Unsupervised Semantic Segmentation by Contrasting Object Mask Proposals: Supplementary Materials

Wouter Van Gansbeke[1*] Simon Vandenhende[1*] Stamatios Georgoulis[2] Luc Van Gool[1,2]
[1]KU Leuven/ESAT-PSI [2]ETH Zurich/CVL, TRACE

The supplementary materials include the pseudocode of our algorithm and the implementation details of the experimental evaluation. Additionally, we provide overclustering results in Section 4 and qualitative results for semi-supervised fine-tuning in Section 6.

1. Pseudo-code

Algorithm 1 shows the pseudo-code of MaskContrast. The saliency masks are obtained by running the public code of existing saliency estimators [6, 7]. The default hyper-parameter settings were used. A PyTorch implementation of our method and pre-computed saliency masks are made publicly available: `github.com/wvangansbeke/Unsupervised-Semantic-Segmentation/`.

2. Pre-training

This section describes the pre-training setup for the models included in the experiments section of the main paper. In the majority of cases, we were able to use the pre-trained weights made available by the authors of the respective works.

Co-Occurence. We adopt the training setup from the original work [4]. The features before the output layer of the network are used for the purpose of training a linear classifier and applying K-Means clustering.

Colorization. The pre-trained colorizer from Zhang *et al*. [11] is used. It is argued that the intermediate representations in the network will extract semantic information in order to solve the colorization task. As a consequence, it is non-trivial from what layer we should tap the features to tackle the semantic segmentation task. To resolve this, we tried using features from various intermediate layers, and report the best results when training a linear classifier or applying K-Means.

CMP. We follow the strategy from the colorization task for training a linear classifier or applying K-Means. The pre-trained model from Zhan *et al*. [10] is used.

Algorithm 1 Pseudocode of **MaskContrast**.

```
# f_q, f_k: encoder-decoder networks for query and key
# queue: dictionary of K prototype keys (CxK)
# m: momentum
# t: temperature
# H, W = height, width of an image x
# P : number of salient pixels in a batch

f_k.params = f_q.params # initialize
for (x, s) in loader:
   # load a batch with N samples and N saliency masks
   # constrain aug s.t. object area > threshold
   x_q, s_q = aug(x, s) # augmented version
   x_k, s_k = aug(x, s) # another augmented version

   q, aux = f_q.forward(x_q) # q: NxCxHxW, aux: NxHxW
   k, _ = f_k.forward(x_k) # k: NxCxHxW

   # salient objects are non zero
   valid_ids = s_q.nonzero() # valid_ids: Px1
   # remap each object to a unique id in {0..N-1}
   s_r = remap(s_q) # s_r: Px1

   # key prototypes: NxC
   p_k = bmm(k.view(N,C,H.W), s_k.view(N,H.W,1))
   p_k = normalize(p_k, dim=1) # L2-normalize
   p_k = p_k.detach() # no gradient to prototypes

   # select embeddings of salient objects: PxC
   q = index_select(q.view(H.WxC), index=valid_ids)

   # positive logits: PxN
   l_pos = mm(q.view(P,C), p_k.view(C,N))

   # negative logits: PxK
   l_neg = mm(q.view(P,C), queue.view(C,K))

   # logits: Px(N+K)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss: positives are the s_r-th
   MaskContrast_loss = CrossEntropyLoss(logits/t, s_r)

   # auxiliary BCE loss to prevent collapse
   aux_loss = BCE(aux, s_q)
   total_loss = MaskContrast_loss + aux_loss

   # SGD update: query network
   total_loss.backward()
   update(f_q.params)

   # momentum update: key network
   f_k.params = m*f_k.params+(1-m)*f_q.params

   # update dictionary
   enqueue(queue, p_k) # enqueue current prototypes
   dequeue(queue) # dequeue earliest prototypes
```

`bmm`: batch matrix multiplication; `mm`: matrix multiplication; `cat`: concatenation; `BCE`: binary cross-entropy loss; `remap`: custom function.

*Authors contributed equally

IIC. We follow the implementation strategy from [5].

Contrastive-Learning Methods. We used the weights from a ResNet-50 model pre-trained on ImageNet. The weights were made available by the authors of the respective works, i.e. the instance discrimination task [9], SWAV [1], MoCo v2 [3] and InfoMin [8]. In some cases, multiple variants of the model were released, e.g. when using different augmentation strategies during training. We chose the best available model each time.

The contrastive learning models were only pre-trained on ImageNet, as we could not see any substantial improvements from further pre-training them on the target dataset, i.e. PASCAL. To obtain dense predictions, we apply dilated convolutions in the last residual block. We use the features from the backbone for training a linear classifier or applying K-Means.

MaskContrast. We use a dilated ResNet-50 model with DeepLab-v3 head as outlined in the main paper. The final 1 x 1 convolutional layer is split into two linear heads. The first head predicts the pixel embeddings, while the second head predicts the saliency mask. During linear evaluation, we replace the final layer by a randomly initialized 1 x 1 convolutional layer. Other details were already provided in the paper.

3. Linear Classifier

This section describes the training setup used for the linear evaluation protocol. We train a 1 x 1 convolutional layer for 60 epochs using batches of size 16. The complete train set is used during training. We optimize the weights through stochastic gradient descent with momentum 0.9, weight decay 0.0001 and initial learning rate 0.1. The learning rate is reduced to 0.01 after 40 epochs. We found that increasing the train time, or modifying the learning rate did not improve the results.

4. Clustering

This section specifies how to obtain discrete class assignments by clustering the representations using K-Means. We follow the evaluation strategy from [5] to calculate the mean IoU metric. In particular, we first match the predicted clusters with the ground-truth classes using a Hungarian algorithm. We subsequently calculate the mean IoU from the re-assigned clusters and the ground-truth labels. We report the average from five runs.

Contrastive based methods. As described in Section 2, we apply K-Means clustering to the backbone features. The cluster assignments are upsampled to match the original image resolution, before applying the Hungarian algorithm.

Init.	MoCo v2		Sup.	
Sup. Sal.	✗	✓	✗	✓
Clusters				
21	35.0	38.9	41.6	44.2
50	41.4	48.8	46.2	51.4
100	43.3	49.5	47.3	52.5
200	45.0	51.1	48.5	53.6
500	48.1	54.2	51.3	57.0

Table 1. Overclustering on PASCAL with MaskContrast (MIoU). We use MoCo or supervised ImageNet initial weights, and supervised (✓) or unsupervised (✗) saliency.

IIC. No specific post-processing is required. We simply match the predicted clusters with the ground-truth classes following the original work [5].

Proxy-task based methods (Co-Occurence, Colorization, CMP). We select the features for applying K-Means as described in Section 2. The predictions are up-sampled to match the original image resolution before applying the Hungarian algorithm.

MaskContrast. We compute the mean embeddings of the foreground objects and apply K-Means using the L2-normalized feature vectors. All pixels belonging to the object are assigned the same label as the mean-pixel embedding after clustering. The predictions from the saliency estimation head are used to identify the background class. We match the predictions with the ground-truth classes using the Hungarian algorithm.

Overclustering results. K-Means does not employ any prior world knowledge, i.e. the ground-truth or target clusters are unknown. Therefore, it is unlikely that the predicted clusters will match the target ones on a complex and imbalanced dataset like PASCAL. To better understand the semantic structure discovered by the embedding space, we apply overclustering. In this case, a many-to-one mapping exists between the predicted and target clusters. Table 1 shows the results. The accuracy improves as we increase the number of predicted clusters. We hypothesize that local neighborhoods in the embedding space contain pixels of the same or visually similar objects, which benefits the performance when overclustering.

5. Semi-Supervised Learning

This section describes the semi-supervised learning setup. In each case, we report the average result for three randomly sampled splits.

ImageNet Pre-Trained Baseline. We load the pre-trained ImageNet weights into a ResNet-50 backbone with dilated convolutions. We use batch size of 8 and stochastic gradient descent with momentum 0.9 and learning rate 0.004 in all data regimes. The learning rate was selected after per-

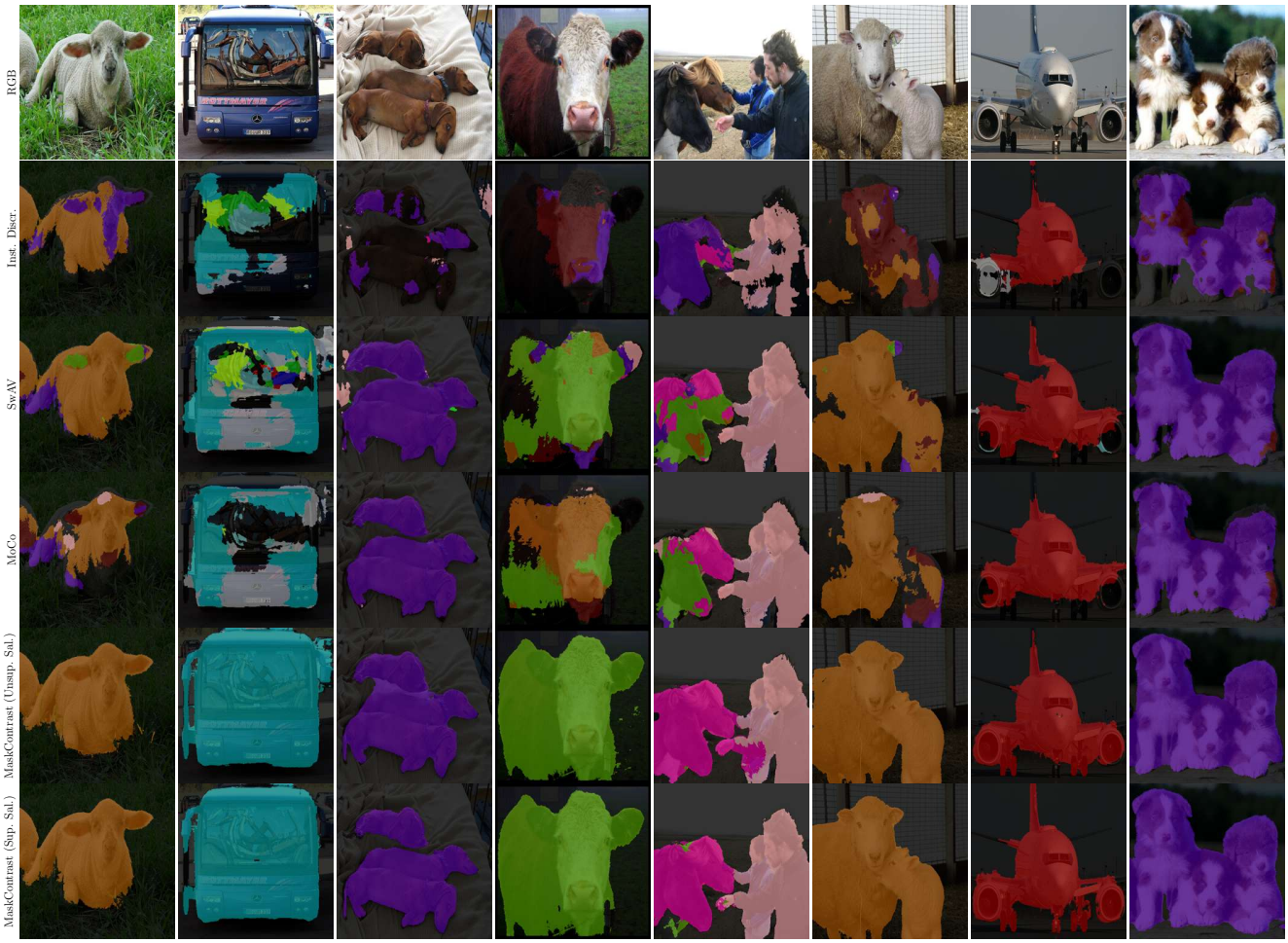

Figure 1. Qualitative comparison of the results after training a linear classifier on PASCAL. We use the MoCo weights to initialize our backbone.

forming a grid search. Additionally, we explored the use of different parameter groups with specific learning rate, e.g. the decoder used 10 times higher learning rate compared to the encoder. However, this did not result in any further improvements. We include a weight decay term 0.0001. A poly learning rate scheduler is used.

MaskContrast. We use a batch size of 8 and learning rate of 0.004 when fine-tuning with 5%, 12.5% and 100% of the labels. Differently, when using 1% and 2% of the labels, the learning rate is set to 0.001 for all layers in the network, except for the final convolutional layer which uses learning rate 0.1. The latter is well-motivated, since the complete network, including both encoder and decoder, were already pre-trained for the semantic segmentation task. The batch norm stats are frozen. We use stochastic gradient descent with momentum 0.9 and a weight decay term 0.0001. The learning rate is decayed using a poly learning rate scheduler.

6. Qualitative Results

Figure 1 shows a qualitative comparison when training a linear classifier on top of the pre-trained representations. We compare the representations learned by our method using an unsupervised (5th row) or supervised (6th row) saliency estimator, against the ones from instance discrimination (2nd row) [9], SWAV (3rd row) [1] and MoCo v2 (4th row) [2]. The qualitative results support the claim that our pixel embeddings learn semantically meaningful information.

References

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

[2] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[3] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

[4] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H Adelson. Learning visual groups from co-occurrences in space and time. *arXiv preprint arXiv:1511.06811*, 2015.

[5] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, 2019.

[6] Tam Nguyen, Maximilian Dax, Chaithanya Kumar Mummadi, Nhung Ngo, Thi Hoai Phuong Nguyen, Zhongyu Lou, and Thomas Brox. Deepusps: Deep robust unsupervised saliency prediction via self-supervision. In *NeurIPS*, 2019.

[7] Xuebin Qin, Zichen Zhang, Chenyang Huang, Chao Gao, Masood Dehghan, and Martin Jagersand. Basnet: Boundary-aware salient object detection. In *CVPR*, 2019.

[8] Yonglong Tian, C. Sun, Ben Poole, Dilip Krishnan, C. Schmid, and Phillip Isola. What makes for good views for contrastive learning. *arxiv preprint arXiv:2005.10243*, 2020.

[9] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018.

[10] Xiaohang Zhan, Xingang Pan, Ziwei Liu, Dahua Lin, and Chen Change Loy. Self-supervised learning via conditional motion propagation. In *CVPR*, 2019.

[11] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016.