CP^2: Copy-Paste Contrastive Pretraining for Semantic Segmentation

Feng Wang[1], Huiyu Wang[2], Chen Wei[2], Alan Yuille[2], and Wei Shen[3⋆]

[1] Department of Automation, Tsinghua University
[2] Department of Computer Science, Johns Hopkins University
[3] MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University

Abstract. Recent advances in self-supervised contrastive learning yield good image-level representation, which favors classification tasks but usually neglects pixel-level detailed information, leading to unsatisfactory transfer performance to dense prediction tasks such as semantic segmentation. In this work, we propose a pixel-wise contrastive learning method called CP^2 (**C**opy-**P**aste **C**ontrastive **P**retraining), which facilitates both image- and pixel-level representation learning and therefore is more suitable for downstream dense prediction tasks. In detail, we copy-paste a random crop from an image (the foreground) onto different background images and pretrain a semantic segmentation model with the objective of 1) distinguishing the foreground pixels from the background pixels, and 2) identifying the composed images that share the same foreground. Experiments show the strong performance of CP^2 in downstream semantic segmentation: By finetuning CP^2 pretrained models on PASCAL VOC 2012, we obtain 78.6% mIoU with a ResNet-50 and 79.5% with a ViT-S. Code and models are available at https://github.com/wangf3014/CP2.

Keywords: dense contrastive learning, semantic segmentation

1 Introduction

Learning invariant *image-level* representation and transferring to downstream tasks has became a common paradigm in self-supervised contrastive learning. Specifically, the objective of these methods is either to minimize the Euclidean (ℓ_2) distance [26,13] or cross entropy [5,6] between the *image-level* features of augmented views of the same image, or to distinguish the positive image feature from a set of negative image features by optimizing an InfoNCE [38] loss [28,12,14,10,11,41].

In spite of the success in downstream classification tasks, these contrastive objectives build on the assumption that every pixel in an image belongs to a single label and lack the perception of spatially varying image content. We argue that these *classification-oriented* objectives are not ideal for downstream dense prediction tasks such as semantic segmentation where the model should

⋆ Corresponding author, shenwei1231@gmail.com

distinguish different semantic labels in an image. For the task of semantic segmentation, current contrastive learning models may easily over-fit to learning the *image-level* representation and neglect pixel-level variances.

Moreover, there is an architectural misalignment in the current pretraining finetuning paradigm for downstream semantic segmentation tasks: 1) The semantic segmentation model usually requires a large atrous rate and a small output stride than those in the classification-oriented pretrained backbones [34,8]; 2) The finetuning of the well pretrained backbone and the randomly initialized segmentation head can be out of sync, *e.g.* the random head may generate random gradients that poison the pretrained backbone, negatively affecting the performance. These two issues prevent the classification-oriented pretrained backbone from facilitating dense prediction tasks such as semantic segmentation.

In this paper, we propose a novel self-supervised pretraining method designed for downstream semantic segmentation, named **C**opy-**P**aste **C**ontrastive **P**retraining (CP^2). Specifically, we pretrain a semantic segmentation model with Copy-Pasted input images which are composed by cropping random crops from a foreground image and pasting them onto different background images. Examples of the composed images are shown in Figures 2. Aside from the image-wise contrastive loss for learning instance discrimination [2,44,47,28], we introduce a pixel-wise contrastive loss to enhance dense prediction. The segmentation model is trained by maximizing cosine similarity between the foreground pixels while minimizing cosine similarity between the foreground and background pixels. Overall, CP^2 yields pixel specific dense representation and has two key advantages for downstream segmentation: 1) CP^2 pretrains both backbone and segmentation head, addressing the issue of architectural misalignment; 2) CP^2 pretrains the model with a dense prediction objective, building up the model's perception of spatially varying information in an image.

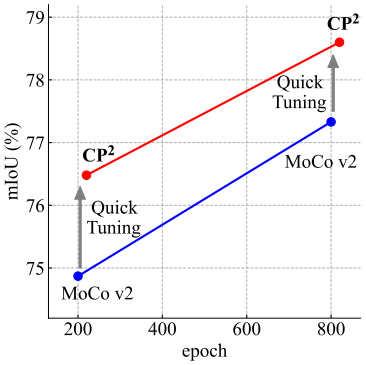

Fig. 1: **Quick Tuning MoCo v2 with CP^2**, evaluated by semantic segmentation on PASCAL VOC. A 20-epoch Quick Tuning with CP^2 yields large mIoU improvements.

Furthermore, we find that a considerably short period of CP^2 training is able to adapt pretrained classification-oriented models quickly to the semantic segmentation task and therefore yields better downstream performance. In particular, we first initialize the backbone with the weights of a pretrained classification-oriented model (*e.g.* a ResNet-50 [29] pretrained by MoCo v2 [12]), attach a randomly initialized segmentation head, and then tune the entire segmentation model by CP^2 for additional 20 epochs. As a result, the performance of the entire segmentation model on downstream semantic segmentation is significantly

improved, *e.g.* +**1.6**% mIoU on PASCAL VOC 2012 [23] dataset. We denote this training protocol as **Quick Tuning**, as it is efficient and practical for transfer learning from image-level instance discrimination to pixel-level dense prediction.

For technical details, we mostly follow MoCo v2 [12], including its architecture, data augmentation, and the instance contrastive loss, in order to fully isolate the effectiveness of our newly introduced copy-paste mechanism and dense contrastive loss, and therefore MoCo v2 [12] serves as a direct baseline to CP^2. In the empirical evaluations of semantic segmentation, the CP^2 200-epoch model achieves 77.6% mIoU on PASCAL VOC 2012 [23], outperforming the MoCo v2 [12] 200-epoch model by +**2.7**% mIoU. Also, as illustrated in Figure 1, the Quick Tuning protocol for CP^2 yields +**1.5**% and +**1.4**% mIoU improvements over the MoCo v2 200-epoch and 800-epoch model respectively. The improvement also generalizes to other segmentation datasets and vision transformers.

2 Related Work

Self-supervised learning and pretext tasks. Self-supervised learning for visual understanding leverages the intrinsic properties of images as the supervisory information for training, for which the capability of visual representation heavily depends on the formulation of pretext tasks. Prior to the recent popularity of instance discrimination [2,44,47,28], people have explored numerous pretext tasks, including image denoising and reconstruction [39,51,3], adversarial learning [19,20,22], and heuristic tasks such as image colorization [50], jigsaw puzzle [36,43], context and rotation prediction [18,33], and deep clustering [4].

The emergence of contrastive learning, or more specifically, the scheme of instance discrimination [2,44,47,28] has made a break-through in unsupervised learning, as MoCo [28] achieves superior transfer performance than supervised training in a wide range of downstream tasks. Inspired by this success, many follow-up works conduct deeper explorations in self-supervised contrastive learning and put forward different optimization objectives [26,41,42,5,6,54], model architectures [13], and training strategies [10,12,14].

Dense contrastive learning. To obtain better adaptation in dense prediction tasks, a recent work [37] extends the image-level contrastive loss into a pixel-level. Despite the extension of contrastive loss helps the model learn finer grained features, it is not able to establish the model's perception of spatially varying information, and therefore the model has to be re-purposed in downstream finetuning. More recent works try to enhance the model's understanding of positional information in images by encouraging the consistency of pixel-level representations [45], or by employing heuristic masks [24,1] and applying a patch-wise contrastive loss [30].

Copy-paste for contrastive learning. Copy-paste, *i.e.*, copying crops of one image and pasting them onto another image, once serves as a data augmentation method in *supervised* instance segmentation and semantic segmentation [25] for its simplicity and significant effect in enriching images' positional and semantic information. Similarly, by mixing images [49,31] or image crops [48] as data

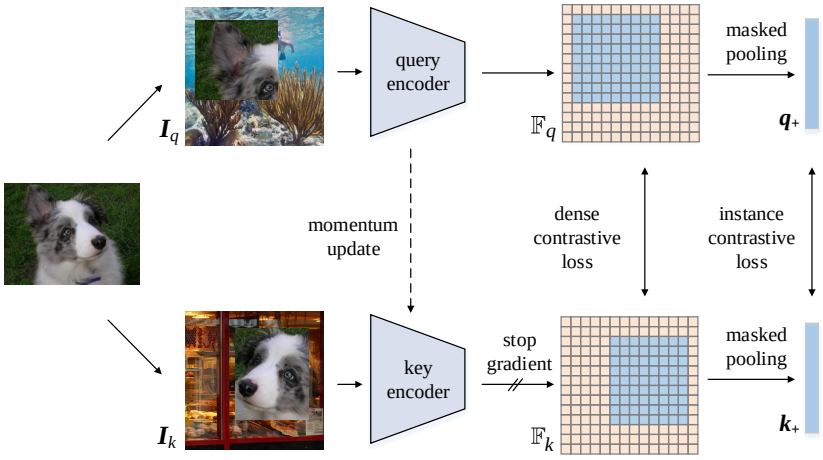

Fig. 2: **Pipeline**. We enrich the spatial information of unannotated images by randomly pasting two crops of foreground images onto different backgrounds. A dense contrastive loss is applied to their encoded feature maps and an instance contrastive loss is applied to the average of the foreground feature vectors (masked pooling). We follow the training architecture of momentum update in MoCo and BYOL.

augmentation, the *supervised* models also attain considerable performance improvements in various tasks. The use of copy-paste is also reported in recent works of self-supervised object detection [46,30]. Inspired by the success of copy-paste, we utilize this approach in our dense contrastive learning method as the self-supervisory information.

3 Method

In this section, we present our CP^2 objective and loss function for learning a pixel-wise dense representation. We also discuss our model architecture and propose a Quick Tuning protocol for efficient training of CP^2.

3.1 Copy-paste contrastive pretraining

We propose a novel pretraining method called CP^2, through which we desire the pretrained model to learn both instance discrimination and dense prediction. To this end, we manually synthesize image compositions by pasting foreground crops onto backgrounds. Specifically, as illustrated in Figure 2, we generate two random crops from the foreground image and then overlay them onto two different background images. The objective of CP^2 is to 1) discriminate the foreground from background within each composed image and 2) identify the composed images with the same foreground from negative samples.

Copy-paste. Given an original foreground image $\boldsymbol{I}^{fore}$, we first generate two different views of it $\boldsymbol{I}_q^{fore}$, $\boldsymbol{I}_k^{fore} \in \mathbb{R}^{224\times224\times3}$ by data augmentation, one being query and the other being the positive key. The augmentation strategy follows SimCLR [10] and MoCo v2 [12], *i.e.*, the image is first randomly resized and cropped to 224×224 resolutions followed by color jittering, gray scale, Gaussian blurring and horizontal flipping. Next, we generate one view for each of two random background images using the same augmentation, denoted as $\boldsymbol{I}_q^{back}$, $\boldsymbol{I}_k^{back} \in \mathbb{R}^{224\times224\times3}$. We compose the image pairs by binary foreground-background masks $\boldsymbol{M}_q, \boldsymbol{M}_k \in \{0,1\}^{224\times224}$, in which each element $m = 1$ denotes a foreground pixel and $m = 0$ denotes a background pixel. Formally, the composed images are generated by

$$\begin{aligned} \boldsymbol{I}_q &= \boldsymbol{I}_q^{fore} \odot \boldsymbol{M}_q + \boldsymbol{I}_q^{back} \odot (1 - \boldsymbol{M}_q)\,, \\ \boldsymbol{I}_k &= \boldsymbol{I}_k^{fore} \odot \boldsymbol{M}_k + \boldsymbol{I}_k^{back} \odot (1 - \boldsymbol{M}_k)\,, \end{aligned} \tag{1}$$

where $\odot$ denotes element-wise product. Now we get two composed images $\boldsymbol{I}_q$ and $\boldsymbol{I}_k$ who share the foreground source image but have different backgrounds.

Contrastive objectives. The composed images are then processed by a semantic segmentation model which we detail in Section 3.2. Given the input $\boldsymbol{I}_q$, the output of the segmentation model is a set of $r \times r$ features $\mathbb{F}_q = \{\boldsymbol{f}_q^i \in \mathbb{R}^C | i = 1, 2, \ldots, r^2\}$, where C is the number of output channels and r is the feature map resolution. For a 224×224 input image, $r = 14$ when the output stride $s = 16$. Among all the output features $\boldsymbol{f}_q \in \mathbb{F}_q$, we denote the foreground features, *i.e.*, the features that correspond to foreground pixels as $\boldsymbol{f}_q^+ \in \mathbb{F}_q^+ \subset \mathbb{F}_q$, where $\mathbb{F}_q^+$ is the foreground feature subset. Similarly, we have all the features $\boldsymbol{f}_k \in \mathbb{F}_k$ for the input image $\boldsymbol{I}_k$, among which the foreground features are denoted as $\boldsymbol{f}_k^+ \in \mathbb{F}_k^+ \subset \mathbb{F}_k$.

We use two loss terms, one *dense* contrastive loss and one *instance* contrastive loss. The contrastive loss $\mathcal{L}_{dense}$ learns local and fine-grained features by distinguishing between foreground and background features, helping with downstream semantic segmentation tasks, while the instance contrastive loss aims to keep the global, instance-level representation.

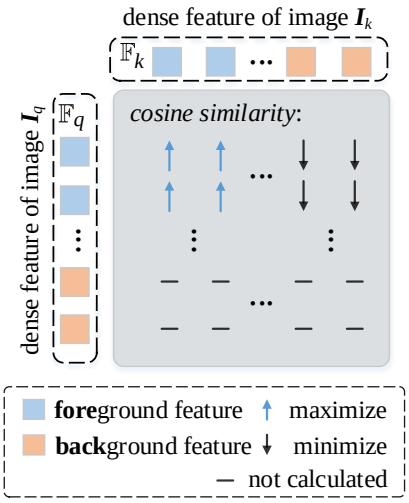

Fig. 3: **Dense contrastive loss** that maximizes the similarity of each foreground pair while minimizes that of each foreground-background pair.

In dense contrastive loss, we desire all the foreground features $\forall \boldsymbol{f}_q^+ \in \mathbb{F}_q^+$ of image $\boldsymbol{I}_q$ to be similar to all the foreground features $\forall \boldsymbol{f}_k^+ \in \mathbb{F}_k^+$ of image $\boldsymbol{I}_k$, and

be dissimilar to the background features $\mathbb{F}_k^- = \mathbb{F}_k \setminus \mathbb{F}_k^+$ of image $\boldsymbol{I}_k$. Formally, for each foreground feature $\forall \boldsymbol{f}_q^+ \in \mathbb{F}_q^+$ and $\forall \boldsymbol{f}_k^+ \in \mathbb{F}_k^+$, the dense contrastive loss is obtained by

$$\mathcal{L}_{dense} = -\frac{1}{|\mathbb{F}_q^+||\mathbb{F}_k^+|} \sum_{\forall \boldsymbol{f}_q^+ \in \mathbb{F}_q^+, \forall \boldsymbol{f}_k^+ \in \mathbb{F}_k^+} \log \frac{\exp\left(\boldsymbol{f}_q^+ \cdot \boldsymbol{f}_k^+ / \tau_{dense}\right)}{\sum_{\forall \boldsymbol{f}_k \in \mathbb{F}_k} \exp\left(\boldsymbol{f}_q^+ \cdot \boldsymbol{f}_k / \tau_{dense}\right)}, \tag{2}$$

where τ_{dense} is a temperature coefficient. This dense contrastive loss is also illustrated in Figure 3. Following supervised contrastive learning methods [32,52], we put the summation outside the log.

Besides the dense contrastive loss, we keep the instance contrastive loss that aims to learn the global, instance-level representation. We mostly follow the practice of MoCo [28,12], where given the query image, the model is required to distinguish the positive key from a memory bank of negative keys. But in our case, we use the composed image $\boldsymbol{I}_q$ as the query image, and the composed image $\boldsymbol{I}_k$ as the positive key image that shares the foreground with image $\boldsymbol{I}_q$. In addition, instead of using the global average pooling feature as the representation in MoCo, we use the normalized masked averaging of only the foreground features as illustrated in Figure 2. Formally, the instance contrastive loss is computed as

$$\mathcal{L}_{ins} = -\log \frac{\exp(\boldsymbol{q}_+ \cdot \boldsymbol{k}_+ / \tau_{ins})}{\exp(\boldsymbol{q}_+ \cdot \boldsymbol{k}_+ / \tau_{ins}) + \sum_{n=1}^{N} \exp(\boldsymbol{q}_+ \cdot \boldsymbol{k}_n / \tau_{ins})}, \tag{3}$$

where $\boldsymbol{q}_+$, $\boldsymbol{k}_+$ are normalized masked averaging of $\mathbb{F}_q^+$ and $\mathbb{F}_k^+$:

$$\boldsymbol{q}_+ = \frac{\sum_{\forall \boldsymbol{f}_q^+ \in \mathbb{F}_q^+} \boldsymbol{f}_q^+}{||\sum_{\forall \boldsymbol{f}_q^+ \in \mathbb{F}_q^+} \boldsymbol{f}_q^+||_2}, \ \boldsymbol{k}_+ = \frac{\sum_{\forall \boldsymbol{f}_k^+ \in \mathbb{F}_k^+} \boldsymbol{f}_k^+}{||\sum_{\forall \boldsymbol{f}_k^+ \in \mathbb{F}_k^+} \boldsymbol{f}_k^+||_2}. \tag{4}$$

$\boldsymbol{k}_n$ denotes the representations of negative samples from a memory bank [28,44] of N vectors, and τ_{ins} is a temperature coefficient.

The total loss $\mathcal{L}$ is simply a linear combination of the dense and the instance contrastive loss

$$\mathcal{L} = \mathcal{L}_{ins} + \alpha \mathcal{L}_{dense}, \tag{5}$$

where α is a trade-off coefficient for the two losses.

3.2 Model architecture

Next, we discuss in detail our CP^2 model architecture that consists of a backbone and a segmentation head for both pretraining and finetuning. Different from existing contrastive learning frameworks [28,12] that pretrain only the backbone, CP^2 enables the pretraining of both the backbone and the segmentation head, almost the same architecture as the one used for downstream segmentation tasks. In this way, CP^2 prevents the finetuning misalignment issue (Section 1), *i.e.*, finetuning the downstream models with a well-pretrained backbone and a randomly initialized head. This misalignment can require careful hyper-parameter

tuning (*e.g.*, a larger learning rate on the head) and result in degradation of the transferring performance, especially when a heavy randomly initialized head is used. Therefore, CP^2 is able to achieve better performance for segmentation and also enables the usage of stronger segmentation heads.

In particular, we study two families of backbones, CNNs [29] and vision transformers [21]. For CNN backbones, we use the original ResNet-50 [29] with a 7×7 convolution as the first layer, instead of an inception stem [40] commonly used in segmentation tasks [7,8,9]. This setting ensures fair comparisons with previous self-supervised learning methods. In order to adapt the ResNet backbone to segmentation, we follow common segmentation settings [7,8,28] and use atrous rate 2 and stride 1 for the 3×3 convolutions in the last stage. For vision transformer backbones, we choose ViT-S [21] with 16×16 patch size, which has a similar number of parameters as ResNet-50. Note that both of our ResNet-50 and ViT-S have an output stride $s = 16$ which makes our backbones compatible with most existing segmentation heads.

Given the backbone output features with an output stride $s = 16$, we study two types of segmentation heads. By default, we employ the common DeepLab v3 [8] segmentation head (*i.e.* ASPP head with image pooling), as it is able to extract multi-scale spatial features and yield very competitive results. In addition to the DeepLab v3 ASPP head, we also study the lightweight FCN head [34] usually adopted for evaluation of self-supervised learning methods.

On top of the backbone and segmentation head that are trained for both pretraining and finetuning, we make as little change as possible. Specifically, for CP^2 pretraining, we add two 1×1 convolution layers to the segmentation head output, projecting the pixel-wise dense features to a 128-dimensional latent space (*i.e.*, $C = 128$). The latent features at each pixel are then ℓ_2 normalized individually. Our dense projection design is analogous to the 2-layer MLP design in common contrastive learning frameworks [12] followed by an ℓ_2 normalization. After the CP^2 training converges, we simply replace the 2-layer convolution projection by a segmentation output convolution that projects the segmentation head feature to the number of output classes, similar to the typical design in image-wise contrastive frameworks [28,10]. Following MoCo [28], we momentum update the key encoder consisting of both the backbone and the segmentation head by the weights in the query encoder.

3.3 Quick Tuning

In order to train our CP^2 models quickly within a manageable computational budget, we propose a new training protocol called Quick Tuning that initializes our backbone with existing backbone checkpoints available online. These backbones typically have been trained by image-wise contrastive loss with extremely long schedules (*e.g.* 800 epochs [12] or 1000 epochs [10]). On top of these existing checkpoints that encode good image-level semantic representations, we apply our CP^2 training for just a few epochs (*e.g.*, 20 epochs) in order to finetune the representation still on ImageNet without human labels but for semantic segmentation. Specifically, we attach a randomly initialized segmentation head

on top of the pretrained backbone with proper atrous rates and train this entire segmentation model with our CP^2 loss function. Finally, the learned segmentation model on ImageNet without using any label is further finetuned on various downstream segmentation datasets for evaluation of the learned representations.

Quick Tuning enables efficient and practical training for self-supervised contrastive learning, as it exploits the heavily-pretrained self-supervised backbones and let them quickly adapt to the desired objective or downstream tasks. According to our empirical evaluations, 20 epochs of Quick Tuning is sufficient to yield significant improvements on various datasets (for example, the finetuning mIoU on PASCAL VOC 2012 is improved by 1.6% after a 20-epoch Quick Tuning). This is particularly helpful for pretraining segmentation models efficiently, because segmentation models are usually much heavier than the backbone in terms of computational cost due to the atrous convolutions in the backbone and the ASPP module. In this case, Quick Tuning saves a large amount of computational resources by demonstrating significant improvements with a short period of segmentation model self-supervised pretraining.

4 Experiments

4.1 Experimental setup

Our MoCo v2 implementation follows the official open source code [12], and our semantic segmentation implementation uses the MMSegmentation [15] library.

Datasets. We pretrain CP^2 and the baselines on ImageNet [17] (∼1.28 million training images) and finetune on semantic segmentation tasks of PASCAL VOC [23], Cityscapes [16], and ADE20k [53]. For PASCAL VOC, we train on the augmented training set [27] with 10582 images and evaluate on VOC2012 validation set. For Cityscapes, we train on the "train-fine" set with 2975 images and evaluate on its validation set. For ADE20k, we train on the training set with 20210 images and evaluate on the validation set.

Segmentation and projection heads. Our DeepLab v3 ASPP head follows the default setting in MMSegmentation which uses 512 output channels for both the atrous convolutions and the output projection. Our CP^2 projection head consists of two layers of 512-channel 1×1 convolutions, ReLU, and a $C = 128$ channel 1×1 convolution. For the FCN head, we follow the settings in prior works [28,30] for fair comparison, *i.e.*, two layers of 256-channel 3×3 convolutions with atrous rate=6 followed by BN and ReLU. The CP^2 projection for the FCN-based model consists of two layers of 256-channel 1×1 convolutions, ReLU, and a $C = 128$ channel 1×1 convolution.

Baselines. We compare CP^2 with the self-supervised contrastive learning methods with classification-oriented [28,12,10,41,26], detection-oriented [46,30], and dense prediction [45] objectives. All the pretrained ResNet-50 models of are downloaded from their official implementations. For InsLoc [46], we use the backbone of its ResNet50-FPN model which has been pretrained for 400 epochs. For DetCon [30], we use the model pretrained 1000 epochs with DetCon-B manner

for the most competitive baseline. The pretrained ViT-S model of MoCo v2 is borrowed from DINO [6]. Moreover, to compare with supervised methods, we load a pretrained ResNet-50 model in torchvision official model zoo, which has a top-1 accuracy of 76.13% on ImageNet validation set [17].

Hyper-parameters. For ResNet-backed models, we pretrain by SGD optimizer with 0.03 learning rate, 0.9 momentum, 0.0001 weight decay, and a mini-batch size of 256 on ImageNet. We finetune them by SGD with 0.9 momuntum, 0.0005 weight decay, and 0.003, 0.01, 0.01 learning rate for PASCAL, Cityscapes, and ADE20k, respectively. For ViT-backed models, we also pretrain with a mini-batch size of 256 on ImageNet but apply an AdamW [35] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, 0.00005 learning rate, 0.01 weight decay for both pretraining and finetuning. We pretrain and finetune with 4 GPUs. We find that for CP^2 pretrained models, the use of weight decay in finetuning stage usually leads to ∼0.2% mIoU decrease. This is possibly because for the baseline methods, the segmentation head is randomly initialized in finetuning and relies on weight decay for better generalization. However, as CP^2 pretrains both the backbone and segmentation head with a proper weight decay, the ewights of segmentation head have been decayed into a lower scale and do not require further decaying during finetuning. Thus, in the finetuning stage, we use weight decay for those baseline models with random segmentation head and turn off weight decay for CP^2 models. We adopt a memory bank of $N = 65536$, $C = 128$ dimensional vectors, with $\boldsymbol{k}_+$, the normalized masked average representation of image I_k in the current mini-batch enqueued and the oldest vectors dequeued. For instance contrastive loss $\mathcal{L}_{ins}$, we set the temperature $\tau_{ins} = 0.2$ in accordance with MoCo v2 [12]. We assign a weight of $\alpha = 0.2$ and set the temperature $\tau_{dense} = 1$ for $\mathcal{L}_{dense}$, according to grid search. For PASCAL VOC, we use crop size 512×512 and train with batch size 16 for 40k iterations. For Cityscapes, we use crop size 512×1024 and train with batch size 8 for 60k iterations. For ADE20k, we use crop size 512×512 and train with batch size 16 for 80k iterations.

4.2 Main results

MoCo v2 [12] is a direct baseline to our method as we follow its model architecture, contrastive formulation, and the technical setups. For ease of reference, we use the following abbreviations to denote MoCo v2 pretrained models:

- **r.200**, **r.800**: ResNet-50 pretrained by MoCo v2 for 200, 800 epochs.
- **v.300**: ViT-S/16 pretrained by MoCo v2 for 300 epochs.

Results with DeepLab v3 segmentation head. We first present the evaluation results of DeepLab v3 semantic segmentation models (a backbone attached by an ASPP head with image pooling) [8]. As summarized in Table 1, CP^2 achieves 77.6% mIoU on PASCAL VOC with 200 epochs pretraining from scratch using a ResNet-50 backbone, which outperforms MoCo v2 by +**2.7**%. Also, the Quick Tuning protocol is demonstrated to be effective as it yields +**1.6**% mIoU on PASCAL VOC when tuning a 200-epoch MoCo v2 checkpoint

Table 1: **Evaluation results (mIoU) with DeepLab v3 segmentation head**. QT denotes Quick Tuning with CP^2 initialized by a MoCo v2 pre-trained backbone. Our results are marked in gray. The best results are **bolded**. Epochs that are consumed by the initialization model are de-emphasized.

method	backbone	epoch	PASCAL	Cityscapes	ADE20k
supervised	ResNet-50	-	76.0	76.3	39.5
MoCo [28]	ResNet-50	200	73.2	75.8	38.6
SimCLR [10]	ResNet-50	1000	77.3	76.5	40.1
BYOL [26]	ResNet-50	300	77.4	76.5	40.2
InfoMin [41]	ResNet-50	800	77.2	76.5	39.6
InsLoc [46]	ResNet-50	400	75.6	76.3	40.3
DetCon [30]	ResNet-50	1000	78.1	77.1	40.6
PixPro [45]	ResNet-50	400	77.5	76.6	40.3
MoCo v2 [12]	ResNet-50	200	74.9	76.2	39.2
CP^2	ResNet-50	200	77.6	77.3	40.5
CP^2 QT r.200	ResNet-50	200+20	76.5	77.2	40.7
MoCo v2 [12]	ResNet-50	800	77.2	76.4	39.7
CP^2 QT r.800	ResNet-50	800+20	**78.6**	**77.4**	**41.3**
MoCo v2 [12]	ViT-S/16	300	78.8	77.2	41.3
CP^2 QT v.300	ViT-S/16	300+20	**79.5**	**77.6**	**42.2**

for only another 20 epochs with CP^2, and +**1.4**% mIoU when tuning an 800-epoch MoCo v2 checkpoint. Moreover, by Quick Tuning the 800-epoch MoCo v2 model, CP^2 achieves the best performance among all ResNet-50 based methods on three evaluated datasets. Notably, it yields +**0.5**% mIoU on PASCAL VOC and +**0.7**% mIoU on ADE20k compared with the most competitive DetCon [30], in spite of DetCon's heavier computational cost and longer training schedule. For ViT based models, CP^2 also outperforms its MoCo v2 baseline by +**0.7**% mIoU on PASCAL, +**0.4**% mIoU on Cityscapes, and +**0.9**% mIoU on ADE20k when Quick Tuning for another 20 epochs.

Results with FCN segmentation head. Table 2 summarizes the evaluation results with the light-weight FCN [34] head (two hidden layers of atrous convolutions and a classification layer). Similarly, CP^2 achieves the highest mIoU on the three datasets with both ResNet-backed and ViT-backed architectures. In particular, compared to the baseline MoCo v2, CP^2 obtains up to +**1.0**% mIoU on PASCAL using ResNet-50 and +**0.9**% mIoU using ViT-S.

Overall, CP^2 yields significant performance improvements in the downstream task of semantic segmentation with both strong (ASPP) and light-weight (FCN) segmentation heads. Aside from demonstrating the effectiveness and robustness of CP^2 in terms of different segmentation heads, we further dissect the performance improvements from various factors and components in our ablation study. The more in-depth discussion and results in the ablation study show that our improvements on downstream segmentation tasks do not merely come from pretraining the segmentation head.

Table 2: **Evaluation results (mIoU) with FCN head**. QT denotes Quick Tuning with CP^2 initialized by a MoCo v2 pre-trained backbone. Our results are marked in gray. The best results are **bolded**. Epochs that are consumed by the initialization model are de-emphasized.

method	backbone	epoch	PASCAL	Cityscapes	ADE20k
supervised	ResNet-50	-	73.7	75.8	37.4
MoCo v2 [12]	ResNet-50	200	74.4	75.8	37.4
CP^2	ResNet-50	200	75.4	76.4	38.4
CP^2 QT r.200	ResNet-50	200+20	75.2	76.4	38.0
MoCo v2 [12]	ResNet-50	800	74.8	75.9	37.9
CP^2 QT r.800	ResNet-50	800+20	**75.7**	**76.5**	**39.2**
MoCo v2 [12]	ViT-S/16	300	77.7	76.6	40.4
CP^2 QT v.300	ViT-S/16	300+20	**78.6**	**77.0**	**41.2**

4.3 Ablation study

In this section, we first question if CP^2 benefits downstream semantic segmentation tasks only because it offers a pretrained segmentation head, or the proposed dense contrastive loss ($\mathcal{L}_{dense}$) also helps? Second, we explore the effect of various types of copy-paste masks, ranging from a simple rectangle mask to masking random patches. Third, we study the effect of the training schedule in Quick Tuning. Fourth, we study the effect of two key hyper-parameters, the loss coefficient (α) and temperature (τ_{dense}) of the dense contrastive loss.

Segmentation head initialization. Intuitively, CP^2 can benefit the downstream segmentation tasks in two aspects. First, CP^2 provides the downstream semantic segmentation with a well-pretrained decoder head. Second, CP^2 pretrains the model with a segmentation-oriented objective (the dense contrastive loss), which is expected to enable the backbone to extract pixel-level features. To ablate the benefit of each component, we dissect the CP^2 trained model and examine the benefits of its backbone and segmentation head respectively.

Table 3 summarizes the results. By pretraining the ResNet-50 based model for 200 epochs from scratch and finetuning on PASCAL, CP^2 achieves 77.6% mIoU, which is 2.7% higher than that of its MoCo v2 baseline. If we use the CP^2 pretrained backbone but still randomly initialize the segmentation head in the finetuning stage, it also attains 1.4% points higher mIoU than MoCo v2, demonstrating that the backbone representation is also improved for downstream segmentation thanks to our segmentation-oriented objective.

Similarly, the same phenomenon is observed for CP^2 Quick Tuning protocol as well. For example, by Quick Tuning the MoCo v2 800-epoch ResNet-50 and finetuning on PASCAL, we obtain +1.4% mIoU over MoCo v2. While finetuning the CP^2 pretrained backbone with a randomly initialized segmentation head also yields +1.0% mIoU over its baseline.

According to the observation above, CP^2 yields both a better backbone and a pretrained segmentation head for downstream semantic segmentation, thanks

Table 3: **Ablation study of segmentation head pretraining** on PASCAL VOC. The results are based on ASPP segmentation head. We use Quick Tuning for CP^2 in the settings of (ResNet-50, 800 epochs) and (ViT-S/16, 300 epochs).

mode	backbone	head	mIoU
ResNet-50, 200 epochs	MoCo v2	random	74.9
	CP^2	random	76.3 (+1.4)
	CP^2	**CP^2**	**77.6** (+**2.7**)
ResNet-50, 800 epochs	MoCo v2	random	77.2
	CP^2 QT	random	78.2 (+1.0)
	CP^2 QT	**CP^2** QT	**78.6** (+**1.4**)
ViT-S/16, 300 epochs	MoCo v2	random	78.8
	CP^2 QT	random	79.3 (+0.5)
	CP^2 QT	**CP^2** QT	**79.5** (+**0.7**)

to our design of CP^2 that enables segmentation head pretraining and employs the segmentation-oriented contrastive objective.

Foreground-background mask. We further explore the effect of various types of copy-paste masking for CP^2. The experiments are conducted using our Quick Transfer protocol, initialized by the 800-epoch MoCo v2 checkpoint when using a ResNet-50 backbone and the 300-epoch MoCo v2 checkpoint when using a ViT-S backbone, on PASCAL VOC dataset.

First, we consider a baseline when copy-paste masking is not applied. Specifically, the augmented views of the foreground image will *not* be composed with random backgrounds but serve as the model inputs directly. And the model will be trained with only the image-wise contrastive loss ($\mathcal{L}_{ins}$) since there is no background to construct the dense contrastive loss. In other words, the segmentation model (a backbone followed by segmentation head) is simply trained with a MoCo loss that operates on average pooled features over the whole image. We denote this setup as *no copy-paste*. As shown in Table 4, *no copy-paste* yields relatively poor performance. Compared to the baseline performance of 77.2% mIoU with ResNet-50 and 78.8% mIoU with ViT-S (Table 1), *no copy-paste* training attains only marginal improvements of 0.4% and 0.1% mIoU respectively. This result indicates that pretraining the segmentation head with classification-oriented objectives cannot yield significant improvements on the downstream performance for semantic segmentation. This suggests that the improvement of CP^2 mainly comes from the copy-paste training and the dense contrastive loss.

We also explore various types of image masking, including the self-attention masks generated by DINO [6] and different shapes of random masking. The random masks include rectangular masks, polygon masks, random blocks, and random patches, for which we provide examples in Figure 4. In order to ablate the influence of mask area, we limit the foreground ratio of each random mask to 0.5∼0.8, which we find usually yields better empirical results. The self-attention masks are generated by the DINO [6] pretrained ViT-B/16 model. Specifically, for each image, we average their 12 heads of last layer self-attentions and then

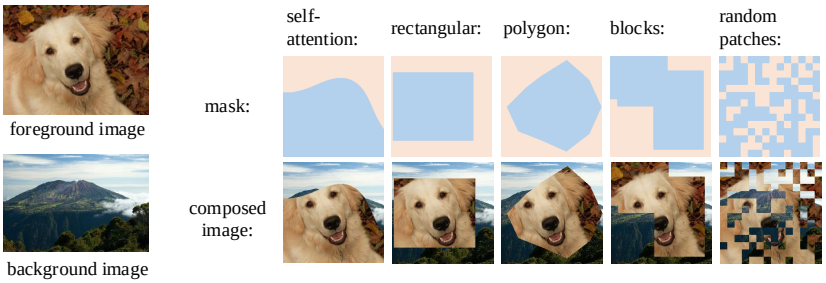

Fig. 4: **Examples of masking strategies and composed images**. The self-attention mask (DINO mask) is smoothed by Gaussian blur.

Table 4: **Evaluation results of foreground-background masks** on PASCAL VOC. Note that for the full mask, the models are trained without dense contrastive loss. Our default setting is marked in gray.

mode	random	mIoU	
		ResNet-50	ViT-S/16
baseline MoCo v2	-	77.2	78.8
no copy-paste	-	77.6	78.9
DINO self-attention mask [6]	✗	77.9	79.3
rectangular mask	✓	**78.6**	**79.5**
polygon mask	✓	78.1	79.0
random blocks	✓	77.3	78.7
random patches	✓	75.3	78.9

up-sample the averaged attention map to the original shape of the image. For denoising purpose, we also apply Gaussian blur to the self-attention DINO masks.

Empirically, the random rectangular mask achieves the highest performance with both ResNet-50 and ViT-S/16 models in Table 4. This is possibly because the rectangular masks contain mostly the real continuous foreground of an image (compared with random patch masks) and also introduce randomness in the masks (compared with DINO masks). This result indicates that simple foreground-background information is sufficient for the models to learn semantic features, and applying the random rectangular masks yields consistent performance gain with both of the backbone architectures. Therefore, we use this simple and easy-to-implement masking strategy in CP^2 for the best performance.

Moreover, the ViT-backed model performs more robustly to various shapes of the mask than the ResNet-backed model. It is worth-noting that the **random patches** mask appears to mislead the ResNet-backed model as it yields 75.3% mIoU, which is 3.3% lower than the result of rectangular mask and even 1.9% lower than the MoCo v2 baseline before Quick Tuning. But the ViT model is robust to this random patch masking although no improvement is observed.

Table 5: **Evaluation results of hyper-parameter search** on PASCAL VOC. The results are based on ResNet50-ASPP models, where the base backbone is loaded from the MoCo v2 pretrained ResNet50 for 800 epochs. Our default setting is marked in gray. The best results are **bolded**.

(a) **loss weight and temperature**

weight	temperature(τ_{dense}) 2	1	0.5	0.2
10	77.4	77.0	76.9	77.2
1	77.3	77.9	77.3	77.4
0.5	77.2	78.0	77.3	77.1
0.2	76.9	**78.6**	77.3	76.7
0.1	76.0	77.7	77.5	75.8

(b) **Quick Tuning epochs**

epoch	mIoU
0	77.2
10	77.7 (+0.5)
20	**78.6 (+1.4)**
40	78.7 (+1.5)

Hyper-parameter search. It is important to consider the trade-off between the image-wise and pixel-wise objective. Two hyper-parameters, the weight and temperature (τ_{dense}) of pixel-wise contrastive loss (setting the weight of image-wise loss to 1), play decisive roles influencing this trade-off. We conduct grid search of these two parameters and summarize the results in Table 5a. As reported, the parameter pair we use in the main experiments, (weight=0.2, τ_{dense}=1), achieves the peak performance. For better efficiency, we recommend a training time of 20 epochs on ImageNet when using Quick Tuning. As listed in Table 5b, 20 epochs of Quick Tuning yields 78.6% mIoU (1.4% higher than MoCo v2) while the 40-epoch Quick Tuning brings only 0.1% extra improvement.

5 Conclusion

In this work, we propose a segmentation-oriented contrastive learning method CP^2, in which we encourage the model to learn both image-level and pixel-level representation by pretraining it with both instance and dense contrastive losses. We point out two key merits of CP^2: First, CP^2 trains the entire semantic segmentation model, pretraining both the backbone and decoder head, which directly addresses the issue of architectural misalignment when finetuning in downstream semantic segmentation. Second, CP^2 is trained on copy-pasted images (images with foreground and background) with a pixel-level dense objective, which helps the model learn localized or spatially varying features that benefit the downstream segmentation task. Our results demonstrate a significant margin over existing methods on semantic segmentation.

Acknowledgements

This work was supported by ONR N00014-21-1-2812, NSFC 62176159, Natural Science Foundation of Shanghai 21ZR1432200 and Shanghai Municipal Science and Technology Major Project 2021SHZDZX0102.

References

1. Arbeláez, P., Pont-Tuset, J., Barron, J.T., Marques, F., Malik, J.: Multiscale combinatorial grouping. In: CVPR (2014) 3
2. Bachman, P., Hjelm, R.D., Buchwalter, W.: Learning representations by maximizing mutual information across views. In: NeurIPS (2019) 2, 3
3. Belghazi, M., Oquab, M., Lopez-Paz, D.: Learning about an exponential amount of conditional distributions. In: NeurIPS (2019) 3
4. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: ECCV (2018) 3
5. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. In: NeurIPS (2020) 1, 3
6. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV (2021) 1, 3, 9, 12, 13
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE TPAMI (2017) 7
8. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. In: CVPR (2017) 2, 7, 9
9. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: ECCV (2018) 7
10. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: ICML (2020) 1, 3, 5, 7, 8, 10
11. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.: Big self-supervised models are strong semi-supervised learners. In: NeurIPS (2020) 1
12. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020) 1, 2, 3, 5, 6, 7, 8, 9, 10, 11
13. Chen, X., He, K.: Exploring simple siamese representation learning. In: CVPR (2021) 1, 3
14. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: ICCV (2021) 1, 3
15. Contributors, M.: MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation (2020) 8
16. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: CVPR (2016) 8
17. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: CVPR (2009) 8, 9
18. Doersch, C., Gupta, A., Efros, A.A.: Unsupervised visual representation learning by context prediction. In: ICCV (2015) 3
19. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. In: ICLR (2016) 3
20. Donahue, J., Simonyan, K.: Large scale adversarial representation learning. In: NeurIPS (2019) 3
21. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.:

An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2021) 7
22. Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., Courville, A.: Adversarially learned inference. In: ICLR (2016) 3
23. Everingham, M., Eslami, S.A., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: A retrospective. IJCV (2015) 3, 8
24. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. IJCV (2004) 3
25. Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T.Y., Cubuk, E.D., Le, Q.V., Zoph, B.: Simple copy-paste is a strong data augmentation method for instance segmentation. In: CVPR (2021) 3
26. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., Valko, M.: Bootstrap your own latent a new approach to self-supervised learning. In: NeurIPS (2020) 1, 3, 8, 10
27. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: ICCV (2011) 8
28. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020) 1, 2, 3, 6, 7, 8, 10
29. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 2, 7
30. Hénaff, O.J., Koppula, S., Alayrac, J.B., Oord, A.v.d., Vinyals, O., Carreira, J.: Efficient Visual Pretraining with Contrastive Detection. In: ICCV (2021) 3, 4, 8, 10
31. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. In: ICLR (2019) 3
32. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. In: NeurIPS (2020) 6
33. Komodakis, N., Gidaris, S.: Unsupervised representation learning by predicting image rotations. In: ICLR (2018) 3
34. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015) 2, 7, 10
35. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019) 9
36. Noroozi, M., Favaro, P.: Unsupervised learning of visual representations by solving jigsaw puzzles. In: ECCV (2016) 3
37. O Pinheiro, P.O., Almahairi, A., Benmalek, R., Golemo, F., Courville, A.C.: Unsupervised learning of dense visual representations. In: NIPS (2020) 3
38. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018) 1
39. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: CVPR (2016) 3
40. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: AAAI (2017) 7
41. Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., Isola, P.: What makes for good views for contrastive learning? In: NeurIPS (2020) 1, 3, 8, 10
42. Wei, C., Wang, H., Shen, W., Yuille, A.: CO2: Consistent contrast for unsupervised visual representation learning. In: ICLR (2021) 3

43. Wei, C., Xie, L., Ren, X., Xia, Y., Su, C., Liu, J., Tian, Q., Yuille, A.L.: Iterative reorganization with weak spatial constraints: Solving arbitrary jigsaw puzzles for unsupervised representation learning. In: CVPR (2019) 3
44. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: CVPR (2018) 2, 3, 6
45. Xie, Z., Lin, Y., Zhang, Z., Cao, Y., Lin, S., Hu, H.: Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In: CVPR (2021) 3, 8, 10
46. Yang, C., Wu, Z., Zhou, B., Lin, S.: Instance localization for self-supervised detection pretraining. In: CVPR (2021) 4, 8, 10
47. Ye, M., Zhang, X., Yuen, P.C., Chang, S.F.: Unsupervised embedding learning via invariant and spreading instance feature. In: CVPR (2019) 2, 3
48. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV (2019) 3
49. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: ICLR (2017) 3
50. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: ECCV (2016) 3
51. Zhang, R., Isola, P., Efros, A.A.: Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In: CVPR (2017) 3
52. Zhao, X., Vemulapalli, R., Mansfield, P.A., Gong, B., Green, B., Shapira, L., Wu, Y.: Contrastive learning for label efficient semantic segmentation. In: ICCV (2021) 6
53. Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., Torralba, A.: Semantic understanding of scenes through the ade20k dataset. IJCV (2019) 8
54. Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A., Kong, T.: iBOT: Image BERT pre-training with online tokenizer. In: ICLR (2022) 3