# Dense Siamese Network for Dense Unsupervised Learning

Wenwei Zhang[1], Jiangmiao Pang[2], Kai Chen[2,3], and Chen Change Loy[1]✉

[1]S-Lab, Nanyang Technological University
[2]Shanghai AI Laboratory [3]SenseTime Research
{wenwei001, ccloy}@ntu.edu.sg {pangjiangmiao, chenkai}@pjlab.org.cn

**Abstract.** This paper presents Dense Siamese Network (DenseSiam), a simple unsupervised learning framework for dense prediction tasks. It learns visual representations by maximizing the similarity between two views of one image with two types of consistency, *i.e.*, pixel consistency and region consistency. Concretely, DenseSiam first maximizes the pixel level spatial consistency according to the exact location correspondence in the overlapped area. It also extracts a batch of region embeddings that correspond to some sub-regions in the overlapped area to be contrasted for region consistency. In contrast to previous methods that require negative pixel pairs, momentum encoders or heuristic masks, DenseSiam benefits from the simple Siamese network and optimizes the consistency of different granularities. It also proves that the simple location correspondence and interacted region embeddings are effective enough to learn the similarity. We apply DenseSiam on ImageNet and obtain competitive improvements on various downstream tasks. We also show that only with some extra task-specific losses, the simple framework can directly conduct dense prediction tasks. On an existing unsupervised semantic segmentation benchmark, it surpasses state-of-the-art segmentation methods by 2.1 mIoU with 28% training costs. Code and models are released at https://github.com/ZwwWayne/DenseSiam.

## 1 Introduction

Dense prediction tasks, such as image segmentation and object detection, are fundamental computer vision tasks with many real-world applications. Beyond conventional supervised learning methods, recent research interests grow in unsupervised learning to train networks from large-scale unlabeled datasets. These methods either learn representations as pre-trained weights then fine-tune on downstream tasks [10, 55] or directly learn for specific tasks [11].

In recent years, unsupervised pre-training has attracted a lot of attention. Much effort has been geared to learn global representation for image classification [6, 8, 10, 20, 21]. As the global average pooling in these methods discards spatial information, it has been observed [6, 8, 20, 55] that the learned representations are sub-optimal for dense uses. Naturally, some attempts conduct similarity

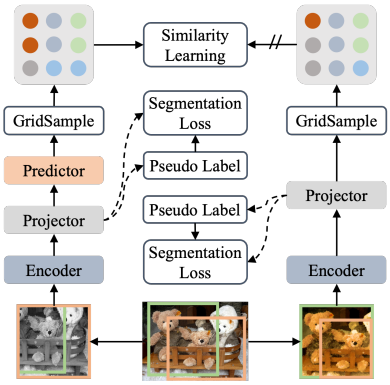


Fig. 1: **Dense Siamese Network** (DenseSiam) unanimously solves unsupervised pre-training and unsupervised semantic segmentation. The pair of images are first processed by the same encoder network and a convolutional projector. Then a predictor is applied on one side, and a stop-gradient operation on the other side. A grid sampling method is used to extract dense predictions in the overlapped area. DenseSiam can perform unsupervised semantic segmentation by simply adding a segmentation loss with pseudo labels produced by the projector.

learning at pixel-level [37, 47, 54] or region-level [25, 39, 51, 52] and maintain the main structures in global ones to learn representations for dense prediction tasks.

Despite the remarkable progress in that field, the development in unsupervised learning for specific tasks is relatively slow-moving. For example, solutions in unsupervised semantic segmentation [11, 28, 36] rely more on clustering methods (such as k-means) that are also derived from unsupervised image classification [5, 6, 28]. When reflecting on these similar tasks from the perspective of unsupervised learning, we observe these tasks share the inherent goal of maximizing the similarity of dense predictions (either labels or embeddings) across images but differ in the task-specific training objectives.

In this paper, we propose *Dense Siamese Network* (DenseSiam) to unanimously solve these dense unsupervised learning tasks within a single framework (Fig. 1). It learns visual representations by maximizing two types of consistency, *i.e.*, pixel consistency and region consistency, with methods dubbed as *PixSim* and *RegionSim*, respectively. The encoder here can be directly fine-tuned for various downstream dense prediction tasks after unsupervised pre-training. By adding an extra segmentation loss to the projector and regarding the argmaxed prediction of projector as pseudo labels, the encoder and projector is capable of learning class-wise representations for unsupervised semantic segmentation.

Specifically, PixSim learns to maximize the pixel-level spatial consistency between the grid sampled predictions. Its training objective is constrained under the exact location correspondence. In addition, the projected feature maps are multiplied with the features from encoder to generate a batch of region embeddings on each image, where each region embedding corresponds to a sub-region in the overlapped area. RegionSim then conducts contrastive learning between pairs of region embeddings and optimizes them to be consistent.

In contrast to previous unsupervised pre-training methods for dense prediction tasks, DenseSiam benefits from the simple Siamese network [10] that does not have negative pixel pairs [47, 54] and momentum encoders [39, 54]. Uniquely, DenseSiam optimizes the consistency of different granularities. The optimization for each granularity is simple as it neither requires heuristic masks [25] nor manual regions crops [39, 51, 52].

Table 1: **Comparison of unsupervised dense representation learning methods**.

| Method | Base | Pixel | Region | Extra Components | Correspondence |
|---|---|---|---|---|---|
| DenseCL [47] | MoCo v2 [9] | ✓ | × | × | feature similarity |
| PixPro [54] | BYOL [20] | ✓ | × | × | coordinate distance |
| VaDER [37] | MoCo v2 | ✓ | × | × | location |
| ReSim [51] | MoCo v2 | × | ✓ | × | dense region crops |
| SCRL [39] | BYOL [20] | × | ✓ | × | sampled region crops |
| DetCo [52] | MoCo v2 | × | ✓ | × | image patches |
| SoCo [48] | BYOL | × | ✓ | selective search [44] | sampled region crops |
| DetCon [25] | SimCLR [8] | × | ✓ | FH masks [17] | heuristic masks |
| CAST [42] | MoCo v2 | × | ✓ | DeepUSPS [34] | saliency map |
| DenseSiam | SimSiam [10] | ✓ | ✓ | × | location + region embeddings |

Extensive experiments show that DenseSiam is capable of learning strong representation for dense prediction tasks. DenseSiam obtains nontrivial 0.4 $AP^{mask}$, 0.7 mIoU, and 1.7 AP improvement in comparison with SimSiam when transferring its representation on COCO instance segmentation, Cityscapes semantic segmentation, and PASCAL VOC detection, respectively. For unsupervised semantic segmentation, DenseSiam makes the first attempt to discard clustering [11, 28] while surpassing previous state-of-the-art method [11] by **2.1** mIoU with only ∼28% of the original training costs.

## 2 Related Work

**Siamese Network.** Siamese network [3] was proposed for comparing entities. They have many applications including object tracking [2], face verification [41, 43], and one-shot recognition [29]. In conventional use cases, Siamese Network takes different images as input and outputs either a global embedding of each image for comparison [3, 29, 41, 43] or outputs feature map of each image for cross-correlation [2]. DenseSiam uses the Siamese architecture to output pixel embeddings and maximizes similarity between embeddings of the same pixel from two views of an image to pre-train dense representations with strong transferability in dense prediction tasks.

**Unsupervised Representation Learning.** Representative unsupervised representation learning methods include contrastive learning [1, 8, 9, 15, 21, 50] and clustering-based methods [5, 6, 45, 56]. Notably, Siamese network has become a common structure in recent attempts [6, 8, 20], despite their different motivations and solutions to avoid the feature 'collapsing' problem. SimSiam [10] studies the minimum core architecture in these methods [6, 8, 20] and shows that a simple Siamese network with stopping gradient can avoid feature 'collapsing' and yield strong representations.

Given the different natures of image-level representation learning and dense prediction tasks, more recent attempts [14, 25, 37, 39, 42, 47, 48, 51–55] pre-train dense representations specially designed for dense prediction tasks. Most of these methods conduct similarity learning with pixels [37, 47, 54], manually cropped

patches of image or features [39, 48, 51–53], and regional features segmented by saliency map [42] or segmentation masks [25] obtained in unsupervised manners. These methods still need a momentum encoder [37,39,47,52,54] or negative pixel samples [37, 47, 54], although they [8, 20] have been proven unnecessary [10] in image-level representation learning.

DenseSiam only needs a Siamese network with a stop gradient operation to obtain strong dense representations, without momentum encoder [37, 39, 47, 52, 54] nor negative pixel pairs [37, 47, 54] (Table 1). It conducts contrastive learning among regional features inside an image emerged via pixel similarity learning. Thanks for the unique design, DenseSiam performs region-level similarity learning without saliency maps [42], region crops [39, 51, 52], nor heuristic masks [25].

**Unsupervised Semantic Segmentation.** Unsupervised semantic segmentation aims to predict labels for each pixel without annotations. There are a few attempts that introduce heuristic masks and conduct similarity learning between segments (regions) [26, 46] with object-centric images. Most methods focus on natural scene images and exploit the assumption that semantic information should be invariant to photometric transformations and equivariant to geometric transformations no matter how the model predicts the labels, which is inherently consistent with the goal of unsupervised representation learning. However, these methods still rely on clustering [11, 28, 36] to predict the per-pixel labels and maximize the consistency of cluster assignments in different views of the image, which is cumbersome and difficult to be used for large-scale data.

DenseSiam conducts unsupervised semantic segmentation by adding a class-balanced cross entropy loss without clustering, significantly reduces the training costs and makes it scalable for large-scale data. RegionSim further boosts the segmentation accuracy by maximizing the consistency between regional features.

## 3 Dense Siamese Network

DenseSiam, as shown in Fig. 2, is a generic unsupervised learning framework for dense prediction tasks. The framework is inspired by SimSiam [10] but differs in its formulation tailored for learning dense representation (Sec. 3.1). In particular, it conducts dense similarity learning by PixSim (Sec. 3.2), which aims at maximizing pixel-level spatial consistency between grid sampled predictions. Based on the region embeddings derived from per-pixel predictions inferred in PixSim, DenseSiam further performs region-level contrastive learning through RegionSim (Sec. 3.3).

### 3.1 Siamese Dense Prediction

As shown in Fig. 2, DenseSiam takes two randomly augmented views $x_1$ and $x_2$ of an image $x$ as inputs. The two views are processed by an encoder network $f$ and a projector network $g$. The encoder network can either be a backbone network like ResNet-50 [24] or a combination of networks such as ResNet-50 with FPN [30]. As the encoder network $f$ does not use global average pooling (GAP), the output of

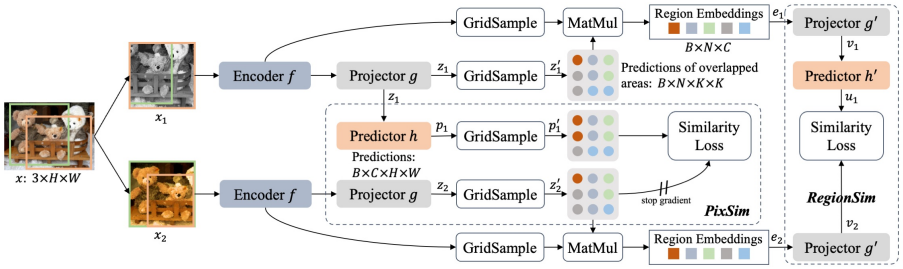


Fig. 2: **Pipeline of DenseSiam.** DenseSiam takes two randomly augmented views of an image $x$ as inputs. The two views are processed by an encoder network $f$ without global average pooling (GAP), followed by a projector network $g$. The predictor network $h$ transforms the dense prediction of the projector of one view. Then GridSample module samples the same pixels inside the intersected regions of the two views and interpolates their feature grids. The similarity between feature grids from two views are maximized by PixSim. Then DenseSiam multiplies the features of encoder and the dense predictions by projector $g$ in the overlapped area to obtain region embeddings of each image. These region embeddings are processed by a new projector $g'$ and new predictor $h'$ for region level similarity learning.

$f$ is a *dense feature map* and is later processed by the projector $g$, which consists of three $1 \times 1$ convolutional layers followed by Batch Normalization (BN) and ReLU activation. Following the design in SimSiam [10], the last BN layer in $g$ does not use learnable affine transformation [27]. The projector $g$ projects the per-pixel embeddings to either a feature space for representation learning or to a labels space for segmentation. Given the dense prediction from $g$, noted as $z_1 \triangleq g(f(x_1))$, the predictor network $h$ transforms the output of one view and matches it to another view [10]. Its output is denoted as $p_1 \triangleq h(g(f(x_1)))$.

The encoder and projector essentially form a Siamese architecture, which outputs two dense predictions ($z_1$ and $z_2$) for two views of an image, respectively. DenseSiam conducts similarity learning of different granularities using the Siamese dense predictions with the assistance of predictor [10,20]. After unsupervised pre-training, only the encoder network is fine-tuned for downstream dense prediction tasks, where the projector is only used during pre-training to improve the representation quality [8]. When directly learning the dense prediction tasks, the encoder and projector are combined to tackle the task.

### 3.2 PixSim

**Dense Similarity Learning.** We formulate the dense similarity learning to maximize the similarity of dense predictions inside the overlapping regions between $p_1$ and $z_2$. Specifically, given the relative coordinates of the intersected region in $x_1$ and $x_2$, we uniformly sample $K \times K$ point grids inside the intersected region from both views as shown in Fig. 3. These points in two views have exactly the same coordinates when they are mapped to the original image

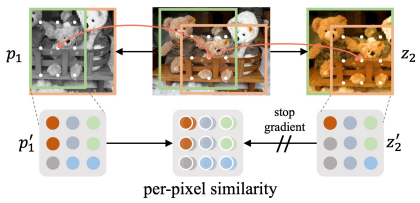


Fig. 3: **Grid Sample module when grid size $K$=3.** Pixels at the same location in the original image should have similar predictions when they are transformed into two views. The Grid Sample module samples these pixels and interpolates their predictions in two views. The predictions of one view are learned to match those in another view.

$x$; thus, their predictions should be consistent to those in another view. Assuming the sampled feature grids are $z_1' \triangleq \texttt{gridsample}(z_1)$ and $p_1' \triangleq \texttt{gridsample}(p_1)$, PixSim minimizes the distance of the feature grids by a symmetrical loss [10,20]:

$$\begin{aligned}\mathcal{L}_{dense} = &\frac{1}{2}\mathcal{D}(p_1', \texttt{stopgrad}(z_2')) \\ &+\frac{1}{2}\mathcal{D}(p_2', \texttt{stopgrad}(z_1')),\end{aligned} \tag{1}$$

where `stopgrad` is the stop-gradient operation to prevent feature 'collapsing' [10] and $\mathcal{D}$ is a distance function that can have many forms [6,10]. Two representative distance functions are negative cosine similarity [10,20]

$$\mathcal{D}(p_1', z_2') = -\frac{p_1'}{\|p_1'\|_2} \cdot \frac{z_2'}{\|z_2'\|_2}, \tag{2}$$

and cross-entropy similarity

$$\mathcal{D}(p_1', z_2') = -\texttt{softmax}(p_1') \cdot \log \texttt{softmax}(z_2'). \tag{3}$$

**Advantages to Image-Level Similarity Learning.** Previous formulation of image-level representation learning faces two issues when transferring their representations for dense prediction tasks. First, it learns global embeddings for each image during similarity learning, where most of the spatial information is discarded during GAP. Hence, the spatial consistency is not guaranteed in its goal of representation learning, while dense prediction tasks like semantic segmentation and object detection rely on the spatial information of each pixel. Second, it is common that $p_1$ and $z_2$ contain different contents (see examples of $x_1$ and $x_2$ in Fig. 2) after heavy data augmentations like random cropping and resizing, but the global embeddings that encode these different contents are forced to be close to each other during training. This forcefully makes the embeddings of different pixel groups to be close to each other, breaking the regional consistency and is thus undesirable for dense prediction tasks.

PixSim resolves the above-mentioned issues by conducting pixel-wise similarity learning inside the intersected regions of two views, where the embeddings at identical locations of different views have different values due to augmentations [8] are forced to be similar to each other in training. Such a process learns

dense representations that are invariant to data augmentations, which are more favorable dense representations for dense prediction tasks.

### 3.3 RegionSim

When using cross-entropy with softmax as the training objective, PixSim implicitly groups similar pixels together into the same regions/segments. The consistency of such regions can be further maximized by the proposed RegionSim to learn better dense representation. The connection between PixSim and RegionSim is seamless - region-level similarity learning can be achieved without heuristic masks [25], saliency map [42], or manually cropping multiple regional features [39, 51, 52] as in previous studies.

**Region Embedding Generation.** As shown in Fig. 2, DenseSiam first obtains the feature grids of the intersected regions of each view by grid sampling from the features produced by the encoder $f$. This restricts RegionSim inside the intersected regions to ensure that the region embeddings to encode the similar contents. For simplicity, the region embeddings $e_1 \in \mathcal{R}^{N \times C}$ are obtained by the summation of multiplication between the masks and features as

$$e_1 = \sum_{i,j}^{K \times K} z_1'[i,j] \cdot \texttt{gridsample}(f(x_1))[i,j], \tag{4}$$

where $N$ and $C$ represent the number of sub-regions and the number of feature channels, respectively. The process yields an embedding for each segment of each pseudo category, which will then be used for contrastive learning to increase the consistency between these region embeddings.

**Region Similarity Learning.** After obtaining the region embeddings, RegionSim transforms them by a projector network $g'$, which is a three-layer MLP head [8, 10, 20]. The output of $g'$ is then transformed by $h'$ for contrasting or matching with another view. RegionSim assumes each region embedding of a sub-region to be consistent with the embedding of the sub-region in another view. Therefore, Eq. 1 can be directly applied to these region embeddings to conduct region-level similarity learning. Meanwhile, we also enforce the region embedding to have low similarities with the embeddings of other region embeddings to make the feature space more compact. Consequently, by denoting the two outputs as $u_1 \triangleq h'(g'(e_1))$ and $v_2 \triangleq g'(e_2)$, RegionSim can also minimize the symmetrized loss function

$$\mathcal{L}_{region} = \frac{1}{2}\mathcal{L}_c(u_1, v_2) + \frac{1}{2}\mathcal{L}_c(u_2, v_1). \tag{5}$$

The contrastive loss function [35] $\mathcal{L}_c$ can be written as

$$\mathcal{L}_c(u_1, v_2) = -\sum_{s}^{N} \log \frac{\exp(u_1^s \cdot v_2^s)}{\sum_{s'}^{N} \exp(u_1^s \cdot v_2^{s'})}. \tag{6}$$

### 3.4 Learning Objective

When conducting unsupervised representation learning, DenseSiam also maintains a global branch to conduct image-level similarity learning to enhance the global consistency of representations, besides PixSim and RegionSim shown in Fig. 2. The architecture of the global branch remains the same as SimSiam [10]. The numbers of channels in the projectors and predictors of PixSim and RegionSim are set to 512 for efficiency. Denoting the loss of the global branch as $\mathcal{L}_{sim}$, DenseSiam optimizes the following loss

$$\mathcal{L}=\mathcal{L}_{sim}+\lambda_1\mathcal{L}_{dense}+\lambda_2\mathcal{L}_{region}, \tag{7}$$

where $\lambda_1$ and $\lambda_2$ are loss weights of $\mathcal{L}_{dense}$ for PixSim and $\mathcal{L}_{region}$ for RegionSim, respectively. The encoder $f$ is used for fine-tuning in downstream dense prediction tasks after pre-training.

## 4 Unsupervised Semantic Segmentation

The proposed framework is appealing in that it can be readily extended to address dense prediction tasks such as unsupervised semantic segmentation, by simply adding a task-specific losses and layers, without needing the offline and cumbersome clustering process [11, 28, 36].

Formally, when using cross-entropy similarity (Eq. 3) in PixSim, the softmax output $\texttt{softmax}(z)$ can be regarded as the probabilities of belonging to each of $N$ pseudo-categories. In such a case, the projector $g$ predicts a label for each pixel, which aligns with the formulation for unsupervised semantic segmentation.

DenseSiam uses ResNet with a simplified FPN [11, 30] as the encoder $g$. The output channel $N$ of PixSim is modified to match the number of classes in the dataset (e.g., 27 on COCO stuff-thing dataset [11, 31]). Our experiment shows that a direct change in the number of output channels in PixSim without any further modification can already yield reasonable performance without feature 'collapsing'. Following previous method [11] that encourages the prediction scores to have a lower entropy (*i.e.*, more like one-hot scores), we add a cross entropy loss, denoted as $\mathcal{L}_{seg}$, when applying PixSim for unsupervised semantic segmentation with the pseudo labels obtained by $\texttt{argmax}(z_1)$.

We also observe that the small number of categories undermines the training stability, which is also a common issue in clustering-based methods [11, 28]. To solve this issue, DenseSiam introduces another set of projector and predictor in PixSim to keep a large number of pseudo categories following the over-clustering strategy [11, 28]. This head only conducts similarity learning using Eq. 1, noted as $\mathcal{L}_{aux}$. Therefore, the overall loss of DenseSiam for unsupervised semantic segmentation is calculated as

$$\mathcal{L}=\lambda_1\mathcal{L}_{dense}+\lambda_2\mathcal{L}_{region}+\lambda_3\mathcal{L}_{seg}+\lambda_4\mathcal{L}_{aux}. \tag{8}$$

We use $\lambda_4=\frac{\log N}{\log N+\log N_{aux}}$ and $\lambda_1=\frac{\log N_{aux}}{\log N+\log N_{aux}}$ to prevent the auxiliary loss from overwhelming the gradients because it uses a larger number $N_{aux}$ of pseudo

categories [11]. Note that RegionSim is only used in training to enhance the region consistency of dense labels predicted by PixSim. Only the encoder $f$ and projector $g$ are combined to form a segmentation model at inference time.

## 5 Experiments

### 5.1 Experimental Settings

**Datasets.** For unsupervised pre-training, we compare with other methods on ImageNet-1k [40] (IN1k) dataset and conduct ablation studies on COCO [31] dataset as it is smaller. COCO and IN1k are two large-scale datasets that contain ∼118K and ∼1.28 million images, respectively. Though having more images, IN1k is highly curated and it mainly consists of object-centric images. Thus, the data is usually used for image classification. In contrast, COCO contains more diverse scenes in the real world and it has more objects (∼7 objects *vs.* ∼1 in IN1k) in one image. Hence, it is mainly used for dense prediction tasks like object detection, semantic, instance, and panoptic segmentation.

For unsupervised semantic segmentation, the model is trained and evaluated on curated subsets [11,28] of COCO `train2017` split and `val2017` split, respectively. The training and validation set contain 49,629 images and 2,175 images, respectively. The semantic segmentation annotations include 80 thing categories and 91 stuff categories [4]. We follow previous methods [11,28] to merge these categories to form 27 (15 'stuff' and 12 'things') categories.

**Training Setup.** We closely follow the pre-training settings of SimSiam [10] when conducting unsupervised pre-training experiments of DenseSiam. Specifically, we use a learning rate of $lr \times \text{BatchSize}/256$ following the linear scaling strategy [19], with a base $lr = 0.05$. The batch size is 512 by default. We use the cosine decay learning rate schedule [33] and SGD optimizer, where the weight decay is 0.0001 and the SGD momentum is 0.9. We pre-train DenseSiam for 800 epochs on COCO and for 200 epochs on IN1k. ResNet-50 [24] is used as default.

For unsupervised semantic segmentation, for a fair comparison, we follow PiCIE [11] to adopt the backbone pre-trained on IN1k in a supervised manner. The backbone is then fine-tuned with DenseSiam for unsupervised semantic segmentation using the same base learning rate, weight decay, and SGD momentum as those used in unsupervised representation learning. The batch size is 256 by default. We empirically find the constant learning schedule works better during training. The model is trained for 10 epochs. ResNet-18 [24] is used as default for a fair comparison with previous methods [11].

**Evaluation Protocol of Transfer Learning.** For unsupervised pre-training, we evaluate the transfer learning performance of the pre-trained representations following Wang *et al* [47]. We select different dense prediction tasks to comprehensively evaluate the transferability of the dense representation, including semantic segmentation, object detection, and instance segmentation. Challenging and popular dataset with representative algorithms in each task is selected.

When evaluating on semantic segmentation, we fine-tune a FCN [32] model and evaluate it with the Cityscapes dataset [13]. The model is trained on the

`train_fine` set (2975 images) for 40k iterations and is tested on the `val` set. Strictly following the settings in MMSegmentation [12], we use FCN-D8 with a crop size of 769, a batch size of 16, and synchronized batch normalization. Results are averaged over five trials.

When evaluating on object detection, we fine-tune a Faster R-CNN [38] with C4-backbone by 24k iterations on VOC 2007 trainval + 2012 train set and is tested in VOC 2007 test set. Results are reported as an average over five trials.

We also evaluate the representation on object detection and instance segmentation on COCO dataset [31]. We fine-tune a Mask R-CNN with FPN [30] with standard multi-scale 1x schedule [7, 49] on COCO `train2017` split and evaluate it on COCO `val2017` split. We apply synchronized batch normalization in backbone, neck, and RoI heads during training [22, 47].

**Evaluation of Unsupervised Semantic Segmentation.** Since the model is trained without labels, a mapping between the model's label space and the ground truth categories needs be established. Therefore, we first let the model predicts on each image on the validation set, then we calculate the confusion matrix between the predicted labels and the ground truth classes. We use linear assignment to build a one-to-one mapping between the predicted labels and ground truth classes by taking the confusion matrix as the assignment cost. Then we calculate mean IoU over all classes based on the obtained mapping [11, 28]. To more comprehensively understand the model's behavior, we also report mean IoU of stuff and things classes, noted as $\text{mIoU}^{\text{St}}$ and $\text{mIoU}^{\text{Th}}$, respectively.

### 5.2 Transfer Learning Results

The comparison on transfer learning in dense prediction tasks between DenseSiam and previous unsupervised representation learning methods [8, 10, 20, 21, 47, 51] is shown in Table 2. The results of scratch, supervised, MoCo v2 [9], DenseCL [47], SimCLR [8], and BYOL [20] are reported from DenseCL [47]. For fair comparison, we fine-tune the model of ReSim-C4 [51] using the similar setting (Sec. 5.1). The model checkpoint is released by the paper authors[1]. We report the results of SimSiam [10], DetCo [52], and PixPro [54] based on our re-implementation.

**COCO Instance Segmentation and Detection.** As shown in the first two columns of Table 2, DenseSiam outperforms SimSiam by 0.4 $\text{AP}^{\text{mask}}$ on COCO instance segmentation. The improvements over SimSiam is on-par with that of DenseCL and DetCo over MoCo v2 (0.4 *vs.* 0.3 $\text{AP}^{\text{mask}}$) and is better than ReSim (0.4 *vs.* 0 $\text{AP}^{\text{mask}}$). This further verifies the effectiveness of DenseSiam. Notably, DenseSiam outperforms ReSim, DetCo, DenseCL, and PixPro by 0.7, 0.4, 0.4, and 0.2 $\text{AP}^{\text{mask}}$, respectively. The results in COCO object detection is consistent with that in instance segmentation.

**Cityscapes Semantic Segmentation.** We compare DenseSiam with previous methods on semantic segmentation on Cityscapes dataset [13] and report the mean IoU (mIoU) of fine-tuned models. DenseSiam surpasses SimSiam by 0.7

[1] https://github.com/Tete-Xiao/ReSim

Table 2: **Transfer Learning**. All unsupervised methods are either based on 200-epoch pre-training in ImageNet ('IN1k'). *COCO instance segmentation* and *COCO detection*: Mask R-CNN [23] (1× schedule) fine-tuned in COCO `train2017`; *Cityscapes*: FCN fine-tuned on Cityscapes dataset [13]; *VOC 07+12 detection*: Faster R-CNN fine-tuned on VOC 2007 trainval + 2012 train, evaluated on VOC 2007 test; COCO `train2017`, evaluated on COCO `val2017`. All Mask R-CNN are with FPN [30]. All Faster R-CNN models are with the C4-backbone [18]. All VOC and Cityscapes results are averaged over 5 trials

| | COCO Instance Seg. | | | COCO Detection | | | Cityscapes | VOC 07+12 Detection | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Pre-train | $AP^{mask}$ | $AP^{mask}_{50}$ | $AP^{mask}_{75}$ | AP | $AP_{50}$ | $AP_{75}$ | mIoU | AP | $AP_{50}$ | $AP_{75}$ |
| scratch | 29.9 | 47.9 | 32.0 | 32.8 | 50.9 | 35.3 | 63.5 | 32.8 | 59.0 | 31.6 |
| supervised | 35.9 | 56.6 | 38.6 | 39.7 | 59.5 | 43.3 | 73.7 | 54.2 | 81.6 | 59.8 |
| BYOL [20] | 34.9 | 55.3 | 37.5 | 38.4 | 57.9 | 41.9 | 71.6 | 51.9 | 81.0 | 56.5 |
| SimCLR [8] | 34.8 | 55.2 | 37.2 | 38.5 | 58.0 | 42.0 | 73.1 | 51.5 | 79.4 | 55.6 |
| MoCo v2 [9] | 36.1 | 56.9 | 38.7 | 39.8 | 59.8 | 43.6 | 74.5 | 57.0 | 82.4 | 63.6 |
| SimSiam [10] | 36.4 | 57.4 | 38.8 | 40.4 | 60.4 | 44.1 | 76.3 | 56.7 | 82.3 | 63.4 |
| ReSim [51] | 36.1 | 56.7 | 38.8 | 40.0 | 59.7 | 44.3 | 76.8 | 58.7 | 83.1 | 66.3 |
| DetCo [52] | 36.4 | 57.0 | 38.9 | 40.1 | 60.3 | 43.9 | 76.5 | 57.8 | 82.6 | 64.2 |
| DenseCL [47] | 36.4 | 57.0 | 39.2 | 40.3 | 59.9 | 44.3 | 75.7 | 58.7 | 82.8 | 65.2 |
| PixPro [54] | 36.6 | 57.3 | 39.1 | 40.5 | 60.1 | 44.3 | 76.3 | **59.5** | **83.4** | **66.9** |
| DenseSiam | **36.8** | **57.6** | **39.8** | **40.8** | **60.7** | **44.6** | **77.0** | 58.5 | 82.9 | 65.3 |

mIoU and outperforms previous dense representation learning methods ReSim, DetCo, DenseCL, and PixPro by 0.2, 0.5, 1.3, and 0.7 mIoU, respectively. Notably, image-level unsupervised method such as SimCLR [8], BYOL [20] and SimSiam [10] show considerable performance gap against dense unsupervised methods including DenseSiam, DenseCL, and ReSim, indicating that pre-training with image-level similarity learning is sub-optimal for dense prediction tasks.

**PASCAL VOC Object Detection.** We further compare DenseSiam with previous methods on PASCAL VOC [16] object detection. We report the original metric $AP_{50}$ (AP calculated with IoU threshold 0.5) of VOC and further report COCO-style AP [49] and $AP_{75}$, which are stricter criteria in evaluating the detection performance. DenseSiam show a large improvement of 1.8 AP in comparison with SimSiam [10]. Notably, DenseSiam exhibits considerable improvements on $AP_{75}$ than $AP_{50}$, suggesting the effectiveness of DenseSiam in learning accurate spatial information. Its improvement over SimSiam is more than that of DetCo [52] (0.8 AP) over MoCo v2, and is on-par with those of DenseCL [47] and ReSim [51] over MoCo v2 [9].

DenseCL, ReSim, and PixPro are 0.2, 0.2, and 1.0 AP better than DenseSiam, respectively. This phenomenon contradicts the results in the benchmarks of COCO dataset, although their improvements over their image-level counterparts are consistent across benchmarks. We hypothesize that the different backbones used in the two benchmarks leads to this phenomenon, where ResNet-50-C4 backbone [38] is used on PASCAL VOC but ResNet-50-FPN [30] is used on COCO. The hypothesis also explains the inferior performance of DetCo [52]:

Table 3: **Unsupervised Semantic Segmentation**. The model is trained and tested on the curated COCO dataset [11, 28]. ' + aux.' denotes PiCIE or DenseSiam is trained with an auxiliary head

| Method | mIoU | $\text{mIoU}^{\text{St}}$ | $\text{mIoU}^{\text{Th}}$ | Time (h) |
|---|---|---|---|---|
| Modified Deep Clustering [5] | 9.8 | 22.2 | 11.6 | - |
| IIC [28] | 6.7 | 12.0 | 13.6 | - |
| PiCIE [11] + aux. | 14.4 | 17.3 | 23.8 | 18 |
| DenseSiam + aux. | **16.4** | **24.5** | **29.5** | 5 |

DetCo conducts contrastive learning on pyramid features but only one feature scale is used when fine-tuning Faster R-CNN on PASCAL VOC dataset.

### 5.3 Unsupervised Segmentation Results

In Table 3 we compare DenseSiam with previous state-of-the-art methods in unsupervised semantic segmentation. DenseSiam achieves new state-of-the-art performance of 16.4 mIoU, surpassing PiCIE by 2 mIoU over all classes. Imbalanced performance is observed in previous methods between thing and stuff classes, *e.g.*, PiCIE [11] and IIC [28] surpass Modified DeepClustering [5] on thing classes but fall behind on stuff classes. In contrast, DenseSiam consistently outperforms previous best results on both thing and stuff classes by more than 2 mIoU. The results reveal that clustering is unnecessary, whereas clustering is indispensable in previous unsupervised segmentation methods [11, 28].

We also calculate the training costs of PiCIE and DenseSiam by measuring the GPU hours used to train the model with similar batch size and backbone. Because PiCIE needs clustering to obtain pseudo labels before each training epoch, its training time is comprised of the time of label clustering, data loading, and model's forward and backward passes. In contrast, DenseSiam does not rely on clustering. In the setting of single GPU training, DenseSiam saves ∼72% training costs in comparison with PiCIE.

### 5.4 Ablation Study

**Unsupervised Pre-training.** We ablate the key components in DenseSiam as shown in Table 4. The baselines in Table 4a-e are SimSiam (53.5 AP).
*i) Effective grid number $K$ in PixSim*: We study the effective number $K$ used in grid sampling in PixSim in Table 4a. The greater the number, the more feature grids will be sampled from the intersected regions and will be used for similarity learning. With zero grid number DenseSiam degenerates to the SimSiam baseline where no pixel similarity learning is performed. The results in Table 4a shows that 7×7 feature grids is sufficient to improve the per-pixel consistency of dense representation. We also compare the training memory used in PixSim. The comparison shows that PixSim only brings 0.1% ∼ 0.3% extra memory cost in comparison with SimSiam.

Table 4: **Ablation studies in unsupervised pre-trainings**. All unsupervised representations are based on 800-epoch pre-training on COCO `train2017`. The representations are fine-tuned with Faster R-CNN [38] (C4-backbone) in VOC 2007 trainval + 2012 train and evaluated on VOC 2007 test. 'Mem.' indicates memory cost measured by Gigabyte (GB). The results of fine-tuning are averaged over 5 trials. Best settings are bolded and used as default settings

(a) The effective number $K$ in PixSim

| Gird Number | AP | $AP_{50}$ | $AP_{75}$ | Mem. |
|---|---|---|---|---|
| 0 | 53.5 | 79.7 | 59.3 | 8.06 |
| 1 | 28.8 | 53.8 | 27.1 | 8.11 |
| 3 | 53.9 | 80.0 | 59.4 | 8.12 |
| 7 | **54.9** | **80.8** | **60.9** | 8.18 |
| 9 | 54.6 | 80.7 | 60.6 | 8.21 |
| 14 | 54.7 | 80.6 | 60.8 | 8.28 |

(b) The effective loss weight of PixSim

| $\lambda_1$ | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| 0 | 53.5 | 79.7 | 59.3 |
| 0.1 | 53.3 | 79.6 | 58.8 |
| 0.3 | 53.5 | 79.9 | 58.8 |
| 0.5 | 54.0 | 80.2 | 60.0 |
| 0.7 | 54.0 | 79.8 | 59.8 |
| 1.0 | **54.9** | **80.8** | **60.9** |

(c) The effective loss weight of RegionSim

| $\lambda_2$ | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| 0 | 53.5 | 79.7 | 59.3 |
| 0.01 | 55.3 | 81.0 | 61.3 |
| 0.05 | 55.0 | 80.9 | 60.6 |
| 0.1 | **55.5** | **81.1** | **61.5** |
| 0.2 | 55.3 | 81.0 | 61.0 |
| 0.5 | 55.3 | 80.9 | 61.1 |

(d) The effective order of grid sample, projector, and predictor in PixSim

| Order | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| - | 53.5 | 79.7 | 59.3 |
| grid. + proj. + pred. | 53.4 | 79.6 | 59.0 |
| proj. + grid. + pred. | 54.7 | 80.6 | 60.4 |
| proj. + pred. + grid. | **54.9** | **80.8** | **60.9** |

(e) Regions to be focused in global branch. 'in.' indicates the intersected regions

| Image-level | Pixel-level | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|
| global | N/A | 53.5 | 79.7 | 59.3 |
| in. | N/A | 0.0 | 0.0 | 0.0 |
| global | in. | **54.9** | **80.8** | **60.9** |
| in. | in. | 35.6 | 61.5 | 35.6 |

(f) Suitable start epoch of RegionSim

| Start epoch | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| never | 54.9 | 80.8 | 60.9 |
| 0.4 | 37.7 | 64.8 | 38.4 |
| 0.5 | **55.5** | **81.1** | **61.5** |
| 0.6 | 55.0 | 80.8 | 60.4 |
| w/o PixSim | 52.8 | 79.2 | 58.0 |

*ii) Loss weight of PixSim*: We further study the loss weight $\lambda_1$ of $\mathcal{L}_{dense}$ used for per-pixel similarity learning. The comparative results in Table 4b show that with $\lambda_1 = 1$ we achieve the best performance, which is equal to the loss weight of the image-level similarity learning.

*iii) Loss weight of RegionSim*: We study the loss weight $\lambda_2$ of RegionSim as shown in Table 4c. We find that 0.1 works best and large value of $\lambda_2$ leads decreased performance.

*iv) Orders of grid sample, projector, and predictor*: As DenseSiam consists of encoder, projector, and predictor, we study the optimal position where the grid sampler should be introduced. The results in Table 4d show that it is necessary to put the grid sample module after the projector, but the predictor can perform equally well when it is before or behind the grid sample module.

*v) Regions to focus in global branch*: We also study the regions that should be focused in global branch in Table 4e, as an image-level similarity learning branch (SimSiam) is kept to facilitate training. The abbreviations 'global' and 'in.' indicate whether the similarity learning is conducted with the whole image or the intersected regions between two views. To maximize per-pixel consistency, PixSim is always conducted with the intersected regions. We find that the focusing on the whole image when conducting image-level similarity learning is always important in both SimSiam and DenseSiam. Only using the intersected regions

Table 5: **Ablation study in unsupervised segmentation**. 'Aux.' indicates auxiliary head

| PixSim | Aux. | CE | Region. | mIoU | $\text{mIoU}^{\text{St}}$ | $\text{mIoU}^{\text{Th}}$ |
|---|---|---|---|---|---|---|
| ✓ | | | | 10.1 | 19.0 | 17.7 |
| ✓ | ✓ | | | 11.1 | 20.4 | 22.3 |
| ✓ | ✓ | ✓ | | 15.0 | 24.8 | 23.4 |
| ✓ | ✓ | ✓ | ✓ | 16.4 | 24.5 | 29.5 |

will lead to feature 'collapsing' (second row with zero accuracy in downstream tasks) in SimSiam since it makes some learning shortcuts for the model.
*vi) Start point of RegionSim*: The start point of RegionSim based on PixSim (54.9) matters as shown in Table 4f. The label prediction quality is not accurate and may lead to a wrong optimization direction if RegionSim is applied at a wrong time. Starting RegionSim at the middle point of training yields the best performance. We further try only using RegionSim without PixSim (last row in Table 4f). Only adding RegionSim degrades fine-tuning results on by 2.7 AP, this also implies that RegionSim needs PixSim to produce meaningful groups.
**Unsupervised Semantic Segmentation.** We also study the effectiveness of the components in DenseSiam for unsupervised semantic segmentation. Directly applying PixSim without any further modification yields 10.1 mIoU, which already surpasses many previous methods [5, 28] (9.8 and 6.7 mIoU). Adding CE loss further improves the performance by making the feature space more compact and discriminative. After adding the auxiliary head, the model already surpasses the previous state-of-the-art method PiCIE. RegionSim further brings 1.4 mIoU of improvement.

## 6 Conclusion

Different dense prediction tasks essentially shares the similar goal of optimizing the spatial consistency between views of images. DenseSiam exploits such a property and unanimously solves unsupervised dense representation learning and unsupervised semantic segmentation within a Siamese architecture. DenseSiam optimizes similarity between dense predictions at pixel level by PixSim and at region level by RegionSim, with neither negative pixel pairs, momentum encoder, manual region crops, nor heuristic masks, which are *all unnecessary* as revealed by DenseSiam to obtain a strong dense representation for downstream tasks. Its unsupervised semantic segmentation performance also achieves the new state-of-the-art.
**Acknowledgements.** This study is supported under the RIE2020 Industry Alignment Fund Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner(s). The work is also suported by Singapore MOE AcRF Tier 2 (MOE-T2EP20120-0001) and NTU NAP Grant. Jiangmiao Pang and Kai Chen are partially supported by the Shanghai Committee of Science and Technology, China (Grant No. 20DZ1100800).

## References


1. Bachman, P., Hjelm, R.D., Buchwalter, W.: Learning representations by maximizing mutual information across views. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) NeurIPS (2019) 3
2. Bertinetto, L., Valmadre, J., Henriques, J.F., Vedaldi, A., Torr, P.H.S.: Fully-convolutional siamese networks for object tracking. In: ECCV (2016) 3
3. Bromley, J., Bentz, J.W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Säckinger, E., Shah, R.: Signature verification using a "siamese" time delay neural network. In: NeurIPS (1993) 3
4. Caesar, H., Uijlings, J., Ferrari, V.: COCO-Stuff: Thing and stuff classes in context. In: CVPR (2018) 9
5. Caron, M., Bojanowski, P., Joulin, A., Douze, M.: Deep clustering for unsupervised learning of visual features. In: ECCV (2018) 2, 3, 12, 14
6. 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, 2, 3, 6
7. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019) 10
8. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.E.: A simple framework for contrastive learning of visual representations. In: ICML (2020) 1, 3, 4, 5, 6, 7, 10, 11
9. Chen, X., Fan, H., Girshick, R.B., He, K.: Improved baselines with momentum contrastive learning. CoRR **abs/2003.04297** (2020) 3, 10, 11
10. Chen, X., He, K.: Exploring simple siamese representation learning. In: CVPR (2021) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
11. Cho, J.H., Mall, U., Bala, K., Hariharan, B.: PiCIE: Unsupervised semantic segmentation using invariance and equivariance in clustering. In: CVPR (2021) 1, 2, 3, 4, 8, 9, 10, 12
12. Contributors, M.: MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation (2020) 10
13. 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) 9, 10, 11
14. Dai, Z., Cai, B., Lin, Y., Chen, J.: UP-DETR: unsupervised pre-training for object detection with transformers. In: CVPR (2021) 3
15. Dosovitskiy, A., Fischer, P., Springenberg, J.T., Riedmiller, M.A., Brox, T.: Discriminative unsupervised feature learning with exemplar convolutional neural networks. TPAMI (2016) 3
16. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (VOC) challenge. IJCV (2010) 11
17. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. IJCV (2004) 3
18. Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K.: Detectron. https://github.com/facebookresearch/detectron (2018) 11
19. Goyal, P., Dollár, P., Girshick, R.B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch SGD: training imagenet in 1 hour. CoRR **abs/1706.02677** (2017) 9

20. Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.Á., Guo, Z., Azar, M.G., Piot, B., Kavukcuoglu, K., Munos, R., Valko, M.: Bootstrap your own latent - A new approach to self-supervised learning. In: NeurIPS (2020) 1, 3, 4, 5, 6, 7, 10, 11
21. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.B.: Momentum contrast for unsupervised visual representation learning. In: CVPR (2020) 1, 3, 10
22. He, K., Girshick, R., Dollar, P.: Rethinking ImageNet pre-training. In: ICCV (2019) 10
23. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: ICCV (2017) 11
24. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016) 4, 9
25. Hénaff, O.J., Koppula, S., Alayrac, J., van den Oord, A., Vinyals, O., Carreira, J.: Efficient visual pretraining with contrastive detection. In: ICCV (2021) 2, 3, 4, 7
26. Hwang, J., Yu, S.X., Shi, J., Collins, M.D., Yang, T., Zhang, X., Chen, L.: Segsort: Segmentation by discriminative sorting of segments. In: ICCV (2019) 4
27. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML (2015) 5
28. Ji, X., Vedaldi, A., Henriques, J.F.: Invariant information clustering for unsupervised image classification and segmentation. In: ICCV (2019) 2, 3, 4, 8, 9, 10, 12, 14
29. Koch, G., Zemel, R., Salakhutdinov, R., et al.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop (2015) 3
30. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR (2017) 4, 8, 10, 11
31. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common objects in context. In: ECCV (2014) 8, 9, 10
32. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015) 9
33. Loshchilov, I., Hutter, F.: SGDR: stochastic gradient descent with warm restarts. In: ICLR (2017) 9
34. Nguyen, D.T., Dax, M., Mummadi, C.K., Ngo, T., Nguyen, T.H.P., Lou, Z., Brox, T.: Deepusps: Deep robust unsupervised saliency prediction via self-supervision. In: NeurIPS (2019) 3
35. van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. CoRR **abs/1807.03748** (2018) 7
36. Ouali, Y., Hudelot, C., Tami, M.: Autoregressive unsupervised image segmentation. In: ECCV (2020) 2, 4, 8
37. Pinheiro, P.O., Almahairi, A., Benmalek, R.Y., Golemo, F., Courville, A.C.: Unsupervised learning of dense visual representations. In: NeurIPS (2020) 2, 3, 4
38. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NeurIPS (2015) 10, 11, 13
39. Roh, B., Shin, W., Kim, I., Kim, S.: Spatially consistent representation learning. In: CVPR (2021) 2, 3, 4, 7
40. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. IJCV (2015) 9
41. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: CVPR (2015) 3

42. Selvaraju, R.R., Desai, K., Johnson, J., Naik, N.: CASTing your model: Learning to localize improves self-supervised representations. In: CVPR (2021) 3, 4, 7
43. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: DeepFace: Closing the gap to human-level performance in face verification. In: CVPR (2014) 3
44. Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M.: Selective search for object recognition. IJCV (2013) 3
45. Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., Van Gool, L.: Scan: Learning to classify images without labels. In: ECCV (2020) 3
46. Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Van Gool, L.: Unsupervised semantic segmentation by contrasting object mask proposals. In: ICCV (2021) 4
47. Wang, X., Zhang, R., Shen, C., Kong, T., Li, L.: Dense contrastive learning for self-supervised visual pre-training. In: CVPR (2021) 2, 3, 4, 9, 10, 11
48. Wei, F., Gao, Y., Wu, Z., Hu, H., Lin, S.: Aligning pretraining for detection via object-level contrastive learning. In: NeurIPS (2021) 3, 4
49. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019) 10, 11
50. Wu, Z., Xiong, Y., Yu, S.X., Lin, D.: Unsupervised feature learning via non-parametric instance discrimination. In: CVPR (2018) 3
51. Xiao, T., Reed, C.J., Wang, X., Keutzer, K., Darrell, T.: Region similarity representation learning. In: ICCV (2021) 2, 3, 4, 7, 10, 11
52. Xie, E., Ding, J., Wang, W., Zhan, X., Xu, H., Li, Z., Luo, P.: DetCo: Unsupervised contrastive learning for object detection. In: ICCV (2021) 2, 3, 4, 7, 10, 11
53. Xie, J., Zhan, X., Liu, Z., Ong, Y.S., Loy, C.C.: Unsupervised object-level representation learning from scene images. In: NeurIPS (2021) 3, 4
54. 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) 2, 3, 4, 10, 11
55. Yang, C., Wu, Z., Zhou, B., Lin, S.: Instance localization for self-supervised detection pretraining. In: CVPR (2021) 1, 3
56. Zhan, X., Xie, J., Liu, Z., Ong, Y., Loy, C.C.: Online deep clustering for unsupervised representation learning. In: CVPR (2020) 3