HCFormer: Unified Image Segmentation with Hierarchical Clustering

Teppei Suzuki

Denso IT Laboratory, Inc.

Abstract

Hierarchical clustering is an effective and efficient approach widely used for classical image segmentation methods. However, many existing methods using neural networks generate segmentation masks directly from per-pixel features, complicating the architecture design and degrading the interpretability. In this work, we propose a simpler, more interpretable architecture, called HCFormer. HCFormer accomplishes image segmentation by bottom-up hierarchical clustering and allows us to interpret, visualize, and evaluate the intermediate results as hierarchical clustering results. HCFormer can address semantic, instance, and panoptic segmentation with the same architecture because the pixel clustering is a common approach for various image segmentation tasks. In experiments, HCFormer achieves comparable or superior segmentation accuracy compared to baseline methods on semantic segmentation (55.5 mIoU on ADE20K), instance segmentation (47.1 AP on COCO), and panoptic segmentation (55.7 PQ on COCO).[1]

1. Introduction

Recently proposed image segmentation methods are basically built on neural networks, including convolutional neural networks [39] and transformers [17, 58], and generate segmentation masks directly from per-pixel features. However, classical image segmentation methods often use a hierarchical approaches (*e.g*., hierarchical clustering) [13, 18, 48, 56, 68], and the hierarchical strategy improves segmentation accuracy and computational efficiency. Nonetheless, the neural network-based approaches do not adopt the hierarchical approach, complicating the architecture design and degrading the interpretability. Therefore, we investigate simpler, more interpretable pipelines for image segmentation from the hierarchical clustering perspective.

In general, segmentation models using neural networks consist of three parts: (i) a backbone model that extracts meaningful features from raw pixels, (ii) a pixel decoder
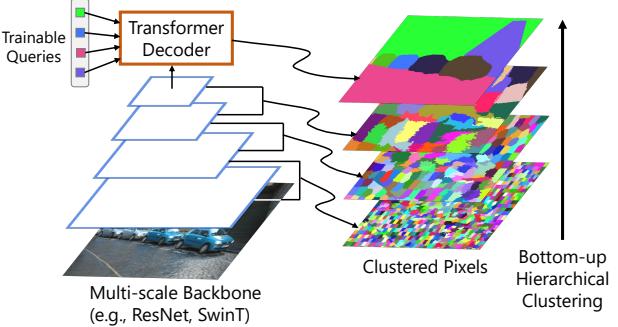

Figure 1. Overview of HCFormer. HCFormer hierarchically groups pixels at downsampling layers in the backbone and then groups clusters obtained from the backbone into an arbitrary number of clusters in the segmentation head.

that recovers the spatial resolution lost in the backbone, and (iii) a segmentation head that generates segmentation masks by a classifier or a cross-attention (or cross-correlation) between queries (or kernels) and feature maps. The pixel decoder is, unlike the others, an inherently unnecessary module to approach image segmentation; in fact, some methods do not adopt it [7, 39, 72]. Nonetheless, the pixel decoder is used in many segmentation models because the models require per-pixel features to generate high-resolution masks. The problem is that the decoding is performed in the high-dimensional feature space, and the high dimensionality makes it difficult to interpret, visualize, and evaluate the decoding process, which is one of the causes of decreasing the interpretability of the segmentation models.

We build the segmentation model based on the hierarchical clustering strategy. Clustering for image segmentation is to group pixels that have the same semantics or ground-truth labels. We can view it as a special case of downsampling if clustering assigns a representative value (*e.g*., a class label or a feature vector representing the cluster) to each cluster. In particular, if elements are grouped based on a fixed window and a representative value is sampled from each group, it is the same as a downsampling scheme. From this perspective, we accomplish image segmentation without the pixel decoder before the segmentation head. If we

---

[1]The code will be publicly available.

1

build the backbone model so that its downsampling layers group pixels and sample representative values, the image segmentation can be accomplished as hierarchical clustering, as shown in Fig. 1. This framework simplifies the architecture design and allows us to interpret, visualize, and evaluate intermediate results as clustering results. Allowing for visualization and evaluation enhances segmentation models' interpretability, that is, the degree to which a human can understand the cause of a decision [43].

To accomplish the hierarchical clustering in deep neural networks, we propose a clustering module using the attention function [58] as an assignment solver and a model using this module, called *HCFormer*. HCFormer groups pixels at downsampling layers by the clustering module and realizes image segmentation by hierarchical clustering, as shown in Fig. 1. We attentively design the clustering module to be easily combined with existing backbone models (*e.g.*, ResNet [20] and Swin Transformer [38]). As a result, the hierarchical clustering scheme can be incorporated into the existing backbone models without a change in their feed-forward path.

HCFormer generates segmentation masks by a matrix multiplication between assignment matrices, and the accuracy of this decoding process can be evaluated by some metrics for assessing pixel clustering methods, such as superpixel segmentation [51]. These properties enable an error analysis and provide some architecture-level insight for improving segmentation accuracy, though one may not comprehend why certain decisions or predictions have been made. For example, when an error occurs at a certain clustering level in HCFormer, at least we know the cause exists in layers before the corresponding downsampling layer in the backbone. Thus, we may be able to resolve an error by adding layers or modules to the relevant stage in the backbone. In contrast, it is difficult for the conventional models to evaluate the decoding accuracy because the pixel decoder upsamples pixels in high-dimensional feature space, and the intermediate results are not comparable to the ground-truth labels. We believe our hierarchical clustering takes the interpretability of segmentation models one step forward, even if it is not a big step.

Since clustering is a common approach for various image segmentation tasks, it can approach many segmentation tasks in the same architecture. Thus, we evaluate HCFormer on three major segmentation tasks: semantic segmentation (ADE20K [73] and Cityscapes [12]), instance segmentation (COCO [37]), and panoptic segmentation (COCO [37]). HCFormer demonstrates comparable or better segmentation accuracy compared to the recently proposed unified segmentation models (*e.g.*, Mask-Former [11], Mask2Former [10], and K-Net [71]) and specialized models for each task, such as Mask R-CNN [19], SOLOv2 [61], SegFormer [63], CMT-Deeplab [70], and

Panoptic FCN [32].

2. Method

We realize hierarchical clustering in deep neural networks by providing a clustering property for downsampling layers. We may be able to do so by clustering pixels and then sampling representative values from obtained clusters instead of conventional downsampling. However, the obtained clusters often do not form regular grid structures, and CNN-based backbones do not allow such irregular grid data as input. Therefore, a straightforward approach, such as downsampling after clustering, is not applicable.

To incorporate the clustering process into existing backbone models while preserving data structures, we propose a *clustering-after-downsampling* strategy. We show our clustering and decoding pipelines in Fig. 2. We assume the downsampling used in existing backbone models is cluster-prototype sampling. Accordingly, we view the pixels in the feature map after downsampling as cluster prototypes and group pixels in the feature map before downsampling. We first show this clustering process can be realized by the attention [58] in Sec. 2.1, and then, we formulate the attention-based clustering module in Sec. 2.2. Finally, we describe our decoding procedure in Sec. 2.3.

2.1. Clustering as Attention

We view the attention function [58] from the clustering perspective. Let $q \in \mathbb{R}^{C \times N_q}$ and $k \in \mathbb{R}^{C \times N_k}$ be a query and a key. N_q and N_k are the number of tokens for the query and key, and C denotes a feature dimension. Then, the attention is defined as follows:

$$\text{Attention}(q, k; s) = \text{Softmax}_{\text{row}}(q^\top k / s), \qquad (1)$$

where $\top$ denotes the transpose of a matrix and s denotes a scale parameter that is usually defined as $\sqrt{C}$ [17, 58]. $\text{Softmax}_{\text{row}}(\cdot)$ denotes the row-wise softmax function. The attention function is generally defined with a query, a key, and a value, but the value is omitted here for simplicity.

When $s \to 0$, eq. (1) is equivalent to the following maximization problem:

$$\operatorname*{arg\,max}_{A \in \{0,1\}^{N_q \times N_k}} \langle A, q^\top k \rangle, \ s.t., \ \sum_m A_{nm} = 1, \qquad (2)$$

where $\langle \cdot, \cdot \rangle$ denotes the Frobenius inner product. We provide the detailed derivation in Appendix A. This maximization problem is interpreted as the clustering problem for q with k as cluster prototypes. With an inner product as a similarity function, $A_{nm} = 1$ if n-th query, q_n, has the maximum similarity to m-th key, k_m, among all key tokens; otherwise, $A_{nm} = 0$. In other words, each row of A indicates the index of the cluster assigned to q_n, known as the assignment matrix. Thus, eq. (2) is an assignment problem, which

is a special case or part of the clustering (*e.g.*, agglomerative hierarchical clustering with a hierarchical level of one and one-step K-means clustering), and the attention in eq. (1) solves the relaxed problem of eq. (2). This insight indicates that we can accomplish clustering after downsampling using the attention in a differentiable form by corresponding the pixels in the feature maps before and after downsampling to queries and keys, respectively.

2.2. Image Segmentation by Hierarchical Clustering

We show the computational scheme of the proposed clustering module in Fig. 2. We obtain an intermediate feature map and its downsampled feature map from a backbone model for clustering. These are fed into layer normalization [2] and a convolution layer with a kernel size of 1×1, as in the transformer block [17, 58]. We define the obtained feature maps as $F^{(i)} \in \mathbb{R}^{C_F \times N^{(i)}}$ and $F_d^{(i)} \in \mathbb{R}^{C_F \times N_d^{(i)}}$, where $i \in \mathbb{N}$ denotes a downsampling factor of the spatial resolution that corresponds to the number of applied downsampling layers, and $N^{(i)}$ and $N_d^{(i)}$ denote the number of pixels in the feature maps before and after downsampling (*i.e.*, $N^{(i)} = \frac{H}{2^i}\frac{W}{2^i}$ and $N_d^{(i)} = \frac{H}{2^{i+1}}\frac{W}{2^{i+1}}$, where H and W are the height and width of an input image). C_F is the number of channels set to 128 in our experiment. Note that we assume that the downsampling halves height and width respectively, and the ℓ_2-norm of the feature vector of each pixel is normalized as 1 to make the inner product the cosine similarity.

Then, the proposed clustering is defined as follows:

$$A^{(i)} = \text{Clustering}(F^{(i)}, F_d^{(i)}; s^{(i)})$$
$$= \text{Softmax}_{\text{row}}((F^{(i)})^\top F_d^{(i)}/|s^{(i)}|), \quad (3)$$

where $A^{(i)} \in (0, 1)^{N^{(i)} \times N_d^{(i)}}$ is an assignment matrix. We define a scale parameter, $s^{(i)} \in \mathbb{R}$, as a trainable parameter. As already described in Sec. 2.1, when $s^{(i)} \to 0$, eq. (3) corresponds to the clustering problem for $F^{(i)}$ with $F_d^{(i)}$ as cluster prototypes. Thus, by computing eq. (3) at every downsampling layer, HCformer hierarchically groups pixels, as shown in Fig. 1.

The obtained feature map from a backbone model is fed into the segmentation head used in [11] (Fig. 4). Specifically, the feature map is fed into the transformer decoder with trainable queries $Q \in \mathbb{R}^{C_q \times N_m}$, and the mask queries $\mathcal{E}_{\text{mask}} \in \mathbb{R}^{C_m \times N_m}$ are generated. The number of queries, N_m, is a hyperparameter. Note that the transformer "decoder" differs from the pixel decoder because one of its roles is to generate the mask queries, not to upsample the feature map. The feature map is also mapped into the C_m-dimensional space by a linear layer, and we define them as $\mathcal{E}_{\text{feature}}^{(i=5)} \in \mathbb{R}^{C_m \times N^{(i)}}$. Then, the output of the segmentation
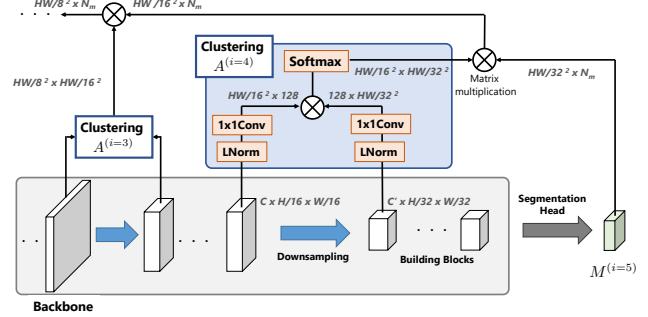

Figure 2. The computational scheme of our clustering module. We can view our clustering module as a variant of the attention module. LNorm and 1×1Conv denote layer normalization [2] and a convolution layer with a kernel size of 1×1.

head is computed as follows:

$$M^{(i=5)} = \text{Sigmoid}((\mathcal{E}_{\text{feature}}^{(i=5)})^\top \cdot \mathcal{E}_{\text{mask}}). \quad (4)$$

Note that we assume that the scale i is 5 because the conventional backbone has five downsampling layers. This segmentation head can be viewed as the clustering, which groups $N^{(i)}$ pixels in the feature map into N_m clusters.[2] Therefore, we also refer to $M^{(i=5)}$ as the assignment matrix. Unlike the previous work [11], we feed a low-resolution feature map into the segmentation head (eq. (4)), which contributes reduction of FLOPs in the segmentation head.

2.3. Decoding

As an output of HCFormer, we obtain the assignment matrices $\{A^{(i)}\}_i$ obtained from the backbone and the output of the segmentation head $M^{(5)}$. We need to decode them as segmentation masks of an input image for evaluation.

One step of the decoding process is defined as a matrix multiplication between $A^{(i)}$ and $M^{(i+1)}$:

$$M^{(i)} = A^{(i)} M^{(i+1)}$$
$$= \text{Softmax}_{\text{row}}\left((F^{(i)})^\top F_d^{(i)}/|s^{(i)}|\right) M^{(i+1)}. \quad (5)$$

By writing down the calculations in $A^{(i)}$ explicitly, we can notice that it is equivalent to the attention function [58], although queries, keys, and values are obtained from different layers. The segmentation mask (*i.e.*, the correspondence between pixels in an input image and the mask queries) is computed by multiplying all $\{A^{(i)}\}_i$ by $M^{(5)}$ as $\prod_i A^{(i)} M^{(5)}$.

---

[2]Eq. (4) uses the sigmoid function, not the softmax function. Thus, eq. (4) does not correspond to eq. (2). However, in the post-processing, the mask with the maximum confidence is selected as the prediction, meaning that the pipeline of the segmentation head, including the post-processing, corresponds to eq. (2). Further details can be found in Appendix E.
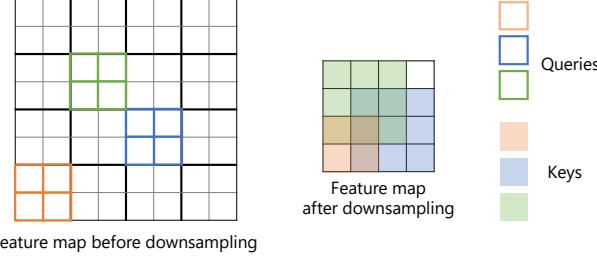
Figure 3. Illustration of the local attention. Each color indicates the correspondence between queries and keys.

From the hard clustering perspective, eq. (2), we can view multiplying $A^{(i)}$ as copying the cluster's representative value to its elements. Thus, this decoding process will be accurate if each cluster is composed of pixels that are annotated with the same ground-truth label. In other words, we can evaluate the accuracy of the decoding from the hard clustering perspective by the undersegmentation error [51] that is used for assessing superpixel segmentation.
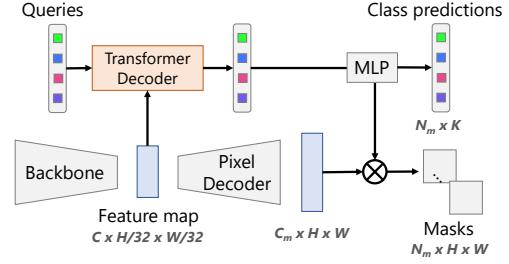
2.4. Efficient Computation

Let N be the number of pixels, and then the computational costs of the proposed clustering are $O(N^2)$, which is the same complexity as the common attention function [58] and is intractable for high-resolution images. Various studies exist on reducing complexity [4, 38, 47, 50, 57, 60, 65], and we approach it with the local attention strategy.

We divide the feature map before downsampling, $F^{(i)}$, into 2×2 windows. The window size is determined by a stride of downsampling, which is assumed to be 2 in this work. Each window corresponds to the pixel in the feature map after downsampling, $F_d^{(i)}$. Then, the attention is computed between a pixel in the window and the corresponding pixel in $F_d^{(i)}$ and its surrounding eight pixels, which is the same technique used in superpixel segmentation [1,23]. We illustrate this local attention in Fig. 3. As a result, the complexity decreases to $O(N)$. The clustering and decoding processes with this local attention can be easily implemented using deep learning frameworks. We show an example PyTorch [45] implementation in Appendix B.
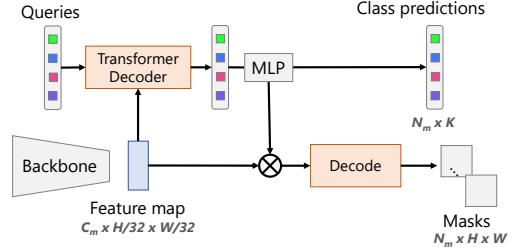
2.5. Architecture

We show a baseline architecture, MaskFormer [11], and our model using the proposed clustering module in Fig. 4. The proposed model removes the pixel decoder from the baseline architecture, and instead, the decoding block, eq. (5), is integrated for obtaining segmentation masks. Optionally, we build a six-layer transformer encoder after the backbone, as in MaskFormer [11]. The mask queries are fed into multi-layer perceptrons and classified into K classes.

The flexibility of the downsampling is important for our

(a) MaskFormer

(b) HCFormer

Figure 4. Comparison between MaskFormer [11] and HCFormer. MaskFormer upsamples a feature map and then predicts masks. In contrast, HCFormer hierarchically groups pixels in every downsampling layer and the segmentation head and then generates segmentation masks by the decoding defined in eq. (5).

clustering module because the downsampled pixels are the cluster prototypes that should have representative values of clusters. However, unlike transformer-based backbones, the receptive field of CNN-based backbones is limited even if the deeper model is used [41, 63]. Thus, the CNN-based backbones may not sample effective pixels for the clustering module. To alleviate this problem, we replace downsampling layers to which the proposed clustering module is attached with DCNv2 [74] with a stride of two, which deforms kernel shapes and modulates weight values. This replacement is only adopted for a CNN-based backbone. We verify its effect in Appendix F.2.

3. Related Work

3.1. Image Segmentation with Deep Neural Networks

Since FCNs [39] have been proposed, deep neural networks have become a de facto standard approach for image segmentation. As main segmentation tasks, semantic, instance, and panoptic segmentation are studied.

Semantic segmentation is often defined as per-pixel classification tasks, and various FCN-based methods have been proposed [3, 6–8, 63, 72]. Instance segmentation has the object detection perspective. Thus, many methods [14, 15, 19, 31, 34] use an object proposal module as the task-specific module, which is used for instance discrimina-

tion. Panoptic segmentation has been proposed in [26], which is a task combining semantic and instance segmentation. In early work, panoptic segmentation is approached with two separated modules for generating semantic and instance masks and then fusing them [9, 25, 26, 64]. To simplify the framework, Panoptic FCN [32] unifies the modules by using dynamic kernels. Panoptic FCN generates kernels from a feature map and produces segmentation masks by cross-correlation between the kernels and feature maps. As a similar approach, cross-attention with the transformer decoder is adopted in other panoptic segmentation methods [5, 10, 11, 33, 59, 70]. Such approaches also simplify the panoptic segmentation pipeline.

While many studies investigate task-specific modules and architectures, they cannot be applied to other tasks. Thus, recent work investigates the unified architectures [10, 11, 71]. These methods use cross-attention-based approaches that can generate segmentation masks regardless of the task definition. They have demonstrated effectiveness in the various segmentation tasks and achieved comparable or better results than the specialized models. Our study also focuses on such unified architectures and builds our model based on MaskFormer [11]. Note that, as seen in eq. (4), the segmentation head used in the unified models can be viewed as a clustering module; namely, the unified models would also be clustering-based frameworks. However, they predict segmentation masks directly from per-pixel features (*i.e.*, not a hierarchical clustering method), and these methods do not allow us to visualize and evaluate the intermediate results.

Several studies define the segmentation tasks as a clustering problem [16, 24, 27, 28, 44, 70], especially in the proposal-free instance segmentation methods. The main focus of these methods is to learn an effective representation for clustering, and their approach is not hierarchical.

Many studies on image segmentation focus on the segmentation head or the pixel decoder. In particular, various decoder architectures have been proposed, thereby improving segmentation accuracy [3, 8, 22, 30, 36, 46, 49, 62]. The decoder would make the segmentation problem complex: models using the decoder have to solve both the upsampling and pixel classification problems internally. Such models suffer from an error in the upsampling process of the pixel decoder. But, it is difficult to know whether the prediction error is due to upsampling or classification because upsampling is performed in the high-dimensional feature space.

Our decoding process can be viewed as an attention-based decoder, as shown in eq. (5), and there are several methods using the attention or similar modules for the pixel decoder [22, 29, 30, 35, 66]. However, existing methods do not have a clustering perspective and do not allow us to interpret, visualize, and evaluate the attention map as the clustering results. In contrast, since HCFormer is built on the

clustering perspective, we can interpret and visualize intermediate results as the clustering results and evaluate their accuracy by comparing ground-truth labels.

There are some methods that do not use the pixel decoder. For example, FCN-32s [39], the simplest variant of FCNs, does not use the trainable decoder, although the generated masks are low-resolution and the segmentation accuracy is somewhat low. Several methods [6, 7, 69, 72] use dilated convolution [6, 69] to keep the resolution in the backbone, which expands the convolution kernel by inserting holes between its consecutive elements and replacing the stride with the dilation rate. The methods using dilated convolution improve the segmentation accuracy at the expense of FLOPs.

3.2. Hierarchical Clustering

Hierarchical clustering approaches are sometimes used to solve image segmentation. For example, some previous methods [13, 18, 21, 48, 53–56, 67, 68] group pixels into small segments using superpixel segmentation [1, 23, 52] as pre-processing; the obtained segments are then merged or classified to obtain the desired cluster. Such a hierarchical approach often reduces computational costs and improves segmentation accuracy compared to directly clustering or classifying pixels.

A recently proposed method, called GroupViT [66], is a bottom-up hierarchical clustering method. However, GroupViT is specialized in vision transformers [17] and semantic segmentation with text supervision. In contrast, our method can be incorporated into almost all multi-scale backbone models, such as ResNet [20] and Swin Transformer [38], and used for arbitrary image segmentation tasks.

4. Experiments

4.1. Implementation Details

We implement our model on the Mask2Former author's implementation.[3] We use the transformer decoder with the masked attention [10], and the number of decoder layers is set to 8. Note that we do not use multi-scale feature maps in the transformer decoder because HCFormer does not have the pixel decoder. Thus, HCFormer is built on top of MaskFormer rather than on top of Mask2Former since the transformer decoder used in HCFormer and MaskFormer is independent of the pixel decoder. We describe the detailed difference between transformer decoders of HCFormer and Mask2Former in Appendix C.

We train HCFormer with almost the same protocol as in Mask2Former [10]; we use the binary cross-entropy loss and the dice loss [42] as the mask loss: $\mathcal{L}_{\text{mask}} = \lambda_{\text{ce}}\mathcal{L}_{\text{ce}} +$

---

[3]https : / / github . com / facebookresearch / Mask2Former

method	backbone	PQ	PQ^{Th}	PQ^{St}	AP^{Th}_{pan}	$mIoU_{pan}$	#params.	FLOPS
Panoptic FCN [32]	R50	44.3	50.0	35.6	-	-	-	-
MaskFormer [11]	R50	46.5	51.0	39.8	33.0	57.8	45M	181G
K-Net [71]	R50	47.1	51.7	40.3	-	-	37M	-
Mask2Former [10]	R50	51.9	57.7	43.0	41.7	61.7	44M	226G
HCFormer	R50	47.7	51.9	41.2	35.7	59.8	38M	87G
HCFormer+	R50	50.2	55.1	42.8	38.4	60.2	46M	96G
MaskFormer [11]	Swin-S	49.7	54.4	42.6	36.1	61.3	63M	259G
Mask2Former [11]	Swin-S	54.6	60.6	45.7	44.7	64.2	69M	313G
HCFormer	Swin-S	50.9	55.7	43.6	38.9	63.1	62M	170G
HCFormer+	Swin-S	53.0	58.1	45.3	41.2	64.4	70M	183G
CMT-Deeplab [70]	Axial-R104-RFN	55.3	61.0	46.6	-	-	270M	1114G
MaskFormer [11]	Swin-L	52.7	58.5	44.0	40.1	64.8	212M	792G
K-Net [71]	Swin-L	54.6	60.2	46.0	-	-	-	-
Mask2Former [10]	Swin-L	57.8	64.2	48.1	48.6	67.4	216M	868G
HCFormer	Swin-L	55.1	60.7	46.7	44.3	66.2	210M	715G
HCFormer+	Swin-L	55.7	62.0	46.1	45.3	66.4	217M	725G

Table 1. Evaluation results for panoptic segmentation with COCO `val`. HCFormer+ stacks a six-layer transformer encoder after the backbone, as in MaskFormer [11].

method	backbone	AP	AP^{S}	AP^{M}	AP^{L}	$AP^{boundary}$	#params.	FLOPs
Mask R-CNN [19]	R50	37.2	18.6	39.5	53.3	23.1	44M	201G
SOLOv2 [61]	R50	38.8	16.5	41.7	56.2	-	-	-
MaskFormer [11]	R50	34.0	16.4	37.8	54.2	23.0	45M	181G
Mask2Former [10]	R50	43.7	23.4	47.2	64.8	30.6	44M	226G
HCFormer	R50	37.4	16.7	40.0	59.7	24.6	38M	87G
HCFormer+	R50	40.1	18.8	43.0	62.3	26.9	46M	96G
Mask2Former [10]	Swin-L	50.1	29.9	53.9	72.1	36.2	216M	868G
HCFormer	Swin-L	46.4	24.1	50.5	70.9	32.6	210M	715G
HCFormer+	Swin-L	47.1	25.6	51.5	70.3	33.3	217M	725G

Table 2. Evaluation results for instance segmentation with COCO `val`. HCFormer+ stacks a six-layer transformer encoder after the backbone.

$\lambda_{dice}\mathcal{L}_{dice}$. The training loss combines mask loss, classification loss, and an additional regularization term for $s^{(i)}$ in eq. (3): $\mathcal{L}_{mask} + \lambda_{cls}\mathcal{L}_{cls} + \lambda_{reg}\mathcal{L}_{reg}$, where $\mathcal{L}_{cls}$ is a cross-entropy loss and $\mathcal{L}_{reg} = \sum_i |s^{(i)}|$. λ_{reg} is set to 0.1. The regularization enforces the proposed attention-based clustering, eq. (3), to be the hard clustering, eq. (2). Following [10], we set $\lambda_{ce} = 5.0$, $\lambda_{dice} = 5.0$, and $\lambda_{cls} = 2.0$. Another training protocol is the same as for Mask2Former [10] (*e.g.*, optimizer, its hyperparameters, and the number of training iterations). The details can be found in Appendix D.

We use the same post-processing as in [11]: we multiply class confidence and mask confidence and use the output as the confidence score. Then, the mask with the maximum confidence score is selected as the predicted mask. The details can be found in Appendix E.

The clustering module defined in eq. (3) is incorporated into every downsampling layer, except for those in the stem block of ResNet [20] and the patch embedding layer in Swin

Transformer [38]. Thus, the hierarchical level is 3 (*i.e.*, $\{A^{(2)}, A^{(3)}, A^{(4)}\}$ are computed). The relation between the hierarchical level and downsampling layers using the clustering module is shown in Fig. II in the appendix.

4.2. Main Results

As baseline methods, we choose the recently proposed methods for unifying segmentation tasks, Mask-Former [11], K-Net [71], and Mask2Former [10], and specialized models for each task [19, 32, 61, 63, 70]. We compare our method to the baselines with several backbones, ResNet-50 [20], Swin-S [38], and Swin-L [38]. Note that ResNet-50 and Swin-S were pretrained with ImageNet-1K, and Swin-L was pretrained with ImageNet-22K. The training procedure and evaluation metrics are following [10]. We trained models three times and reported medians.

We first show the evaluation results for panoptic segmentation on the COCO validation dataset [37] in Tab.

1. HCFormer outperforms the unified architectures, Mask-Former and K-Net, and Panoptic FCN for all metrics with fewer parameters. The additional transformer encoder (*i.e.*, HCFormer+) boosts the segmentation accuracy for the relatively small backbone models and outperforms CMT-Deeplab with fewer parameters and FLOPs. Although the transformer decoder used in HCFormer is different from that used in MaskFormer in this comparison, we verify HCFormer still outperforms MaskFormer even when MaskFormer's transformer decoder is applied (see Appendix F.3).

Mask2Former shows the best accuracy, though it sacrifices the FLOPs. According to [10], using multi-scale features in the transformer decoder improves 1.7 PQ for Mask2Former with the ResNet-50 backbone (*i.e.*, PQ of Mask2Former without the multi-scale feature maps is 50.2, which is the same as PQ of HCFormer+ with the ResNet-50 backbone in Tab. 1). Thus, we believe the gap in PQ between HCFormer and Mask2Former is due to the design of the transformer. In other words, PQ of HCFormer is comparable to that of Mask2Former in the same setting, but HCFormer shows lower FLOPs and has the interpretability. The detailed analysis is described in Appendix C.

Tab. 2 shows the average precision on instance segmentation tasks. HCFormer also outperforms MaskFormer and the specialized model, Mask R-CNN and SOLOv2. Notably, HCFormer significantly improves AP^L from MaskFormer, which indicates HCFormer can generate well-aligned masks for large objects. For recovering the large objects by the pixel decoder, the model needs to capture the long-range dependence, which may be difficult for the simple pixel decoder. Thus, the conventional pipelines need a well-design decoder to recover the large objects, such as that used in Mask2Former. However, since HCFormer groups pixels locally and hierarchically, it may not need long-range dependence to capture the large objects, compared to the conventional methods. As a result, HCFormer outperforms MaskFormer, especially in terms of AP^L.

Tabs. 3 and 4 show the results on semantic segmentation. On ADE20k, HCFormer outperforms MaskFormer and the specialized model, SegFormer, and Mask2Former shows the best mIoU. However, on Cityscapes [12] (Tab. 4), mIoU of HCFormer is worse than that of SegFormer, although HCFormer improves mIoU from MaskFormer. Unlike COCO and ADE20K, the Cityscapes dataset contains only urban street scenes, and there are many thin or small objects, such as poles, signs, and traffic lights. MaskFormer and HCFormer do not have specialized modules to capture such small and thin objects; hence such objects would be missed in an intermediate layer.

For further analysis, we visualize intermediate clustering results and a predicted mask using HCFormer with Swin-L in Fig. 5. From the visualization of undersegmentation er-

method	backbone	crop size	mIoU	FLOPs
MaskFormer [11]	R50	512	44.5	53G
Mask2Former [10]	R50	512	47.2	73G
HCFormer	R50	512	45.5	29G
HCFormer+	R50	512	46.9	32G
SegFormer [63]	MiT-B2	512	46.5	62G
MaskFormer [11]	Swin-S	512	49.8	79G
Mask2Former [10]	Swin-S	512	51.3	98G
HCFormer	Swin-S	512	48.8	56G
HCFormer+	Swin-S	512	50.1	58G
SegFormer [63]	MiT-B5	640	51.0	184G
MaskFormer [11]	Swin-L	640	54.1	375G
Mask2Former [10]	Swin-L	640	56.1	403G
HCFormer	Swin-L	640	55.2	338G
HCFormer+	Swin-L	640	55.5	342G

Table 3. Evaluation results for semantic segmentation with ADE20K `val`. HCFormer+ stacks a six-layer transformer encoder after the backbone.

method	backbone	mIoU	FLOPs
MaskFormer [11]	R50	76.5	405G
Mask2Former [10]	R50	79.4	527G
HCFormer	R50	76.7	196G
HCFormer+	R50	78.3	225G
SegFormer [63]	MiT-B2	81.0	717G
MaskFormer [11]	Swin-S	78.5	599G
Mask2Former [10]	Swin-S	82.6	727G
HCFormer	Swin-S	79.3	398G
HCFormer+	Swin-S	80.0	427G
SegFormer [63]	MiT-B5	82.4	1460G
MaskFormer [11]	Swin-L	81.8	1784G
Mask2Former [10]	Swin-L	83.3	1908G
HCFormer	Swin-L	81.6	1578G
HCFormer+	Swin-L	82.0	1607G

Table 4. Evaluation results for semantic segmentation with Cityscapes `val`. HCFormer+ stacks a six-layer transformer encoder after the backbone.

ror [51], we find that the model groups pixels well except for the boundaries in intermediate clustering, but it misclassifies clusters for the small objects in this image (*e.g.*, poles and persons in the center). HCFormer cannot extract semantically discriminative features for such small objects, although the obtained features are discriminative in terms of clustering. Thus, to solve errors for the small objects, we may improve the transformer decoder or stack more layers on the backbone for the coarsest scale. We believe that this analysis plays one of the roles in interpretability, and conventional models do not allow for this type of analysis. We present further results and analysis on other datasets in Appendix F.4.
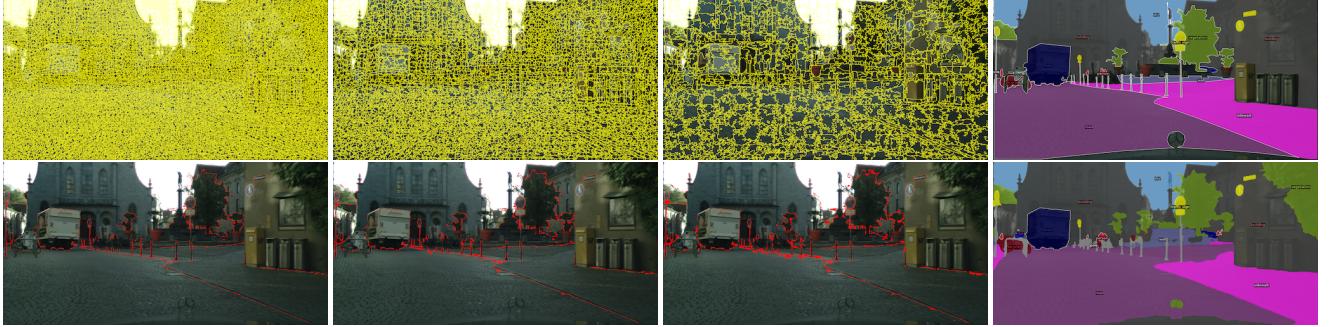
Figure 5. Hierarchical clustering results for Cityscapes. From left to right, the top row shows the cluster boundaries obtained from downsampling layers and ground truth. The bottom row shows undersegmentation errors [51] corresponding to the top row results as red regions, which measure the "leakage" of clusters for ground truth, and a predicted mask.

Hierarchical Level	0	1	2	3
PQ	41.9	45.6	46.7	47.7
FLOPs	82G	83G	84G	87G
#Params	37M	38M	38M	38M

Table 5. Evaluation results for COCO `val.` with various hierarchical levels.

#Decoder Layers	1	2	4	8	16
PQ	42.0	44.8	46.7	47.7	48.0
FLOPs	85G	85G	86G	87G	90G
#Params	27M	29M	32M	38M	51M

Table 6. Evaluation results for COCO `val.` with various numbers of transformer decoder layers.

4.3. Ablation Study

To investigate the effect of the hierarchical level and the more effective architecture, we evaluate PQ, FLOPs, and the number of parameters of our model with various hierarchical levels and different numbers of transformer decoder layers. We use the COCO dataset and the ResNet-50 backbone for evaluation. The additional ablation study can be found in Appendix F.3.

The results are shown in Tabs. 5 and 6. The higher the hierarchical level, the higher the resolution of the predicted masks, increasing PQ and FLOPs. The hierarchical level of 2 would be preferred regarding the balance between accuracy and computational costs in practice, though we adopt the hierarchical level of 3 in Sec. 4.2. In particular, PQ in our model is still comparable to that of MaskFormer [11] even when the hierarchical level is 2.

Our model does not work well with only one decoder layer, as with other models using the transformer decoder [5, 10, 11]. Regarding the balance between accuracy and computational costs, four or eight layers would be suitable for our model.

5. Limitations

HCFormer accomplishes the hierarchical clustering framework by removing the pixel decoder before the segmentation head. However, we do not argue that the pixel decoder is useless, especially in improving segmentation accuracy. Mask2Former uses the multi-scale feature maps obtained from the pixel decoder in the transformer decoder, significantly improving the segmentation accuracy. Also, there are many techniques leveraging the pixel decoder to improve segmentation accuracy [8, 30, 36, 63]. Thus, one of the limitations of HCFormer is that it cannot take advantage of existing techniques using the pixel decoder to improve the segmentation accuracy. This limitation causes the gap between the accuracy of HCFormer and Mask2Former. We will explore alternatives to the modules leveraging the pixel decoder to improve the accuracy in future work.

6. Conclusion

We proposed an attention-based clustering module easily incorporated into existing backbone models, such as ResNet and Swin Transformer. As a result, we accomplished image segmentation via hierarchical clustering in deep neural networks and simplified the segmentation architecture design by removing the pixel decoder before the segmentation head. In experiments, we verified that our method achieves comparable or better accuracy than baseline methods for semantic, instance, and panoptic segmentation. Since our hierarchical clustering allows interpretation, visualization, and evaluation of the intermediate results, we believe that the hierarchical clustering strategy enhances the interpretability of the segmentation models.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE*

transactions on pattern analysis and machine intelligence, 34(11):2274–2282, 2012. 4, 5

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 3

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017. 4, 5

[4] Irwan Bello. Lambdanetworks: Modeling long-range interactions without attention. *arXiv preprint arXiv:2102.08602*, 2021. 4

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 5, 8, 15

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 4, 5, 14

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 1, 4, 5, 14, 16

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018. 4, 5, 8

[9] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12475–12485, 2020. 5

[10] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *CVPR*, 2022. 2, 5, 6, 7, 8, 12, 13, 15

[11] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. volume 34, 2021. 2, 3, 4, 5, 6, 7, 8, 12, 13, 14, 15, 16

[12] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 7

[13] Camille Couprie, Clément Farabet, Laurent Najman, and Yann LeCun. Indoor semantic segmentation using depth information. *arXiv preprint arXiv:1301.3572*, 2013. 1, 5

[14] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3992–4000, 2015. 4

[15] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3150–3158, 2016. 4

[16] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017. 5

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 3, 5

[18] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2012. 1, 5

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2, 4, 6, 13

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 5, 6, 14, 16

[21] Shengfeng He, Rynson WH Lau, Wenxi Liu, Zhe Huang, and Qingxiong Yang. Supercnn: A superpixelwise convolutional neural network for salient object detection. *International journal of computer vision*, 115(3):330–344, 2015. 5

[22] Shihua Huang, Zhichao Lu, Ran Cheng, and Cheng He. Fapn: Feature-aligned pyramid network for dense image prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 864–873, 2021. 5

[23] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 352–368, 2018. 4, 5

[24] Tommi Kerola, Jie Li, Atsushi Kanehira, Yasunori Kudo, Alexis Vallet, and Adrien Gaidon. Hierarchical lovász embeddings for proposal-free panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14413–14423, 2021. 5

[25] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 5

[26] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019. 5

[27] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from

edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5008–5017, 2017. 5

[28] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9018–9028, 2018. 5

[29] Hanchao Li, Pengfei Xiong, Jie An, and Lingxue Wang. Pyramid attention network for semantic segmentation. *arXiv preprint arXiv:1805.10180*, 2018. 5

[30] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maoke Yang, Kuiyuan Yang, Shaohua Tan, and Yunhai Tong. Semantic flow for fast and accurate scene parsing. In *European Conference on Computer Vision*, pages 775–793. Springer, 2020. 5, 8

[31] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2359–2367, 2017. 4

[32] Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Fully convolutional networks for panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 214–223, 2021. 2, 5, 6

[33] Zhiqi Li, Wenhai Wang, Enze Xie, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, Ping Luo, and Tong Lu. Panoptic segformer: Delving deeper into panoptic segmentation with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1280–1289, 2022. 5

[34] Xiaodan Liang, Yunchao Wei, Xiaohui Shen, Zequn Jie, Jiashi Feng, Liang Lin, and Shuicheng Yan. Reversible recursive instance-level object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 633–641, 2016. 4

[35] Huaijia Lin, Xiaojuan Qi, and Jiaya Jia. Agss-vos: Attention guided single-shot video object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3949–3957, 2019. 5

[36] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 5, 8, 13, 14

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 6, 14, 15

[38] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2, 4, 5, 6, 15, 16, 17

[39] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1, 4, 5

[40] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 13

[41] Wenjie Luo, Yujia Li, Raquel Urtasun, and Richard Zemel. Understanding the effective receptive field in deep convolutional neural networks. *Advances in neural information processing systems*, 29, 2016. 4

[42] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016. 5

[43] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020. 2

[44] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019. 5

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 4, 12

[46] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European conference on computer vision*, pages 75–91. Springer, 2016. 5

[47] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention. *arXiv preprint arXiv:2202.08791*, 2022. 4

[48] Xiaofeng Ren and Jitendra Malik. Learning a classification model for segmentation. In *Computer Vision, IEEE International Conference on*, volume 2, pages 10–10. IEEE Computer Society, 2003. 1, 5

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5

[50] Zhuoran Shen, Mingyuan Zhang, Haiyu Zhao, Shuai Yi, and Hongsheng Li. Efficient attention: Attention with linear complexities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3531–3539, 2021. 4

[51] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: An evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, 2018. 2, 4, 7, 8, 16

[52] Teppei Suzuki. Superpixel segmentation via convolutional neural networks with regularized information maximization. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2573–2577. IEEE, 2020. 5

[53] Teppei Suzuki. Implicit integration of superpixel segmentation into fully convolutional networks. *arXiv preprint arXiv:2103.03435*, 2021. 5

[54] Teppei Suzuki, Shuichi Akizuki, Naoki Kato, and Yoshimitsu Aoki. Superpixel convolution for segmentation. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 3249–3253. IEEE, 2018. 5

[55] Shota Takayama, Teppei Suzuki, Yoshimitsu Aoki, Sho Isobe, and Makoto Masuda. Tracking people in dense crowds using supervoxels. In *2016 12th International Conference on Signal-Image Technology Internet-Based Systems (SITIS)*, pages 532–537, 2016. 5

[56] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 1, 5

[57] Ashish Vaswani, Prajit Ramachandran, Aravind Srinivas, Niki Parmar, Blake Hechtman, and Jonathon Shlens. Scaling local self-attention for parameter efficient visual backbones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12894–12904, 2021. 4

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 2, 3, 4

[59] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. MaX-DeepLab: End-to-end panoptic segmentation with mask transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5463–5474, 2021. 5

[60] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021. 4

[61] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic and fast instance segmentation. *Advances in Neural information processing systems*, 33:17721–17732, 2020. 2, 6

[62] Zbigniew Wojna, Vittorio Ferrari, Sergio Guadarrama, Nathan Silberman, Liang-Chieh Chen, Alireza Fathi, and Jasper Uijlings. The devil is in the decoder. In *British Machine Vision Conference 2017, BMVC 2017*, pages 1–13. BMVA Press, 2017. 5

[63] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 4, 6, 7, 8

[64] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019. 5

[65] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A nystöm-based algorithm for approximating self-attention. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 35, page 14138. NIH Public Access, 2021. 4

[66] Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18134–18144, 2022. 5

[67] Fengting Yang, Qian Sun, Hailin Jin, and Zihan Zhou. Superpixel segmentation with fully convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13964–13973, 2020. 5

[68] Jian Yao, Marko Boben, Sanja Fidler, and Raquel Urtasun. Real-time coarse-to-fine topologically preserving segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2947–2955, 2015. 1, 5

[69] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 5, 14

[70] Qihang Yu, Huiyu Wang, Dahun Kim, Siyuan Qiao, Maxwell Collins, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Cmt-deeplab: Clustering mask transformers for panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2560–2570, 2022. 2, 5, 6

[71] Wenwei Zhang, Jiangmiao Pang, Kai Chen, and Chen Change Loy. K-Net: Towards unified image segmentation. *Advances in Neural Information Processing Systems*, 34, 2021. 2, 5, 6, 13

[72] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 1, 4, 5, 14, 16

[73] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 2

[74] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 4, 14

11

A. Detailed derivation of Eq. (2)

Let $v \in \mathbb{R}^n$ be an n-dimensional vector fed into the softmax function, which corresponds to a row of $q^\top k$ in eq. (1), and then i-th output of $\mathrm{softmax}(v)$ with scale s is:

$$a_i = \frac{\exp(v_i/s)}{\sum_j \exp(v_j/s)}. \tag{6}$$

We define $\hat{v} = \max_j v_j$ and modify eq. (6) as follows:

$$\begin{aligned} a_i &= \frac{\exp(v_i/s)}{\sum_j \exp(v_j/s)} \cdot \frac{\exp(-\hat{v}/s)}{\exp(-\hat{v}/s)} \\ &= \frac{\exp((v_i - \hat{v})/s)}{\sum_j \exp((v_j - \hat{v})/s)}, \end{aligned} \tag{7}$$

where $v_i - \hat{v} <= 0$. For numerical calculations, when $s \to 0$, $\exp((v_i - \hat{v})/s) = \exp(-\infty) = 0$ if $v_i \neq \hat{v}$, and $\exp((v_i - \hat{v})/s) = \exp(0/s) = 1$ if $v_i = \hat{v}$. In this case, eq. (6) is equivalent to the following problem:

$$\underset{a \in \{0,1\}^n}{\arg\max} \; a^\top v, \; s.t., \; \sum_i a_i = 1. \tag{8}$$

The solution of this problem is a one-hot vector that holds $a_l = 1$, $l = \arg\max_j v_j$. By considering eq. (2) for each row, eq. (2) is equivalent to eq. (8). Hence, when $s \to 0$, eq. (1) is equivalent to eq. (3).

B. Example PyTorch implementation

We show example PyTorch [45] implementation of the proposed clustering and decoding with the local attention in Listings 1 and 2. We can easily implement the local attention by using the unfold and einsum functions.

Listing 1. The proposed soft clustering (eq. (3))

```python
import torch
import torch.nn.functional as F


def clustering(feat, downsampled_feat, scale):
    batch_size, emb_dim, height, width = feat.shape
    # get 9 candidate clusters and corresponding pixel
        features
    candidate_clusters = F.unfold(downsampled_feat,
        kernel_size=3, padding=1).reshape(batch_size,
        emb_dim, 9, -1)
    feat = F.unfold(feat, kernel_size=2, stride=2).
        reshape(batch_size, emb_dim, 4, -1)
    # calculate similarities
    similarities = torch.einsum('bkcn,bkpn->bcpn', (
        candidate_clusters, feat)).reshape(batch_size*9,
        4, -1)
    similarities = F.fold(similarities, (height, width),
         kernel_size=2, stride=2).reshape(batch_size, 9,
        height, width)
    # normalize
    soft_assignment = (similarities / scale.abs()).
        softmax(1)
    return soft_assignment
```
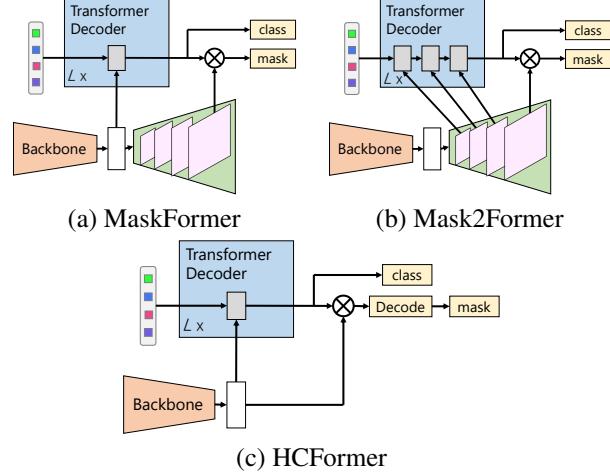

(a) MaskFormer (b) Mask2Former

(c) HCFormer

Figure I. The architecture of MaskFormer [11], Mask2Former [10], and HCFormer.

Listing 2. The decoding (eq. (5))

```python
import torch
import torch.nn.functional as F


def decode(x, A):
    batch_size, _, height, width = A.shape
    n_channels = x.shape[1]
    # get 9 candidate clusters and corresponding
        assignments
    candidate_clusters = F.unfold(x, kernel_size=3,
        padding=1).reshape(batch_size, n_channels, 9, -1)
    A = F.unfold(A, kernel_size=2, stride=2).reshape(
        batch_size, 9, 4, -1)
    # decoding
    decoded_features = torch.einsum('bkcn,bcpn->bkpn', (
        candidate_clusters, A)).reshape(batch_size,
        n_channels * 4, -1)
    decoded_features = F.fold(decoded_features, (height,
         width), kernel_size=2, stride=2)
    return decoded_features
```

C. Difference between the transformer decoders of HCFormer and Mask2Former

The architectures of MaskFormer [11], Mask2Former [10], and HCFormer are shown in Fig. I. Mask2Former feeds the Multi-scale feature maps obtained from the pixel decoder into the transformer decoder, unlike MaskFormer and HCFormer. The difference between HCFormer and MaskFormer is only the decoding process, but the difference between HCFormer and Mask2Former is, in addition to the decoding process, the inputs to the transformer decoder. Therefore, in terms of the architecture design, we believe HCFormer is built on MaskFormer rather than Mask2Former.

In our experiments, the accuracy of HCFormer is inferior to that of Mask2Former. However, the PQ of HCFormer is comparable to that of Mask2Former without using multi-scale feature maps in the transformer decoder (*i.e.*, the input

	Mask2Former [10]	HCFormer+
PQ	50.2	50.2
FLOPs	213G	96G

Table I. Comparison between Mask2Former *without* using multi-scale feature maps in the transformer decoder and HCFormer+ on the COCO dataset. We use ResNet-50 as the backbone architecture.

	Mask2Former [10]	HCFormer+
PQ	50.7	50.2
FLOPs	226G	96G

Table II. Comparison between Mask2Former with FPN [36] as the pixel decoder and HCFormer+ on the COCO dataset. We use ResNet-50 as the backbone architecture.

to the transformer decoder is the same for both HCFormer and Mask2Former), as shown in Tab. I. This result indicates HCFormer is not inferior to Mask2Former, and the gap of PQ would be due to the transformer decoder design.

In addition, we report PQ of Mask2Former replacing the well-designed pixel decoder with a simple pixel decoder, FPN [36], in Tab. II. FPN is one of the most simple pixel decoders used in MaskFormer [11] and K-Net [71]. This replacement reduces the PQ of Mask2Former from 51.9 to 50.7.

As seen in Tabs. I and II, the pixel decoder contributes the improvement of PQ. There are some effective pixel decoder architectures and techniques leveraging the pixel decoder (*e.g.*, the use of multi-scale feature maps as in Mask2Former) to improve the accuracy. Thus, removing the pixel decoder makes the segmentation pipeline simple and interpretable but may lose some means to improve the segmentation accuracy, which is one of the limitations of HCFormer.

D. Detailed experimental setup

We trained models with the Swin-L backbones for the COCO dataset on $4\times$ NVIDIA A100 GPUs and the other models on $2\times$ NVIDIA RTX8000 GPUs. The training protocol follows Mask2Former [10] except for the number of training epochs for HCFormer+. We describe the details as follows.

D.1. Panoptic and instance segmentation on COCO

For HCFormer with the ResNet-50 and Swin-S backbones, we set the number of training epochs and the trainable queries that are fed into the transformer decoder to 50 and 100, respectively. For HCFormer with the Swin-L backbone, these parameters are set to 100 and 200. For HCFormer+, the number of epochs is set to 100 for the ResNet-50 and Swin-S backbones and 200 for the Swin-L

backbone, and other parameters are the same as HCFormer. We use an initial learning rate of 0.0001 and a weight decay of 0.05 for all backbones. A learning rate multiplier of 0.1 is applied to the backbone, and we decay the learning rate at 0.9 and 0.95 fractions of the total number of training steps by a factor of 10. We use AdamW optimizer [40] with a batch size of 16. We initialize the scale parameter $s^{(i)}$ with 0.1. For data augmentation, we use the same policy as in Mask2Former [10]. For inference, we use the Mask R-CNN inference setting [19], where we resize an image with a shorter side to 800 and a longer side up to 1333.

D.2. Semantic segmentation on AD20K and Cityscapes

We use AdamW [40] and the poly learning rate schedule with an initial learning rate of 0.0001 and a weight decay of 0.05. A learning rate multiplier of 0.1 is applied to the backbones. A batch size is set to 16. We initialize the scale parameter $s^{(i)}$ with 0.1. For data augmentation, we use random scale jittering between 0.5 and 2.0, random horizontal flipping, random cropping, and random color jittering. For the ADE20K dataset, if not stated otherwise, we use a crop size of 512×512 and train HCFormer for 160k iterations and HCFormer+ for 320k iterations. For the Cityscapes dataset, we use a crop size of 512×1024 and train HCFormer for 90k iterations and HCFormer+ for 180k iterations. The number of trainable queries is set to 100 for all models and both datasets.

E. Post-processing for the MaskFormer's segmentation head and the relation to the clustering

Let $P \in \Delta^{N_m \times K+1}$ be the $(K+1)$ class probability for masks computed by eq. (4). For a post-processing, we assign a pixel at $j = [h, w]$ to one of the N_m predicted probability-assignment pairs via $\arg\max_{i:c_i \neq \varnothing} P_{i,c_i} M_{j,i}^{(0)}$, where $\varnothing$ denotes the `no object` class and c_i is the most likely class label, $c_i = \arg\max_{c \in \{1,...,K,\varnothing\}} P_{i,c}$, for each probability-assignment pair i. Note that $M^{(0)}$ denotes the decoded masks (*i.e.*, $M^{(0)} = \prod_i A^{(i)} M^{(5)}$).

The maximization problem in post-processing, $\arg\max_{i:c_i \neq \varnothing} P_{i,c_i} M_{j,i}^{(0)}$, is equivalent to the clustering problem, eq. (2). Specifically, $P_{i,c_i} M_{j,i}$ is a similarity between the i-th mask query and the pixel at $j = [h, w]$, and the maximization problem selects the most similar mask query. This procedure is the clustering problem for pixels with the mask queries as a prototype. Thus, we can view MaskFormer's segmentation head as one of the clustering modules.

F. Additional results

F.1. Per-pixel classification with hierarchical clustering

The per-pixel classification head, which is the segmentation head used in many semantic segmentation models, can also be viewed from the clustering perspective. In this head, the weight of the linear classifier with the softmax activation that produces class probability can be viewed as a set of class prototypes. The linear classifier groups pixels based on the class prototypes with the inner product as the similarity. Specifically, let $w \in \mathbb{R}^{C \times K}$ be a weight matrix of the linear classifier, which corresponds to the K-class prototypes with C-dimensional features, and let $f \in \mathbb{R}^{C \times M}$ be a feature map with M pixels. Then, the per-pixel classification models classify pixels as $\mathrm{Softmax}_{\mathrm{row}}(f^{\top}w)$, which is the same formula as the attention in eq. (1) (the norm of w can be viewed as the inverse of the scale parameter s). Therefore, we can also view this as the soft clustering problem and naturally incorporate our hierarchical clustering scheme into the per-pixel classification models.

We evaluate our method using the per-pixel classification model. As a baseline, we use FCN-32s [36], FPN [36], PSPNet [72], and Deeplabv3 [7], and we combine the proposed module with FCN-32s, PSPNet, and Deeplabv3. We refer to the combined models as HC-FCN-32s, HC-PSPNet, and HC-Deeplabv3, respectively. We use ResNet-101 [20] as the backbone for all models. Note that PSPNet and Deeplabv3 use dilated convolution [6, 69] in the backbone, and their output stride is 8, meaning that the number of downsampling layers in the backbone is 3. HC-PSPNet and HC-Deeplabv3 use the same backbone architecture as FCN-32s; their output stride is 32. The training protocol follows [72]. We set the crop size for ADE20K to 512×512.

We show the evaluation results for the validation set in Tab. III. The latency of HC-PSPNet and HC-Deeplabv3 is lower than that of PSPNet and Deeplabv3, and HC-PSPNet shows a comparable result for PSPNet for both datasets. However, mIoU of HC-Deeplabv3 on Cityscapes is lower than that of Deeplabv3. Deeplabv3 uses atrous spatial pyramid pooling (ASPP) that uses several dilated (atrous) convolution layers with different dilation, and the maximum dilation is 24, which corresponds to the convolution with a kernel size of 49. For HC-Deeplabv3, the resolution of the feature map fed into the ASPP layer would be quite low. As a result, HC-Deeplabv3 degrades mIoU from Deeplabv3.

Compared to FCN-32s, HC-FCN-32s significantly improves mIoU because it can preserve detailed information, such as object boundaries, via the clustering module. In addition, HC-FCN-32s shows a superior mIoU than FPN, that is, FCN-32s with the simple pixel decoder, which is also used in MaskFormer [11], while the latency is higher than FPN. This result is consistent with the results of the comparison between HCFormer and MaskFormer.

Models	mIoU	Pixel Acc.	msec/image
PSPNet	42.7	80.9	48.7
HC-PSPNet	42.6	80.9	19.0
Deeplabv3	42.7	81.0	62.5
HC-Deeplabv3	42.8	81.1	22.5

(a) ADE20K

Method	mIoU	Pixel acc.	msec/image
FCN-32s	71.7	94.9	71.7
FPN	75.1	95.8	82.0
HC-FCN-32s	76.1	96.1	87.2
PSPNet	77.7	96.2	298.7
HC-PSPNet	77.6	96.2	88.1
Deeplabv3	78.4	96.3	380.8
HC-Deeplabv3	77.6	96.3	93.9

(b) Cityscapes

Table III. Evaluation results on semantic segmentation with per-pixel models. The inference time is measured with NVIDIA Quadro RTX8000.

F.2. Effect of downsampling modules

In the experiments, we use the deformable convolution v2 (DCNv2) [74] as the downsampling for the ResNet backbone. We verify its effect by comparing the ResNet backbone with and without DCNv2. We train models using the same training protocol for both models with and without DCNv2 (Sec. D.1).

We show the evaluation results for COCO [37] in Tab. IV. Note that the hierarchical level is shown in Fig. II. Since the downsampling layer is replaced with DCNv2 only for the layers to which the clustering module is attached, both results for the hierarchical level of 0 are the same. DCNv2 significantly improves PQ, especially for hierarchical levels of 2 and 3. The higher the hierarchical level, the more sampling error is accumulated. Thus, the improvement is larger for the higher level.

Note that we do *not* use DCNv2 as downsampling for the transformer-based backbone (Swin-S and Swin-L), and the accuracy of HCFormer with the transformer-based backbone is higher than that of MaskFormer. Thus, the effectiveness of HCFormer is significant, although its accuracy of HCFormer with the CNN-based backbone is almost the same as that of MaskFormer.

F.3. Additional ablation study

We show the additional ablation study. Basically, we use ResNet-50 as a backbone model, except for the result in Tab. VIII, and the training protocol is the same as described in Sec. D.
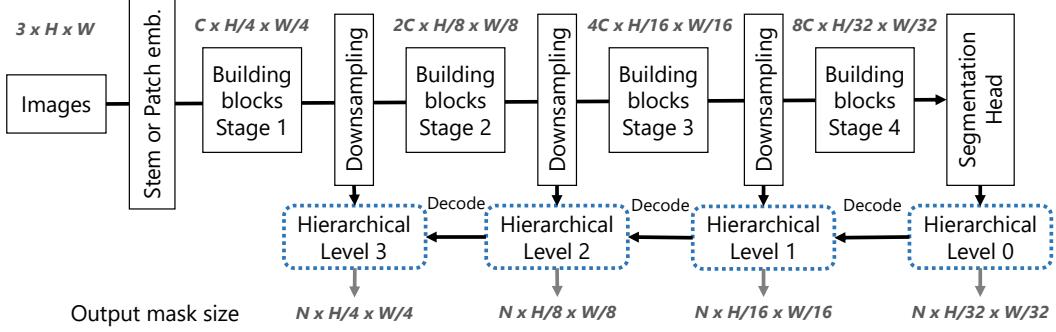
We evaluate HCFormer with the transformer decoder

Figure II. Backbone architecture with hierarchical clustering.

Hierarchical Level	0	1	2	3
PQ	41.9	45.6	46.7	47.7
FLOPs	82G	83G	84G	87G
#Params	37M	38M	38M	38M

(a) w/ DCNv2

Hierarchical Level	0	1	2	3
PQ	41.9	44.0	44.8	45.3
FLOPs	82G	83G	84G	87G
#Params	37M	38M	38M	38M

(b) w/o DCNv2

Table IV. Evaluation results on COCO `val.` with various hierarchical levels.

method	PQ	FLOPs
MaskFormer [11]	46.5	181G
Mask2Former [10]	47.1	213G
HCFormer w/o DCNv2	46.8	97G
HCFormer w/ DCNv2	47.6	97G

Table V. Evaluation results of HCFormer with the standard transformer decoder used in [5, 11]. Note that Mask2Former does not use masked attention but uses multi-scale feature maps in the transformer decoder in this comparison.

#Queries	20	50	100	150	200
COCO (PQ)	42.2	46.3	47.7	47.9	47.9
ADE20K (mIoU)	44.4	45.5	45.5	44.8	44.4
Cityscapes (mIoU)	75.9	75.4	76.7	76.2	76.4

Table VI. Evaluation results with various numbers of trainable queries.

Hierarchical Level	0	1	2	3
mIoU	42.2	43.4	44.0	45.5
FLOPs	28G	28G	28G	29G
#Params	37M	38M	38M	38M

Table VII. Evaluation results on ADE20K `val.` with various hierarchical levels.

Hierarchical Level	0	1	2	3
PQ	47.5	49.7	50.5	50.9
FLOPs	167G	167G	168G	170G
#Params	62M	62M	62M	62M

Table VIII. Evaluation results on COCO `val.` with various hierarchical levels for the Swin-S backbone.

used in MaskFormer [11], although the masked transformer decoder [10] was used in the experiments in the main manuscript. As shown in Tab. V, HCFormer with the standard transformer decoder still outperforms MaskFormer even though the DCNv2 is not used. The difference between HCFormer w/o DCNv2 and MaskFormer in Tab. V is the segmentation process: MaskFormer generates segmentation masks directly from the per-pixel features, and HCFormer generates them based on the hierarchical strategy. This indicates that our hierarchical clustering scheme generates more accurate masks than direct inference from the per-pixel features. We also show the results of Mask2Former without the masked attention in the transformer decoder. PQ of HCFormer without DCNv2 is slightly worse than that of Mask2Former, but we believe that due to the use of multi-scale feature maps, as described in Sec. C.

We evaluate HCFormer using a various number of trainable queries (Tab. VI). The segmentation accuracy is saturated at about 100 or 150. For memory efficiency, 50 or 100 queries would be preferred.

We evaluate HCFormer with the various hierarchical levels on the semantic segmentation (ADE20K). We verify the improvement of the accuracy on the semantic segmentation, as shown in Tab. VII. In addition, we evaluate HCFormer with the Swin-S [38] backbone with various numbers of hierarchical levels on COCO [37]. Regardless of the backbone type and dataset, a higher hierarchical level leads to higher accuracy.
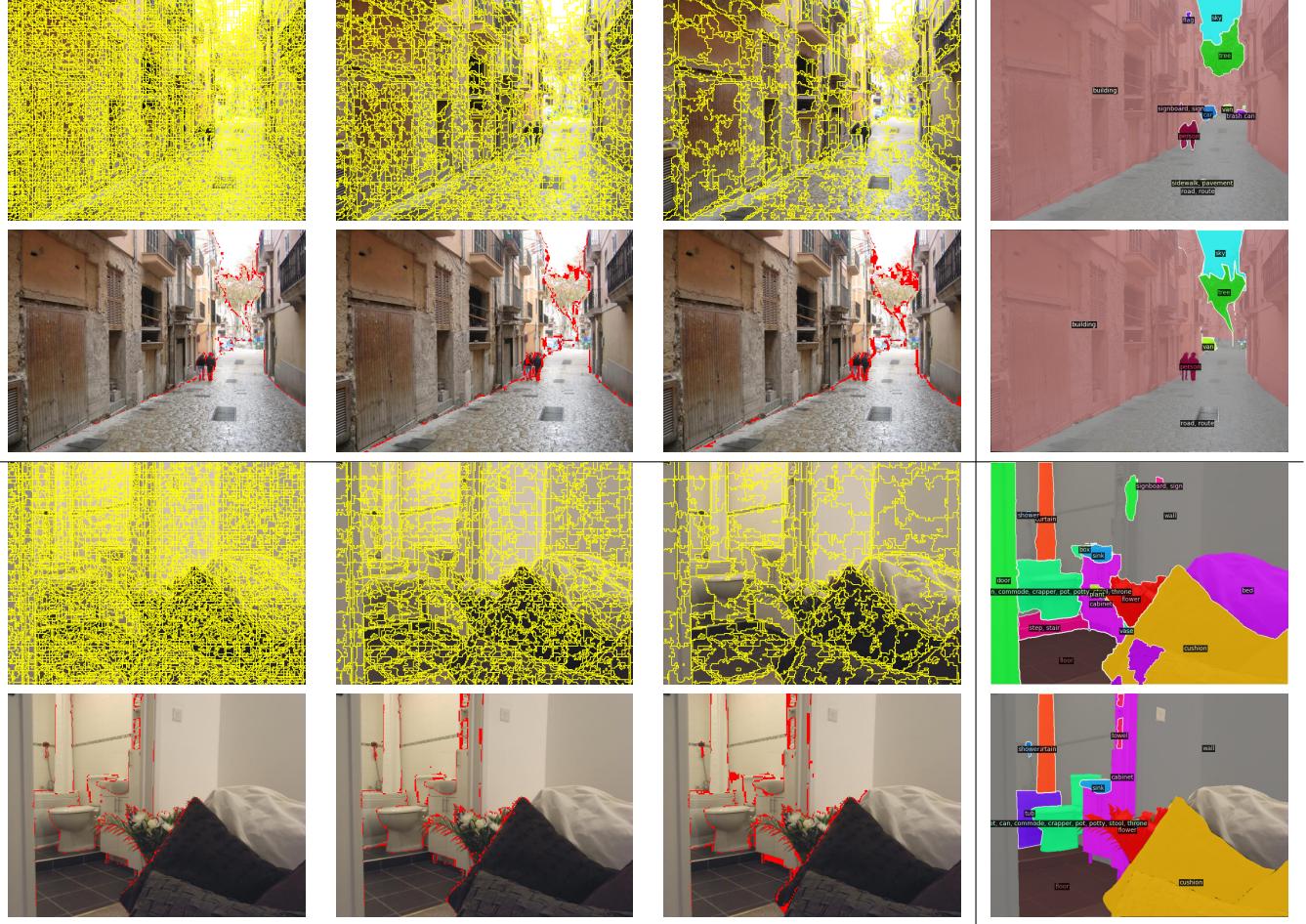
15

Figure III. Example results on ADE20K. The top row shows the cluster boundaries and predicted masks, and the bottom row shows the undersegmentation error [51] as red regions and the ground-truth label.

#Decoder Layers	1	2	4	8	16
mIoU	43.2	44.4	44.5	45.5	46.1
FLOPs	28G	28G	28G	29G	31G
#Params	27M	29M	32M	38M	51M

Table IX. Evaluation results on ADE20K `val.` with various numbers of transformer decoder layers.

We evaluate HCFormer with the ResNet-50 backbone with various numbers of transformer decoder layers on ADE20K. MaskFormer [11] reports reasonable semantic segmentation performance (43.0 mIoU on ADE20K) with only one transformer decoder layer. HCFormer with only one transformer decoder layer also shows reasonable results (43.2 mIoU on ADE20K), as shown in Tab. IX. This result is better than the per-pixel classification baselines with ResNet-101 [20], such as PSPNet [72] and Deeplabv3 [7], as shown in Tab. III. The deeper transformer decoder improves mIoU, and four or eight layers would be sufficient for HCFormer.

F.4. Visualization

We show example results on ADE20K by HCFormer with Swin-L [38] in Fig. III. Basically, the undersegmentation error appears near the boundaries because of ambiguity and downsampling in stem and patch embedding layers of the backbone to which the clustering module is not attached. In the outdoor image (top row), the undersegmentation error appears in the tree region (green region) at the coarser clustering levels, but this would be due to ambiguity in the annotation. In the indoor image (bottom row), the undersegmentation error appears below the cabinet region (pink region) in all the levels because the pixels in the wall and floor regions are grouped. In addition, HCFormer misclassifies the cabinet and the towel as wall and door, respectively. The backbone model would not recognize their semantics, although it was able to discriminate between pixels in the cabinet and pixels in the wall because the undersegmenta-
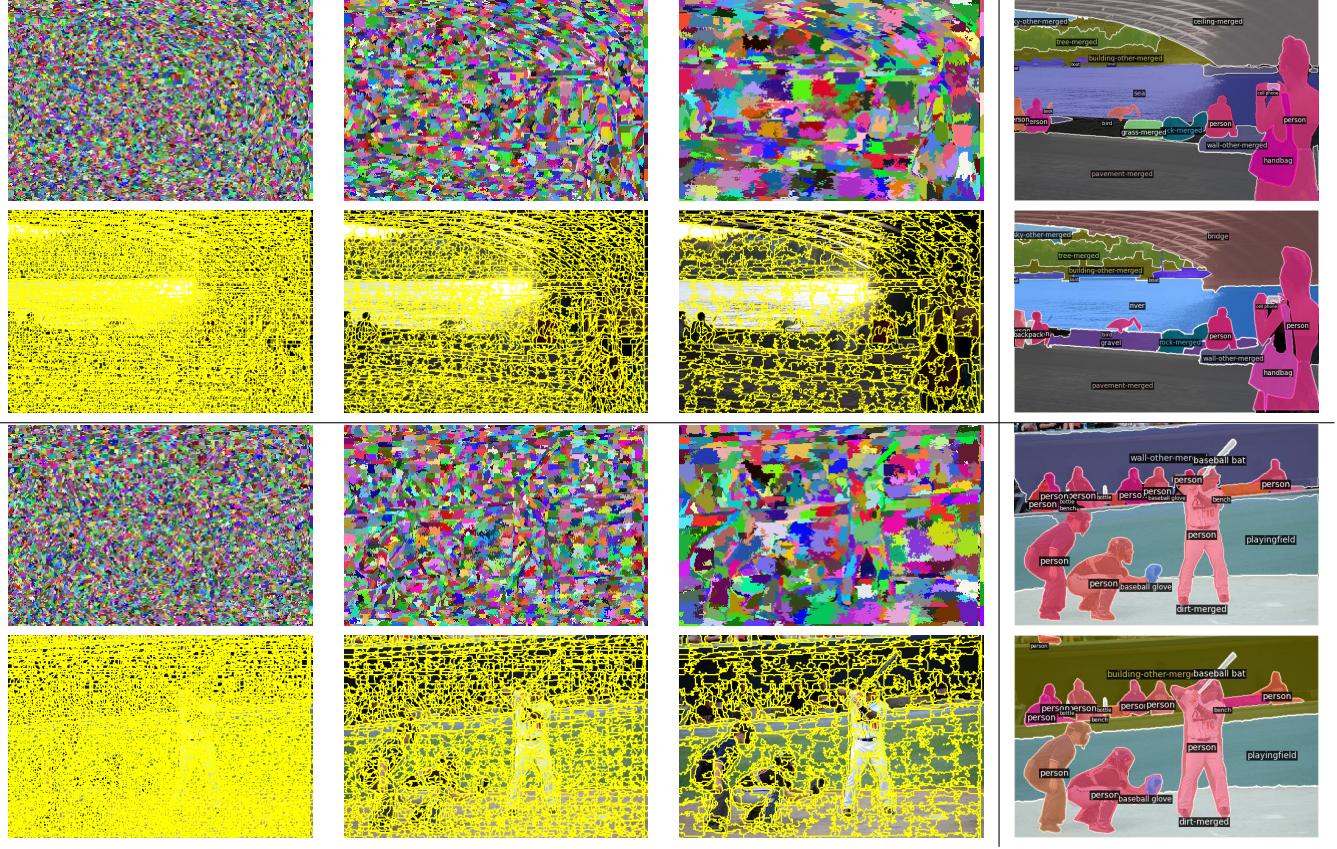
Figure IV. Example results on COCO. The top row shows hierarchical clustering results to which a random color is assigned and prediction, and the bottom row shows cluster boundaries and ground truth.

tion error only appears in their boundaries.

We show example results for COCO panoptic segmentation by HCFormer+ with Swin-L [38] in Fig. IV. HCFormer groups pixels in images well, although there are some misclassifications. To solve such a misclassification error, we might need to improve the transformer decoder and/or incorporate stronger backbones.