ClustViT: Clustering-based Token Merging for Semantic Segmentation

Fabio Montello, Ronja Güldenring and Lazaros Nalpantidis

***Abstract*— This is a reduced version of the original paper accepted at ICRA 2026 [1]. Vision Transformers can achieve high accuracy and strong generalization across various contexts, but their practical applicability on real-world robotic systems is limited due to their quadratic attention complexity. Recent works have focused on dynamically merging tokens according to the image complexity. Token merging works well for classification but is less suited to dense prediction. We propose ClustViT, where we expand upon the Vision Transformer (ViT) backbone and address semantic segmentation. Within our architecture, a trainable Cluster module merges similar tokens along the network guided by pseudo-clusters from segmentation masks. Subsequently, a Regenerator module restores fine details for downstream heads. Our approach achieves up to $2.18\times$ fewer GFLOPs and $1.64\times$ faster inference on three different datasets, with comparable segmentation accuracy. Our code and models are made publicly available[1].**

I. Introduction

Reducing the computational cost of Vision Transformers (VIT) [2] in dense prediction tasks, such as semantic segmentation, is essential when considering autonomous robotic systems that need to perceive their operational environments. However, one major limitation of Transformers is that their computational complexity grows quadratically with respect to the input size. Even though existing approaches have successfully reduced token redundancy for classification tasks, they often struggle to generalize to dense prediction settings, where preserving spatial and semantic detail is critical.

In this paper, we focus on semantic segmentation and argue that the computational cost of ViTs can be significantly reduced without compromising segmentation accuracy by incorporating appropriate token clustering. More precisely, the semantic information embedded in segmentation masks is not only used to supervise the segmentation task. We also use it to train a token clustering component integrated between Transformer blocks of a ViT backbone, which allows the model to identify and merge semantically similar tokens.

Our approach has several characteristics that make it well-suited for the task of reducing the computational complexity of ViTs in semantic segmentation. First, it introduces a clustering module that is trained end-to-end within the Transformer backbone, allowing the model to dynamically identify and merge semantically similar tokens during inference. Alike tokens get grouped from there on into representative tokens. This reduces the number of active tokens without disrupting the computation graph, leading to faster processing, as shown in Fig. 1. Second, by leveraging pseudo-clusters derived from segmentation masks, the clustering process is guided by semantic information rather than low-level token similarity, which tailors the compression to the content of the image. Third, our architecture includes a regenerator module that reconstructs individual token representations from their clustered counterparts, ensuring compatibility with dense prediction heads. Together, these components enable efficient computation while maintaining high-quality segmentation outputs in a variety of datasets, as shown in Fig. 2. The *contribution* of this work is threefold:

- We present the concept of merging tokens according

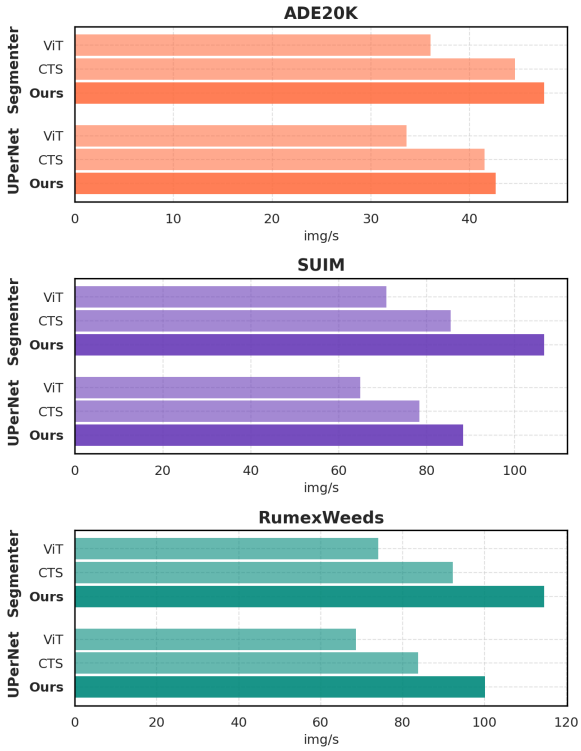

Fig. 1: **Comparison of segmentation speed (*img/s*)** across three datasets (*ADE20K*, *SUIM*, and *RumexWeeds*). Each plot shows results for different segmentation backbones: *Segmenter* (top) and *UPerNet* (bottom). For each dataset, we compare three models: *ViT*, *CTS*, and our model. Across both backbones and all datasets, our model consistently achieves the highest image throughput. The improvements are most pronounced for datasets with few subjects and dominated by background (see ablation study).

This work has been supported by Innovation Fund Denmark through the project "Safety and Autonomy for Vehicles in Agriculture (SAVA)", 2105-00013A.

All authors are with the Department of Electrical and Photonics Engineering, DTU - Technical University of Denmark, Kgs. Lyngby, Denmark. {fabmo, ronjag, lanalpa}@dtu.dk

[1]https://github.com/DTU-PAS/clustvit

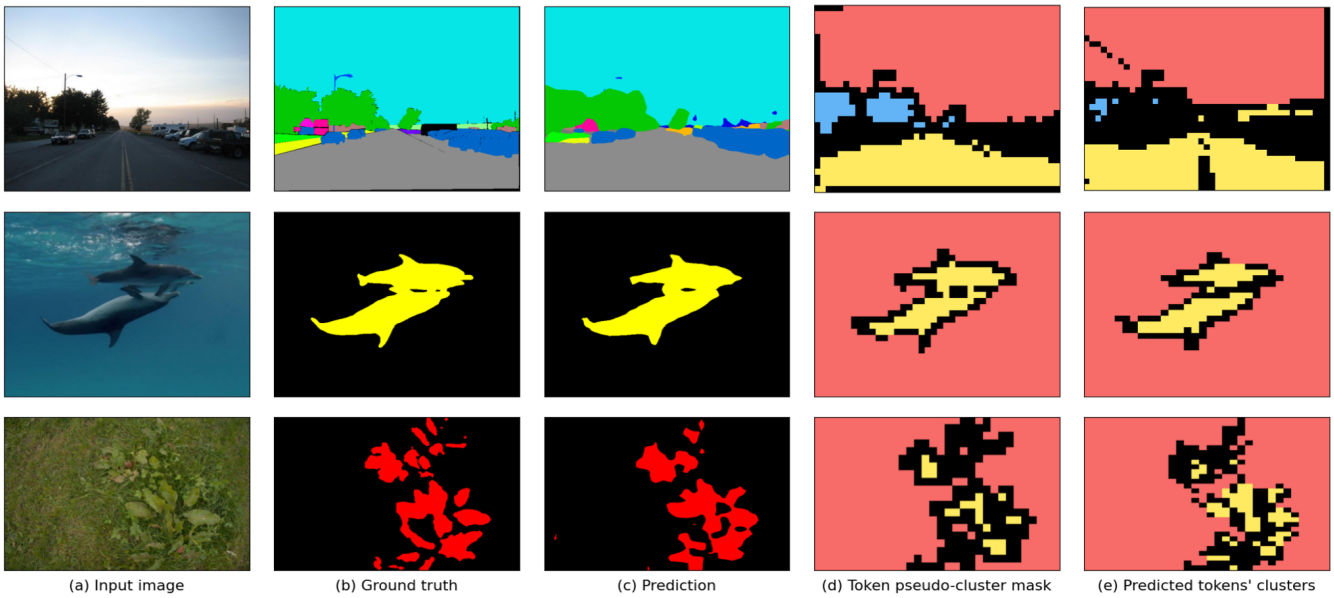

Fig. 2: **Examples from the *ADE20K* [3] (top), *SUIM* [4] (middle), and *RumexWeeds* [5] (bottom) datasets**. Columns: (a) Input image, (b) Ground truth semantic segmentation, (c) Model prediction, (d) Mask for the token clustering generated from the ground truth, (e) Predicted cluster for each token. Starting from the output of (e), regions with the same non-black color belong to the same cluster and get merged into a single token for the subsequent Transformer layers, while tokens in black regions are kept intact to preserve fine details. Our model configuration used to obtain these results is ClustViT-b$_{k3,ip3}$.

to the semantic content of the image—information that can be derived from the segmentation mask. This allows unstructured compression, simplifying processing.

- We present ClustViT, an end-to-end trainable backbone that can identify and merge semantically similar tokens. We instill the standard ViT architecture with a component that clusters tokens based on image semantics. A regenerator component reconstructs tokens at the end of the encoder to enable the use of off-the-shelf segmentation heads.
- Our approach significantly reduces the computational cost compared to state-of-the-art methods on datasets with few subjects and dominated by background—common in robotics—while achieving comparable segmentation accuracy on more complex datasets.

This document is a reduced version of the original paper accepted at ICRA 2026 [1]. For the complete manuscript, including extended background literature, detailed architectural specifications, and in-depth ablations, please refer to the full published work.

II. Related Work

A. Semantic Segmentation with Vision Transformers

Semantic segmentation classifies each pixel in an image into its semantic category. It is a core problem in computer vision with applications in autonomous driving, image editing, robotics, and image analysis [6], [7], [8]. ViTs have recently emerged and significantly surpass previous convolutional approaches in various vision tasks, including semantic segmentation [9]. SETR [10] is the first to successfully replace the traditional CNN backbone with a ViT backbone, though it still relies on a CNN decoder. Segmenter [11] showed that a pure encoder-decoder Transformer approach is a viable solution by designing a lightweight decoder mask Transformer that uses attention to produce class-specific masks. SegFormer [12] improved scalability with a hierarchical Transformer encoder. SegViT [13] uses a plain ViT with the use of an attention mechanism for the creation of Transformer masks. Finally, Mask2Former [14] introduces one architecture to address any possible segmentation task (panoptic, instance, or semantic) by constraining cross-attention to predicted mask regions.

B. Token skimming for semantic segmentation

Whereas most existing methods target image classification, some solutions have been proposed to address specifically segmentation. What makes dense prediction tasks different from classification is that all tokens are needed at the encoder output for it to be compatible with off-the-shelf segmentation heads. Yuan et al. in [15] present a token clustering layer and a token reconstruction layer into a pretrained ViT to merge and unmerge neighboring tokens based on a local k-means clustering. This method uses a structured per-block approach (fixed amount of compressed tokens). In our case, the focus is on tackling the problem based on semantic similarity instead of distance similarity among tokens.

The work of [16] combines token skimming and early exits. After each Transformer block in the encoder, an auxiliary head halts *simple* tokens, sending them to an early decoder, while the rest continue through the full encoder.

This work is relevant to ours but approaches the problem differently by sending partial amounts of tokens earlier to the decoder head; each head sees only partial context at each exit. In contrast, we work on reconstructing all the tokens to be passed to the decoder head.

Most related to our approach, [17] proposes Content-aware Token Sharing (CTS) that uses a class-agnostic policy network attached before the tokenizer to predict whether 2×2 neighbor patches contain the same semantic class. In that case, the patch shares a token, effectively reducing resolution. The original resolution is then reconstructed after the backbone for CNN decoders or expanded after the decoder for Transformer decoders. This work is closest in spirit to our own approach, however, the token compression is structured (at most group of 4 tokens in each region) and predefined as a hyperparameter. In contrast, our method has the freedom of compressing an arbitrary amount of tokens. Furthermore, in CTS, a policy network needs to be trained aside from the main network. Our solution allows end-to-end training of the clustering component as a part of the architecture.

Finally, different from all existing approaches, we explore clustering guided by prior knowledge from the segmentation mask, enabling token-level classification for clustering and merging similar ones. Our backbone is an optimized ViT, so comparisons naturally focus on this architecture.

III. Method

In this section, we detail the core idea of clustering semantically similar tokens. We first briefly revisit the Transformer architecture (III-A), followed by an overview of how our ClustViT method works overall (III-B). We then delve into details of the different ClustViT characteristics: the clustering module (III-C), the regenerator module (III-D), and the clustering training from the generation of pseudo-clusters and integration into a combined loss (III-E).

A. Preliminaries

Our approach expands upon the ViT [2] architecture and leverages some of its unique properties. Assuming an RGB image $X \in \mathbb{R}^{H \times W \times 3}$, ViT starts by splitting it into $P \times P$ non-overlapping patches, flattening them, and projecting each into a D-dimensional space. These projected patches, called tokens, are enriched with a positional encoding, and a specific classification token (x_{cls}) is concatenated at the beginning of the sequence to facilitate information flow. The resulting sequence is $Z \in \mathbb{R}^{(N+1) \times D}$, where N is the total number of patches. ViT then applies stacked attention blocks, each composed of Multi-Head Self-Attention (MHSA) and the Feed-Forward Network (FFN), both wrapped with Layer Normalization (LN) and residual connections. Subsequent attention blocks are called layers, indexed as $l = \{1, ..., N\}$, for which the complete set of operations in each block is the following:

$$Z'_l = \text{MHSA}(\text{LN}(Z_{l-1})) + Z_{l-1} \tag{1}$$

$$Z_l = \text{FFN}(\text{LN}(Z'_l)) + Z'_l \tag{2}$$

ViT is commonly used as an encoder with task-specific decoders. For semantic segmentation, we experimented with two different segmentation heads: (*i*) Segmenter [11], which reprojects tokens into a latent space, concatenates learned class embeddings, and processes them with Transformer layers; and (*ii*) UPerNet [18], which uses a feature pyramid and multi-scale fusion as its decoder, chosen for a fair comparison since it is part of the original encoder-decoder combination of CTS [17]. In both cases, outputs are upsampled by bilinear interpolation to the original resolution at the end.

B. ClustViT

Our proposed ClustViT architecture modifies the standard ViT with a clustering mechanism to dynamically merge semantically similar tokens (cf. Fig. 3). More precisely, in this work, we introduce two novel components to the standard ViT architecture: the Cluster and the Regenerator components. In between these two components, the Transformer blocks remain unchanged but operate on a reduced (compressed) number of tokens.

As shown in Fig. 3, the input image is split into patches, converted to tokens, augmented with positional embeddings, and prepended with the x_{cls} token. The sequence is fed through the early layers of the encoder. All these steps are identical to the vanilla ViT. However, in our ClustViT, in between layers and at a predefined injection point (ip_l), the tokens enter our Cluster block (details in III-C) which assigns each token to one of $k \in \mathbb{N}$ clusters or leaves it unclustered. Clustered tokens get aggregated into k representative tokens, which attend subsequent standard Transformer blocks with the unclustered tokens, reducing sequence length and computational cost. After all encoder layers, our Regenerator block (details in III-D) propagates updates from the representative tokens back to their original tokens. The restored sequence is then passed to the chosen off-the-shelf decoder head for prediction.

C. Cluster Block

This section introduces our cluster block (cf. left side of Fig. 3), which compresses semantically similar tokens.

MLP $\mathcal{C}$ (1) takes the tokens Z_{l-1} as input and outputs cluster logits L_C. It is composed of two FFN with a ReLU activation in between:

$$L_C = \mathcal{C}(Z_{l-1}) = \text{Linear}_2(\text{ReLU}(\text{Linear}_1(Z_{l-1}))) \tag{3}$$

where $\text{Linear}_1 : \mathbb{R}^D \to \mathbb{R}^H$ and $\text{Linear}_2 : \mathbb{R}^H \to \mathbb{R}^{k+1}$, with H being the hidden dimension and k being a hyperparameter defining the number of clusters.
Thus, $L_C \in \mathbb{R}^{B \times N \times (k+1)}$. The output dimension $k + 1$ corresponds to k active clusters, plus one additional category for unclustered tokens.
The cluster assignments $C \in \{0, 1, \ldots, k\}^{B \times N}$ are obtained by applying an argmax operation along the last dimension of the cluster logits L_C (2). This assigns each patch token

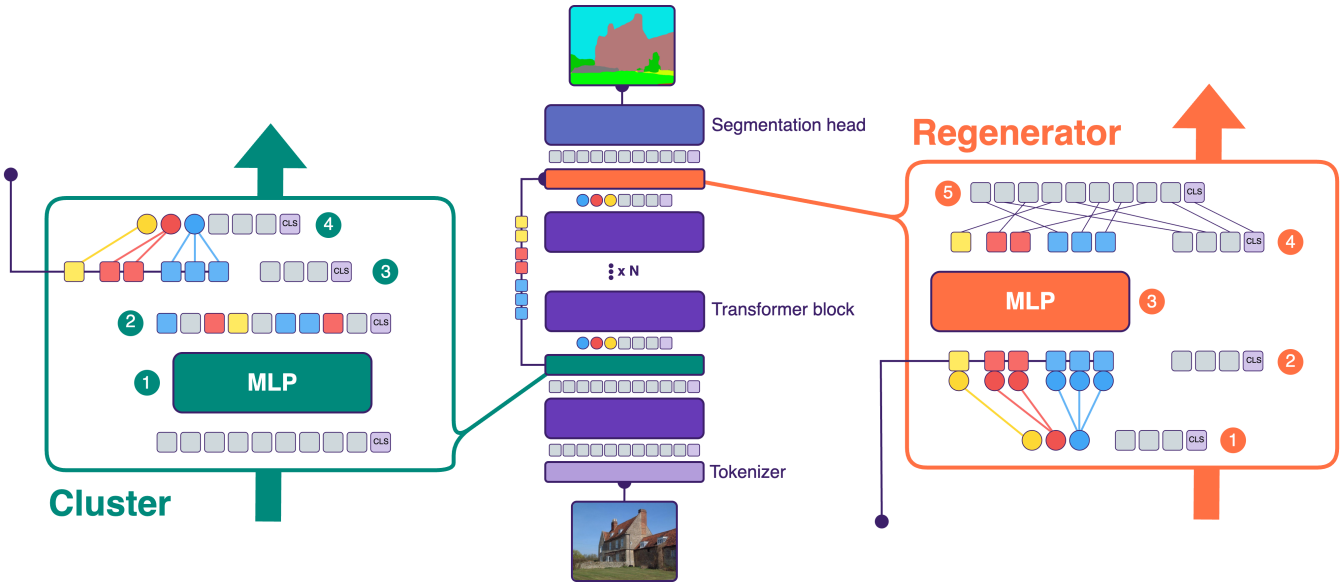

Fig. 3: **ClustViT overview.** The standard Transformer pipeline is executed (center, from bottom to top) through the tokenizer and few Transformer blocks until the Cluster module is encountered. Subsequently, the Transformer backbone proceeds with a reduced amount of tokens. Before being passed to the segmentation head, the tokens are reconstructed by the Regenerator module. **Cluster module (left)**: 1 An MLP predicts the probability of a token belonging to a cluster. 2 Tokens of the same cluster (color coded) are grouped; unclustered (gray) tokens are kept intact. 3 Tokens within each group are aggregated into a single representative token. 4 The reduced token set (cluster representatives + unclustered tokens + CLS) is fed through the remaining Transformer blocks, lowering compute. **Regenerator module (right)**: 1 Takes the reduced sequence. 2 Uses stored assignments to expand each representative back to its original token positions. 3 An MLP refines the reinstated per-token features. 4 Reconstructed full-resolution tokens and preserved unclustered tokens are combined. 5 The restored sequence is delivered to the segmentation head.

to a specific cluster ID (0 for unclustered, 1 to k for active clusters):

$$C_{b,n} = \text{argmax}_{k\in\{0,\dots,k\}}(L_C)_{b,n,k} \tag{4}$$

where b is the batch index and n is the token index.
Based on the cluster assignments C, the tokens not belonging to any cluster are left untouched, while for each cluster k the tokens get compacted into a representative token $E_k \in \mathbb{R}^{B\times D}$, computed as the mean of all patch tokens assigned to that cluster (3 and 4):

$$E_{b,k} = \frac{1}{|\{n' \mid C_{b,n'} = k\}|} \sum_{n'|C_{b,n'}=k} Z_{l-1,b,n'} \quad \text{where } |\{n' \mid C_{b,n'} = k\}| > 0 \tag{5}$$

The final reduced sequence to be passed to the subsequent Transformer layers is obtained by concatenating the kept patch tokens, and the representative tokens:

$$Z_{red} = [Z_{l-1,\text{unclustered}}; E] \tag{6}$$

and it is of shape $Z_{red} \in \mathbb{R}^{B\times(N_{\text{unclustered}}+k)\times D}$.
The original matrix of the clustered tokens $Z_{l-1,\text{clustered}}$ is passed as residuals to the regenerator module for reconstruction with respect to the updated representatives. The classification token x_{cls} is excluded during clustering and concatenated back afterward; this detail is omitted in the formalization and Fig. 3 for simplicity.

D. Regenerator Block

This section describes the steps to regenerate compressed tokens from the updated representatives and the residual clustered tokens (cf. right side of Fig. 3).

Let $Z_{out} \in \mathbb{R}^{B\times(N_{\text{unclustered}}+k)\times D}$ be the output sequence from the Transformer layers after layer l that processed the reduced sequence. We recover only the representative tokens by deconcatenating them from the unclustered tokens (1). The two resulting matrices will be $Z_{repr} \in \mathbb{R}^{B\times k\times D}$ and $Z_{unclustered} \in \mathbb{R}^{B\times N_{\text{unclustered}}\times D}$.

Subsequently, we expand Z_{repr} in such a way that for each token in the residual matrix $Z_{l-1,\text{clustered}} \in \mathbb{R}^{B\times(N_{\text{clustered}}+k)\times D}$, the corresponding representative token from Z_{repr} gets placed into an empty matrix $Z_{reprexp} \in \mathbb{R}^{B\times(N_{\text{clustered}}+k)\times D}$. We then proceed to concatenate the two matrices (2) on the token embedding dimension D and pass the resulting matrix to (3) a refining MLP:

$$Z_{\text{refined}} = \text{Linear}_2(\text{GELU}(\text{Linear}_1(f_{\text{concat}}))) \tag{7}$$

TABLE I: **Comparison of segmentation performance** across different heads (*Segmenter*, *UPerNet*) and backbones (*ViT-b*, *CTS-b*, *ClustViT-b*) on *ADE20K*, *SUIM*, and *RumexWeeds* datasets. Metrics reported include mIoU (higher is better), image throughput (higher is better), and GFLOPs (lower is better). Bold values indicate the best performance in each category. GFLOPs include standard deviation as ±.

Head	Backbone	mIoU (↑)	img/s (↑)	GFLOPs (↓)
		ADE20K		
Segmenter	ViT-b	**49.22**	36.06	473.15 ± 99.30
	CTS-b	45.96	44.62	347.11 ± 72.85
	ClustViT-b$_{k3,ip3}$	46.10	**47.56**	**321.28 ± 88.83**
	ClustViT-b$_{k3,ip4}$	46.19	45.52	338.68 ± 89.91
	ClustViT-b$_{k1,ip4}$	48.20	41.66	399.13 ± 99.86
UPerNet	ViT-b	**47.53**	33.63	637.51 ± 72.85
	CTS-b	46.85	41.55	511.48 ± 107.35
	ClustViT-b$_{k3,ip3}$	44.70	**42.65**	**494.67 ± 119.59**
	ClustViT-b$_{k3,ip4}$	44.16	41.64	508.52 ± 120.07
	ClustViT-b$_{k1,ip4}$	45.92	37.69	566.97 ± 131.10
		SUIM		
Segmenter	ViT-b	**69.91**	70.80	252.33 ± 0.00
	CTS-b	66.33	85.41	183.40 ± 0.00
	ClustViT-b$_{k3,ip3}$	61.95	106.65	**115.49 ± 9.00**
	ClustViT-b$_{k4,ip3}$	63.26	115.95	134.19 ± 14.54
	ClustViT-b$_{k4,ip4}$	63.86	**116.56**	132.94 ± 8.95
UPerNet	ViT-b	**70.01**	64.9	384.24 ± 0.00
	CTS-b	68.05	78.31	279.35 ± 0.00
	ClustViT-b$_{k3,ip3}$	63.77	88.31	**221.02 ± 15.04**
	ClustViT-b$_{k3,ip4}$	64.98	97.86	236.53 ± 8.30
	ClustViT-b$_{k2,ip4}$	65.02	**103.54**	224.51 ± 16.06
		RumexWeeds		
Segmenter	ViT-b	51.29	74.10	252.07 ± 0.00
	CTS-b	48.54	92.32	183.18 ± 0.00
	ClustViT-b$_{k3,ip3}$	49.82	**114.53**	122.23 ± 46.76
	ClustViT-b$_{k2,ip3}$	50.45	115.62	**118.73 ± 41.80**
	ClustViT-b$_{k3,ip4}$	**51.53**	107.34	135.54 ± 37.98
UPerNet	ViT-b	**51.56**	68.61	348.23 ± 0.00
	CTS-b	49.13	83.76	279.34 ± 0.00
	ClustViT-b$_{k3,ip3}$	50.55	**100.18**	221.14 ± 34.20
	ClustViT-b$_{k2,ip3}$	49.63	99.99	**221.00 ± 34.22**
	ClustViT-b$_{k3,ip4}$	51.20	94.17	237.91 ± 35.87

where $\text{Linear}_1 : \mathbb{R}^{2D} \to \mathbb{R}^D$ and $\text{Linear}_2 : \mathbb{R}^D \to \mathbb{R}^D$. Finally, we can recompose back the refined tokens into their original position before the merging. We take into account also the processed unclustered tokens (4). Our final output sequence is $Z_{\text{final}} \in \mathbb{R}^{B\times(1+N)\times D}$ (5). This sequence now matches the input sequence in length and is ready for the decoder head. As before, the classification token x_{cls} is excluded during processing and concatenated back afterwards.

E. Pseudo-clusters and combined loss for model training

To train the cluster module MLP, we generate pseudo-clusters from the segmentation mask. We split the segmentation mask into patches, as done by the tokenizer. For each patch, we check if all the pixels inside belong to the same semantic class. If all pixels belong to the same semantic class, that class is assigned; otherwise, a value of 0 indicates mixed classes. Next, among patches of a single class, we keep only the top-k most frequent classes, where k is the hyperparameter for the number of clusters. Classes outside of the top-k are set to 0; top-k classes get a label number $l \in 1, ..., k$, ordered from the most to the least frequent. Examples of pseudo-clusters are shown in Fig. 2(d).

Once pseudo-clusters are computed, we define a composite loss that accounts for both segmentation accuracy and cluster behavior. For both tasks, we use cross-entropy:

$$\mathcal{L} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c}\log(\hat{y}_{n,c}) \tag{8}$$

where for the segmentation case N is the number of pixels, C is the number of classes, $y_{n,c}$ is the ground truth, while $\hat{y}_{n,c}$ is the predicted class value. In the case of clustering, N is the total number of patches, $C = k+1$, where k is the number of clusters; $y_{n,c}$ is the pseudo-cluster for a specific patch, and $\hat{y}_{n,c}$ is the predicted cluster. The final loss is then composed of:

$$\mathcal{L} = \mathcal{L}_{segm} + \lambda\,\mathcal{L}_{clust} \tag{9}$$

where λ balances the relative importance of the clustering loss.

IV. Experiments

A. Datasets and Metrics

We evaluate our backbone on three different semantic segmentation datasets. The first one, **ADE20K** [3], contains *20k* training and *2k* validation images across 150 classes in indoor and outdoor scenes; results are reported on the validation set. While *ADE20K* tests generalizability, it is less representative of real-world robotics scenarios. For this reason, we also use the **SUIM** dataset [4] containing underwater scenes and **RumexWeeds** [5] containing agricultural images. SUIM is a dataset for underwater robotics composed of *1525* training images (split 85/15 for test and validation) and *110* test images, with 8 different classes. RumexWeeds is composed of data collected in grasslands and targets two weed species with *2796* training, *1411* validation and *1303* test images.

Segmentation accuracy is measured via mean Intersection over Union (**mIoU**). Inference speed is reported as image throughput (**img/s**) and Giga Floating Point Operations (**GFLOPs**) are calculated to estimate the computational cost; in the case of GFLOPs, standard deviation is included to reflect fluctuations from model dynamics and input image size. It remains fixed with respect to repeated iterations (in contrast to *img/s* that requires warmup).

B. Experimental setup

Model and loss configuration. We use the base ViT architecture for model comparisons and hyperparameter ablation, while also showing how our optimized solution scales

across different ViT sizes. All architectures use a patch size of 16. For the combined loss, after empirical testing, we found that $\lambda = 0.1$ works best to balance the resolution of both segmentation and clustering problems. As mentioned, we compare directly with CTS [17], considered the state-of-the-art architecture. While our approach could support full-network token compression (including both encoder and decoder), as seen in CTS, we focus only on the backbone to allow standard segmentation heads to be used interchangeably. Among CTS's token-sharing settings, we adopt the configuration achieving the highest accuracy, which compresses 30% of tokens.

Training setup. We use *mmsegmentation*[2] as base framework. Following Segmenter [11] we train with SGD (learning rate 0.001, momentum 0.9, weight decay 0.0005) and a polynomial decay of power 0.9 to a minimum of 0.0001 over 160k iterations for *ADE20K* and 80k for *RumexWeeds* and *SUIM*. Batch size is 8 during training; inference is on single images. All *mIoU* scores use single-scale inputs. *ADE20K* images retain their original resolution, while *RumexWeeds* and *SUIM* images are resized to 640×640. Backbones are initialized with ImageNet-pretrained [19] ViT weights (384×384, patch size 16) from *mmsegmentation*. Our approach supports multiple segmentation heads: Segmenter (transformer-based) and UPerNet (CNN-based). Segmenter uses only the ViT output tokens, which we provide after reconstruction. UPerNet also uses intermediate tokens (after layers 4, 6, and 8 for base ViT), which are reconstructed at each stage via the regenerator module.

Infrastructure. All trainings are performed on a server on a single Nvidia H100 GPU with 80GB of VRAM, while image throughput and GFLOPs were measured on a workstation with an Intel i9-14900KF CPU, 96GB of RAM and a Nvidia 3090 GPU with 24GB of VRAM.

C. Results

Table I reports results for different backbones and heads across the three datasets, showing the top three configurations per dataset in terms of *mIoU*. We provide different configurations of our ClusterViT architecture based on the number of clusters k and the injection point ip where tokens are compressed. For example, ClustViT-b$_{k4,ip2}$ is trained to perform at most 4 clusters ($k = 4$) and the Cluster module is placed after the second Transformer block ($ip = 2$) in the architecture. On *ADE20K*, ClustViT-b$_{k3,ip3}$ achieves the lowest computational cost (321.28 GFLOPs), a $1.47\times$ improvement over the baseline, with a trade-off in terms of accuracy of 1.66 mIoU points (about 3.7%). The best-performing model still achieves $1.15\times$ speedup with respect to the baseline, with only 2% accuracy decrease. Using UPerNet as the head shows a similar pattern, although the accuracy seems to be affected more by our compression method.

On *SUIM*, which contains large background areas (water), Segmenter predictions decrease notably: ClustViT-b$_{k4,ip3}$ drops in mIoU of 9.5% compared to the baseline and 4.6% compared to CTS, likely due to token merging reducing details representation. However, speed improvements are substantial: $1.64\times$ speedup in terms of FPS and $2.18\times$ increase in terms of GFLOPs in the respective best cases. Even when compared to CTS, we can observe $+21$ FPS increase and $1.58\times$ increase of GFLOPs. UPerNet confirms this trend. Notably, fastest FPS does not always correspond to lowest GFLOPs, likely due to architectural optimizations for specific operations, which can run in parallel and be aggregated.

On *RumexWeeds*, a dataset composed mostly of grassy background, Segmenter slightly outperforms even the baseline ($+0.24$% in mIoU) while achieving $1.54\times$ FPS and $2.12\times$ GFLOPs gains. This highlights ClustViT's strength in scenarios with few subjects and largely uniform backgrounds, common in many robotics applications.

To further understand the accuracy trade-offs of token clustering, we evaluated global size-stratified recall (Table II) at different sizes relative to the image area: Small (<5%), Medium (5–15%), Large (15–30%), and Huge (>30%). While *ClustViT-b* maintains near-baseline accuracy on "Huge" elements, it causes a minor consistent performance drop across "Small," "Medium," and "Large" objects, instead of disproportionately penalizing smaller classes.

TABLE II: **Global size-stratified recall evaluation** between *ViT-b* and ClustViT-b$_{k3,ip3}$ across the datasets. Values represent the recall percentage for each relative object size category. Missing ground truth is indicated with "–".

Model	**Small** (<5%)	**Medium** (5–15%)	**Large** (15–30%)	**Huge** (>30%)
		ADE20K		
ViT-b	65.99	81.06	86.45	88.57
ClustViT-b-3,3	63.48	78.93	84.84	87.52
		SUIM		
ViT-b	68.25	82.50	87.01	92.70
ClustViT-b-3,3	63.60	73.67	77.68	91.42
		RumexWeeds		
ViT-b	49.21	87.26	–	99.51
ClustViT-b-3,3	45.09	84.25	–	99.50

V. Conclusions

In this paper, we presented a novel approach to token compression that exploits semantic information from segmentation masks to guide the token merging process. Building on top of the ViT backbone, we introduced ClustViT, a model that integrates a clustering mechanism to adaptively merge tokens in between Transformer layers. A regenerator block is also introduced to restore the full representation for compatibility with standard segmentation heads. Experiments highlight how semantically guided token compression improves the tradeoff between efficiency and accuracy, and point toward broader opportunities for incorporating token clustering strategies into vision models for robotics.

[2]https://github.com/open-mmlab/mmsegmentation

REFERENCES

[1] F. Montello, R. Güldenring, and L. Nalpantidis, "Clustvit: Clustering-based token merging for semantic segmentation," in *2026 IEEE/RSJ International Conference on Robotics and Automation (ICRA)*. IEEE, 2026.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[3] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene Parsing through ADE20K Dataset," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, July 2017, pp. 5122–5130.

[4] M. J. Islam, C. Edge, Y. Xiao, P. Luo, M. Mehtaz, C. Morse, S. S. Enan, and J. Sattar, "Semantic Segmentation of Underwater Imagery: Dataset and Benchmark," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 1769–1776.

[5] R. Güldenring, F. K. Van Evert, and L. Nalpantidis, "Rumexweeds: A grassland dataset for agricultural robotics," *Journal of Field Robotics*, vol. 40, no. 6, pp. 1639–1656, 2023.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[7] R. Güldenring, E. Boukas, O. Ravn, and L. Nalpantidis, "Few-leaf learning: Weed segmentation in grasslands," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Prague, Czech Republic, 2021.

[8] J. Jain, J. Li, M. Chiu, A. Hassani, N. Orlov, and H. Shi, "OneFormer: One Transformer to Rule Universal Image Segmentation," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Vancouver, BC, Canada: IEEE, June 2023, pp. 2989–2998.

[9] X. Li, H. Ding, H. Yuan, W. Zhang, J. Pang, G. Cheng, K. Chen, Z. Liu, and C. C. Loy, "Transformer-Based Visual Segmentation: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 12, pp. 10 138–10 163, Dec. 2024.

[10] S. Zheng, J. Lu, H. Zhao, X. Zhu, Z. Luo, Y. Wang, Y. Fu, J. Feng, T. Xiang, P. H. Torr, and L. Zhang, "Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA: IEEE, June 2021, pp. 6877–6886.

[11] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for Semantic Segmentation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 7242–7252.

[12] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers," in *Advances in Neural Information Processing Systems*, vol. 34. Curran Associates, Inc., 2021, pp. 12 077–12 090.

[13] B. Zhang, Z. Tian, Q. Tang, X. Chu, X. Wei, C. Shen, *et al.*, "Segvit: Semantic segmentation with plain vision transformers," *Advances in Neural Information Processing Systems*, vol. 35, pp. 4971–4982, 2022.

[14] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention Mask Transformer for Universal Image Segmentation," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, LA, USA: IEEE, jun 2022, pp. 1280–1289.

[15] Y. Yuan, W. Liang, H. Ding, Z. Liang, C. Zhang, and H. Hu, "Expediting Large-Scale Vision Transformer for Dense Prediction Without Fine-Tuning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 1, pp. 250–266, Jan. 2024.

[16] Q. Tang, B. Zhang, J. Liu, F. Liu, and Y. Liu, "Dynamic Token Pruning in Plain Vision Transformers for Semantic Segmentation," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. Paris, France: IEEE, Oct. 2023, pp. 777–786.

[17] C. Lu, D. De Geus, and G. Dubbelman, "Content-aware Token Sharing for Efficient Semantic Segmentation with Vision Transformers," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Vancouver, BC, Canada: IEEE, June 2023, pp. 23 631–23 640.

[18] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, "Unified Perceptual Parsing for Scene Understanding," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Cham: Springer International Publishing, 2018, vol. 11209, pp. 432–448.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 248–255.