Unsupervised Hierarchical Semantic Segmentation with Multiview Cosegmentation and Clustering Transformers

Tsung-Wei Ke  Jyh-Jing Hwang  Yunhui Guo  Xudong Wang  Stella X. Yu
UC Berkeley / ICSI

Abstract

Unsupervised semantic segmentation aims to discover groupings within and across images that capture object- and view-invariance of a category without external supervision. Grouping naturally has levels of granularity, creating ambiguity in unsupervised segmentation. Existing methods avoid this ambiguity and treat it as a factor outside modeling, whereas we embrace it and desire hierarchical grouping consistency for unsupervised segmentation.

We approach unsupervised segmentation as a pixel-wise feature learning problem. Our idea is that a good representation shall reveal not just a particular level of grouping, but any level of grouping in a consistent and predictable manner. We bootstrap feature learning with co-segmentation among multiple views of the same image, and enforce semantic consistency across the grouping hierarchy with clustering transformers between coarse- and fine-grained features.

We deliver the first data-driven unsupervised hierarchical semantic segmentation method called Hierarchical Segment Grouping (HSG). Capturing visual similarity and statistical co-occurrences, HSG also outperforms existing unsupervised segmentation methods by a large margin on five major object- and scene-centric benchmarks.

1. Introduction

Semantic segmentation requires figuring out the semantic category for each pixel in an image. Learning such a segmenter from unlabeled data is particularly challenging, as neither pixel groupings nor semantic categories are known.

If pixel groupings are known, semantic segmentation is reduced to an unsupervised image (segment) recognition problem, to which contrast learning methods [9, 20, 59, 62] could apply, on computed segments instead of images.

If semantic categories are known, semantic segmentation is reduced to a weakly supervised segmentation problem with coarse annotations of image-level tags; pixel labeling can be predicted from image classifiers [32, 34].

The fundamental task of unsupervised semantic segmentation is grouping, not semantics in terms of naming, which is unimportant other than the convenience of tagging segments in the same or different groups. The challenge of unsupervised semantic segmentation is to discover groupings within and across images that capture object- and view-invariance of a category without external supervision, so that (Fig. 1): 1) A baby’s face and body are parts of a whole...
in the same image; 2) The whole baby is separated from the rest of the image; 3) A baby instance is more similar to another baby instance than to a cat instance, despite their different poses, illuminations, and backgrounds.

Several representative approaches have been proposed for tackling this challenge under different assumptions.

• **Visual similarity:** SegSort [26] first partitions each image into segments based on contour cues and then by segment-wise contrastive learning discovers clusters of visually similar segments. However, semantics by visual similarity is far too restrictive: A semantic whole is often made up of visually dissimilar parts. Parts of body such as head and torso look very different; it is not their visual similarity but their spatial adjacency and statistical co-occurrence that bind them together.

• **Spatial stability:** IIC [29] maximizes the mutual information between clusterings from two views of the same image related by a known spatial transformation, enforcing stable clustering while assuming that a fixed number of clusters are equally likely within an image. It works best for coarse and balanced texture segmentation and has major trouble scaling up with the scene complexity.

• **Image-wise feature learning:** [56, 60] train representations on object-centric datasets with multiscale cropping to sharpen the representation within the image. These methods do not work well on scene-centric datasets where an image has more than one dominant semantic class.

Grouping as well as semantics naturally have different levels of granularity: A hand is an articulated configuration of a palm and five fingers, likewise a person of a head, a torso, two arms, and two legs. Such an inherent grouping poses a major challenge: Which level should an unsupervised segmentation method target at and what is the basis for such a determination? Existing methods avoid this ambiguity and treat it as either a factor outside the segmentation modeling, or an aspect of secondary concern.

Our key insight is that the inherent hierarchical organization of visual scenes is not a nuisance for scene parsing, but a universal property that we can exploit and desire for unsupervised segmentation. This idea has previously led to a general image segmenter that handles texture and illusory contours through edges entirely without any explicit characterization of texture or curvilinearity [65]. We now advance the concept to data-driven representation learning: A good representation shall reveal not just a particular level of grouping, but any level of grouping in a consistent and predictable manner across different levels of granularity.

We approach unsupervised semantic segmentation as an unsupervised pixel-wise feature learning problem. Our objective is to best produce a consistent hierarchical segmentation for each image in the entire dataset based entirely on hierarchical clusterings in the feature space (Fig. 1). Specifically, given the pixel-wise feature, we perform hierarchical groupings within and across images and their transformed versions (i.e., views). In turn, groupings at each level impose a desire on how the feature should be improved to maximize the discrimination among different groups.

Our model has two novel technical components: 1) **Multiview cosegmentation** is to not only enforce spatial consistency between segmentations across views, but also bootstrap feature learning from visual similarity and co-occurrences in a simpler clean setting; 2) **Clustering transformers** are used to enforce semantic consistency across different levels of the feature grouping hierarchy.

To summarize, our work makes three contributions.

1. **We deliver the first unsupervised hierarchical semantic segmentation** method that can produce parts and wholes in a data-driven manner from an arbitrary collection of images, whether they come from object-centric or scene-centric datasets.

2. **We are the first to embrace the ambiguity of grouping granularity** and exploit the inherent grouping hierarchy of visual scenes to learn a pixel-wise feature representation for unsupervised segmentation. It can thus discover semantics based on not only visual similarity but also statistical co-occurrences.

3. **We outperform existing unsupervised (hierarchical) semantic segmentation methods by a large margin** on not only object-centric but also scene-centric datasets.

### 2. Related Work

**Image segmentation** refers to the task of partitioning an image into visually coherent regions. Traditional approaches often consist of two steps: extracting local features and clustering them based on different criteria, e.g., mode-finding [3, 10], or graph partitioning [16, 42, 52, 66, 67].

**Hierarchical image segmentation** has been supervisedly learned from how humans perceive the organization of an image [2]: While each individual segmentation targets a particular level of grouping, the collection of individual segmentations present the perceptual hierarchy statistically.

A typical choice for representing a hierarchical segmentation is contours: They are first detected to sharply localize region boundaries [25, 63] and can then be removed one by one to reveal coarser segmentations (OWT-UCM [2]).

Such models are trained on individual ground-truth segmentations, hoping that coarse and fine-grained organization would emerge automatically from common and rare contour occurrences respectively in the training data.

In contrast, our model is trained on multi-level segmentations unsupervisedly discovered by feature clustering, and it also operates directly on segments instead of contours. **Semantic segmentation** refers to the task of partitioning an image into regions of different semantic classes. Most
deep learning models treat segmentation as a spatial extension of image recognition and formulate it as a pixel-wise classification problem. They are often based on Fully Convolutional Networks [7, 36, 40], incorporating information from multiple scales [8, 18, 22–24, 31–33, 35, 45, 53, 64].

SegSort [26] does not formulate segmentation as pixel-wise labeling, but pixel-segment contrastive learning that operates directly on segments delineated by contours. It learns pixel-wise features in a non-parametric way, with or without segmentation supervision. SPML [32] extends it to unify segmentation with various forms of weak supervision: image-level tags, bounding boxes, scribbles, or points.

**Unsupervised semantic segmentation** has been modeled by non-parametric methods using statistical features and graphical models [39, 49, 54]. For example, [49] proposes to discover region boundaries by mining the statistical differences of matched patches in coarsely aligned images.

There are roughly three lines of recent unsupervised semantic segmentation methods. 1) One way is to increase the location sensitivity of the feature learned from images [9, 20, 59, 62], by either adding an additional contrastive loss between pixels based on feature correspondences across views [60], or using stronger augmentation and constrained cropping [51, 56]. 2) A pixel-level feature encoder can be learned directly by maximizing discrimination between pixels based on either contour-induced segments [26] or region hierarchies [68] derived from OWT-UCM [2]. Segmentation is indicated by pixel feature similarity and semantic labels can be inferred from retrieved nearest neighbours in a labeled set. 3) A pixel-wise cluster predictor can be directly learned by maximizing the mutual information between cluster predictions on augmented views of the same instance at corresponding pixels [29, 47].

Our model advances pixel-wise feature learning methods [26, 32, 69]: It contrasts features based on feature-induced hierarchical groupings themselves, and most strikingly, directly outputs consistent hierarchical segmentations.

### 3. Hierarchical Segment Grouping (HSG)

We approach unsupervised semantic segmentation as an unsupervised pixel-wise feature learning problem (Fig. 2). The basic idea is that, once every pixel is transformed into a point in the feature space, image segmentation becomes a point clustering problem.

Semantic segmentation and feature clustering form a pair of dual processes: 1) Clustering of feature $X$ defines segmentation $G$ in each image: Pixels with features in the same (different) clusters belong to the same (different) semantic regions. This idea is used to co-segment similar images given handcrafted features [30, 37, 48]. 2) Segmentation $G$ defines the similarity of feature $X$: A pixel should be mapped close to its own segment group and far from other segment groups in the feature space. This idea is used to learn the pairwise feature similarity [44] and pixel-wise feature [26, 32] given segmentations.

Our key insight is that a good representation shall reveal not just a particular level of grouping – as past cosegmentation methods have explored, but any level of grouping in a consistent and predictable manner. If we embrace the ambiguity of grouping granularity that all previous methods have avoided and desire the consistency of hierarchical semantic segmentation on the pixel-wise feature, we address not only the shortcoming of cosegmentation, but also provide a joint feature-segmentation learning solution.

Specifically, while there is no supervision available for either feature $X$ or segmentation $G$, we can desire that: 1) each segmentation separates features well and 2) the coarser segmentation defined by next-level feature clusters simply merges the current finer segmentation. These strong constraints guide the feature learning towards quality hierarchical segmentations, thereby better capturing semantics.

Our model has two components: 1) multi-view cosegmentation to robustify feature clustering against spatial transformation and appearance variations of visual scenes, and 2) clustering transformers to enforce consistent semantic segmentations across different levels of the feature grouping hierarchy. Both are necessary for mapping pixel features to segmentations, which in turn impose desired pairwise attraction and repulsion on the pixel features.

In the following, we first introduce our contrastive feature learning loss given any grouping $G$, and then describe
how we obtain three kinds of groupings within and across images, and how we evaluate their goodness of grouping and enforce their consistency.

### 3.1. Pixel-Segment Contrastive Feature Learning

We learn a pixel-wise feature extraction function $f$ as a convolutional neural network (CNN) with parameters $\theta$. It transforms image $I$ to its pixel-wise feature $V$. Let $v_i$ be the unit-length feature vector at pixel $i$ of image $I$:

$$v_i = f_i(I; \theta), \quad \|v_i\| = 1. \quad (1)$$

Suppose that $I$ is partitioned into segments (Fig. 3). Let $u_s$ be the feature vector for segment $s$, defined as the (length-normalized) average pixel feature within the segment:

$$u_s \propto \text{mean}(v_i : i \text{ in segment } s), \quad \|u_s\| = 1 \quad (2)$$

Consider a batch of images and their pixel groupings $\{ (I, G) \}$. We want to learn the right feature mapper $f$ such that all the pixels form distinctive clusters in the feature space, each corresponding to a different semantic group.

We follow [26, 32] to formulate desired feature-wise attraction and repulsion not between pixels, but between pixels and segments. Such contrastive learning across granularity levels reduces computation, improves balance between attraction and repulsion, and is more effective [59].

Our contrastive feature learning loss to minimize is:

$$L_f(G) = \sum_i -\log \frac{\sum_{s \in G_i^+} \exp \frac{v_i^\top u_s}{T}}{\sum_{s \in G_i^+} \exp \frac{v_i^\top u_s}{T} + \sum_{s \in G_i^-} \exp \frac{v_i^\top u_s}{T}} \quad (3)$$

where $T$ is a temperature hyper-parameter that controls the concentration level of the feature distribution. Ideally, $v_i$ should be attracted to segments in the positive set $G_i^+$ and repelled by segments in the negative set $G_i^-$. Our batch of images consists of several augmented views of some training instances. For pixel $i$ in a particular view of image $I$, $G_i^+$ includes segments of the same semantic group in any view of image $I$ except $i$’s own segment, in order to achieve within-instance invariance, whereas $G_i^-$ includes segments of different semantic groups in any view of $I$, and segments of training instances other than $I$, in order to maximize between-instance discrimination [26, 62].

### 3.2. Consistent Segments by View & Hierarchy

From pixel feature $V$, we compute feature grouping $G_0$ and cluster feature $X_0$. Our initial pixel grouping $G_0$ is based on OWT-UCM edges detected in the image. Next-level cluster feature $X_{l+1}$ and grouping $G_{l+1}$ are predicted from $G_l$ with ensured consistency. We use three levels for the sake of illustration (Fig. 3), but our procedure can be repeated for more (coarser) levels.

**Base cluster feature $X_0$ and grouping $G_0, G_c$.** We segment each view of $I$ by clustering pixel features, resulting in base grouping $G_0$ and cluster (centroid) feature $X_0$ (Fig. 2).

During training but not testing, we segment image $I$ into a fixed number of coherent regions according to its OWT-UCM edges [14], based on which we split each $G_0$ region to obtain edge-conforming segments [26] marked by white lines in Fig. 3. For training, we obtain pixel grouping $G_c$ by inferring the coherent region segmentation according to how each view is spatially transformed from $I$.

Minimizing $L_f(G_c)$ encourages the feature to be similar not only for different pixels of similar appearances in the image, but also for corresponding pixels of different appearances across views of $I$. The former grounds the feature $f$ at respecting low-level appearance coherence, whereas the latter develops view invariance in the feature.

**Next-level cluster feature $X_{l+1}$ and grouping $G_{l+1}$.** Now we have grouping $G_0$ in the feature space of $V$, and for each cluster, we obtain its centroid feature in $X_0$. We model how cluster feature $X_l$ maps to cluster feature $X_{l+1}$, which cor-

![Figure 3. We co-segment multiple views (Column 1) of the same image by OWT-UCM edges ($G_c$, Column 2) or by feature clustering at fine and coarse levels ($G_1, G_2$, Columns 3-4). White lines mark the segments derived from pixel feature clustering and OWT-UCM edges. The color of feature points (pixels) mark grouping at fine and coarse levels ($G_1, G_2$), Columns 3-4). White lines mark the segments derived from pixel feature clustering and OWT-UCM edges.](image-url)
responds to how segmentation at level $l$ maps to segmentation at level $l+1$ in the image.

We adopt a probabilistic framework, where any feature point $x$ has a (soft) assignment probability belonging to a group determined by its cluster centroid. Let $P_l(a)$ be the probability of $x$ in group $a$ at level $l$:

$$P_l(a) = \text{Prob}(G_l = a | x).$$

(4)

To ensure that feature points in the same group remain together at the next level, we introduce group transition probability $C_l^{l+1}(a, b)$, the transition probability from group $a$ at level $l$ to group $b$ at level $l+1$:

$$C_l^{l+1}(a, b) = \text{Prob}(G_{l+1} = b | G_l = a).$$

(5)

Per the Bayesian rule, we have:

$$P_{l+1}(b) = \sum_a P_l(a) \cdot C_l^{l+1}(a, b).$$

(6)

Writing $P_l$ as a row vector, we can derive the soft group assignment $P_{l+1}$ for cluster feature $X_0$ at level $l+1$:

$$P_{l+1} = P_l \times C_l^{l+1} = P_0 \times C_0^{l+1} \times \cdots \times C_l^{l+1}.$$  

(7)

**Clustering Transformers.** $C_l^{l+1}$ is defined on multiview cosegmentation of each instance. We learn a function, in terms of a transformer [5], to naturally capture group transitions for all the training instances. It enables more consistent grouping compared to non-parametric clustering methods such as KMeans, NCut [58], and FINCH [50].

Our clustering transformer from level $l$ to $l+1$ maps group centroid feature $X_l$ to the next-level group centroid feature $X_{l+1}$, and simultaneously outputs the group transition probability $C_l^{l+1}$ (Fig. 4).

**Consistent feature groupings.** At level $l = 0$, $P_0$ has binary values, indicating hard grouping $G_0$. For next level $l$, we compute $P_{l+1}$ by propagating $P_l$ with our clustering transformer $C_l^{l+1}$, which also outputs $X_{l+1}$. We obtain $G_{l+1}$ by binarizing $P_{l+1}$ with winner-take-all. By decreasing the number of groups as $l$ increases, we obtain consistent fine to coarse segmentations $G_1, G_2$ (Fig. 2).

Minimizing $L_f(G_1)$ and $L_f(G_2)$ encourages the feature $f$ to capture semantics at multiple levels and produce consistent hierarchical segmentations (Fig. 3).

**3.3. Goodness of Grouping**

While clustering transformers ensure grouping consistency across levels, we still need to drive feature learning towards good segmentations. We follow [55] and supervise our transformer with modularity maximization [46] and collapse regularization. The former seeks a partition that results higher (lower) in-cluster (out-cluster) similarity than the total expectation, whereas the latter encourages partitions of equal sizes. We additionally maximize the separation between cluster centroids.

We first build a sparsified graph based on pairwise feature similarity for $X_0$. Let $e$ be the number of edges in this graph, $n_l$ the number of centroids in $X_l$, $A$ the $n_0 \times n_0$ connection matrix for edges, $D$ the $n_0 \times 1$ degree vector of $A$, $M_l$ the $n_0 \times n_l$ soft assignment matrix where each row is $P_l$ for a centroid of $X_0$, and $z_{l,k}$ the normalized $k$-th feature of $Z_l$ in Fig. 4. Our goodness of grouping loss is:

$$L_g = \sum_{l \geq 1} \frac{1}{2e} \text{trace}(M_l (A - \frac{1}{2e} D D^\top) M_l) + \frac{\sqrt{n_{l+1}}}{n_0} \| M_l \|_F^2 - 1$$

$$+ \frac{1}{n_l} \sum_k \log \left( \sum_{j} \exp(z_{l,k}^j) \right)$$

$$\left(8\right)$$

**3.4. Model Overview: Training and Testing**

Our model (Fig. 5) is trained with the contrastive feature learning losses given edge-based grouping $G_e$ and multi-level feature-based grouping $G_l$, and the goodness of group-
Figure 5. Our model consists of two essential components: 1) multiview cosegmentation and 2) hierarchical grouping. We first produces pixel-wise feature $V$, from which we cluster to get base cluster feature $X_0$ and grouping $G_0$. Each $G_0$ region is split w.r.t coherent regions derived by OWT-UCM procedure, which is marked by the white lines. We create three groupings–$G_e$, $G_1$ and $G_2$ in multiview cosegmentation fashion. We obtain $G_e$ by inferring the coherent region segmentation according to how each view is spatially transformed from the original image. Starting with input $X_0$ of an image and its augmented views, we conduct feature clustering to merge $G_0$ into $G_1$, and then, $G_1$ into $G_2$. Based on $G_e$, $G_1$ and $G_2$, we formulate a pixel-to-segment contrastive loss for each grouping. Our HSG learns to generate discriminative representations and consistent hierarchical segmentations for the input images.

\[ L(f) = \lambda_E L_f(G_e) + \lambda_F \sum_{l \geq 1} L_f(G_l) + \lambda_G L_g. \] (9)

For testing, the same pipeline with the pixel feature CNN and clustering transformers predicts hierarchical segmentations $\{G_l\}$. To benchmark segmentation performance given a labeled set, We follow [26] and predict the labels using k-nearest neighbor search for each segment feature.

4. Experiments

We benchmark our model on two tasks: unsupervised semantic segmentation and hierarchical image segmentation, the first on five major object- and scene-centric datasets and the second on Pascal VOC. We conduct ablation study to understand the contributions of our model components.

We adopt FCN-ResNet50 as the common backbone architecture. The FCN head consists of $1 \times 1$ convolution, BatchNorm, ReLU, and $1 \times 1$ convolution. Specifically, we follow DeepLabv3 [8] to set up the dilation and strides in ResNet50. We set Multi_Grid to $(1, 2, 4)$ in res5. The output_stride is set to 16 and 8 during training and testing. We do not use any pre-trained models, but train our models from scratch on each dataset. Ground-truth annotations are not for training but only for testing and evaluation’s sake.

Pascal VOC 2012 [15] is a generic semantic segmentation dataset of 20 object category and a background class. It consists of 1, 464 and 1, 449 images for training and validation. We follow [7] to augment the training data with additional annotations [19], resulting in 10, 582 training images. Following [56], we do not train but only inference on VOC.

MSCOCO [38] is a complex scene parsing dataset with 80 object categories. Objects are embedded in more complex scenes, with more objects per image than Pascal (7.3 vs. 2.3). Following [56, 60], we use train2017 split (118, 287 images) for training and test on the VOC validation set.

Cityscapes [11] is an urban street scene parsing dataset, with 19 stuff and object categories. Unlike MSCOCO and VOC where classes are split by scene context, Cityscapes contains similar street scenes covering almost all 19 categories. The train/test split is 2, 975/500.

KITTI-STEP [61] is a video dataset for urban scene understanding, instance detection and object tracking. It has pixel-wise labels of the same 19 categories as Cityscapes. There are 12 and 9 video sequences for training and validation, or 5, 027 and 2, 981 frames.

COCO-stuff [4] is a scene texture segmentation dataset, a subset of MSCOCO. As [29, 47], we use 15 coarse stuff categories and reduce the dataset to 52K images with at least 75% stuff pixels. The train/test split is 49, 629/2, 175.

Potsdam [17] is a dataset for aerial scene parsing. The raw
6000 × 6000 image is divided into 8550 RGBIR 200 × 200 patches. There are 6 categories (roads, cars, vegetation, trees, buildings, clutter). The train/test split is 7, 695/855.

### Table 1

<table>
<thead>
<tr>
<th>Training set</th>
<th>MSCOCO</th>
<th>Cityscapes</th>
<th>KITTI-STEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation set</td>
<td>VOC</td>
<td>Cityscapes</td>
<td>KITTI-STEP</td>
</tr>
<tr>
<td>Method</td>
<td>mIoU</td>
<td>Acc</td>
<td>mIoU</td>
</tr>
<tr>
<td>Moco [20]</td>
<td>28.1</td>
<td>-</td>
<td>15.3</td>
</tr>
<tr>
<td>DenseCL [60]</td>
<td>35.1</td>
<td>-</td>
<td>12.7</td>
</tr>
<tr>
<td>Revisit [56]</td>
<td>35.1</td>
<td>-</td>
<td>17.1</td>
</tr>
<tr>
<td>SegSort [26]</td>
<td>11.7</td>
<td>75.1</td>
<td>24.6</td>
</tr>
<tr>
<td>Our HSG</td>
<td>41.9</td>
<td>85.7</td>
<td>32.5</td>
</tr>
</tbody>
</table>

Table 1. Our method delivers better performance on different types of datasets. The results are reported on VOC, KITTI-STEP and Cityscapes val set, using IoU and pixel accuracy metrics. In VOC, object categories are separated according to image scenes. In Cityscapes and KITTI-STEP, images all come from urban street scene and thus contain mostly the same set of categories. Instance-discrimination methods apply image-wise contrastive loss, and learn less optimally on Cityscapes and KITTI-STEP, as image scenes are similar. Our HSG instead learns to discriminate regions at different scales and performs well on both types of datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>COCO-stuff</th>
<th>Potsdam</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU</td>
<td>Acc.</td>
</tr>
<tr>
<td>Isola 2016 [28]</td>
<td>-</td>
<td>24.3</td>
</tr>
<tr>
<td>IIC [29]</td>
<td>-</td>
<td>27.7</td>
</tr>
<tr>
<td>AC [47]</td>
<td>-</td>
<td>30.8</td>
</tr>
<tr>
<td>SegSort [26]</td>
<td>16.4</td>
<td>49.9</td>
</tr>
<tr>
<td>Our HSG</td>
<td>23.8</td>
<td>57.6</td>
</tr>
</tbody>
</table>

Table 2. Our method outperforms baselines on both stuff region and aerial scene parsing datasets. The results are reported on COCO-stuff and Potsdam test set, using IoU and pixel accuracy metrics. We evaluate our model using nearest neighbor search. Our HSG achieves superior performance.

<table>
<thead>
<tr>
<th>Method</th>
<th>KMeans</th>
<th>NCut [58]</th>
<th>FINCH [50]</th>
<th>Our Transformer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mIoU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>41.2</td>
<td>41.3</td>
<td>40.6</td>
<td>41.9</td>
</tr>
</tbody>
</table>

Table 3. Regularizing with our goodness of grouping loss and pixel-to-segment contrastive losses improves learned features. The results are reported over VOC val set, using IoU metric. Our resulted pixel features encode better semantic information.

Results on unsupervised semantic segmentation. All the models are trained from scratch and evaluated by IoU and pixel accuracy. For VOC, we follow baselines [56] to train on MSCOCO. Table 1 shows that our method outperforms baselines by 6.8%, 7.9% and 2.5% in mIoU on VOC, Cityscapes, and KITTI-STEP validation sets respectively.

Note that methods relying on image-wise instance discrimination do not work well on Cityscapes and KITTI-STEP. Both datasets have urban street scenes with similar categories in each image. Our method can still discover semantics by discriminating regions among these images.

For texture segmentation on COCO-stuff and Potsdam,
Figure 7. Our clustering transformers capture semantics at different levels of granularity. **Top:** We compare to other clustering algorithms on VOC val set, using Normalized Foreground Coverings as metric. We exclude background regions for evaluation. Our HSG overlaps with ground truths more accurately. **Bottom:** We present visual results to compare our hierarchical segmentation (top row) with SE [14]-OWT-UCM procedure (bottom row). We also show the detected edges at the leftmost figure in the bottom row. Each image is segmented into 12, 6, and 3 regions. Our method reveals low-to-high level of semantics more consistently.

Tab. 2 shows that our method achieves huge gains, +26.8% and +18.1% over IIC [29] and AC [47] respectively.

**Results on hierarchical segmentation.** We benchmark hierarchical segmentation with respect to ground-truth segmentation. We evaluate the overlapping of regions between predicted segmentations and ground truth within each image, known as Segmentation Covering [2]. However, such a metric scores performance with the number of pixels within each segment, and is thus easily biased towards large regions. For object-centric dataset VOC, a trivial all-foreground mask would rank high by the Covering metric.

We propose a Normalized Foreground Covering metric, by focusing on the foreground region and the overlap ratio instead of the overlap pixel count. To measure the average foreground region overlap ratio of a ground-truth segmentation \( S \) by a predicted segmentation \( S' \), we define:

\[
\text{NFCovering}(S' \rightarrow S_{fg}) = \frac{1}{|S_{fg}|} \sum_{R \in S_{fg}} \max_{R' \in S'} \frac{|R \cap R'|}{|R \cup R'|}
\]

where \( S_{fg} \) denotes the set of ground-truth foreground regions. Given a hierarchical segmentation, we report NFCovering at each level in the hierarchy. Fig. 7 shows that our clustering transformers produce segmentations better aligned with the ground-truth foreground at every level.

**Visualization.** Fig. 6 shows sample semantic segmentations on VOC (trained on MSCOCO), Cityscapes and KITTI-STEP. Compared to SegSort [26], our method retrieves same-category segments more accurately. For larger objects or stuff categories, such as **airplane** or **road**, our results are more consistent within the region. Our segmentations are also better at respecting object boundaries.

We also compare our hierarchical segmentations with SE [14]-OWT-UCM, an alternative based entirely on low-level cues. Fig. 7 bottom shows that, when partitioning an image into 12, 6 and 3 regions, our segmentations follow the semantic hierarchy more closely.

**Ablation study.** Tab. 3 shows that our model improves consistently by adding the feature learning loss based on hierarchical groupings and the goodness of grouping loss. It also shows that multiview cosegmentation significantly improves the performance over a single image.

Tab. 4 shows that our clustering transformers provide better regularization with hierarchical groupings than alternative non-parametric clustering methods.

**Summary.** We deliver the first unsupervised hierarchical semantic segmentation method based on multiview cosegmentation and clustering transformers. Our unsupervised segmentation outperforms baselines on major object- and scene-centric benchmarks, and our hierarchical segmentation discovers semantics far more accurately.

**Acknowledgements.** This work was supported, in part, by Berkeley Deep Drive, Berkeley AI Research Commons with Facebook, NSF 2131111, and a Bosch research gift.
References


[15] Mark Everingham, Luc Van Gool, Christopher KI Williams,
scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 3


Unsupervised Hierarchical Semantic Segmentation with Multiview Cosegmentation and Clustering Transformers

Tsung-Wei Ke  Jyh-Jing Hwang  Yunhui Guo  Xudong Wang  Stella X. Yu
UC Berkeley / ICSI

5. Supplementary

We propose the first unsupervised hierarchical semantic segmentation method. Our core idea has two folds: 1) imposing grouping hierarchy in the feature space with clustering transformers and 2) enforcing spatial grouping consistency with multiview cosegmentation. We demonstrate state-of-the-art performance on unsupervised semantic segmentation and hierarchical segmentation. In this supplementary, we include more details on the following aspects:

• We present the visual results of the attention maps from our clustering transformers in 5.1.
• We present the qualitative results of hierarchical semantic segmentation on DensePose dataset in 5.2.
• We present the visual results of contextual information encoded in our learned feature mappings in 5.3.
• We showcase unsupervised semantic segmentation on COCO-stuff and Potsdam datasets in 5.4.
• We present per-category results on VOC with ImageNet-trained models in 5.5.
• We present inference latency of our clustering transformers in 5.6.
• We describe more details of our clustering transformers in 5.7.
• We describe the details of our experimental settings and hyper-parameters in 5.8.
• We detail the experimental settings on VOC using ImageNet-trained models in 5.9.

5.1. Visual Results on Attention Maps from Decoder

We visualize the multi-head attention maps in the decoder of our clustering transformer. Such attention maps correspond to the correlation among cluster centroids and input segments. As shown in Fig. 8, we observe that each cluster attends to their cluster members, e.g. face and hair of the head region. Interestingly, we also see these clusters correlate better with image segments that carry more similar semantic meanings. For example, the ‘head’ cluster attends more to body parts than background regions. Such correlation information implies the next-level groupings: ‘head’ will be grouped with ‘torso’ instead of ‘background’.

5.2. Hierarchical Semantic Segmentation

We show that hierarchical segmentation is needed to pick out semantics at different levels of granularity. On DensePose [1] dataset, we process ground-truth labels at two levels of semantics: person and body parts. Body-part-level labels include head, torso, upper and lower limb regions. As shown in the top of Fig. 9, coarse (person-level) and fine (body-part-level) labels are best picked out with higher and lower levels of segmentations. The evaluation metric is based on F-score of region matching.

We next demonstrate the efficacy of our hierarchical clustering transformer. As shown in the bottom of Fig. 9, our predicted segmentations outperform others by large margin at every level in the hierarchy on both the fine and coarse sets of semantic labels. Hierarchical consistency provides regularizations which help us to obtain better segmentation results at any granularity.

5.3. Visual Results on Contextual Retrievals

We reveal the encoding of visual context in our learned feature representations. We first conduct hierarchical segmentation using our clustering transformers to partition an image into fine and coarse regions. We then compute the unit-length average feature within each region and perform nearest neighbor search among the training dataset. Fig. 10 shows nearest neighbor retrievals at coarse (cyan) and fine (red) segmentations. The query and retrieved segments are generated at same level of partitioning. Strikingly, the feature representations at each level of grouping correlate with multiple levels of semantic meanings such as baseball players and their body parts.

We next demonstrate the contextual information of co-occurring objects encoded in our feature representations. We visualize the length-normalized average features of the
Figure 8. The multi-head attention maps reveal the fine-to-coarse semantic relationships among image segments. From left to right: input image, our feature-induced segmentation, attention maps in the decoder of our clustering transformers. We use a clustering transformer to partition each image into 8 clusters, and show the attention map (colored in viridis color maps) of each cluster to all the image segments. We observe these clusters correlate better with image segments that carry more similar semantic meanings, e.g., the ‘head’ cluster attends more to body parts than background regions. Such correlation information implies the next-level groupings: ‘head’ will be grouped with ‘torso’ instead of ‘background’.

Table 5. Our method outperforms SegSort [26] and MaskContrast [57] among most categories on VOC val dataset. We adopt PSPNet [70]-ResNet50 as our backbone CNN. All models are supervisedly pre-trained on ImageNet [12]. Per-category performance is evaluated using IoU metric. We use the officially trained model and released code for inference MaskContrast. Our HSG demonstrates superior performance for semantic segmentation.

Table 6. Our method imposes less runtime overhead than other hierarchical clustering methods. All methods are conducted on a 640 × 640 image, which is hierarchically partitioned into 25, 16, 9 and 4 segments. While major latency comes from the pixel embedding network, HSG is still 17% faster than KMeans.


Fig. 11, we observe clusters of similar co-occurring object categories, such as a person riding a horse (in cerise) or a 'person' category region on Pascal VOC 2012 dataset using tSNE [41]. We represent each 'person' feature with the co-occurring object categories, and observe that features in the similar semantic context are clustered. As shown in
We iterate KMeans and NCut for 30 steps. Our hierarchical segmentations outperform others, which better pick out semantics at different levels of granularity. On DensePose [1], ground-truth labels are processed at two levels of semantics: person and body parts. a) F-score of region matching among ground truths and our hierarchical segmentations, b) F-score on fine (body-part-level) labels and c) F-score on coarse (person-level) labels. Hierarchical segmentation is needed to capture semantics across different granularity and our HSG outperforms others by large margin at every level.

### 5.4. Visual Results on Semantic Segmentation

We show some visual results of semantic segmentation on COCO-stuff and Potsdam in Fig. 12. Compared to SegSort [26], our results are more accurate and consistent. Our predicted segmentations also preserve boundaries more precisely than the baseline.

### 5.5. Unsupervised Semantic Segmentation

In the main paper, we perform unsupervised semantic segmentation by training from scratch on each dataset. Here, we carry out experiments by following the settings used in SegSort [26] and MaskContrast [57]. We summarize the quantitative results of unsupervised semantic segmentation using ImageNet-trained models on VOC in Table 5 according to IoU metric. We demonstrate the efficacy of our method, which outperforms SegSort [26] and MaskContrast [57] consistently among most categories by large margin. Most strikingly, our method is able to capture categories with complex structures better than baselines, e.g., chair (+5.8%), table (+14.8%) and plant (+10.6%).

### 5.6. Inference Latency on Clustering Transformer

We present the inference latency of different hierarchical clustering methods. We test on a 640 × 640 image, which is hierarchically partitioned into 25, 16, 9 and 4 segments. We iterate KMeans and NCut for 30 times. As shown in Table 6, our HSG imposes less runtime overhead than other clustering methods. While major latency comes from the backbone CNN, HSG is still 17% faster than KMeans.

### 5.7. Hierarchical Clustering Transformer

We mostly follow [5] to implement the transformer. The detailed architecture of the clustering transformer is presented in Fig. 13. The \((l + 1)^{th}\)-level transformer takes \(X_l\) as inputs and forwards to the encoder. The encoder contextually updates \(X_l\) to \(Y_l\) based on the pairwise correlation information of \(X_l\). Meanwhile, the decoder takes a set of query embeddings \(Q_{l+1}\) as inputs and outputs the next-level cluster centroids. \(Q_{l+1}\) can be considered as the initial representations of next-level clusters. As the clusterings should adapt with input statistics, we calculate the ‘mean’ and ‘std’ of \(Y_l\), followed by fc layers, and sum them with \(Q_{l+1}\) before inputting to the decoder. The decoder contextually updates \(Q_{l+1}\) to the next-level cluster centroids \(X_{l+1}\), which become the inputs to the next-level transformer. To calculate the clustering assignment, we do not use \(X_{l+1}\) but \(Z_{l+1}\), which shares the decoder layers with \(X_{l+1}\) but transformed by a separate fc layer. The soft clustering assignments are calculated as: \(C_{l+1}^{fc} = \text{softmax}\left(\frac{1}{m}Y_{l}^{T}Z_{l+1}\right); m\) is the feature dimension.

In particular, we replace LayerNorm with BatchNorm. We set number of heads to 4 in the attention module, and use 2 encoder (decoder) layers in each encoder (decoder) module. We set drop-out rate to 0.1 during training. For query embeddings \(Q_l\) at level \(l\), we randomly initiate and update them thru SGD.

In the clustering loss, the affinity matrix \(A\) among base level feature \(X_0\) is required to compute the modularity maximization loss. We construct \(A\) as a \(k\)-nearest neighbor (sparse) graph using the similarity of \(X_0\), where the entry value is set to 1. \(A\) is a binarized affinity matrix of a sparsified graph. For MSCOCO/VOC/COCO-stuff/Potsdam, we set \(k\) to 2 within an image and its augmented views, respectively. In such a manner, we encourage segment groupings across views. On Cityscapes/KITTI-STEP, cropped patches from each image instance are less likely to overlap. Without enforcing groupings across views, we search top 4 nearest neighbors among views (\(k = 4\)).

### 5.8. Hyper-Parameters and Experimental Setup

We next describe the same set of hyper-parameters shared across different datasets, and summarize the different settings in Table 7.

For all the experiments, we set the dimension of output embeddings to 128, temperature \(T\) to \(\frac{1}{2}\). We apply step-wise decay learning rate policy, with which learning rate is decayed by 32%, 56% and 75% of total training epochs. We obtain base-level grouping \(G_0\) by iterating spherical KMeans algorithm over pixel-wise feature \(V\) for 15 steps.
Figure 10. Sample retrieval results in MSCOCO for two images, baseball (Rows 1-3) and wii sport (Rows 4-6), based on our CNN features. Column 1 shows a query segment and Columns 2-5 are its nearest neighbour retrievals at the same level of the hierarchy. Segments at a coarser / finer level are shown in cyan (Rows 1,4) / red (Rows 2-3, 5-6). Coarser segment retrievals show that our feature learned from hierarchical groupings are reflective of the visual scene layout (For example, Row 1 all has the 3-person baseball pitching configuration despite drastic appearance variations), whereas finer segment retrievals show that our learned feature is precise at characterizing both the segment itself and the visual context around it (For example, the feature of the query segment (legs) in Row 3 is indicative of the pitcher pose on the baseball field). Such a holistic yet discriminative feature representation is discovered in a pure data-driven fashion without any semantic supervision.

and partition each cropped input to $4 \times 4$ segments. During training not testing, $G_0$ is then refined by coherent regions
Figure 11. Our visual representations encode contextual information of co-occurring objects. We visualize the average feature of person category region on Pascal VOC 2012 dataset using tSNE [41]. We use the feature mappings extracted with models trained from scratch on MSCOCO. We represent each person category region with the co-occurring object categories, and observe that features in the similar semantic context are clustered.

generated from the OWT-UCM procedure. For $G_1$ and $G_2$, we set $n_1$ and $n_2$ to 8 and 4. The whole framework is optimized using SGD. Notably, we only adopt rescaling, cropping, horizontal flipping, color jittering, gray-scale conversion, and Gaussian blurring for data augmentation. All the other different settings are presented in Table 7. For fair comparison with corresponding baselines, we apply different settings for training on MSCOCO and COCO-stuff.

Particularly, for MSCOCO, we adopt a two-stage learning strategy. We first train the model with smaller crop size ($224 \times 224$) and larger batch size (128), then fine-tune with larger crop size ($448 \times 448$) and smaller batch size (48). The models are trained and fine-tuned for 380 and 8 epochs. We do not use spherical KMeans to generate image oversegmentation in the first stage of training.

For inference, we only use single-scale image. For unsupervised semantic segmentation, we follow [26] to conduct nearest neighbor search to predict the semantic segmentation. We apply spherical KMeans algorithm over $V$ to derive pixel grouping $G_0$ and base cluster feature $X_0$. We search nearest neighbors using $X_0$ from the whole training dataset. We set $n_0$ the number of centroids in $G_0$ to $6 \times 6, 12 \times 24$ and $6 \times 12$ on Pascal/COCO-stuff/Potsdam, Cityscapes and KITTI-STEP dataset. On Cityscapes and
Our framework delivers better semantic segmentation on textural and aerial region parsing datasets. From top to bottom every three rows are visual results from COCO-stuff and Potsdam. The results are predicted via segment retrievals. Our results are more consistent and accurate within each category. Additionally, our segmentation predictions are better aligned with the boundaries. Our pixel-wise features encode more precise semantic information than baselines.

KITTI-STEP, we train the baselines with officially released code and test with our inference procedure. Otherwise, we report the numbers according to their papers.

We follow [26] and adopt the UCM-OWT procedure [2] to generate coherent region segmentations from contours. For MSCOCO and COCO-stuff, we follow [68] to detect edges by SE [14]. The detector is first pre-trained on BSDS dataset [43] with ground-truth edge labels. We start with threshold as 0.25 to binarize the UCM, followed by OWT-UCM to generate the segmentations. We gradually increase the threshold until the number of regions is smaller than 48. For Cityscapes/KITTI-STEP and Potsdam, we use PMI [27] to predict edges. The detector only considers co-occurring statistics among paired colors, and does not require any ground-truth label. The initial threshold is 0.05 and 0.5, which is increased until the number of regions is smaller than 1024 and 128.

5.9. Unsupervised Semantic Segmentation with ImageNet-trained Models

We use PSPNet [70] based on ResNet50 [21] as backbone CNN. The ResNet model is supervisedly pre-trained on ImageNet [12] and fine-tuned on Pascal VOC 2012. We follow mostly the same hyper-parameters, except that, we set batch size to 16, crop size to 448, base learning rate to 0.002 and weight decay to 0.0005. The models are fine-tuned for 15 epochs. We set $\lambda_E$, $\lambda_G$ and $\lambda_F$ to 1.0, 1.0 and 0.1, respectively.