Clustering as Attention: Unified Image Segmentation with Hierarchical Clustering

Teppei Suzuki
Denso IT Laboratory, Inc.

Abstract

We propose a hierarchical clustering-based image segmentation scheme for deep neural networks, called HCFormer. We interpret image segmentation, including semantic, instance, and panoptic segmentation, as a pixel clustering problem, and accomplish it by bottom-up, hierarchical clustering with deep neural networks. Our hierarchical clustering removes the pixel decoder before the segmentation head and simplifies the segmentation pipeline, resulting in improved interpretability. 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. Introduction

Image segmentation is a task of clustering pixels; for example, the semantic segmentation groups pixels that have the same semantics, and instance-level segmentation groups pixels in the same instance. However, many methods using deep neural networks define it as per-pixel classification tasks: models for the semantic segmentation classify each pixel into pre-defined categories [36, 68, 7, 60], and models for the instance segmentation classify each pixel in the region of interest into foreground or background [19, 30, 15]. These classification approaches make the model architecture fragmented and task-specific because the classifier depends on the task definition, although recent studies [11, 13, 67] have attempted to design unified architectures that approach various segmentation tasks with the same architecture. Therefore, we revisit the segmentation architecture from the clustering perspective and investigate simpler, more interpretable architectures for unified image segmentation.

In general, segmentation models using neural networks consist of three parts: (i) a backbone model that aggregates spatial information, (ii) a pixel decoder 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. Existing segmentation heads can be interpreted as the pixel clustering modules (see Sec. 2.2), and clustering has the downsampling property. From this perspective, the segmentation pipeline may be somewhat redundant because for downsampling pixels (i.e., clustering in the segmentation head) the decoder upsamples pixels of the feature map already downsampled in the backbone. This insight provides the possibility of removing the pixel decoder before the segmentation head: if downsampling layers in the backbone have the clustering property, we can remove the pixel decoder before the seg-

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.

1Code: https://github.com/DensoITLab/HCFormer
mentation head and achieve image segmentation by hierarchically grouping pixels at every downsampling layer and the segmentation head.

To do so, we propose a clustering module by interpreting the attention function [54] as a clustering solver and also propose a model using the proposed 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 [55]). As a result, the hierarchical clustering scheme can be incorporated into the existing backbone models without a change in their feed forward path.

HCFormer decodes the low-resolution masks after the segmentation head, and the decoding process can be visually interpretable. Moreover, we can evaluate the decoding by some metrics for pixel clustering methods such as superpixel [47]. In contrast, it is difficult for the conventional models to evaluate accuracy of the decoding because the pixel decoder upsamples pixels in high-dimensional feature space and the intermediate results cannot be compared to the ground-truth. The evaluation of the decoding may be useful to resolve an error. In other words, 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.

Since HCFormer uses the segmentation head used in the unified architecture [11], it can approach many segmentation tasks in the same architecture. Thus, we evaluate HCFormer on three major segmentation tasks: semantic segmentation (ADE20K [69] and Cityscapes [12]), instance segmentation (COCO [34]), and panoptic segmentation (COCO [34]). HCFormer demonstrates comparable or better segmentation accuracy compared to the recently proposed unified models (e.g., MaskFormer [11], Mask2Former [10], and K-Net [67]) and specialized models for each task, such as Mask R-CNN [19], SegFormer [60], and Panoptic FCN [31].

2. Method

We realize the hierarchical clustering in deep neural networks by providing a clustering property for downsampling layers. Simply, 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 cannot compute data with irregular grid structures. Therefore, a straightforward approach, such as downsampling after clustering, is not applicable.

To incorporate the clustering procedure into existing backbone models while preserving data structures, we propose a clustering-after-downsampling strategy. We view downsampling used in existing backbone models as 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 [54], and then, we propose HCFormer that segments images by the hierarchical clustering scheme.

2.1. Clustering as Attention

We view the attention function [54] 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:

\[
\text{Attention}(q, k; s) = \text{Softmax}_{k \in \mathbb{R}^{C \times N_k}}(q^\top k/s), \tag{1}
\]

where \( s \) denotes a scale parameter that is usually defined as \( \sqrt{C} [17, 54] \). \( \text{Softmax}_{k \in \mathbb{R}^{C \times N_k}}(\cdot) \) denotes the row-wise softmax function. In general, the attention function is defined with a query, a key, and a value, but the value is omitted here for simplicity.

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

\[
\arg \max_{A \in \{0, 1\}^{N_q \times N_k}} \langle A, q^\top k \rangle, \; s.t., \; \sum_j A_{ij} = 1, \tag{2}
\]

where \( \langle \cdot, \cdot \rangle \) denotes the Frobenius inner product. We provide the detailed derivation in the Appendix. This maximization problem is interpreted as the clustering problem for \( q \) with \( k \) as cluster prototypes. With an inner product as a similarity function, \( A_{ij} = 1 \) if \( q_i \) has the maximum similarity to \( k_j \) among all key tokens; otherwise, \( A_{ij} = 0 \). In other words, each row of \( A \) indicates the cluster index assigned to \( q_i \), which is known as the assignment matrix. Thus, the attention in eq. (1) is a relaxed matrix of \( A \) obtained from eq. (2), and we can view the attention, eq. (1), as the assignment process, which is a special case of the soft clustering.

2.2. Image Segmentation by Hierarchical Clustering

We show the computational scheme of the proposed clustering module in Fig. 2. For clustering, we obtain an intermediate feature map and its downsampled feature map from a backbone model. These are fed into layer normalization [2] and convolution layers with a kernel size of \( 1 \times 1 \), as in the transformer block [17, 54]. We define the obtained feature maps as \( F(i) \in \mathbb{R}^{C_P \times H^i / 2^i \times W^i / 2^i} \) and \( F_d(i) \in \mathbb{R}^{C_P \times H^i / 2^i \times W^i / 2^i} \). \( C_P \) is the output channel of the \( 1 \times 1 \) convolution, which is set to 128 in our experiment; \( H \) and
W are the height and width of an input image; and \( i \in \mathbb{N} \) is a scale factor of the spatial resolution that corresponds to the number of applied downsampling layers. We assume that the downsampling halves height and width respectively, and the \( \ell_2 \)-norm of the feature vector of each pixel is normalized as 1, which gives the inner product a perspective as the cosine similarity.

Then, the proposed clustering is defined as follows:

\[
\text{Clustering}(F(i), F_d(i), s(i)) = \text{Softmax}_\text{row} \left( (F(i)^T F_d(i) / |s(i)|) \right). \tag{3}
\]

We define \( s(i) \in \mathbb{R} \) 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, we can provide the clustering property for the downsampling layers by computing eq. (3) after downsampling and accomplish the hierarchical clustering in deep neural networks, as shown in Fig. 1.

The obtained feature map from a backbone model (hereafter referred to as cluster features) is further clustered by the segmentation head used in [11], and segmentation masks are obtained. The cluster features are 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” is different from the pixel decoder because one of its roles is to generate the mask queries, not to upsample the feature map. The cluster features are also mapped into the \( C_m \)-dimensional space by a linear layer, and we define them as \( \mathcal{E}^{(i=5)}_{\text{cluster}} \in \mathbb{R}^{C_m \times \frac{H}{2} \times \frac{W}{2}} \). Then, the predicted masks are computed as follows:

\[
M^{(i=5)} = \text{Sigmoid}(\mathcal{E}_{\text{mask}}^T \cdot \mathcal{E}^{(i=5)}_{\text{cluster}}). \tag{4}
\]

Note that we assume that the scale \( i \) is 5 because the conventional backbone has five downsampling layers. This segmentation head can also be viewed as the clustering, which groups \( \frac{H}{2} \times \frac{W}{2} \) pixels in the cluster features into \( N_m \) clusters.

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 \( \text{Softmax}_\text{row}(f^T w) \), 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 in the Appendix.

2.3. Decoding

The obtained masks \( M^{(i=5)} \in \mathbb{R}^{N_m \times \frac{H}{2} \times \frac{W}{2}} \) are low-resolution, and each spatial index corresponds not to a pixel but to a cluster generated in the backbone by eq. (3). Thus, we have to decode it based on the clustering results to obtain masks in the input image space. The decoding process is defined as follows:

\[
(M^{(i)})^T = \text{Clustering}(F(i), F_d(i); s(i))(M^{(i+1)})^T = \text{Softmax}_\text{row} \left( (F(i)^T F_d(i) / |s(i)|) (M^{(i+1)})^T \right), \tag{5}
\]

which corresponds to the attention function [2], although queries, keys, and values are obtained from different layers. From the hard clustering perspective, eq. (2) we can view the decoding as an operation of 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.

Unlike a conventional pixel decoder, we can evaluate accuracy of our decoding from the hard clustering perspective by the undersegmentation error [47] that is used for evaluating superpixel segmentation. In addition, the low-resolution feature maps are fed into the segmentation head because our decoding is applied after predicting masks. It contributes reduction of FLOPs in the segmentation head.

Note that we assume that the scale \( i \) is 5 because the conventional backbone has five downsampling layers. This segmentation head can also be viewed as the clustering, which groups \( \frac{H}{2} \times \frac{W}{2} \) pixels in the cluster features into \( N_m \) clusters. 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 \( \text{Softmax}_\text{row}(f^T w) \), 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 in the Appendix.

2.3. Decoding

The obtained masks \( M^{(i=5)} \in \mathbb{R}^{N_m \times \frac{H}{2} \times \frac{W}{2}} \) are low-resolution, and each spatial index corresponds not to a pixel but to a cluster generated in the backbone by eq. (3). Thus, we have to decode it based on the clustering results to obtain masks in the input image space. The decoding process is defined as follows:

\[
(M^{(i)})^T = \text{Clustering}(F(i), F_d(i); s(i))\left( M^{(i+1)} \right)^T = \text{Softmax}_\text{row} \left( (F(i)^T F_d(i) / |s(i)|) \left( M^{(i+1)} \right)^T \right), \tag{5}
\]

which corresponds to the attention function [2], although queries, keys, and values are obtained from different layers. From the hard clustering perspective, eq. (2) we can view the decoding as an operation of 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.

Unlike a conventional pixel decoder, we can evaluate accuracy of our decoding from the hard clustering perspective by the undersegmentation error [47] that is used for evaluating superpixel segmentation. In addition, the low-resolution feature maps are fed into the segmentation head because our decoding is applied after predicting masks. It contributes reduction of FLOPs in the segmentation head.
2.4. Efficient Computation

Let $M$ be the number of pixels and then computational costs of the proposed clustering are $O(M^2)$, which is the same complexity as the common attention function [54], and it is intractable for high-resolution images. Various studies exist on reducing complexity [35, 46, 4, 62, 56, 53, 43], and we approach it with the local attention strategy.

We divide the feature map before downsampling, $F(i)$, into $2 \times 2$ 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 [23] that is a pixel clustering algorithm. We illustrate this local attention in Fig. 3. As a result, the complexity decreases to $O(M)$.

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 decodes the predicted masks using eq. (5). Optionally, we build a six-layer transformer encoder after the backbone, as in MaskFormer [11]. For class prediction, the mask queries are fed into multilayer perceptrons and classified into $K$ classes.

The flexibility of the downsampling is important because for our clustering module, 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 [38, 60]. Thus, the CNN-based backbones may not sample effective pixels for the clustering module. To improve the flexibility, we replace downsampling layers (e.g., strided convolution) to which the proposed clustering module is attached with a deformable convolution v2 [70] that deforms kernel shapes and modulates weight values. This replacement is only adopted for a CNN-based backbone. We verify its effect in the Appendix.

Figure 3: Illustration of the local attention. Each color indicates the correspondence between queries and keys.

Figure 4: Comparison between MaskFormer [11] and HCFormer. MaskFormer upsamples a feature map and then predicts masks. In contrast, HCFormer directly predicts masks from downsampled feature maps. Instead of the pixel decoder, HCFormer recovers the resolution by cluster-based decoding, which corresponds to eq. (5).

3. Related Work

3.1. Image Segmentation with Deep Neural Networks

Since FCNs [36] 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 for semantic segmentation [68, 6, 3, 7, 8, 60]. In instance segmentation, many methods [19, 15, 32, 14, 30] depend on the object proposal, which is used for instance discrimination because instance segmentation has the object detection perspective. The pipeline of many proposal-based methods is somewhat complex, and thus proposal-free methods have been studied to simplify the pipeline, which uses the clustering approach including cross-attention (cross-correlation) [27, 40, 24, 16, 28, 57, 58]. 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 masks and instance masks and then fusing them [26, 25, 9, 61]. To simplify the framework, Panoptic FCN [31] 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 [55, 11, 10]. Such approaches also simplify the panoptic segmentation pipeline.

While many studies investigate task-specific modules and architectures, they cannot be applied models for other tasks. Thus, recent work investigates the unified architectures [11, 62, 10]. These methods use cross-attention-based approaches that can generate segmentation masks regardless of task definition. They have demonstrated effectiveness in the various segmentation tasks and have achieved comparable or better results compared to the specialized models. Our study also focuses on such unified architectures and builds our model based on MaskFormer [11], which is a unified approach.

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 accuracies [33, 42, 3, 32, 29, 59, 8]. However, the decoder would make the segmentation problem complex: models using the decoder have to solve not only the segmentation problem but also the upsampling problem internally. Such models suffer from an error in the upsampling process of the pixel decoder. In addition, it is difficult to interpret whether the prediction error is due to upsampling or classification because upsampling is performed in the high-dimensional feature space. Several methods use the attention [54] or similar modules for the pixel decoder [29, 10].

The pixel decoder is not necessarily required for image segmentation with deep neural networks. In fact, FCN-32s [35], which is 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. To keep the resolution in the backbone, several methods [6, 60, 68, 7] use dilated convolution [66, 8], which expands the convolution kernel by inserting holes between its consecutive elements, and replace the stride with the dilation rate. Our clustering and decoding process can be viewed as the attention-based pixel decoder. However, unlike other attention-based decoders [29, 22], our method does decoding after the segmentation head, and it enables visualization and interpretation of attention maps as the hierarchical clustering results, which is conceptually different from the other attention-based decoders.

3.2. Hierarchical Clustering

Hierarchical clustering approaches are sometimes used to solve image segmentation. For example, some previous methods [13, 18, 21, 51, 64, 65, 50, 39, 52, 44] group pixels into small segments by using superpixel segmentation [1, 48, 23] for 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 [63], is a bottom-up, hierarchical clustering method. However, GroupViT is specialized for vision transformers [17] and for semantic segmentation with text supervision. In contrast, our method can be incorporated into most multi-scale backbone models, such as ResNet [20] and Swin Transformer [35], and used for arbitrary image segmentation tasks.

4. Experiments

4.1. Implementation Details

We implement our model on the Mask2Former author’s implementation. 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. Our transformer decoder is independent of the pixel decoder, similar to MaskFormer. Thus, HCFormer is built on top of MaskFormer rather than on top of Mask2Former. We compare HCFormer to MaskFormer with the same transformer decoder architecture in the appendix.

We train HCFormer with almost the same protocol as in Mask2Former [10]. We use the binary cross-entropy loss and the dice loss [39] as the mask loss: $L_{\text{mask}} = \lambda_{\text{ce}} L_{\text{ce}} + \lambda_{\text{dice}} L_{\text{dice}}$. The training loss is a combination of mask loss, classification loss, and an additional regularization term for $s^{(i)}$ in eq. (3): $L_{\text{mask}} + \lambda_{\text{cls}} L_{\text{cls}} + \lambda_{\text{reg}} L_{\text{reg}}$, where $L_{\text{cls}}$ is a cross-entropy loss and $L_{\text{reg}} = \sum_i |s^{(i)}|$, $\lambda_{\text{reg}}$ is set to 0.1. The regularization enforces the proposed attention-based clustering, eq. (5), to be the hard clustering, eq. (2). Following [10], we set $\lambda_{\text{ce}} = 5.0, \lambda_{\text{dice}} = 5.0$, and $\lambda_{\text{cls}} = 2.0$. Other training protocol is also the same as for Mask2Former [10] (e.g., optimizer, its hyperparameters, and the number of training iterations). The details can be found in the Appendix.

We use the same post-processing as [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 at each pixel.

The clustering module defined in eq. (5) is incorporated into every downsampling layer, except for those in the stem blocks of ResNet [20] and the patch embedding layer in...
Swin Transformer [35]. Thus, the hierarchical level, namely the number of clustering modules, is 3.

4.2. Main Results

As baseline methods, we choose the recently proposed methods for unifying segmentation tasks, MaskFormer [11], K-Net [67], and Mask2Former [10] and specialized models for each task [31, 55, 60, 19, 58]. We compare our method to the baselines with several backbones, ResNet-50 [20], Swin-S [35], and Swin-L [35]. 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 train models three times and report medians.

We first show the evaluation results for panoptic segmentation on the COCO validation dataset [34] in Tab. 1. HCFormer outperforms the unified architectures, MaskFormer and K-Net, and the specialized models, Panoptic FCN and Max-Deeplab for all metrics with fewer parameters. In addition, the additional transformer encoder (i.e., HCFormer+) boosts the segmentation accuracy for the relatively small backbone models. MaskFormer and K-Net use a simple pixel decoder [33], and compared to such a simple decoder, hierarchical clustering scheme is a better approach for image segmentation in terms of the accuracy.

Mask2Former shows the best accuracy although it sacrifices the FLOPs. According to [10], the use of multi-scale features in the transformer decoder improves 1.7 PQ for the ResNet-50 backbone, which is the same as the gap between PQ of the HCFormer+ and Mask2Former with the ResNet-50 backbone. Therefore, we believe the gap in PQ between HCFormer and Mask2Former is due to the design of the transformer.

Tab. 2 shows the average precision on instance segmentation tasks. HCFormer also outperforms MaskFormer and the specialized model, Mask R-CNN. Notably, HCFormer significantly improves AP$^L$ from MaskFormer, which indicates HCFormer can generate the 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, poor pixel decoder, and the well-design decoder and the transformer decoder such as that used in Mask2Former is needed. However, since HCFormer groups pixels locally and hierarchically, it may not need long-range dependence compared to the conventional methods. As a result, HCFormer outperforms MaskFormer, especially in terms of AP$^L$.

Tabs. 3 and 4 shows 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, and 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-
Table 2: Evaluation results for instance segmentation with COCO val. HCFormer+ stacks six-layer transformer encoder after the backbone.

| 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 [58]    | 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    |
| 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 3: Evaluation results for semantic segmentation with ADE20K val. HCFormer+ stacks six-layer transformer encoder after the backbone.

| method         | backbone | mIoU | FLOPs  |
|----------------|----------|------|--------|
| MaskFormer [11]| R50      | 44.5 | 53G    |
| Mask2Former [10]| R50     | 47.2 | 73G    |
| HCFormer+      | R50      | 35.5 | 29G    |
| SegFormer [60] | MIT-B2   | 46.5 | 62G    |
| MaskFormer [11]| Swin-S   | 49.8 | 79G    |
| Mask2Former [10]| Swin-S  | 51.3 | 98G    |
| HCFormer+      | Swin-S   | 50.1 | 58G    |
| SegFormer [60] | MIT-B5   | 51.0 | 184G   |
| MaskFormer [11]| Swin-L   | 54.1 | 375G   |
| Mask2Former [10]| Swin-L  | 56.1 | 403G   |
| HCFormer+      | Swin-L   | 55.5 | 342G   |

Table 4: Evaluation results for semantic segmentation with Cityscapes val. HCFormer+ stacks 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   |
| SegFormer [60] | MIT-B2   | 81.0 | 196G   |
| MaskFormer [11]| Swin-S   | 78.5 | 599G   |
| Mask2Former [10]| Swin-S  | 82.6 | 727G   |
| HCFormer+      | Swin-S   | 79.5 | 398G   |
| SegFormer [60] | 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  |

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

| 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 6: Evaluation results for COCO val. with various numbers of transformer decoder layers.

| #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  |

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. We show an additional ablation study in the Appendix. The results are shown in Tabs. 5 and 6.
Since the resolution of the predicted masks also increases in line with the hierarchical level, the model can predict high-resolution masks. The hierarchical level of 2 would be preferred based on the balance between accuracy and computational costs in practice, although 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 [11, 10, 5]. In terms of the balance between accuracy and computational costs, four or eight layers would be suitable for our model.

5. Discussion

As described in the experimental section, the segmentation accuracy of our method is lower than the well-designed model with the pixel decoder [10]. While many findings exist for the pixel decoder to improve accuracy, no such finding has yet been made in our hierarchical clustering model.

However, we can provide a insight for improving the architecture. In the hierarchical clustering, early stages group pixels based on low-level properties such as color, as shown in Fig. 5. Thus, the backbone does not need to allocate the representational capacity to them, and the capacity should be allocated to the deeper stages. However, many existing backbones allocate more layers to the stage 3 (e.g., a ratio of the number of layers is 3:4:6:3 in ResNet-50 and 1:1:9:1 in Swin Transformer), and fewer layers at the deepest stage. Hence, we may be able to improve the segmentation accuracy by investigating the better ratio for HCFormer, but designing the backbone architecture for the hierarchical clustering is outside the scope of this work. We will investigate more effective architectures or modules for the hierarchical clustering models in terms of accuracy in future work.

6. Conclusion

We proposed a method that introduces the clustering property into conventional downsampling layers in the deep neural networks. As a result, we accomplished image segmentation via hierarchical clustering in deep neural networks and simplified the segmentation pipeline by removing the pixel decoder before the segmentation head. In experiments, we verified that our method achieves comparable or better accuracies compared to baseline methods for semantic, instance, and panoptic segmentation. Since our hierarchical clustering allows visualization of the segmentation process, we believe that the hierarchical clustering not only simplifies the segmentation models but also enhances their interpretability.
A. 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-mask pairs via $\arg \max_{c \in \{1, \ldots, K + 1\}} P_{i, c}$, where $c_i$ is the most likely class label, $c_i = \arg \max_{c \in \{1, \ldots, K + 1\}} P_{i, c}$, for each probability-mask pair $i$.

The maximization problem in post-processing, $\arg \max_{i} P_{i, c}$, corresponds to the clustering problem for the $(i, c)$ label, $i = \arg \max_{c \in \{1, \ldots, K + 1\}} P_{i, c}$, for each probability-mask pair $i$.

B. Detailed derivation of Eq. (2)

We provide the detailed derivation of eq. (2). We consider the softmax function for a vector, because the softmax function in eq. (1) is computed independently for each row.

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

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

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

$$a_i = \frac{\exp(v_i/s)}{\sum_j \exp(v_j/s)} - \frac{\exp(-\hat{v}/s)}{\exp(-\hat{v}/s)}$$

$$= \frac{\exp((v_i - \hat{v})/s)}{\sum_j \exp((v_j - \hat{v})/s)} \quad (7)$$

where $v_i - \hat{v} \leq 0$. When $s \to 0$, $\exp((v_i - \hat{v})/s) = \exp(-\infty) = 0$ if $v_i \neq \hat{v}$, and if $v_i = \hat{v}$, $\exp((v_i - \hat{v})/s) = \exp(0/s) = 1$. In this case, eq. (6) corresponds to the following problem:

$$\arg \max_{a \in \{0, 1\}^n} a^T v, \text{ s.t. } \sum_i a_i = 1. \quad (8)$$

The solution of this problem indicates a one-hot vector that holds $a_i = 1$, $i = \arg \max_j v_j$. By considering eq. (2) for each row, eq. (2) corresponds to eq. (8). Hence, when $s \to 0$, eq. (1) corresponds to eq. (5).

C. Example PyTorch implementation

We show example PyTorch implementation of the soft clustering and the cluster-based decoding. We can easily implement the local attention by using the unfold and einsum functions.

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

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

def clustering(feat, downsampled_feat, scale):
    candidate_clusters = F.unfold(downsampled_feat, kernel_size=3, padding=1).reshape(batch_size, n_channels, 9, -1)
    feat = F.unfold(feat, kernel_size=2, stride=2).reshape(batch_size, n_channels, 2, -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
```

Listing 2: The cluster-based decoding (eq. (5))

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

def decode(x, A):
    batch_size, _, height, width = A.shape
    # get 9 candidate clusters and corresponding assignments
    candidate_clusters = F.unfold(A, kernel_size=2, stride=2).reshape(batch_size, n_channels, 9, 4, -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, feats)).reshape(batch_size, n_channels, 9, height, width)
    # normalize
    soft_assignment = (similarities / scale.abs()).softmax(1)
    return decoded_features
```

D. Detailed experimental setup

We trained models with the Swin-L backbones for the COCO dataset on 4× NVIDIA A100 GPUs and the other models on 2× NVIDIA RTX8000 GPUs. The training protocol is following 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 \cite{37} 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 \cite{10}. For inference, we use the Mask R-CNN inference setting \cite{19}, where we resize an image with shorter side to 800 and longer side up to 1333.

D.2. Semantic segmentation on AD20K and Cityscapes

We use AdamW \cite{37} 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 $\times$ 512 and train HCFormer for 160k iterations and HCFormer+ for 320k iterations. For the Cityscapes dataset, we use a crop size of 512 $\times$ 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. Additional results

E.1. Per-pixel classification with hierarchical clustering

As described in Sec. 2.2, our hierarchical clustering scheme can be naturally incorporated into per-pixel classification models. Thus, we evaluate the proposed method combined with the per-pixel classification models. As a baseline, we use FCN-32s \cite{33}, FPN \cite{33}, PSPNet \cite{68}, and Deeplabv3 \cite{4}, 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 \cite{20} as the backbone for all models. Note that PSPNet and Deeplabv3 use dilated convolution \cite{69, 66} 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 remove the dilated convolution and use the same backbone architecture as FCN-32s; their output stride is 32. The training protocol is following \cite{68}. We set the crop size for ADE20K to 512 $\times$ 512.

We show the evaluation results for the validation set in Tab. I The latency of HC-PSPNet and HC-Deeplabv3 is lower than that of PSPNet and Deeplabv3, and HC-PSPNet shows the 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 the superior mIoU than FPN, that is, FCN-32s with the simple pixel decoder, which is also used in MaskFormer \cite{11}, while the latency is higher than FPN. This result is consistent with the results of the comparison between HCFormer and MaskFormer.

E.2. Effect of downsampling

In the experiments, we use the deformable convolution v2 (DCNv2) \cite{70} 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). Note that the hierarchical level is shown in Fig. 4.

We show the evaluation results for COCO \cite{34} in Tab. I.

| Models       | mIoU | Pixel Acc. | msec/image |
|--------------|------|------------|------------|
| PSPNet       | 42.7 | 80.9       | 42.7       |
| HC-PSPNet    | 42.6 | 80.9       | 42.6       |
| Deeplabv3    | 42.7 | 81.0       | 42.7       |
| HC-Deeplabv3 | 42.8 | 81.1       | 42.8       |

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

### E.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. VI and the training protocol is the same as described in Sec. D.

We evaluate HCFormer with the transformer decoder used in MaskFormer [11], although the masked transformer decoder [10] was used in the experiments in the main manuscript. As shown in Tab. II, 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. II is the decoding process. Thus, our hierarchical clustering scheme generates more accurate masks than the pixel decoder, namely the upsampling and then clustering scheme. Moreover, the training iteration for HCFormer is one-third that for MaskFormer. The simplification of the architecture may allow faster convergence.

We evaluate HCFormer using various number of trainable queries (Tab. IV). 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. V. In addition, we evaluate HCFormer with the Swin-S [35] backbone with various numbers of hierarchical level on COCO [34]. Regardless of the backbone type and dataset, higher hierarchical level leads to higher accuracy.

We evaluate HCFormer with the ResNet-50 backbone with various numbers of transformer decoder layers. 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. VII. This result is better than the per-pixel classification baselines with ResNet-101 [20], such as PSPNet [68] and Deeplabv3 [7], as shown in Tab. I. The deeper transformer decoder improves mIoU, and four or eight layers would be sufficient for HCFormer.

### E.4. Visualization

We show example results on ADE20K by HCFormer with Swin-L [35] in Fig. II. Basically, the undersegmentation error appears near the boundaries between regions 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) in the coarser clustering levels, but this would be due to the annotation error. 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 undersegmentation error only appears in their boundaries.

We show example results for COCO panoptic segmentation by HCFormer+ with Swin-L [35] in Fig. III. HCFormer segments the image well albeit with some misclassification. Thus, to solve such misclassification error, we
| 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 II: Evaluation results on COCO val. with various hierarchical levels.

| method                      | PQ     | FLOPs |
|-----------------------------|--------|-------|
| MaskFormer [11]             | 46.5   | 181G  |
| HCFormer w/o DCNv2          | 46.8   | 97G   |
| HCFormer w/ DCNv2           | 47.6   | 97G   |

Table III: Evaluation results of HCFormer with the standard transformer decoder used in [5, 11].

| #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 IV: 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 V: 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 VI: Evaluation results on COCO val. with various hierarchical levels for the Swin-S backbone.

| #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 VII: Evaluation results on ADE20K val. with various numbers of transformer decoder layers.

would need to improve the transformer decoder and/or incorporate stronger backbones.

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.

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

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

[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.

[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.

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

[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.

[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.

[10] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. *CVPR*, 2022.

[11] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *volume 34*, 2021.

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe
Figure II: Example results on ADE20K. The top row shows the cluster boundaries and predicted masks, and the bottom row shows the undersegmentation error as red regions and the ground-truth label.

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.

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

[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.

[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.

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

[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.

[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.

[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.

[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.

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

[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.

[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.

[24] Tommi Kerola, Jie Li, Atsushi Kancheira, 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.

[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.

[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.

[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.

[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.

[29] Xiangtai Li, Ansheng You, Zhen Zhu, Houlong Zhao, Maokong 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.

[30] 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.

[31] Yanwei Li, Hengshuang Zhao, Xiaojun 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.

[32] Xiaodan Liang, Yunchao Wei, Xiaohui Shen, Zequn Jie, Jiashii 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.

[33] 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.

[34] 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.

[35] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Xiujuan 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.

[36] 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.

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

[38] 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.

[39] 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.

[40] 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.

[41] 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’Alch´e-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019.

[42] 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.

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

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In European Conference on Computer Vision, pages 649–665. Springer, 2020.
[58] 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.

[59] 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.

[60] 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.

[61] 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.

[62] Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nystromformer: 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.

[63] 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.

[64] 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.

[65] 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.

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

[67] Wenwei Zhang, Jianguo Peng, Kai Chen, and Chen Change Loy. K-Net: Towards unified image segmentation. *Advances in Neural Information Processing Systems*, 34, 2021.

[68] 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.

[69] 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.

[70] 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.