This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Region Similarity Representation Learning

Tete Xiao[1*] Colorado J Reed[1*] Xiaolong Wang[2] Kurt Keutzer[1] Trevor Darrell[1]

[1]UC Berkeley [2]UC San Diego

Abstract

*We present Region Similarity Representation Learning (ReSim), a new approach to self-supervised representation learning for localization-based tasks such as object detection and segmentation. While existing work has largely focused on solely learning global representations for an entire image, ReSim learns both regional representations for localization as well as semantic image-level representations. ReSim operates by sliding a fixed-sized window across the overlapping area between two views (*e.g.*, image crops), aligning these areas with their corresponding convolutional feature map regions, and then maximizing the feature similarity across views. As a result, ReSim learns spatially and semantically consistent feature representation throughout the convolutional feature maps of a neural network. A shift or scale of an image region,* e.g.*, a shift or scale of an object, has a corresponding change in the feature maps; this allows downstream tasks to leverage these representations for localization. Through object detection, instance segmentation, and dense pose estimation experiments, we illustrate how ReSim learns representations which significantly improve the localization and classification performance compared to a competitive MoCo-v2 baseline: $+2.7$ AP^{bb}_{75} VOC, $+1.1$ AP^{bb}_{75} COCO, and $+1.9$ AP^{mk} Cityscapes. Code and pre-trained models are released at:* https://github.com/Tete-Xiao/ReSim

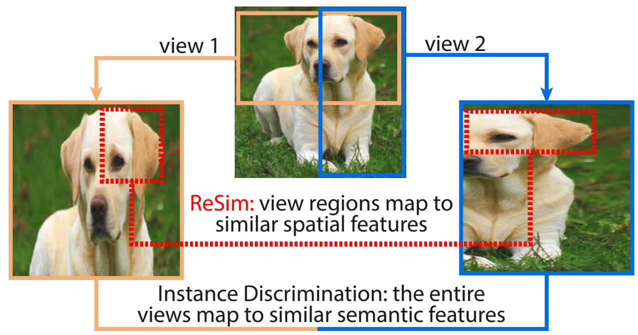

Figure 1. Existing instance discrimination-based self-supervised learning frameworks learn representations by augmenting an image into two different views (*e.g*., cropping/scaling the input image) and then maximizing the similarity between the image features for the entire views. In this work, we present Region Similarity Representation Learning (ReSim), which learns representations by maximizing the similarity of corresponding sub-image *regions* throughout the convolutional layers of a network. In the above example, instance discrimination learns to map both views to the same features, despite the fact that the dog's eyes and ears are in different locations. On the other hand, ReSim learns features which explicitly align these changes with corresponding changes in the convolutional feature maps.

1. Introduction

Recently, self-supervised pre-training has outperformed supervised pre-training for a number of computer vision applications such as image classification and object detection [4, 5, 6]. Much of this recent progress comes from exploiting the *instance discrimination* task [1, 14, 37, 52, 56], in which a network learns image-level features that are invariant to certain image augmentations. Specifically, instance discrimination maximizes the similarity of two *views* of an image, where each view is an augmented version of an image, while minimizing its similarity to views which originate from other images [4, 5, 7].

Chen et al. [5] compared a diverse set of possible augmentations, and found that random cropping and scaling have the largest impact on downstream ImageNet [9] classification performance. Several follow-up works have further explored augmentation policies and confirmed this finding [43, 47, 53]. Through cropping and scaling augmentations and similarity maximization, the network learns to map various scales and crops of an image to the same feature representations. For example, an image crop of the top half of a dog's body and the right half of a dog's body would map to the same representation in the embedding space, which a downstream task could then classify as "dog".

However, instance discrimination uses global image-level feature representations for these views, which is obtained by average pooling the final convolutional feature

*: Equal contribution.

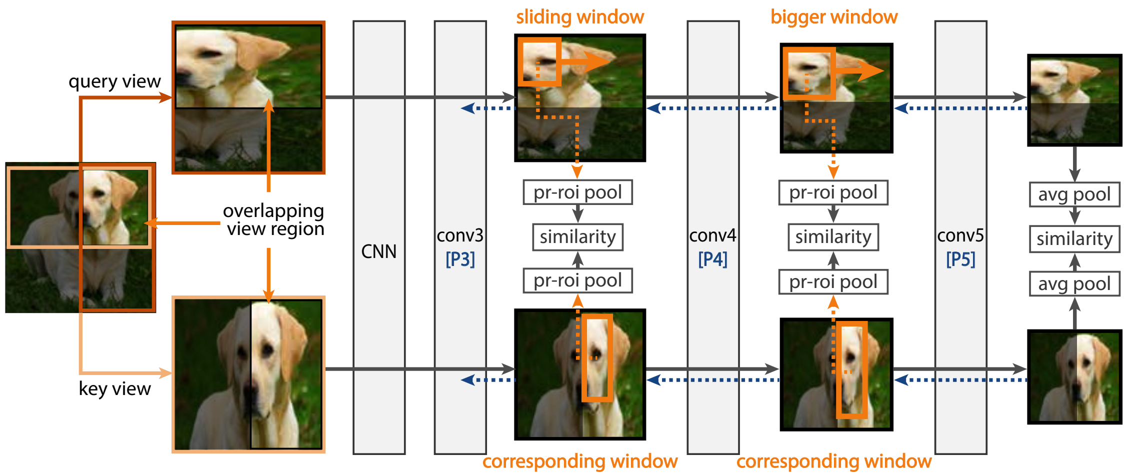

Figure 2. ReSim takes two different views of an image as input, *i.e.*, a *query* and *key* view obtained by cropping/scaling an image. The views have an associated overlapping area as highlighted in each of the above views. Both views are encoded using the same network (CNN), *e.g.*, a ResNet-50 [27]. Before the final convolutional layers in the network, ReSim slides a fixed-size window over the overlapping area between the two views, aligns window regions with corresponding regions in the convolutional feature maps using Precise RoI Pooling from [30], and then maximizes the similarity between these features. Earlier layers use smaller sliding windows as the feature maps have higher spatial resolution. Furthermore, similar to Feature Pyramid Networks [33], the feature maps are combined with semantic top-down features from later convolution layers, as indicated by the blue arrows and "[P3]/[P4]/[P5]" layers. The feature maps following the final convolutional layer are used for instance discrimination learning following either [6] or [7].

map. It does not enforce any type of spatial consistency in the convolutional features (see Figure 1, "instance discrimination" path). For example, different crops that scale and shift the dog's ear will not necessarily have a corresponding scale and shift in the convolutional feature maps throughout the network – instance discrimination only optimizes the final globally pooled features. This is problematic for downstream tasks such as object detection that leverage the spatial information from the convolutional feature maps for object localization.

To address this issue, we introduce Region Similarity Representation Learning (ReSim): a self-supervised pre-training method which learns spatially consistent features across multiple convolutional layers. Inspired by the Region Proposal Network (RPN) used in Faster-RCNN [44], ReSim operates by sliding a fixed-size window across the overlapping region between two image views, mapping the corresponding regions in each view to their associated regions in the convolutions layers throughout the network, and then maximizing the similarity of these convolution feature regions, along with the global similarity objective. See Figure 1 for a high-level difference between ReSim and existing instance discrimination techniques, and see Figure 2 for a detailed description of the full ReSim pipeline.

As we show, maximizing the similarity of these convolutional feature map regions leads to representations that improve object localization for downstream detection and instance segmentation tasks. Furthermore, we extend the framework to learn features at various scales by using sliding windows of multiple sizes at different feature maps. We adopt the Feature Pyramid Network (FPN) design from Lin et al. [33], a design which naturally incorporates feature hierarchies and propagates stronger semantic features to earlier convolutional layers through the top-down path. Region-level self-supervised similarity learning trains feature pyramid layers without labeled supervision and leads to further improvement on downstream tasks.

We conduct object detection, instance segmentation, and dense pose estimation experiments on PASCAL VOC [17], COCO [34], and Cityscapes [8] and show that ReSim learns representations which significantly improve the classification and localization performance compared to a MoCo-v2 baseline, *i.e.*, $+2.7$ $\text{AP}^{\text{bb}}_{75}$ on VOC, $+1.1$ $\text{AP}^{\text{bb}}_{75}$ on COCO, and $+1.9$ AP^{mk} on Cityscapes.

2. Related work

Self-supervised representation learning. The goal of representation learning is to reveal the intrinsic qualities of data in such a way that they are informative and effective for a desired task [2]. Practically, this often manifests as pre-training a deep network so that it can be finetuned for a particular downstream task, see [10, 12, 15, 21, 25, 28, 31, 42, 57]. Recently, SimCLR [5] and MoCo [6, 25] demonstrated substantial improvements by using similar forms of instance contrastive learning where a network was trained to identify a pair of views originating from the same image when contrasted with a large set of views from other images. Following SimCLR and MoCo, later works, such as SwAV [4] and BYOL [23], reported substantial improvements for image classification tasks, but as several follow-up works have shown [11, 54, 55], SwAV and BYOL do not tend to lead to improvements on localization-based tasks such as object detection and segmentation. This decrease in performance indicated that strictly optimizing global, image-level representations could decrease performance for localization.

In contrast to prior work that leveraged instance discrimination to learn global, image-level representations, we propose a region-based pretext task to learn representations for tasks which require both localization and semantic classification. In earlier work, several authors proposed representation learning at the pixel or region level via color prediction [48, 58], key-point encoding [45, 38], optical flow similarity [13], and cycle consistency or frame prediction in videos [32, 49]. We on the other hand build on instance discrimination pretext task and simultaneously learn semantic features from an entire image as well as regional features across multiple scales, shifts, and resolutions.

Object detection and instance segmentation. Object detection and segmentation localize and classify object bounding boxes or pixels within an image – see [35] for a survey and review. Many commonly used object detection and instance segmentation techniques such as Fast-RCNN [20], Faster-RCNN [44], and Mask-RCNN [26] operate through a two-stage process to regress and classify the location of the region to align with a ground truth bounding box.

Existing self-supervised methods have largely focused on learning strong semantic representations by average pooling the final convolutional layer, and they do not explicitly maintain spatial consistency throughout the convolutional layers. In this work, we propose a self-supervised learning method which learns both spatially consistent and semantically sensitive regional features.

Pixel and grid-based contrastive learning. Recently, several related papers emerged: [41] explored pixel-level contrastive learning, whereas our work examines region-level consistency across the convolutional features. Wang et al. [50] proposed contrastive learning using a grid of feature vectors over the feature maps. Rather than maximizing the similarity of the regions, their solution was more akin to clustering, in that they compared all combination of feature vectors from the feature grid and maximized the similarity of the most similar pair. As we show below, our method demonstrates superior performance over these models.

3. Region Similarity Representation Learning

3.1. Preliminaries

State-of-the-art self-supervised representation learning frameworks overwhelmingly exploit instance discrimination as their pretext task, see [29]. In instance discrimination, a reference image is augmented by a set of predefined

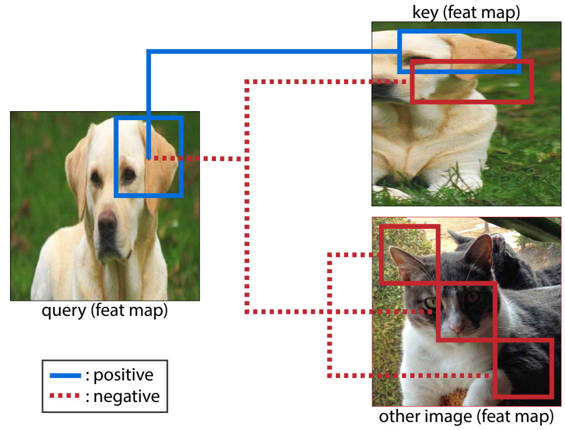

Figure 3. At each of the final convolutional layers, ReSim maximizes the similarity of aligned convolutional feature map regions across the query and key views of an image – the *positive* samples. ReSim simultaneously minimizes the similarity with a set of *negative* samples: the non-positive regions from the same key view (potentially overlapping with the positive region) and any other region from other images.

augmentation modules $\mathcal{T} = [\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_n]$, yielding two views: query and key. For instance, Figure 2 shows the query and key view after cropping and scaling augmentations. The features from the final convolutional layer from the two views are average-pooled and then enforced to be similar in the embedding space by a learning objective, such as contrastive loss [5] or cosine similarity [7].

In the context of representation learning, an ideal encoder for localization-sensitive tasks should consistently encode the same image region, *e.g.*, the same object components across different views of an image, to the same point in the representational embedding space. This is necessary for tasks such as object detection with Faster-RCNN [44] or related detectors [26, 35], which localize and then classify regions within the convolutional feature maps. When regressing the location of the regional features, small adjustments to the region within the convolutional feature map can have a large impact on the object classification prediction at the pixel level, *e.g.*, shifting a feature map region by a few pixels may cause the region to overlap with different objects in the input pixel space, changing the ground-truth classification and regression target.

Similarly, different image regions, *e.g.*, different object components across two different views of an image, should map to different points in the embedding space. By aligning similar image regions to the same representations and dissimilar image regions to dissimilar representations, downstream tasks can leverage the representations to localize and classify the underlying image components. As illustrated in Figure 2, the key and query views of the dog image correspond to the same image even though the two views are scaled and translated differently.

When learning representations for image classification, existing works map all regions of an image to similar representations [5, 6]. This occurs because the widely adopted set of augmentations from Chen et al. [5] include Random Resized Crop (RRC), which enforces that the query and key views always correspond to different regions in the original image. Therefore, we propose a region-level similarity learning component (Figure 2) which enforces that the same regions within the views encode to the same spatially and semantically consistent feature representations.

3.2. Framework Overview

The full ReSim framework, shown in Figure 2, operates as follows: First, ReSim augments an input image into two different views – a query and key, where each view is passed through a common encoder, *e.g.*, a ResNet-50. On a subset of the final convolutional layers, *e.g.*, `conv3` and `conv4`, ReSim slides a fixed-size window over the query view, but only within the overlapping region between the query and key view – this area is highlighted in Figure 2. Following a foundational idea from feature pyramids [33], earlier layers use a smaller window because they have a higher spatial resolution. ReSim then uses Precise RoI Pooling [30] to extract a feature vector from the associated feature map region for both views.

ReSim uses a similarity optimization function to maximize the similarity of the aligned feature vectors from the corresponding view regions. We chose a contrastive optimization function from [25], which also takes dissimilar regions from the same image as well as other images as negative examples, as illustrated in Figure 3. Note: we also explored a cosine similarity optimization function from [7] which does not require negative examples – see §4.

Finally, following the final convolutional layer, ReSim adopts the global average pool of the final feature map for an instance-level contrastive optimization from [25] or a cosine similarity optimization from [7]. As shown by the blue dotted arrows in Figure 2, ReSim uses 1x1 convolutional layers to propagate semantically strong features to the earlier convolutional layers; these are known as *top-down* connections in the original FPN work [33].

3.3. Region-level Similarity

We formulate a region as a rectangular sub-area of an image $\mathcal{I}$ described by its top, left, bottom, and right coordinates (t, l, b, r). An encoder f yields the representation of any region of a given image, *i.e.*, $f(\mathcal{I}, (t, l, b, r))$. Our objective is to perform self-supervised pre-training of the encoder in such a way that the learned representations are tailored for downstream tasks such as object detection and instance segmentation.

Aligning corresponding regions. As shown in Figure 2, ReSim performs self-supervised similarity learning on sub-image level regions across two views of an image. Given two augmented views $\mathcal{I}_q$ and $\mathcal{I}_k$ from the same input image $\mathcal{I}$, and a region in the query view denoted by its coordinates (t_q, l_q, b_q, r_q), we need to find its corresponding region in the key view. From the set of widely adopted augmentations listed in the previous subsection, the only spatial transformations which affect the image coordinates are RRC and horizontal flips. We ensure that the query and key views are *either both horizontally flipped or both not flipped*, as otherwise they should have difference ground-truth regression term for downstream tasks (see object regression in [20]). Thus, it leaves us with RRC alone. Denote the operator as $\mathcal{R}$, the problem can be formulated as, given $\mathcal{I}_q = \mathcal{R}_q(\mathcal{I})$, $\mathcal{I}_k = \mathcal{R}_k(\mathcal{I})$ and (t_q, l_q, b_q, r_q), find $(t_k, l_k, b_k, r_k) = \mathcal{R}_{q\to k}(t_q, l_q, b_q, r_q)$. The solution is trivial as RRC is simple linear translations, therefore $\mathcal{R}_{q\to k} = \mathcal{R}_k(\mathcal{R}_q^{-1})$, where $\mathcal{R}^{-1}$ is the inverse translation of the applied spatial transformations.

Generating candidate regions and extracting features. ReSim generates candidate regions for the region-level contrastive learning by sliding a small window across the overlapping region between the query and key view – see Figure 2. The overlapping areas between the query and key views are selected as valid areas for the sliding window because they can be mapped between the views.

Rather than cropping regions from query/key images and feeding them into encoders, we encode an entire image and extract region features through Precise RoI Pooling [30] which takes in a rectangular window and a convolutional feature map, and yields features of the input window in the size of $h \times w \times c$, where we set $h = 1$ and $w = 1$ to create a feature vector from the region, and c is the same as the number of input feature map channels. Given a candidate region on the input views, ReSim performs region-level similarity across multiple convolutional feature maps at the end of the network, e.g., using $C3$ and $C4$ as shown in Figure 2.

Region similarity on feature pyramids. Lin et al. [33] proposed Feature Pyramid Networks (FPNs), an object detection architecture which combined the low-resolution, semantically strong features from later convolutional layers with high-resolution, semantically weak features from earlier layers via *top-down connections*. As shown by the blue dotted arrows in Figure 2, we add top-down connections to propagate semantically strong features to the earlier convolutional layers used for region similarity learning — we call this variant ReSim-FPN. We follow the proposed methodology from [33]: we use lateral convolutional connections to map the output of the convolutional layers to FPN layers (indicated with $\{P3, P4, P5\}$ in Figure 2), and then directly add the top-down layers after using nearest neighbor sampling to match the spatial resolutions and 1×1 convolutional layers to match the channel dimensions.

Similarity learning objectives. Denote the 4-tuple of a region (t, l, b, r) as u, the set of valid region pairs in query and key from image $\mathcal{I}$ as $\{(u_q^1, u_k^1), (u_q^2, u_k^2), \dots, (u_q^n, u_k^n)\}$, and the region-level similarity function

$$E_{i,j}^{\{k_+,k_-\}} = \exp\left(f(\mathcal{I}_q, u_q^i) \cdot f(\mathcal{I}_{\{k_+,k_-\}}, u_{\{k_+,k_-\}}^j)/\tau\right), \quad (1)$$

where the positive sample $\mathcal{I}_q = \mathcal{T}_q(I), \mathcal{I}_k = \mathcal{T}_k(I)$, $f(I, u)$ is the Precise RoI-pooled feature of region u from image $\mathcal{I}$, and the negative pairs can be *any* image and region pairs not identical to $\mathcal{I}_q$ and u_q – see Figure 3. τ is a temperature parameter to regularize the similarity distribution. We learn the region-level similarity for a query via the following objective:

$$\mathcal{L}_q^{\text{rs}} = \frac{1}{n}\sum_{i=1}^{n} \frac{E_{i,i}^{k+}}{E_{i,i}^{k+} + \sum_{j \neq i} E_{i,j}^{k+} + \sum_{k_-,j} E_{i,j}^{k_-}}, \quad (2)$$

Denote the image-level similarity function with global average pooling (GAP) as

$$D^{\{k_+,k_-\}} = \exp\left(f(\mathcal{I}_q, \text{GAP}) \cdot f(\mathcal{I}_{k\{+,-\}}, \text{GAP})/\tau\right). \quad (3)$$

We use the global image-level similarity from [25]:

$$\mathcal{L}_q^{\text{is}} = \frac{D^{k_+}}{D^{k_+} + \sum_{k_-} D^{k_-}}. \quad (4)$$

We combine the region-level similarity and global image-level similarity objective via a weighting hyperparameter, λ, to yield the final objective:

$$\mathcal{L}_q = \mathcal{L}_q^{\text{rs}} + \lambda \mathcal{L}_q^{\text{is}}. \quad (5)$$

3.4. Framework Configurations

We use two 3×3 convolutions on $C4/P4$ or $P3$ to project the feature maps to 128 channels. The first convolution is followed by Batch Normalization (BN) and ReLU activation. The similarity boxes shown in Figure 2 compute the similarity between aligned convolutional feature map regions. For this similarity computation, we use a momentum contrastive similarity with the associated settings from [6] as default, while we also investigated a cosine similarity with settings from [7] – see §4.1.

For the region-based contrastive similarity, we take negative samples to be both the non-positive regions from the same key view and any other region from other images (Figure 3). WWe find that using a momentum-based queue to maintain a large number of negative samples is unnecessary for region-level samples as there are a large number of negative region samples within each batch, *i.e.*, we can generate over 30,000 negatives within a mini-batch of 256 images on $C4$ – see the appendix for more details.

pre-train	AP	AP_{50}	AP_{75}
random init.	33.8	60.2	33.1
supervised	44.0	72.8	45.5
MoCo-v2	54.4	80.1	60.0
ReSim-C4	**55.9** (+1.5)	**81.3** (+1.2)	**62.1** (+2.1)

Table 1. **Comparisons of ReSim and MoCo-v2 [25] on PASCAL VOC object detection task.** The models are pre-trained on *IN-100*, the weights of which are transferred to a Faster R-CNN R50-C4 subsequently finetuned on VOC `trainval07+12`, and evaluated on `test2007`. AP_k is the average precision at k IoU threshold, and AP is the COCO-style metric (`[0.5:0.95:0.05]`). In the brackets are the deltas to the MoCo-v2 baseline.

We design two variations of ReSim: 1) ReSim-C4, which applies region-similarity learning on ResNet $C4$ feature map *without* FPN; and 2) ReSim-FPN, which applies region-similarity learning on FPN $P3$ and $P4$ feature maps. When applying the Precise RoI Pooling operator to obtain the features for the convolutional feature map regions, we apply the pooling operator on $C4/P4$ with a down-sampling rate of 16, and on feature map $P3$ with a down-sampling rate of 8. By default, we use a sliding window of 48 pixels with a stride of 32 pixels on the input image on $C4/P4$, and a sliding window of 32 pixels with a stride of 24 pixels on $P3$. We ablate these settings in § 4.1.

4. Experiments

We study the transfer capability of ReSim for localization-dependent downstream tasks. We perform *self-supervised pre-training* on two splits of the ImageNet dataset [9]: 1) 1000-category ImageNet (IN-1K), the standard ImageNet training set containing ∼1.25M images; and 2) 100-category ImageNet (IN-100), a subset of IN-1K split containing ∼125k images, following previous works [46, 53]. The pre-trained model is subsequently finetuned on PASCAL VOC [18] for object detection, COCO [34] and Cityscapes [8] for instance segmentation, and DensePose [24] for dense pose estimation. For the ablation studies, we use IN-100 for pre-training. We report full IN-1K results for selected best models.

Training. Our hyperparameters closely follow the adopted self-supervised learning framework MoCo-v2 [6]. We use a ResNet-50 [27] backbone, pre-training for 200 epochs for IN-1K, 500 epochs for IN-100, with a mini-batch size of 256 on 8 GPUs. See A.1 for more details. ReSim-FPN takes ∼62 hours to train 200 IN1K-epochs with 8 NVIDIA V100 GPUs, compared to ∼53 hours for MoCo. Note that the design of ReSim does *not* affect the complexity of its transferred downstream model.

4.1. IN-100 Study

Setup. Throughout these experiments, we adopt the commonly used PASCAL VOC object detection task [16]. We

pre-train	AP	AP_{50}	AP_{75}
SimSiam	54.6	80.6	60.7
+ReSim-C4	55.7 (+1.1)	81.2 (+0.6)	61.9 (+1.2)
MoCo-v2	54.4	80.1	60.0
+ReSim-C4	**55.9** (+1.5)	**81.3** (+1.2)	**62.1** (+2.1)

Table 2. **Building ReSim on various self-supervised representation learning frameworks**, *i.e.*, SimSiam [7] and MoCo-v2. All models are obtained by self-supervised pre-training on IN-100, transferring weights to a Faster R-CNN R50-C4 detector subsequently finetuned on VOC `trainval07+12`, and evaluated on `test2007`. We use the prediction and projection heads as in [7] for ReSim on SimSiam. The improvement over SimSiam shows that ReSim is applicable to multiple frameworks.

use a Faster R-CNN [44] with R50-C4 [26] backbone. As in [25], we unfreeze all BN layers and synchronizing their statistics across GPUs [40] during training. The short-side length of input images is randomly selected from [480, 800] pixels during training and fixed at 800 for inference. Training is performed on `trainval07+12` set (~16.5k images), and testing is performed on `test2007` set (~4.9k images), unless otherwise specified. The network is trained on 8 GPUs of mini-batch size 16. Training takes 24,000 iterations with a base learning rate 0.02. The learning rate is decreased by a factor of $1/10$ at the 18,000 and 22,000 iteration. All settings follow [25].

Comparisons with baselines. We first compare the ReSim framework with similarity learning on C4 (ReSim-C4) to the MoCo-v2 baseline. Since ReSim is built on MoCo-v2, the key difference between the two frameworks is the presence of similarity learning across the convolutional feature map regions in ReSim. Table 1 shows the VOC transfer results of the two frameworks, as well as the models which are supervise pre-trained on ImageNet and randomly initialized. Albeit competing against a strong reference, ReSim surpasses MoCo-v2 by a large margin of 1.5, 1.2 and 2.1 at AP, AP_{50}, and AP_{75}, respectively. The performance gap at high AP metric (AP_{75}) is larger than it at low AP metric (AP_{50}) which is less localization-dependent.

Self-supervised learning frameworks. We investigated ReSim with an additional self-supervised learning framework. Specifically, we selected the recently proposed Siamese-based framework, SimSiam [7], for its simplicity and effectiveness. We use identical designs of its projection and prediction heads (see their reference for details) for ReSim as the image-level similarity learning heads presented in SimSiam for consistency. Note that since those heads have several sequentially connected 2048 channel layers rather than a single 2048-128 channel connection used in MoCo-v2, the design significantly increases computational complexity over ReSim on MoCo, *i.e.*, ReSim-FPN based on SimSiam takes 46.7 hours to train on IN-100, compared to 15.5 hrs for ReSim-FPN based on MoCo.

pre-train	window	AP	AP_{50}	AP_{75}
MoCo-v2	N/A	54.4	80.1	60.0
ReSim-C4	W16-S16	55.0	80.7	61.3
ReSim-C4	W48-S16	55.8	81.2	61.6
ReSim-C4	W48-S32	55.9	**81.3**	62.1
ReSim-C4	W64-S48	**56.0**	81.0	**62.7**

(a) Full finetuning setting on VOC

pre-train	window	AP	AP_{50}	AP_{75}
MoCo-v2	N/A	47.1	75.4	50.6
ReSim-C4	W16-S16	48.7	76.0	53.2
ReSim-C4	W48-S16	49.5	76.7	54.1
ReSim-C4	W48-S32	**50.0**	**77.2**	**55.0**
ReSim-C4	W64-S48	49.7	**77.2**	54.3

(b) Frozen-backbone setting, finetuning on VOC

Table 3. **Comparisons of various sliding window sizes and strides under (a) full-finetuning and (b) frozen-backbone settings.** "Wl-Sk" denotes sliding a window of size $l \times l$ at stride k. ReSim-C4 by default uses W48-S32 setting. Inspired by the linear classification metric for classification, in the frozen-backbone setting the backbone of object detector is frozen from finetuning to directly evaluate the quality of features for object detection. The more significant improvement in the frozen-backbone setting implies ReSim learns features of better quality for localization-dependent task.

Table 2 shows the results. ReSim built upon SimSiam improves the baseline by 1.1, 0.6 and 1.2 at AP, AP_{50}, and AP_{75}, respectively, and is comparable to ReSim built upon MoCo. Due to the extra pre-training time, we choose MoCo as our backbone framework.

Frozen backbone for finetuning on VOC. Linear classification, *i.e.*, training a linear classifier on a frozen backbone is widely adopted to evaluate the representations *for classification* [4, 5, 7, 25, 23, 52]. We design a similar feature evaluation experiment on object detection. After transferring the self-supervise pre-trained weights to a Faster R-CNN object detector, we freeze the backbone and finetune the RPN and object classifier head on labeled object data, as RPN is randomly initialized and object classifier is task-specific. For the Faster R-CNN R50-C4 detector, specifically, all weight layers from $C1$ to $C4$ feature map, along with statistics of BN layers are frozen; weights in RPN and C5 (object classification head) are finetuned. According to Goyal et al. [22], this type of evaluation is ideal for representation learning because finetuning an entire network "evaluates not only the quality of the representations but also the initialization and optimization method."

Table 3(b) shows the results. ReSim improves the baseline MoCo-v2 (default W48-S32) by 2.9, 1.8, 4.4 at AP, AP_{50}, and AP_{75}, respectively, a much larger margin compared to the full-finetune setting. The improvement is particularly significant at high IoU metric AP_{75}. It indicates ReSim yields features of higher quality for localization-dependent tasks, particularly when limiting the impact of the initialization and optimizations used during finetuning.

pre-train	AP	AP$_{50}$	AP$_{75}$
random init.	33.8	60.2	33.1
supervised	53.5	81.3	58.8
Jigsaw [22]	48.9 (−4.6)	75.1 (−6.2)	52.9 (−5.9)
Rotation [22]	46.3 (−7.2)	72.5 (−8.8)	49.3 (−9.5)
NPID++ [39]	52.3 (−1.2)	79.1 (−2.2)	56.9 (−1.9)
SimCLR [5]	51.5 (−2.0)	79.4 (−1.9)	55.6 (−3.2)
PIRL [39]	54.0 (+0.5)	80.7 (−0.6)	59.7 (+0.9)
BoWNet [19]	55.8 (+2.3)	81.3 (+0.0)	61.1 (+2.3)
MoCo [25]	55.9 (+2.4)	81.5 (+0.2)	62.6 (+3.8)
MoCo-v2 [6]	57.0 (+3.5)	82.4 (+1.1)	63.6 (+4.8)
SwAV [4]	56.1 (+2.6)	82.6 (+1.3)	62.7 (+3.9)
DenseCL [50]	58.7 (+5.2)	82.8 (+1.5)	65.2 (+6.4)
ReSim-C4	58.7 (+5.2)	**83.1** (+1.8)	**66.3** (+7.5)
ReSim-FPN	**59.2** (+5.7)	82.9 (+1.6)	65.9 (+7.1)

Table 4. **Comparison with previous works on PASCAL VOC** `trainval07+12`**, evaluated on** `test2007`. Jigsaw and Rotation results are from [39]. SimCLR result is from [50]. Both ReSim-C4 and ReSim-FPN improve MoCo-v2 baseline, and achieve state-of-the-art over competitive concurrent work DenseCL [50]. In the brackets are the deltas to the ImageNet supervised pre-training baseline. **Green**: deltas at least +1.0.

Region size and stride. We vary the size and stride of sliding windows in ReSim-C4 to determine the optimal combination. The results are reported in Table 3(a,b). "Wl-Sk" denotes sliding a window of size $l \times l$ at stride k. By considering the performance both at frozen-backbone and standard full-finetune settings, we choose W48-S32 setting as the default for ReSim-C4.

RoI Align vs. Precise RoI Pooling. We compare Precise RoI Pooling [30] and RoI Align [26] as the region feature extraction operator. On VOC, ReSim using RoI Align achieves AP/AP$_{50}$/AP$_{75}$ of 55.7/81.0/62.0 with all parameters finetuned, and 48.9/76.6/53.0 with a frozen backbone. Comparing these results with Table 3 indicates that both RoI Align and Precise RoI Pooling work well, but in the frozen-backbone setting, RoI Align performs worse than Precise RoI Pooling. We conjecture this is because Precise RoI Pooling does not heuristically set the number of sampling points, enabling a more robust and simpler optimization.

4.2. IN-1K Results

Following the IN-100 experiments, we studied pre-training on the full IN-1K and report the results on PASCAL VOC, Cityscapes, COCO detection and COCO dense pose estimation. We evaluate two models as introduced in §3.4: 1) ReSim-C4, which performs region-level similarity on $C4$ feature map of ResNet; and 2) ReSim-FPN, which performs region-level similarity on the $P3$ and $P4$ feature maps on the top-down path of FPN. We transfer the backbone ResNet weights for finetuning downstream tasks; *if the network of a downstream task adopts the FPN design*, we can transfer the FPN weights from ReSim-FPN to the new network. We term this as ReSim-FPN^T, and report the results on selected downstream tasks which use FPN.

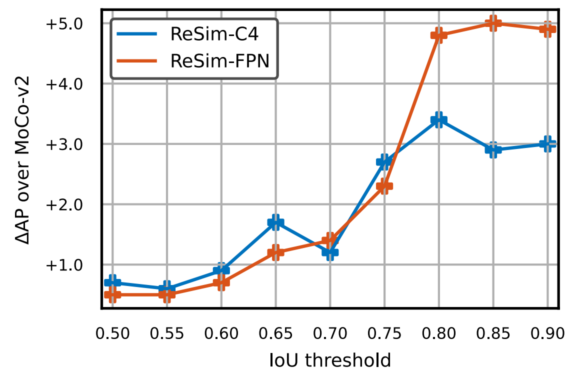

Figure 4. **ΔAP of ReSim over MoCo-v2 at various IoU thresholds on PASCAL VOC.** As the IoU threshold increases, both ReSim methods perform considerably better than MoCo-v2, especially at higher IoU threshold, indicating that the ReSim features have better localization capability.

PASCAL VOC. We use the Faster R-CNN R50-C4 detector on VOC and adopt the same setup for PASCAL VOC as described in §4.1. We first report the results of finetuning on `trainval07+12` and evaluating on `test2007`. Table 4 compares our method with a series of previous state-of-the-arts. Respectively, ReSim-C4 and ReSim-FPN improve over their baseline MoCo-v2 by $1.7/0.7/2.7$ and $2.2/0.5/2.3$ at AP/AP$_{50}$/AP$_{75}$, and claims state-of-the-art results over the competitive concurrent work DenseCL [50].

In Figure 4 we show the AP improvements of ReSim over MoCo-v2 at IoU threshold from 0.5 to 0.9. Not only do both ReSim methods outperform MoCo-v2, the larger improvement at high IoU indicates superior localization accuracy, particular for ReSim-FPN. The growing performance gap as the IoU threshold increases indicates that ReSim has led to improved localization of the objects.

We also report the results of finetuning on PASCAL VOC `trainval2007` and evaluating on `test2007`. ReSim-C4 achieves $48.9/75.4/53.5$ at AP/AP$_{50}$/AP$_{75}$, in comparison with $46.6/72.8/50.9$ for MoCo-v2.

COCO Object Detection and Instance Segmentation. We use Mask-RCNN [26] (R50-FPN) with Sync-BN as our base model. The short-side length of input images is randomly selected from [640, 800] pixels during training and fixed at 800 for inference. Training is performed on `train2017` split, and testing is performed on `val2017` split. Two metrics are used, Average Precision on bounding-boxes (AP$^{\text{bb}}$), and on masks (AP$^{\text{mk}}$). We report finetuning results on the standard 1x and 2x schedules, the latter of which trains twice as long as the former.

Table 5 shows the results. All variations of ReSim improve over its MoCo-v2 counterpart under both box and mask metrics, outperforming the supervised pre-training baseline. Unlike MoCo-v2, ReSim improves with the $1\times$ schedule. We note that ReSim-FPN outperforms ReSim-

pretrain	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init	31.0	49.5	33.2	28.5	46.8	30.4
supervised	38.9	59.6	42.7	35.4	56.5	38.1
MoCo-v2	38.9 (+0.0)	59.2 (−0.4)	42.4 (−0.3)	35.4 (+0.0)	56.2 (−0.3)	37.8 (−0.3)
VADeR [41]	39.2 (+0.3)	59.7 (+0.1)	42.7 (+0.0)	35.6 (+0.2)	56.7 (+0.2)	38.2 (+0.1)
DenseCL [50]†	39.4 (+0.5)	59.9 (+0.3)	42.7 (+0.0)	35.6 (+0.2)	56.7 (+0.2)	38.2 (+0.1)
ReSim-C4	39.3 (+0.4)	59.7 (+0.1)	43.1 (+0.4)	35.7 (+0.3)	56.7 (+0.2)	38.1 (+0.0)
ReSim-FPN	39.5 (+0.6)	59.9 (+0.3)	43.3 (+0.6)	35.8 (+0.4)	57.0 (+0.5)	38.4 (+0.3)
ReSim-FPN^T	**39.8** (+0.9)	**60.2** (+0.6)	**43.5** (+0.8)	**36.0** (+0.6)	**57.1** (+0.6)	**38.6** (+0.5)
ReSim-FPN^T (400 ep)	**40.3** (+1.4)	**60.6** (+1.0)	**44.2** (+1.5)	**36.4** (+1.0)	**57.5** (+1.0)	**38.9** (+0.8)

(a) Mask R-CNN R50-FPN, 1× schedule

pretrain	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init	36.7	56.7	40.0	33.7	53.8	35.9
supervised	40.6	61.3	44.4	36.8	58.1	39.5
MoCo-v2	40.9 (+0.3)	61.5 (+0.2)	44.6 (+0.2)	37.0 (+0.2)	58.4 (+0.3)	39.6 (+0.1)
VADeR [41]	-	-	-	-	-	-
DenseCL [50]†	41.2 (+0.6)	61.9 (+0.6)	45.1 (+0.7)	37.3 (+0.5)	58.9 (+0.8)	40.1 (+0.6)
ReSim-C4	41.1 (+0.5)	61.5 (+0.2)	44.8 (+0.4)	37.1 (+0.3)	58.6 (+0.5)	39.8 (+0.3)
ReSim-FPN	**41.4** (+0.8)	61.8 (+0.5)	45.4 (+1.0)	**37.5** (+0.7)	**59.1** (+1.0)	**40.4** (+0.9)
ReSim-FPN^T	**41.4** (+0.8)	**61.9** (+0.6)	**45.4** (+1.0)	**37.5** (+0.7)	**59.1** (+1.0)	40.3 (+0.8)
ReSim-FPN^T (400 ep)	**41.9** (+1.3)	**62.4** (+1.1)	**45.9** (+1.5)	**37.9** (+1.1)	**59.4** (+1.3)	**40.6** (+1.1)

(b) Mask R-CNN R50-FPN, 2× schedule

Table 5. **Results on COCO object detection and instance segmentation tasks.** The models are pre-trained on IN-1K for 200 epochs (except the final entry, which shows extended performance out to 400 epochs), the weights of which are transferred to Mask R-CNN R50-FPN model, subsequently finetuned on `train2017` (1× and 2× schedules) and evaluated on `val2017`. Averaging precision on bounding-boxes (AP^{bb}) and masks (AP^{mk}) are used as benchmark metrics. All ReSim models outperform VADeR [41] and the MoCo-v2 counterpart, surpassing the supervised pre-training under all metrics (we report VADeR results from their paper which did not include a 2x schedule). In the brackets are the gaps to the ImageNet supervised pre-training baseline. **Green**: deltas at least +0.5. †: Pre-trained weights are downloaded from the official releases of the authors and trained on our COCO frameworks, as [50] adopted a different COCO finetuning baseline settings from the common approach used in this paper and other previous works [25].

pre-train	COCO dense pose estimation AP^{gpsm}	AP^{gpsm}_{50}	AP^{gpsm}_{75}
supervised	65.8	92.6	77.8
MoCo-v2	66.2 (+0.4)	93.1 (+0.5)	77.4 (−0.4)
ReSim-C4	66.6 (+0.8)	92.9 (+0.3)	78.5 (+0.7)
ReSim-FPN	**67.1** (+1.3)	**93.2** (+0.6)	**80.0** (+2.2)
ReSim-FPN^T	66.8 (+1.0)	92.8 (+0.2)	79.2 (+1.4)

pre-train	Cityscapes Instance Seg. AP^{mk}	AP^{mk}_{50}
supervised	33.0	59.8
MoCo-v2	33.7 (+0.7)	61.8 (+2.0)
ReSim-C4	35.4 (+2.4)	63.1 (+3.3)
ReSim-FPN	**35.6** (+2.6)	**63.4** (+3.6)
ReSim-FPN^T	35.4 (+2.4)	63.3 (+3.5)

Table 6. **Results on COCO dense pose estimation and Cityscapes instance segmentation tasks.** The performance deltas with the supervised baseline is shown in the brackets. **Green**: deltas at least +0.5.

C4, and in turn ReSim-FPN^T, in which FPN weights are transferred, outperforms ReSim-FPN, demonstrating the effectiveness of our framework on feature pyramids. Furthermore, the final row in Table 5 shows a continued increase in performance after 400 epochs of pre-training.

COCO DensePose Estimation. We use DensePose R-CNN [24] (R50-FPN) as our base model. We add Sync-BN layers in FPN. We use the DensePose implementation in [51]. The model is trained on `train2014` split (1× schedule) and evaluated on `val2014`. We report results under masked-GPS (AP^{gpsm}). Table 6 upper shows the comparisons of ReSim versus MoCo-v2 and supervised pre-training counterparts. Our ReSim-FPN improves over the supervised baseline by 2.2 points under AP^{gpsm}_{75}.

Cityscapes Instance Segmentation. We use Mask R-CNN [26] with R50-FPN as our base model, add Batch Normalization layers before the FPN, and finetune and sync all such layers during training. The model is trained on the standard splits and training settings from [51]. Table 6 lower shows the comparisons of ReSim versus MoCo-v2 and supervised pre-training counterparts. All of the ReSim variants substantially improve over the supervised baseline, *e.g.*, by 3.3-3.6 points under AP^{mk}_{50}.

5. Conclusion

We presented ReSim, a novel self-supervised representation learning algorithm for localization-based tasks such as object detection and segmentation. ReSim performs both global and region-level contrastive learning to simultaneously learn semantic and spatial feature representations. The spatially consistent feature representations are learned on regions of the convolutional feature maps, while the global representations are learned over the entire image. In order to learn hierarchical representations, we further incorporated feature pyramids into the ReSim pre-training, and showed how these parameters can also be transferred to popular FPN-based frameworks for downstream tasks. Our method shows significant improvements for various localization-based tasks, such as object detection, instance segmentation, and dense pose estimation.

Acknowledgements Prof. Darrell's group was supported in part by DoD including DARPA's XAI, LwLL, and SemaFor programs, as well as BAIR's industrial alliance programs. Prof. Keutzer's group was supported in part by Alibaba, Amazon, Google, Facebook, Intel, and Samsung as well as BAIR and BDD at Berkeley. Prof. Wang's group was supported, in part, by gifts from Qualcomm and TuSimple. We would additionally like to thank Amir Bar, Ani Nrusimha, and Ilija Radosavovic for feedback and discussions. Wandb [3] graciously provided academic research accounts to track our experimental analyses.

References

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

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

[7] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[11] Jian Ding, Enze Xie, Hang Xu, Chenhan Jiang, Zhenguo Li, Ping Luo, and Gui-Song Xia. Unsupervised pretraining for object detection by patch reidentification. *arXiv preprint arXiv:2103.04814*, 2021.

[12] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014.

[13] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[14] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in neural information processing systems*, pages 766–774, 2014.

[15] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[16] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge 2007 (voc2007) results. 2007.

[18] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[19] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Learning representations by predicting bags of visual words. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6928–6938, 2020.

[20] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[22] Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6391–6400, 2019.

[23] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[24] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7297–7306, 2018.

[25] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[26] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[28] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

[29] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2021.

[30] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018.

[31] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[32] Xueting Li, Sifei Liu, Shalini De Mello, Xiaolong Wang, Jan Kautz, and Ming-Hsuan Yang. Joint-task self-supervised learning for temporal correspondence. *arXiv preprint arXiv:1909.11895*, 2019.

[33] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[34] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[35] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[36] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[37] Tomasz Malisiewicz and Alexei A Efros. Recognition by association via learning per-exemplar distances. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[38] Matthias Minderer, Chen Sun, Ruben Villegas, Forrester Cole, Kevin Murphy, and Honglak Lee. Unsupervised learning of object structure and dynamics from videos. *arXiv preprint arXiv:1906.07889*, 2019.

[39] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[40] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6181–6189, 2018.

[41] Pedro O Pinheiro, Amjad Almahairi, Ryan Y Benmaleck, Florian Golemo, and Aaron Courville. Unsupervised learning of dense visual representations. *arXiv preprint arXiv:2011.05499*, 2020.

[42] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training.

[43] Colorado J Reed, Sean Metzger, Aravind Srinivas, Trevor Darrell, and Kurt Keutzer. Selfaugment: Automatic augmentation policies for self-supervised learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2021.

[44] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[45] James Thewlis, Hakan Bilen, and Andrea Vedaldi. Unsupervised learning of object frames by dense equivariant image labelling. *arXiv preprint arXiv:1706.02932*, 2017.

[46] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

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

[48] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 391–408, 2018.

[49] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2566–2576, 2019.

[50] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. *arXiv preprint arXiv:2011.09157*, 2020.

[51] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.

[52] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3733–3742, 2018.

[53] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *International Conference on Learning Representations*, 2021.

[54] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. *arXiv preprint arXiv:2102.04803*, 2021.

[55] Ceyuan Yang, Zhirong Wu, Bolei Zhou, and Stephen Lin. Instance localization for self-supervised detection pretraining. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3987–3996, 2021.

[56] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019.

[57] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[58] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.