
+ ALGM (ours)	**79.5**	8.6	766
+ ALGM* (ours)	79.1	10.3	654

(b) **Cityscapes [11].**

Method	mIoU (%)↑	Im/sec↑	GFLOPs↓
Seg-S [38]	53.0	172	32.1
+ CTS [29]	52.9	220	22.4
+ ALGM (ours)	**53.2**	217	24.6
+ ALGM* (ours)	53.0	**263**	**20.2**
Seg-L [38]	57.7	21	343
+ CTS [29]	57.6	32	233
+ ToMe [2]	57.6	22	263
+ ELViT [23]	57.5	25	257
+ ELViT‡ [23]	57.9	25	257
+ ALGM (ours)	**58.0**	30	247
+ ALGM* (ours)	57.7	**34**	**222**

(c) **Pascal-Context [31].**

Table 2. **Main results on COCO-Stuff, Cityscapes and Pascal-Context.** ALGM applied to Segmenter (Seg) across 3 ViT backbones and 3 datasets. ALGM* is the same trained model as ALGM, but uses the threshold τ during inference that achieves the best efficiency while maintaining the mIoU w.r.t. the baseline. ‡Indicates a training-free method, applied directly to the baseline model.

Method	No token reduction		With token reduction	
	mIoU↑	GFLOPs↓	mIoU↑	GFLOPs↓
SETR-B + DToP [39]	47.0	108	47.0 (**+0.0**)	81 (**-25%**)
SETR-B + ALGM (ours)	46.4	108	47.1 (**+0.7**)	86 (**-20%**)
SETR-B + ALGM* (ours)	46.4	108	46.5 (**+0.1**)	75 (**-30%**)
Seg-S + DoViT [26]	46.2	26.6	45.8 (**-0.4**)	21.8 (**-18%**)
Seg-S + ALGM (ours)	45.3	38.6	46.4 (**+1.1**)	26.3 (**-32%**)
Seg-S + ALGM* (ours)	45.3	38.6	45.5 (**+0.2**)	20.9 (**-46%**)

Table 3. **ALGM vs. DToP and DoViT [26, 39].** Applied to SETR [52], and Segmenter (Seg) [38] on ADE20K [53].

Method	Decoder	mIoU (%)↑	Im/sec↑	GFLOPs↓
EVA [14]	Mask2Former	61.5	1.9	4080
EVA [14] + ALGM‡	Mask2Former	61.5	**2.4**	3538
EVA [14] + ALGM	Mask2Former	**61.7**	**2.4**	**3537**
BEiT-3 [9, 41]	Mask2Former	**62.0**	-	-
BEiTv2-L [9, 32]	Mask2Former	61.2	-	-
BEiT-L [1, 9]	Mask2Former	59.4	-	-
SwinV2-G [27]	UperNet [45]	59.3	-	-

Table 4. **Application to state-of-the-art model.** ALGM is applied to SOTA method EVA + ViT-Adapter + Mask2Former [9, 10, 14] and evaluated over the ADE20K validation set, with single-scale testing. ‡Directly applied to the backbone without fine-tuning.

that, the smaller the local window is, the more likely it is that a high token similarity indicates that tokens depict the same category and can thus share a token without harming the segmentation quality. On the other hand, using smaller window sizes means that fewer tokens are merged, limiting the overall efficiency improvement.

Impact of merging modules. To assess the impact of the individual CLAP and GBM merging modules, we evaluate various configurations in Tab. 5b. We find that only applying CLAP leads to modest improvements in both the mIoU and the throughput, showing that local merging is effective. If we use the GBM module in the first layer instead, we find that the throughput increases but at the cost of segmentation quality, confirming our findings in Sec. 3.2 and Fig. 2b that global token similarities in the first layer should not be used to identify tokens for merging. Conversely, placing the GBM module in layer 5 does yield an improved mIoU, albeit with a lower efficiency gain. Applying GBM in both layer 1 and layer 5 results in a significantly better efficiency, but the incorrectly merged tokens in layer 1 are then merged even further in layer 5, harming the segmentation quality. Finally, combining CLAP with GBM yields the best mIoU while achieving a significant efficiency gain, showing the power of applying both early local merging and later global merging. Fig. 5 visualizes the tokens that are merged by these modules. See the supplementary material for more examples.

GBM module position. Tab. 5c shows the effect of positioning the GBM module at various transformer layers. We find that, for ViT-S, layer 5 yields the best trade-off between mIoU and efficiency, which again shows that early layers should not be used for global merging, and applying it in later layers gives diminishing efficiency returns.

5.4. Detailed analyses

Cause of segmentation quality improvement. The main results in Tab. 1 and Tab. 2 have shown that ALGM not only enhances efficiency, but also improves the segmentation quality. This improvement is most significant for complex datasets, and we hypothesize that it has two causes: (1) **Balancing**: As tokens that depict the same category are merged, large and frequently-occurring categories are represented by fewer tokens, meaning that they play a less dominant role in the self-attention operation, causing a more balanced attention distribution with respect to rare classes. To assess if this is true, in Tab. 6, we evaluate a setting in which we do not reduce the number of tokens and therefore do not balance the attention process, but instead replicate the average token embedding across the tokens that would otherwise be merged. We find that this causes a drop in mIoU, indicating that attention balancing is indeed a factor in the mIoU improvement of ALGM. (2) **Denoising**: As tokens are merged, we take the average of their values. This denoises the tokens, which could facilitate the learning

Window size	mIoU (%)↑	Im/sec↑	GFLOPs↓
Baseline	*45.3*	*134*	*38.6*
2×2	46.4	192	26.3
4×4	44.6	212	22.5
8×8	37.2	264	13.3
2×1	46.5	181	28.3
2×4	45.4	208	23.6

(a) **CLAP module window size.**

Layer 1	Layer 5	mIoU (%)↑	Im/sec↑	GFLOPs↓
Baseline		*45.3*	*134*	*38.6*
CLAP	-	45.6	161	31.9
GBM	-	45.0	183	28.5
-	GBM	45.8	144	36.1
GBM	GBM	44.2	216	23.7
CLAP	GBM	46.4	192	26.3

(b) **Merging modules.**

Layer	mIoU (%)↑	Im/sec↑	GFLOPs↓
Baseline	*45.3*	*134*	*38.6*
2	45.1	201	25.1
4	45.8	195	25.9
5	46.4	192	26.2
6	46.3	185	26.8
8	46.5	176	27.6

(c) **GBM module position.**

Table 5. **Ablations for ALGM.** We apply ALGM to Segmenter [38] with ViT-S [13] and evaluate on the ADE20K validation set [53]. **CLAP**: Conditional Local Average Pooling (local merging); **GBM**: Global Bipartite Matching [2] (global merging).

Denoising	Balancing	Token selection	Token reorganization	mIoU (%)↑
✓	✓	Take average	Reduction	46.4
✓	✗	Take average	No reduction	45.6
✗	✓	Pick random token	Reduction	45.0
✗	✗	Pick random token	No reduction	44.4

Table 6. **Analyzing mIoU improvement.** ALGM is applied to Segmenter with ViT-S [13, 38] on ADE20K [53]. **Balancing** refers to attention balancing; **Denoising** refers to token denoising.

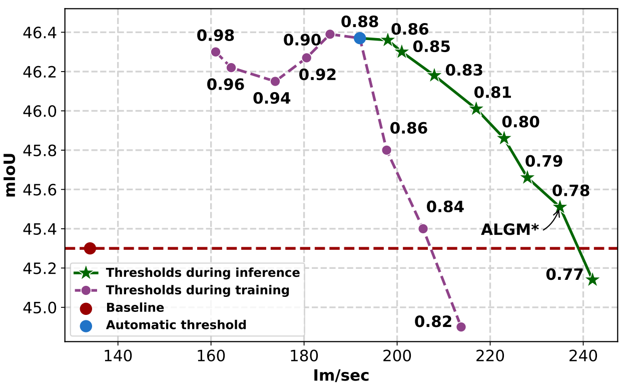
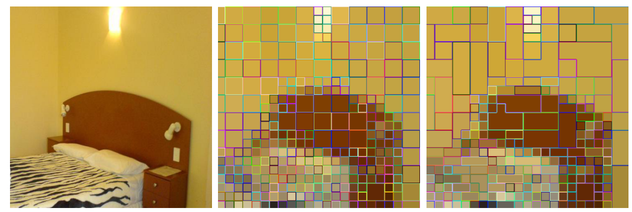


Figure 4. **Similarity thresholds for token merging.** ALGM applied to Segmenter [38] with ViT-S [13] on ADE20K [53].

process. To evaluate this aspect, we evaluate a configuration where we do not take the average of tokens but instead pick one random token from each set of mergeable tokens. Tab. 6 shows that doing so also results in a considerable drop in mIoU. Finally, when disabling both denoising and balancing, the mIoU is at its worst. This implies that both denoising and balancing play an important role, and that the merging of dominant tokens in ALGM is a potent approach to rectify attention imbalances and reduce token noise to enhance the segmentation quality.

Similarity threshold. As explained in Sec. 3.3, we compute an automatic similarity threshold τ to select tokens that can be merged. In Fig. 4, we show the performance of ALGM with different thresholds. We find that our automatic threshold finds an optimal balance between efficiency and segmentation quality. Interestingly, taking the automatic threshold during training and using lower thresholds during inference leads to less significant mIoU drops than training with a lower τ. We expect that this difference arises from the fact that early-training embeddings are less accurate, leading to overly aggressive merging during training, which the network cannot recover from; naturally, this problem does not occur during inference. Overall, these results show the versatility of ALGM at test time, as it can be used to optimize efficiency while keeping the mIoU the same like we do with ALGM*, but also to achieve a higher mIoU than the baseline. We observe similar results for other datasets, see the supplementary material.



(a) Input image (b) Local merging (c) Global merging

Figure 5. **Merged tokens.** We depict tokens that are merged as a result of the local CLAP and global GBM merging modules.

6. Discussion

In this work, we propose a token reduction method for semantic segmentation that combines early local merging with later global merging. This is motivated by the finding that, using this merging strategy, we predominantly merge tokens that contain redundant information, meaning that they can be merged without compromising the segmentation quality. With extensive experiments, we show that our approach is indeed able to find a very good balance between efficiency and segmentation quality, outperforming existing work. Interestingly, we also find that using ALGM for token reduction leads to substantial mIoU improvements for complex datasets. With some first analyses, we find that this is likely caused by improved attention balancing and token denoising. However, further research is required to fully understand the causes of these phenomena and their potential applicability to other networks and tasks. Another interesting avenue for future research could be to examine whether token reduction is similarly effective on more complex tasks like panoptic and video segmentation.

Acknowledgements This work was funded by the EU project MODI, grant no. 101076810, and the KDT JU EdgeAI project, grant no. 101097300, and utilized the Dutch national e-infrastructure, supported by the SURF Cooperative under grant no. EINF-5197, funded by the Dutch Research Council (NWO).

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers. In *ICLR*, 2022. 7

[2] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT But Faster. In *ICLR*, 2023. 1, 2, 3, 4, 5, 6, 7, 8

[3] Maxim Bonnaerens and Joni Dambre. Learned Thresholds Token Merging and Pruning for Vision Transformers. *TMLR*, 2023. 1, 2

[4] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-Stuff: Thing and Stuff Classes in Context. In *CVPR*, 2018. 5, 6, 7

[5] Qingqing Cao, Bhargavi Paranjape, and Hannaneh Hajishirzi. PuMer: Pruning and Merging Tokens for Efficient Vision Language Models. In *ACL*, 2023. 2

[6] Shuning Chang, Pichao Wang, Ming Lin, Fan Wang, David Junhao Zhang, Rong Jin, and Mike Zheng Shou. Making Vision Transformers Efficient from A Token Sparsification View. In *CVPR*, 2023. 2

[7] Mengzhao Chen, Mingbao Lin, Ke Li, Yunhang Shen, Yongjian Wu, Fei Chao, and Rongrong Ji. CF-ViT: A General Coarse-to-Fine Method for Vision Transformer. In *AAAI*, 2023. 1

[8] Mengzhao Chen, Wenqi Shao, Peng Xu, Mingbao Lin, Kaipeng Zhang, Rongrong Ji Fei Chao, Yu Qiao, and Ping Luo. DiffRate: Differentiable Compression Rate for Efficient Vision Transformers. In *ICCV*, 2023. 1, 2

[9] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. Vision Transformer Adapter for Dense Predictions. In *ICLR*, 2023. 1, 6, 7

[10] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. In *CVPR*, 2022. 1, 6, 7

[11] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 2016. 5, 6, 7

[12] Evann Courdier, Prabhu Teja Sivaprasad, and François Fleuret. PAUMER: Patch Pausing Transformer for Semantic Segmentation. In *BMVC*, 2022. 1, 3

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 3, 5, 8

[14] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA: Exploring the Limits of Masked Visual Representation Learning at Scale. In *CVPR*, 2023. 6, 7

[15] Mohsen Fayyaz, Soroush Abbasi Kouhpayegani, Farnoush Rezaei Jafari, Eric Sommerlade, Hamid Reza Vaezi Joze, Hamed Pirsiavash, and Juergen Gall. Adaptive token sampling for efficient vision transformers. *ECCV*, 2022. 1, 2

[16] Joakim Bruslund Haurum, Sergio Escalera, Graham W. Taylor, and Thomas B. Moeslund. Which Tokens to Use? Investigating Token Reduction in Vision Transformers. In *ICCV Workshops*, 2023.

[17] Jung Hwan Heo, Arash Fayyazi, Mahdi Nazemi, and Massoud Pedram. A Fast Training-Free Compression Framework for Vision Transformers. *arXiv preprint arXiv:2303.02331*, 2023. 2

[18] Huaibo Huang, Xiaoqiang Zhou, Jie Cao, Ran He, and Tieniu Tan. Vision Transformer with Super Token Sampling. In *CVPR*, 2023. 1

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment Anything. In *ICCV*, 2023. 1

[20] Zhenglun Kong, Peiyan Dong, Xiaolong Ma, Xin Meng, Wei Niu, Mengshu Sun, Xuan Shen, Geng Yuan, Bin Ren, Hao Tang, et al. SPViT: Enabling Faster Vision Transformers via Latency-Aware Soft Token Pruning. In *ECCV*, 2022. 1, 2

[21] Jin Li, Yaoming Wang, Xiaopeng Zhang, Bowen Shi, Dongsheng Jiang, Chenglin Li, Wenrui Dai, Hongkai Xiong, and Qi Tian. AiluRus: A Scalable ViT Framework for Dense Prediction. In *NeurIPS*, 2023. 1, 3, 6, 7

[22] Ling Li, David Thorsley, and Joseph Hassoun. SaiT: Sparse Vision Transformers through Adaptive Token Pruning. *arXiv preprint arXiv:2210.05832*, 2022. 1

[23] Weicong Liang, Yuhui Yuan, Henghui Ding, Xiao Luo, Weihong Lin, Ding Jia, Zheng Zhang, Chao Zhang, and Han Hu. Expediting Large-Scale Vision Transformer for Dense Prediction without Fine-tuning. In *NeurIPS*, 2022. 1, 2, 3, 6, 7

[24] Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not All Patches are What You Need: Expediting Vision Transformers via Token Reorganizations. In *ICLR*, 2022. 1, 2

[25] Yifei Liu, Mathias Gehrig, Nico Messikommer, Marco Cannici, and Davide Scaramuzza. Revisiting Token Pruning for Object Detection and Instance Segmentation. In *WACV*, 2024. 1

[26] Yuang Liu, Qiang Zhou, Jing Wang, Fan Wang, Jun Wang, and Wei Zhang. Dynamic Token-Pass Transformers for Semantic Segmentation. In *WACV*, 2024. 1, 3, 6, 7

[27] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin Transformer V2: Scaling Up Capacity and Resolution. In *CVPR*, 2022. 7

[28] Sifan Long, Zhen Zhao, Jimin Pi, Shengsheng Wang, and Jingdong Wang. Beyond Attentive Tokens: Incorporating Token Importance and Diversity for Efficient Vision Transformers. In *CVPR*, 2023. 1, 2, 3

[29] Chenyang Lu, Daan de Geus, and Gijs Dubbelman. Content-aware Token Sharing for Efficient Semantic Segmentation with Vision Transformers. In *CVPR*, 2023. 1, 2, 3, 6, 7

[30] Lingchen Meng, Hengduo Li, Bor-Chun Chen, Shiyi Lan, Zuxuan Wu, Yu-Gang Jiang, and Ser-Nam Lim. AdaViT:

Adaptive Vision Transformers for Efficient Image Recognition. In *CVPR*, 2022. 1, 2

[31] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *CVPR*, 2014. 5, 6, 7

[32] Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. BEiT v2: Masked Image Modeling with Vector-Quantized Visual Tokenizers. *arXiv preprint arXiv:2208.06366*, 2022. 7

[33] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. In *NeurIPS*, 2021. 1, 2

[34] Cedric Renggli, Andre Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to Merge Tokens in Vision Transformers. *arXiv preprint arXiv:2202.12015*, 2022. 2

[35] Meta Research. fvcore. https://github.com/facebookresearch/fvcore, 2023. 5

[36] Michael S. Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: What Can 8 Learned Tokens Do for Images and Videos? In *NeurIPS*, 2021. 1, 2

[37] Cai Jianfei Shi Hengcan, Hayat Munawar. Transformer Scale Gate for Semantic Segmentation. In *CVPR*, 2023. 1

[38] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for Semantic Segmentation. In *ICCV*, 2021. 1, 3, 5, 6, 7, 8

[39] Quan Tang, Bowen Zhang, Jiajun Liu, Fagiu Liu, and Yifan Liu. Dynamic Token Pruning in Plain Vision Transformers for Semantic Segmentation. In *ICCV*, 2023. 1, 3, 6, 7

[40] Hongjie Wang, Bhishma Dedhia, and Niraj K. Jha. Zero-TPrune: Zero-Shot Token Pruning through Leveraging of the Attention Graph in Pre-Trained Transformers. *arXiv preprint arXiv:2305.17328*, 2023. 1, 2

[41] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, and Furu Wei. Image as a Foreign Language: BEiT Pretraining for Vision and Vision-Language Tasks. In *CVPR*, 2023. 7

[42] Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not All Images are Worth 16x16 Words: Dynamic Transformers for Efficient Image Recognition. In *NeurIPS*, 2021. 1

[43] Siyuan Wei, Tianzhu Ye, Shen Zhang, Yao Tang, and Jiajun Liang. Joint Token Pruning and Squeezing Towards More Aggressive Compression of Vision Transformers. In *CVPR*, 2023. 2, 3

[44] Xinjian Wu, Fanhu Zeng, Xiudong Wang, Yunhe Wang, and Xinghao Chen. PPT: Token Pruning and Pooling for Efficient Vision Transformers. *arXiv preprint arXiv:2310.01812*, 2023. 1, 2

[45] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified Perceptual Parsing for Scene Understanding. In *ECCV*, 2018. 7

[46] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers. In *NeurIPS*, 2021. 1

[47] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. GroupViT: Semantic Segmentation Emerges from Text Supervision. In *CVPR*, 2022. 1

[48] Hongxu Yin, Arash Vahdat, Jose Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive Tokens for Efficient Vision Transformer. In *CVPR*, 2022. 1, 2

[49] Lu Yu and Wei Xiang. X-Pruner: eXplainable Pruning for Vision Transformers. In *CVPR*, 2023.

[50] Wang Zeng, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Not All Tokens Are Equal: Human-centric Visual Analysis via Token Clustering Transformer. In *CVPR*, 2022. 1

[51] Bowen Zhang, Zhi Tian, Quan Tang, Xiangxiang Chu, Xiaolin Wei, Chunhua Shen, et al. SegViT: Semantic Segmentation with Plain Vision Transformers. In *NeurIPS*, 2022. 1, 5, 6

[52] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers. In *CVPR*, 2021. 1, 5, 6, 7

[53] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene Parsing through ADE20K Dataset. In *CVPR*, 2017. 3, 5, 7, 8

[54] Zhuofan Zong, Kunchang Li, Guanglu Song, Yali Wang, Yu Qiao, Biao Leng, and Yu Liu. Self-slimmed Vision Transformer. In *ECCV*, 2022. 1, 2