RegionCL: Can Simple Region Swapping Contribute to Contrastive Learning?

Yufei Xu1∗ Qiming Zhang1∗ Jing Zhang1 Dacheng Tao2,1
1The University of Sydney 2JD Explore Academy

Abstract

Self-supervised methods (SSL) have achieved significant success via maximizing the mutual information between two augmented views, where cropping is a popular augmentation technique. Cropped regions are widely used to construct positive pairs, while the left regions after cropping have rarely been explored in existing methods, although they together constitute the same image instance and both contribute to the description of the category. In this paper, we make the first attempt to demonstrate the importance of both regions in cropping from a complete perspective and propose a simple yet effective pretext task called Region Contrastive Learning (RegionCL). Specifically, given two different images, we randomly crop a region (called the paste view) from each image with the same size and swap them to compose two new images together with the left regions (called the canvas view), respectively. Then, contrastive pairs can be efficiently constructed according to the following simple criteria, i.e., each view is (1) positive with views augmented from the same original image and (2) negative with views augmented from other images. With minor modifications to popular SSL methods, RegionCL exploits those abundant pairs and helps the model distinguish the regions features from both canvas and paste views, therefore learning better visual representations. Experiments on ImageNet, MS COCO, and Cityscapes demonstrate that RegionCL improves MoCo v2, DenseCL, and SimSiam by large margins and achieves state-of-the-art performance on classification, detection, and segmentation tasks. The code will be available at Code.

1. Introduction

Self-supervised learning (SSL) has become an active research topic in computer vision because of its ability to learn generalizable representations from large-scale unlabeled data and offer good performance in downstream tasks, e.g., classification, detection, segmentation, etc. Contrastive learning, one of the popular directions in SSL, has attracted a lot of attention due to its ease of use in pretext designing and capacity to generalize across various visual tasks.

Current contrastive learning methods typically use augmented views of the same image as positive pairs and maximize their mutual information. Cropping is by far the most popular augmentation technique. By randomly cropping regions from the same images and treating the cropped regions as positive pairs, the methods in [5–7, 9, 15, 18] have shown promising results in image classification. Multipitch [3,4] has been investigated as a way to improve performance even further by generating more diverse candidates and facilitating the model learning a better feature representation. Constrained cropping strategies have recently been developed by some methods [28, 37–39, 44] to ensure that two cropped views contain shared regions of a specific size and to improve models’ performance on dense prediction tasks by constructing contrastive pairs within the shared regions. These methods have achieved superior performance on a variety of visual tasks by leveraging various cropping strategies to construct contrastive pairs during pretraining. However, the left regions after cropping have received lit-
tle attention, despite the fact that the cropped and left regions together make up the same image instance and both contribute to the category’s description. We argue that using both regions during pretraining would help the model learn better complete visual representations of object instances, which will improve the model’s performance on downstream classification and dense prediction tasks.

Based on this motivation, we propose a simple yet effective pretext task called Region Contrastive Learning (RegionCL). Specifically, given two different images, RegionCL randomly crops a region (called the paste view) from each image with the same size and swaps them to compose two new images together with the left regions (called the canvas view), respectively. It is worth noting that the two views that compose the new images are from different source images. Then, contrastive pairs can be constructed following the simple criteria, i.e., each view is (1) positive with views augmented from the same original image and (2) negative with views augmented from other images. In this way, RegionCL generates abundant pairs that contain not only the instance-level pairs as other methods [5, 7] but also the region-level pairs, e.g., the paste and canvas views in the composite images. By exploiting these pairs in popular SSL frameworks, RegionCL helps the models learn better feature representations of object instances owing to the abundant contrastive supervisory signals at both instance and region levels, delivering better performance on various downstream tasks. As shown in Figure 1, RegionCL helps MoCo v2 [7], DenseCL [33], and SimSiam [9] improve their linear classification accuracy by 2%~5% on the ImageNet [13] dataset and object detection performance by 0.8~1.0 mAP on the MS COCO [8] dataset, simultaneously.

In summary, the contribution of the paper is threefold:

1. We make the first attempt to demonstrate the importance of both regions, i.e., the cropped and left regions in cropping, from a complete perspective for self-supervised learning.
2. We propose a simple yet effective pretext task, i.e., RegionCL. It is compatible with various popular SSL methods with minor modifications and improves their performance on many downstream visual tasks.
3. Extensive experimental results with MoCo v2, SimSiam, and DenseCL on the ImageNet, MS COCO, and Cityscapes datasets demonstrate the effectiveness of the proposed RegionCL on classification, detection, and instance and semantic segmentation tasks.

2. Related Work

Self-supervised learning has shown great potential in learning visual representations that can generalize to a series of downstream visual tasks. Early works [21, 24, 26, 43] generate pseudo labels using specific tasks such as image corruption and restoration, reordering, re-colorization. However, the models pretrained in these tasks may be too coupled with the designed tasks and the transfer results on other visual tasks may not be competitive.

Recently, contrastive learning [5–7, 9, 15, 18, 31] has made rapid progress and shown promising transfer performance. Typically, they take augmented views from the same (different) images as positive (negative) pairs and learn to pull the features from positive pairs while pushing away those from the negative pairs via a contrastive loss. Among the augmentation techniques, cropping plays an important role in improving the performance, as shown in [5]. Taking the cropped augmented views as input, SimCLR [5] obtains superior results on image classification. MoCo [7, 18] utilizes a momentum encoder to better utilize the cropped views during pretraining, as it provides consistent optimization direction. However, as cropping at a single resolution may not provide enough descriptions of the target object, a multi-crop strategy is explored in [3, 4] by fusing several cropped views at different resolutions. Such a strategy helps the model learn a better feature representation at different scales and boost the performance on image classification.

On the other hand, [27, 28, 33, 38] focus on advancing the performance on dense prediction tasks by establishing dense correspondences between the augmented cropped views. Some methods [28, 44] further design constrained cropping strategies during pretraining to improve the transfer results on detection, e.g., they require the two cropped views have some shared regions and attract the dense positive features within the shared regions based on explicit spatial correspondences. By exploring different properties of cropping-based augmented views, these methods obtain superior performance. However, the left regions after cropping have rarely been explored. Different from them, we make the first attempt to investigate the importance of both regions in this study. Our RegionCL uses a simple region swapping strategy to generate abundant contrastive pairs at...
both instance and region levels, from which the model can learn better visual representations of object instances.

Although several methods also explore region-level contrastive learning, they have not yet explored the complementary left regions after cropping, i.e., the canvas view in our paper. For example, SCRL [28], DUPR [14], and MaskCo [44] incorporate bounding boxes generation and alignment between the shared area of two cropped views during pretraining. InstLoc [39] further introduces anchors with bounding boxes augmentations to boost the transfer results on dense prediction tasks at the cost of decreased image classification accuracy. DetCo [37] designs delicate cropping strategies to generate separate patches at different resolutions and uses extra memory banks to capture patch features. In addition to the above-mentioned differences, the proposed RegionCL requires no extra information such as bounding boxes alignment and is compatible with popular SSL frameworks such as MoCo v2 [7], SimSiam [9], and DenseCL [33], with only minor modifications to them. It is also noteworthy that although RegionCL shares a similar implementation like CutMix [42], a popular data augmentation technique in supervised image classification, they have many differences. First, ground truth labels are required in CutMix to generate fused labels while RegionCL generates contrastive pairs naively according to the simple criteria. Second, CutMix helps supervised learning through the weighted cross-entropy loss derived from fused labels, while RegionCL works in the SSL domain with the contrastive loss derived from the abundant contrastive pairs.

### 3. Method

The details of the region swapping strategy for constructing region-level contrastive pairs are firstly introduced in this section. Then, taking MoCo v2 [7] as an example, we discuss the proposed RegionCL in depth. The extension of RegionCL to other representative SSL methods will also be presented, such as DenseCL [33] and SimSiam [9], where DenseCL focuses on dense prediction tasks and SimSiam does not require negative pairs during training.

#### 3.1. The region swapping strategy

Different from current methods that only use the cropped regions, we take both the cropped and left regions into consideration for self-supervised learning. Given two different images, we randomly crop a region with the same size from each image and swap them to compose two new images. As shown in Figure 2-C, the composite image after region swapping contains two views: one is the paste view (the cropped region), i.e., the dog’s face, and the other is called the canvas view (the left region), i.e., the cat’s head. Specifically, we first sample a size of the paste view, i.e., the height and width, and then determine the coordinates of the origin point from which the cropping starts. We make sure the size and location of the cropped region match the network’s downsampling ratio $R$ during region swapping so that the region’s feature can be directly extracted from the feature map by a simple operation of mask pooling. Specifically, the height and width are determined by $R$ and a discrete uniform distribution $C \sim U(C_L, C_U)$, where the ratio $R$ is typically 32 for ResNet [20] and $C_L, C_U$ are two predefined hyper-parameters shared for both spatial dimensions for simplicity. They are set to 3 and 5 in the paper unless specified. We sample twice from the distribution $C$ and get two observations $c_h$ and $c_w$. Then, we calculate the width and height as $r_h = c_h \times R$, $r_w = c_w \times R$, respectively. Then we uniformly sample the origin point coordinates $(r_x, r_y)$ from a valid range that guarantees there is enough remaining area to crop a patch of size $r_w \times r_h$. In this way, the candidate region is determined by $(r_x, r_y, r_w, r_h)$. It is note-
worthy that within a mini-batch of the training images, we use the same coordinates \((r_x, r_y, r_w, r_h)\) for efficient batch-wise implementation during training.

### 3.2. The Region contrastive learning

**The architecture.** We take MoCo v2 as an example here to describe the proposed RegionCL method in depth, denoted as RegionCL-M. The overall architecture of RegionCL-M is presented in Figure 3. As can be seen, RegionCL-M has exactly the same architecture with MoCo v2 and only requires marginal modifications to the inputs and learning objectives, i.e., a region-level branch in the middle.

RegionCL-M uses a Siamese network structure in pre-training. Given image instances \(x\), RegionCL-M first creates two randomly augmented views, i.e., the query view \(x^q\) and the key view \(x^k\) following the same augmentation strategy as in MoCo v2. The online network processes the query view, and the other branch, i.e., the momentum updated network, processes the key view. Unlike other methods that also utilize region-level contrastive learning [14, 28, 44], we follow the same cropping strategies as in MoCo v2 and do not need the two views \(x^q, x^k\) to have a sufficiently large overlap, which keeps the diversity of the contrastive pair candidates. We construct the region-level contrastive pairs using the region-swapping strategy. Specifically, given two image instances from the query view \(x^q\), we randomly crop a region with the same size in each image instance and swap them to compose two new images \(x^p, x^c\), where the cropped region after swapping and the left region in the new images are the paste view \(x^p\) and canvas view \(x^c\), respectively.

**The region- and instance-level contrastive loss.** In this way, we have a total of four different views, i.e., the query view, the paste view, the canvas view, and the key view, denoted as \(x^q, x^p, x^c, x^k\), respectively. RegionCL-M projects these views into the corresponding feature representations \(q, p, c, k\), among which the features \(q, k\) are instance-level feature representations while the features \(p, c\) are region-level feature representations. Note that features of the paste view and canvas view are extracted from the feature maps of \(x^p, x^c\) via mask pooling, where the mask is obtained according to the coordinates \((r_x, r_y, r_w, r_h)\) as described in Section 3.1. The other views’ features are from the global average pooling upon the corresponding feature maps. Then we can efficiently construct the contrastive pairs for these views according to the simple criteria, i.e., each view is (1) positive with views augmented from the same original image and (2) negative with views augmented from other images. We follow the practice of MoCo v2 in our implementation and ignore the positive pairs whose features are both generated by the online network to stabilize the training.

We use contrastive loss [16, 18] as the learning objectives, which can be thought of as training an encoder for a dictionary lookup task at both instance and region levels. We first introduce the instance-level contrastive loss and then present the region-level one. Assume that we have a set of encoded samples \(\{k_i | i = 1, 2, ..., K\}\) as keys of a dictionary. For each query feature \(q\), if there is a single key \((k^+)\) that matches the query \(q\), the contrastive loss aims to increase the similarity between \(q\) and \(k^+\) meanwhile reducing the similarity between \(q\) and all other keys (considered as the negative counterparts for \(q\)). We use L2-normalized dot product to measure the similarity between the queries and keys, and the contrastive loss, i.e., the InfoNCE [25] loss, is therefore formulated as:

\[
\mathcal{L}_{ins} = -\log \frac{\exp(q \cdot k^+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)},
\]

where \(\tau\) is a temperature hyper-parameter (set to 0.2 by default) [18, 34]. Following MoCo v2, the dictionary keys \(\{k_i | i = 1, 2, ..., K\}\) in RegionCL-M are maintained using a first-in-first-out queue with a predefined maximum number of samples \((K)\), which is set to 65,536 as in MoCo v2. The queue is progressively updated using the features of the key view \(k\) during self-supervised learning. This form of contrastive loss is the exact one that appeared in MoCo v2 [18], while it can also have other forms for different SSL methods [9, 25]. Apart from the instance-level contrastive loss, the features of other views build the region-level contrastive pairs with the following modified contrastive loss:

\[
\mathcal{L}_{reg} = -\frac{1}{2} \log \frac{\exp(p \cdot k_i^+ / \tau)}{\sum_{i=0}^{K} \exp(p \cdot k_i / \tau) + \exp(p \cdot sg(c) / \tau)} - \frac{1}{2} \log \frac{\exp(c \cdot k_i^+ / \tau)}{\sum_{i=0}^{K} \exp(c \cdot k_i / \tau) + \exp(c \cdot sg(p) / \tau)}.
\]

where the features \(p, c\) are obtained from the identical composite image \(x^p\) (thus the term is divided by \(\frac{2}{3}\) for normalization). \(sg(\cdot)\) represents ‘stop gradient’, which helps stabilize the training. \(p\) and \(c\) are indeed hard negative pairs since they involve some context information from each other due to convolution and pooling operations, thereby helping the model to learn robust and discriminative feature representations. Thus the total contrastive loss is formulated as:

\[
\mathcal{L}_{total} = \mathcal{L}_{ins} + \mathcal{L}_{reg}.
\]

Since the query view, canvas view, and paste view share the online network, we believe that the features of these views should be in the same feature space. Thus, RegionCL-M only needs a single queue to provide negative samples for features of all the three views, in contrast to the usage of multiple queues as in [37, 39].

### 3.3. Extension to other SSL methods

As RegionCL defines a model-agnostic pretext task and requires minor modifications to the SSL methods, we also choose two other representative approaches, i.e.,
DenseCL [33] and SimSiam [9], to validate its effectiveness, denoted as RegionCL-D and RegionCL-S, respectively. Specifically, DenseCL [33] focuses on dense prediction tasks and has two learning objectives, i.e., the instance-level contrastive loss as in MoCo v2 and the pixel-level dense loss. Therefore, RegionCL-D includes the proposed region-level loss seamlessly as in RegionCL-M and remains the pixel-level loss. For SimSiam [9], it only adopts instance-level positive pairs \( \{p, k^+\} \) for training. Thus, we only enrich the set of positive pairs by collecting those abundant instance- and region-level positive pairs provided by RegionCL-S for training while retaining the other components. Please refer to the supplementary for details.

4. Experiments

To thoroughly validate the improvements brought by introducing both regions into pretraining, we incorporate the RegionCL in representative state-of-the-art SSL methods, \textit{i.e.}, MoCo v2 [7], DenseCL [33], and SimSiam [9], and propose the RegionCL compatible models, \textit{i.e.}, RegionCL-M, RegionCL-D, and RegionCL-S. We evaluate their performance on popular datasets including ImageNet [13], MS COCO [8], and Cityscapes [12] for classification, object detection, and instance and semantic segmentation. The models are pretrained following the same settings as their own base methods, \textit{i.e.}, we train RegionCL-M and RegionCL-D for 200 epochs, and RegionCL-S for 100 epochs, with SGD [30] optimizer and corresponding augmentations, respectively. All the methods are based on ResNet-50 [20] backbone. Please refer to the supplementary material for more details. The code will be released.

4.1. Image classification on ImageNet

Settings. We benchmark the RegionCL for image classification on ImageNet, which contains 1.28M images in the training set and 50K images in the validation set from 1,000 classes, respectively. After pretraining the models with the proposed RegionCL, we freeze the backbone weights and initialize a new linear classification layer at the top of the models to evaluate the models’ transferability for the image classification task. The pretrained models of other SSL methods are either obtained from their authors or reproduced using their official codes. The performance of Top-1 and Top-5 accuracy on a single crop is reported. We have two experimental settings regards training:

- **Linear classification on ImageNet.** We follow the default setting in MoCo v2 [7, 18] and SimSiam [9] to finetune the linear layer in RegionCL-M/RegionCL-S, \textit{i.e.}, using SGD [30]/LARS [40] optimizer for 100/90 epochs, respectively. The finetuning of RegionCL-D is the same as the setting adopted in MoCo v2.

- **Linear few-shot finetuning.** We finetune the model using randomly sampled 1% and 10% data per class from the training set. All the models are finetuned for 50 epochs with an initial learning rate of 30, which then decreases by a factor of 10 at the 30th and 40th epoch.

| Methods     | 1% Data | 10% Data |
|-------------|---------|----------|
|             | Top-1   | Top-5    | Top-1 | Top-5 |
| MoCo v2 [7] | 43.6    | 70.9     | 58.8  | 82.4  |
| RegionCL-M  | 46.1    | 72.9     | 60.4  | 83.5  |
| RegionCL-D  | 38.9    | 66.2     | 54.0  | 79.3  |
| SimSiam [9] | 47.8    | 74.0     | 60.4  | 83.1  |
| RegionCL-S  | 32.8    | 61.5     | 51.8  | 77.7  |

4.2. Linear classification. Table 1 presents the results of different methods at the linear few-shot finetuning setting. Thanks to the abundant contrastive pairs brought by RegionCL, the models pretrained by RegionCL have learned better feature representations from a complete

\begin{table}[h]
\centering
\begin{tabular}{lcccr}
\hline
Methods & 1% Data & 10% Data \\
\hline
MoCo v2 [7] & 43.6 & 70.9 & 58.8 & 82.4 \\
RegionCL-M & 46.1 & 72.9 & 60.4 & 83.5 \\
RegionCL-D & 38.9 & 66.2 & 54.0 & 79.3 \\
SimSiam [9] & 47.8 & 74.0 & 60.4 & 83.1 \\
RegionCL-S & 32.8 & 61.5 & 51.8 & 77.7 \\
\end{tabular}
\caption{Linear classification results on ImageNet [13].}
\end{table}

Results with linear classification. We report the linear classification results of different methods in Table 1. ‘Real’ indicates that the labels used for evaluation are provided by [1]. From the table, we can see that RegionCL improves the aforementioned SSL baseline methods significantly by a large margin: +1.9% for RegionCL-M, +4.8% for RegionCL-D, and +3.2% for RegionCL-S. This proves RegionCL is compatible with various SSL methods and helps them learn better feature representations owing to the abundant contrastive supervisory signals. Besides, RegionCL-S reaches the best at 71.3% Top-1 accuracy using only 100 epochs, while the vanilla SimSiam requires a significantly longer training schedule of 800 epochs, proving the effectiveness of RegionCL in accelerating the model convergence and improving the performance. It is noteworthy that DenseCL focuses on dense prediction tasks and does not perform that well on classification. In contrast, RegionCL brings a large improvement on DenseCL for image classification, indicating that RegionCL is not only compatible with classification favored SSL methods but also generalizes well on dense prediction favored approaches.

Table 2. Linear classification on ImageNet [13] 1% and 10% data with MoCo v2 [7], DenseCL [33], SimSiam [9], and RegionCL.

| Methods     | 1% Data | 10% Data |
|-------------|---------|----------|
|             | Top-1   | Top-5    | Top-1 | Top-5 |
| MoCo v2 [7] | 43.6    | 70.9     | 58.8  | 82.4  |
| RegionCL-M  | 46.1    | 72.9     | 60.4  | 83.5  |
| RegionCL-D  | 38.9    | 66.2     | 54.0  | 79.3  |
| SimSiam [9] | 47.8    | 74.0     | 60.4  | 83.1  |
| RegionCL-S  | 32.8    | 61.5     | 51.8  | 77.7  |

Results with linear few-shot finetuning.
perceptive and can generalize well on classification tasks, thus delivering much more significant improvements over their baselines when only a limited number of data are available for finetuning, i.e., a gain of +2.5%, +8.9%, +9.5% for 1% data and 1.8%, 6.4%, 8.1% for 10% data achieved by RegionCL-M, RegionCL-D, RegionCL-S, respectively.

4.2. Detection and segmentation on MS COCO

Settings. We show the detection performance of the models pretrained with the RegionCL pretext task. The experiments are conducted on the MS COCO dataset [8], which contains about 118K images with bounding boxes and instance segmentation annotations and covers 80 object categories in total. We choose two representative detectors: the two-stage detector Mask-RCNN [19] and the one-stage detector RetinaNet [22]. For Mask-RCNN, we choose two backbones, i.e., ResNet50-C4 and ResNet50-FPN, respectively. RetinaNet is trained with focal loss. We also use Mask-RCNN to evaluate the models’ transfer performance on instance segmentation, an important topic in dense prediction tasks. All the experiments are conducted under the 1x and 2x training schedules, i.e., 90K and 180K iterations respectively, following the same settings as in [37, 44].

Results of Mask-RCNN on MS COCO. Table 3 and Table 4 summarize the Mask-RCNN results on 1x and 2x schedules respectively, where the RegionCL variants are highlighted in bold. We can see that RegionCL has significantly improved all approaches with ResNet50-C4 and ResNet50-FPN backbones, confirming the generalization of the proposed RegionCL pretext task on various SSL methods. According to the tables, MoCo v2 serves as a strong baseline and outperforms the supervised counterparts on both object detection and instance segmentation. Nevertheless, incorporating RegionCL into MoCo v2 (RegionCL-M) can further improve the performance over the MoCo v2 baseline with both backbones. It is also noticeable that RegionCL-M has already surpassed the previous representative SSL methods, including DetCo [37] and DenseCL [33], which are specifically designed for the dense prediction tasks. More importantly, when incorporating RegionCL into DenseCL, RegionCL-D achieves the best scores for all metrics in both the 1x and 2x settings. It suggests that with the exploration of both cropped and left regions, RegionCL can still help the dense prediction favored methods to learn more discriminative features.

Results of RetinaNet on MS COCO. The results of RetinaNet on MS COCO using different SSL methods are presented in Table 5. From the table, we can see that the improvement brought by RegionCL still holds in all metrics and at all the training settings. Similarly, RegionCL-D achieves the best results at 38.8 AP and 40.6 AP for the two training schedules respectively, significantly surpassing the supervised baseline by 1.4 AP and 1.7 AP. It is also noted that the improvement in the more stringent metric APbb 75 is more significant than that in the APbb 50 metric, demonstrating that leveraging both cropped and left regions for contrastive learning contributes to learning better feature representations for object detection and thus improving the detection accuracy. These results show that RegionCL helps existing SSL methods to achieve a better trade-off between the classification and detection tasks (see Figure 1).

4.3. Segmentation on Cityscapes

Settings. Further, we evaluate the models’ transfer performance for both instance and semantic segmentation on the Cityscapes [12] dataset, which contains over 5K well-annotated images of street scenes from 50 different cities. We follow the same setting as in MoCo v2 [18] for instance segmentation, i.e., using Mask-RCNN as the segmentation framework and training the models for 24K iterations. For semantic segmentation, we utilize UPerNets [36] with the self-supervised pretrained models are further trained for 40K and 80K iterations, respectively.

Results on Cityscapes. Table 6 presents the performance of different SSL methods and their variants with RegionCL. The second and third columns show the performance for instance and semantic segmentation, respectively. According to the table, RegionCL consistently improves the three representative SSL methods by large margins. For example, RegionCL-M reaches the best on instance segmentation at 34.9 AP and 62.5 AP 75, while RegionCL-D outperforms the others on semantic segmentation at 78.7 mIoU and 79.5 mIoU with different training schedules.

4.4. Ablation Study

We conduct the ablation studies with the proposed RegionCL-M. All models are trained for 100 epochs. We adopt a k-NN classifier to evaluate their classification performance on ImageNet [13] and train these models for 12K iterations on MS COCO [8] to evaluate their object detection and instance segmentation performance.

The size of the paste view. We investigate the influence of the size of the paste view by varying the lower and upper bounds $C_L$ and $C_U$. Note that the image size is set as 224 during the training and the downsampling ratio for the backbone network ResNet-50 [20] is 32, thus $C_L$, $C_U$ are valid in the range [1, 7]. The optimal hyper-parameters are determined through two steps. (1) We first fix the upper bound $C_U$ to 5 and search different configurations for the lower bound $C_L$. As shown in Table 7, the performance on both classification and detection peaks with $C_L = 3$. (2) Then we fix $C_U$ to 3 and search for $C_L$. It is interesting to see that decreasing $C_U$ from 5 to 4 slightly improves the performance on classification but degrades that on detection. This suggests that the optimal configurations of $C_L$, $C_U$ for classifi-
Table 3. Object detection results on the MS COCO [8] dataset with Mask-RCNN [19] C4 and FPN (1x).

| Method          | AP_{bb} | AP_{50} | AP_{75} | AP_{bb} | AP_{50} | AP_{75} | AP_{bb} | AP_{50} | AP_{75} |
|-----------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Rand Init       | 38.4    | 44.2    | 4.9     | 38.6    | 43.8    | 36.8    | 38.7    | 42.1    | 37.0    |
| Supervised      | 35.6    | 37.9    | 35.2    | 35.8    | 37.6    | 35.6    | 35.6    | 37.6    | 35.6    |
| InsDis [34]     | 38.5    | 43.3    | 36.4    | 38.7    | 43.7    | 36.7    | 38.7    | 43.7    | 36.7    |
| PIRL [23]       | 38.5    | 43.5    | 36.4    | 38.7    | 43.8    | 36.7    | 38.7    | 43.8    | 36.7    |
| SwAV [13]       | 38.6    | 43.6    | 36.5    | 38.8    | 43.9    | 36.7    | 38.8    | 43.9    | 36.7    |
| MoCo v2 [7]     | 38.9    | 43.8    | 36.6    | 38.8    | 43.9    | 36.7    | 38.8    | 43.9    | 36.7    |
| DetCo [37]      | 39.0    | 44.0    | 36.7    | 38.9    | 44.0    | 36.8    | 38.9    | 44.0    | 36.8    |
| DetCo-AA [37]   | 39.0    | 44.1    | 36.7    | 38.9    | 44.1    | 36.8    | 38.9    | 44.1    | 36.8    |

Table 4. Object detection results on the MS COCO [8] dataset with Mask-RCNN [19] C4 and FPN (2x).

| Method          | AP_{bb} | AP_{50} | AP_{75} | AP_{bb} | AP_{50} | AP_{75} | AP_{bb} | AP_{50} | AP_{75} |
|-----------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Rand Init       | 38.4    | 44.2    | 4.9     | 38.6    | 43.8    | 36.8    | 38.7    | 42.1    | 37.0    |
| Supervised      | 35.6    | 37.9    | 35.2    | 35.8    | 37.6    | 35.6    | 35.6    | 37.6    | 35.6    |
| InsDis [34]     | 38.5    | 43.3    | 36.4    | 38.7    | 43.7    | 36.7    | 38.7    | 43.7    | 36.7    |
| PIRL [23]       | 38.5    | 43.5    | 36.4    | 38.7    | 43.8    | 36.7    | 38.7    | 43.8    | 36.7    |
| SwAV [13]       | 38.6    | 43.6    | 36.5    | 38.8    | 43.9    | 36.7    | 38.8    | 43.9    | 36.7    |
| MoCo v2 [7]     | 38.9    | 43.8    | 36.6    | 38.8    | 43.9    | 36.7    | 38.8    | 43.9    | 36.7    |
| DetCo [37]      | 39.0    | 44.0    | 36.7    | 38.9    | 44.0    | 36.8    | 38.9    | 44.0    | 36.8    |
| DetCo-AA [37]   | 39.0    | 44.1    | 36.7    | 38.9    | 44.1    | 36.8    | 38.9    | 44.1    | 36.8    |

Table 5. Detection results on MS COCO [8] with RetinaNet [22].

| Method          | AP_{bb} | AP_{50} | AP_{75} | AP_{bb} | AP_{50} | AP_{75} | AP_{bb} | AP_{50} | AP_{75} |
|-----------------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| Rand Init       | 38.4    | 44.2    | 4.9     | 38.6    | 43.8    | 36.8    | 38.7    | 42.1    | 37.0    |
| Supervised      | 35.6    | 37.9    | 35.2    | 35.8    | 37.6    | 35.6    | 35.6    | 37.6    | 35.6    |
| InsDis [34]     | 38.5    | 43.3    | 36.4    | 38.7    | 43.7    | 36.7    | 38.7    | 43.7    | 36.7    |
| PIRL [23]       | 38.5    | 43.5    | 36.4    | 38.7    | 43.8    | 36.7    | 38.7    | 43.8    | 36.7    |
| SwAV [13]       | 38.6    | 43.6    | 36.5    | 38.8    | 43.9    | 36.7    | 38.8    | 43.9    | 36.7    |
| MoCo v2 [7]     | 38.9    | 43.8    | 36.6    | 38.8    | 43.9    | 36.7    | 38.8    | 43.9    | 36.7    |
| DetCo [37]      | 39.0    | 44.0    | 36.7    | 38.9    | 44.0    | 36.8    | 38.9    | 44.0    | 36.8    |
| DetCo-AA [37]   | 39.0    | 44.1    | 36.7    | 38.9    | 44.1    | 36.8    | 38.9    | 44.1    | 36.8    |

Table 6. Semantic segmentation results on Cityscapes [12].

| Method          | Instance Seg AP | Semantic Seg 40K (mIoU) | Semantic Seg 80K (mIoU) |
|-----------------|------------------|-------------------------|------------------------|
| Supervised      | 32.9             | 77.1                    | 78.2                   |
| MoCo v2 [7]     | 33.9             | 69.6                    | 77.8                   |
| RegionCL-M      | 34.3             | 62.5                    | 78.1                   |
| DenseCL [33]    | 34.5             | 62.3                    | 78.7                   |
| RegionCL-D      | 34.8             | 62.7                    | 79.5                   |
| SimSiam [9]     | 33.6             | 61.0                    | 76.2                   |
| RegionCL-S      | 34.9             | 61.6                    | 77.8                   |

Table 7. The influence of the size of the paste view.

| Configuration | ImageNet [14] | MS COCO [8] |
|---------------|---------------|-------------|
| C_L, C_U     | AP_{bb}       | AP_{50}     |
| 1 5           | 51.8          | 49.8        |
| 2 5           | 53.0          | 51.2        |
| 3 5           | 54.7          | 52.7        |
| 4 5           | 53.9          | 51.8        |
| 3 4           | 55.1          | 52.8        |
| 3 6           | 54.3          | 52.5        |
| 20-NN         | 26.3          | 24.0        |
| 100-NN        | 27.3          | 24.9        |
| 20-NN         | 27.9          | 25.5        |
| 100-NN        | 28.0          | 25.6        |
| 20-NN         | 27.9          | 25.5        |
| 100-NN        | 27.8          | 25.5        |

The influence of using paste and canvas views. We further investigate the importance of using both paste and canvas views during pretraining. The results are concluded in Table 8, where \( \checkmark \) under Paste or Canvas denotes whether to use the former or latter term in Eq. (2). The ‘Negative’ option means whether to treat the canvas and paste counterpart views from the same composite image \( x^{mc} \) as negative pairs, \( \text{i.e., } \exp(c \cdot sg(p)/r) \) or \( \exp(p \cdot sg(c)/r) \) in the denominator in Eq. (2). With all columns marked \( \times \), the method be-
Table 8. The influence of using paste and canvas views. ‘Paste’ and ‘Canvas’ denote using paste and canvas views to construct positive pairs. ‘Negative’ means using the canvas and paste counterpart views in the composited images as negative pairs.

| Configuration | ImageNet [13] | MS COCO [8] |
|---------------|---------------|-------------|
|               | 20-NN | 100-NN | AP\^bb | AP\^fork |
| Paste Canvas Negative | | | | |
| × × | 49.3 | 47.3 | 26.3 | 24.0 |
| ✓ × × | 51.8 | 50.0 | 26.9 | 24.7 |
| ✓ × × | 50.0 | 50.2 | 27.0 | 24.7 |
| ✓ ✓ × | 54.6 | 52.4 | 27.8 | 25.5 |
| ✓ ✓ ✓ | 54.7 | 52.7 | 28.0 | 25.6 |

comes standard MoCo v2. From the first three rows in the table, we can see that using either the paste or canvas views can bring performance gains, and the performance will be further boosted when considering both regions. For example, in the 4th row, the model gains more than 5% accuracy improvement over MoCo v2 in both ImageNet 20-NN and 100-NN classification. We attribute it to that incorporating both regions during pretraining can help the model learn better category feature representations from a complete perceptive. Comparing the 4th row with the last row, where intra-image negative pairs are utilized, the performance on both tasks is slightly improved. It demonstrates that the usage of negative pairs within images can help the model learn more discriminative features between different regions, again validating the importance of introducing region-level contrastive pairs in self-supervised learning.

4.5. Visual inspection

To further analyze the performance gains brought by the proposed RegionCL, we randomly select 10 categories from the ImageNet [13] dataset and extract the features from the models pretrained by MoCo v2 [7], DenseCL [33], SimSiam [9], and their RegionCL counterparts. Then we visualize the features using t-SNE [32] as in Figure 4. With the complementary cropped and left regions used in contrastive learning, the features extracted from RegionCL models are better clustered, demonstrating that models can learn better and more discriminative features with abundant contrastive pairs at both instance and region levels.

![Figure 4](image1.png)  
(a) MoCo v2 [7] and RegionCL-M, (b) DenseCL [33] and RegionCL-D, and (c) SimSiam [9] and RegionCL-S. Best viewed in color.

![Figure 5](image2.png)  
(a) MoCo v2 [7] (b) and RegionCL-M (c) on the input images (a).

Besides, we apply Grad-CAM [29] on the features from the last layer of the pretrained backbone models to qualitatively inspect them. The visualization results are provided in Figure 5, which further demonstrate that with the help of both regions, the pretrained models can learn better features to discriminate the target objects in the images.

5. Limitation and Discussion

By proposing a simple yet effective RegionCL pretext task, we make the first attempt to demonstrate the importance of considering both cropped and left regions in SSL. With minor modifications to representative SSL methods, RegionCL consistently improves them on various tasks, including classification, detection, and instance and semantic segmentation. However, there is still much to be explored in utilizing the complementary regions to help the model learn better feature representations, e.g., introducing additional bounding box selection and alignment modules or adopting multi-level supervision during pretraining. Besides, although the proposed RegionCL aims to improve the performance of self-supervised pretrained models on both classification and dense prediction tasks, it is also worthy of further research efforts on how to design specific task-oriented pretext tasks to utilize the complementary regions.

6. Conclusion

This paper demonstrates the importance of using both cropped and left regions after cropping for self-supervised learning. A simple yet effective pretext task RegionCL is proposed to help the models learn better category feature representation from a complete perceptive. Experimental
results on image classification, object detection, and instance and semantic segmentation benchmarks demonstrate the effectiveness of the proposed RegionCL and its compatibility to representative self-supervised learning methods. We hope that this study will provide valuable insights into the subsequent studies of self-supervised learning in exploring region-based contrast methodology.

**Broad impacts.** RegionCL successfully improves the performance of SSL methods, but it still requires large scales of unlabeled data during training as others. The data collection process, which may compromise the privacy or rights of individuals, could be regularized by following specific rules, e.g., GDPR, to mitigate the compromise issues.

### A. Appendix

#### A.1. Results for more training epochs

Table S1. Results of MoCo v2 [7] and RegionCL-M trained for 200, 400, and 800 epochs. *" denotes that we end-to-end finetune RegionCL-M pretrained models for 50 epochs [2, 17].

| Models       | ImageNet Top-1 | MS COCO AP\textsuperscript{\textsubscript{bbox}} | MS COCO AP\textsuperscript{\textsubscript{mask}} |
|--------------|----------------|-----------------------------------------------|-----------------------------|
| MoCo v2      | 67.5 72.0 74.1 | 40.9 43.2 43.9                                 | 37.0 38.7 39.4              |
| RegionCL-M   | 69.4 75.2 76.2 | 41.6 43.8 44.5                                 | 37.7 39.3 39.7              |
| RegionCL-M*  | 76.8 79.7 80.4 | - - -                                         | - - -                       |

We also investigate the influence of different training epochs by extending the epochs to 200, 400, and 800 respectively. We train MoCo v2 [7] and the corresponding RegionCL-M respectively and present their results in Table S1. We evaluate these models’ image classification performance on the ImageNet [13] dataset with linear probing. Their object detection and instance segmentation performance are also evaluated on the MS COCO [8] dataset with ResNet50-FPN and Mask-RCNN [19]. The detection and segmentation models are trained following the $2 \times$ scheduler, i.e., the models are trained for 180K iterations in total.

As can be seen, with only 200 epochs for training, the proposed RegionCL-M obtains competitive results compared with MoCo v2 trained for 400 epochs, no matter on classification or dense prediction tasks. The performance of RegionCL-M increases with the total training epochs increasing, and RegionCL-M with 400 epochs has significantly outperformed MoCo v2 trained with 800 epochs, confirming the good property of convergence brought by RegionCL. Besides, RegionCL-M sees an further improvement especially for classification (by 1% accuracy) when extending to 800 training epochs, reaching 73.1% Top-1 accuracy for classification, 42.1 AP for object detection and 38.2 AP for instance segmentation. Such observation demonstrates that the abundant contrastive pairs with both cropped and left regions can not only improves the model's convergence but also effectively enhance the model's representation capacity with more training epochs.

#### A.2. Results for models with variant sizes

To investigate the effectiveness of the proposed RegionCL method on models with variant sizes, we adopt ResNet50 [20], ResNet50-w2 (2×parameters), and ResNet50-w4 (4×parameters) as backbone networks and train them for 200 epochs with MoCo v2 and RegionCL-M, respectively. The results of linear probing on ImageNet along with object detection and instance segmentation on MS COCO are reported in Table S2. We use Mask-RCNN with ResNet50-FPN as the object detection and instance segmentation framework and train them for 180K iterations, following the $2 \times$ scheduler. It can be observed that RegionCL-M with ResNet50-w2 outperforms MoCo v2 with ResNet50-w4 on image classification and obtains competitive performance on both object detection and instance segmentation tasks on MS COCO. RegionCL-M with ResNet50-w4 obtains the best performance on all tasks. It indicates that the proposed RegionCL method is scalable to large models and can improve their performance on both classification and dense prediction tasks, further validating the importance of using both cropped and left regions in self-supervised learning.

#### A.3. Influence of the region swapping operation

Table S3. Influence of the region swapping strategy.

| Configuration | ImageNet 20-NN | ImageNet 100-NN | MS COCO AP\textsuperscript{\textsubscript{bbox}} | MS COCO AP\textsuperscript{\textsubscript{mask}} |
|---------------|----------------|-----------------|-----------------------------|-----------------------------|
| w/o swapping  | 52.2 50.2      | 27.1 25.0       | 28.0 25.6                   |
| w/ swapping   | 54.7 52.7      | 28.0 25.6       |                             |

To investigate the performance gains brought by the region swapping operation, we adopt a simple RegionCL variant without the swapping operation, i.e., simply cropping a region from candidate images and filling zeros into the cropped regions. Using RegionCL-M as the base, we train RegionCL-M and its variant without the region swapping operation for 100 epochs, and evaluate their performance on the ImageNet [13] dataset with $k$-NN classifier and on
the MS COCO [8] dataset with ResNet50-C4 and Mask-RCNN [19] trained for 12K iterations. The results are available in Table S3. Without the region swapping operations, the proposed RegionCL still improves the model’s performance on both classification and dense prediction tasks, while the region swapping operation can further boost the performance on both tasks by a large margin, i.e., 2.5% accuracy for 20-NN classification, 0.9 AP for object detection, and 0.6 AP for instance segmentation. It indicates that although taking both regions into consideration can facilitate the models learning, composing the hard negative samples, i.e., the paste and canvas views since their features share some context from each other during network forward calculation, in the same images can further help the model learn better and discriminative feature representations from both instance- and region-level pairs.

A.4. Influence of different batch size

Table S4. The influence of batch size of MoCo v2 and RegionCL.

|                | Batch Size | LR  | ImageNet Top-1 | ImageNet Top-5 | MS COCO AP\(_{10k}\) | MS COCO AP\(_{1k}\) |
|----------------|------------|-----|----------------|----------------|------------------------|----------------------|
| MoCo v2        | 256        | 0.03| 67.5           | -              | 40.9                   | 37.0                 |
| MoCo v2        | 1024       | 0.15| 67.5           | 82.2           | 41.0                   | 37.2                 |
| RegionCL-M     | 256        | 0.03| 70.0           | 90.0           | 41.6                   | 37.8                 |
| RegionCL-M     | 1024       | 0.15| 69.4           | 89.6           | 41.6                   | 37.7                 |

As the number of negative pairs plays an important role in the InfoNCE [25] loss and affects the pretrained model’s performance as pointed in [5], MoCo v2 maintains a huge memory queue to provide enough negative samples, which makes the calculation of the InfoNCE loss and the training process not coupled with the batch size. Thus we accelerate the training of MoCo v2 [7] and RegionCL-M by increasing the batch size from 256 to 1024. We train the models for 200 epochs with an initial learning rate 0.15 (around linear growth w.r.t. the batch size) and a cosine learning rate scheduler, while the origin training setting is 200 epochs with an initial learning rate 0.03 and a cosine learning rate scheduler. The other settings are exactly the same, including the data augmentation strategies, optimizers, and the values of hyper-parameters. We validate the performance difference of the two training settings and present the results in Table S4. It can be observed that MoCo v2’s performance are consistent for both classification and dense prediction tasks with both training settings, confirming the rationality of the batch size to be 1,024 for MoCo v2. Thus, we choose such batch size for MoCo v2-based models in our paper.

We also conduct similar experiments for RegionCL-M as shown in the last two rows in Tab. S4. Similar conclusion can be observed in the evaluation of RegionCL-M with different batch size for training. Besides, as we adopt the batch-wise implementation for region swapping, RegionCL-M with small batch size and thus more iterations can see more diverse paste views and canvas views in terms of different sizes and locations, thus learning better feature representations. As a result, RegionCL-M with a batch size of 256 obtains slightly better performance for image classification by 0.6% Top-1 accuracy. Nevertheless, we choose the batch size of 1024 in this paper for acceleration purpose.

A.5. Results on pose estimation

Besides the evaluation on detection and segmentation tasks, we also evaluate the models’ performance on both human pose estimation and animal pose estimation tasks on MS COCO [8] and AP-10K [41] datasets. We adopt SimpleBaseline [35] as the base pose estimation framework and utilizes backbone models pretrained by MoCo v2 [7], DenseCL [33], SimSiam [9], and their RegionCL compatible counterparts. We train these models for 210 epochs. We adopt an Adam optimizer with initial learning rate at 1e-4, which decreases by a factor of 10 at the 170 and 200 epochs respectively, following the same setting as in mm-pose [11]. The results are available in Table S5. It can be observed that the SSL pretrained models outperforms the supervised counterpart. Besides, with both cropped and left regions taken into consideration, RegionCL improves the pretrained models’ transfer performance on both pose estimation tasks, especially on the smaller animal pose dataset AP-10k. Such observation further demonstrates that exploiting supervisory signals from both instance and region levels can help the model obtain a better trade-off on both classification and dense prediction tasks.

A.6. Architecture details

We present the details of the proposed RegionCL-M (MoCo v2), RegionCL-D (DenseCL), and RegionCL-S (SimSiam) in this section. We also provide the pseudo codes for RegionCL-M, RegionCL-D, and RegionCL-S as in Algorithm 1, 2, and 3, respectively, with red color denoting the modifications of RegionCL compared with the base architecture.

RegionCL-D. As shown in Algorithm 2 and Figure S2, DenseCL [33] adopts both instance-level and pixel-level losses during pretraining. The modifications from DenseCL
Figure S1. Illustration of the proposed RegionCL with the MoCo v2 framework, i.e., RegionCL-M. Taking the two augmented views $x^q, x^k$ as inputs, RegionCL employs region swapping among the batch of $x^j$ to generate the composite images with paste views $x^p$ and canvas views $x^c$. Then, for the composite images, mask pooling is used to extract the features belonging to the paste and canvas views, respectively. The pooled region-level features (with stripes in the figure) are batched with the instance-level features and processed by the projector. The projected features $q, p, c, k$, and features from the memory queue form both instance- and region-level contrastive pairs.

Figure S2. Illustration of the proposed RegionCL with the DenseCL framework, i.e., RegionCL-D. Taken the instance-level views $x^q$ and $x^k$, and the region-level views $x^c$ and $x^p$ as inputs, RegionCL-D firstly extract the instance- and region-level features using the same way as RegionCL-M. The extracted and projected features $q, p, c, k$ are constructed the contrastive pairs. The instance-level views $x^q$ and $x^k$ are also used to enhance dense feature correspondences before the average pooling operation, with another projector for pixel-wise feature projection and memory queue.

Figure S3. Illustration of the proposed RegionCL with the SimSiam framework, i.e., RegionCL-S. Taking the two augmented views as inputs, RegionCL-S extracts the region-level features in the same way as RegionCL-M. The pooled region-level features are then batched with the instance-level features and processed by the projector. Following SimSiam, the projected features $q, p, c, k$ build positive pairs.
Algorithm 1: Example code of RegionCL-M.

**Input:** Two augmented views $x^a, x^b$
**Input:** The negative Queue $Q$
**Output:** The contrastive loss $L$

```plaintext
/* Feature extraction */
// Online Branch
1 $x^a_\text{reg} = \text{RegionSwapping}(x^a)$
2 $q = \text{Projector}(\text{AvgPool}(\text{Encoder}(x^a)))$
3 $p,c = \text{Projector}(\text{MaskPool}(\text{Encoder}(x^b)))$

// Momentum Branch
4 $k = \text{Projector}_M(\text{AvgPool}(\text{Encoder}_M(x^b)))$
/* Loss computation */
// Eq. 1
5 $L_{\text{ins}} = \text{Loss}(q,k|Q)$
// Eq. 2
6 $L_{\text{dense}} = \text{DenseLoss}(q_d,k_d|Q_{\text{dense}})$
// Eq. 2
7 $L_{\text{reg}} = (\text{Loss}(p,k|Q,sg(c)) + \text{Loss}(c,k|Q,sg(p)))/2$
8 $L = L_{\text{ins}} + L_{\text{reg}}$
```

Algorithm 2: Example code of RegionCL-D.

**Input:** Two augmented views $x^a, x^b$
**Input:** The instance negative Queue $Q_{\text{ins}}$
**Input:** The dense negative Queue $Q_{\text{dense}}$
**Output:** The contrastive loss $L$

```plaintext
/* Feature extraction */
// Online Branch
1 $x^a_\text{reg} = \text{RegionSwapping}(x^a)$
2 $q = \text{Projector}(\text{AvgPool}(\text{Encoder}(x^a)))$
3 $p,c = \text{Projector}(\text{MaskPool}(\text{Encoder}(x^b)))$

// Extract dense feature
4 $q_d = \text{Projector}_d(\text{Encoder}(x^b))$
// Momentum Branch
5 $k = \text{Projector}_M(\text{AvgPool}(\text{Encoder}_M(x^b)))$
/* Loss computation */
// Eq. 1
6 $L_{\text{ins}} = \text{Loss}(q,k|Q_{\text{ins}})$
// Eq. 2
7 $L_{\text{dense}} = \text{DenseLoss}(q_d,k_d|Q_{\text{dense}})$
// Eq. 2
8 $L_{\text{reg}} = (\text{Loss}(p,k|Q,sg(c)) + \text{Loss}(c,k|Q,sg(p)))/2$
9 $L = L_{\text{ins}} + L_{\text{reg}} + L_{\text{dense}}$
```

Algorithm 3: Example code of RegionCL-S.

**Input:** Two augmented views $x^a, x^b$
**Output:** The contrastive loss $L$

```plaintext
/* Feature extraction */
// 1st Branch
1 $x^a_\text{reg} = \text{RegionSwapping}(x^a)$
2 $q = \text{Projector}(\text{AvgPool}(\text{Encoder}(x^a)))$
3 $p,c = \text{Projector}(\text{MaskPool}(\text{Encoder}(x^b)))$

// 2nd Branch, weight sharing with the 1st Branch
4 $k = \text{Projector}(\text{AvgPool}(\text{Encoder}(x^b)))$
/* Loss computation */
// Eq. 1
5 $L_{\text{ins}} = \text{Loss}(q,k|Q)$
// Eq. 2
6 $L_{\text{reg}} = (\text{Loss}(p,q,sg(c)) + \text{Loss}(c,q,sg(p)))/2$
7 $L = L_{\text{ins}} + L_{\text{reg}}$
```

to RegionCL-D appears at the instance-level branch in a same way as the modifications from MoCo v2 [7] to RegionCL-M, \textit{i.e.,} we use the encoder, mask pooling, and the instance-level projector to extract the features belonging to the canvas and paste views, separately. Then, the contrastive pairs are also enriched by the regions while the dense correspondences related loss functions are remained the same as in DenseCL.

**RegionCL-S.** As shown in Algorithm 3 and Figure S3, SimSiam [9] does not require the negative pairs during pretraining and focuses on attracting the features among positive pairs. The modifications from SimSiam [9] to RegionCL-S are simply providing abundant positive pairs from both instance and region levels. Specifically, given the augmented views $x^a, x^b$ and the composite images with the canvas view $x^c$ and the paste view $x^p$ as inputs, RegionCL-S adopts an encoder, average pooling (mask pooling) layer, a projector, and a predictor to get the instance-level (region-level) features $q$ and $p$ and $c$. The other view $x^k$ is processed by the weight-shared encoder and projector to get the feature $k$, where an stop gradient operation is applied on $k$ to stabilize the training. Thus, there are three cases of positive pairs in the modified RegionCL-S, \textit{i.e.,} the instance-level pairs $q$ and $k$ as in origin SimSiam, the region-level pairs $c$ and $k$, and $p$ and $k_p$, and we keep the learning objectives and architecture the same as in SimSiam.

**A.7. Implementation details**

In this section we give the implementation details of all the three RegionCL models, \textit{i.e.,} RegionCL-M (MoCo v2 [7]), RegionCL-D (DenseCL [33]), and RegionCL-S (SimSiam [9]), respectively. We conduct all the experiments on NVIDIA A100 40G GPUs.

**A.7.1 RegionCL-M and RegionCL-D**

- **Training settings.** To accelerate the training, we train the RegionCL-M and RegionCL-D with a total batch size of 1,024 and initial learning rate 0.15, which is slightly different from the original setting but does not affect the performance and our conclusion as shown in Appendix A.4. The other settings are the same as in MoCo v2 [7] and DenseCL [33]. For example, the input images are first randomly cropped and resized to 224 x 224 by remaining 20% \textrightarrow 100% regions, which is followed by color jitter with a probability of 0.8, grayscale with a probability of 0.2, Gaussian blur with
a probability of 0.5, and random horizontal flips, which are the same as the origin settings in the two base methods. The SGD [30] optimizer is adopted with weight decay at 1e-4 and momentum at 0.9.

**Architecture settings.** There is no difference in the model’s architectures comparing the base models and RegionCL models. The instance-level and region-level features share the same projector, which has the structure of \( \text{Linear}(2048, 2048) \rightarrow \text{ReLU} \rightarrow \text{Linear}(2048, 128) \), where 2048 is the hidden dimension and 128 is the output dimension. The dense correspondences projectors in DenseCL and RegionCL-D have the structure of \( \text{Conv2d}(2048, 2048) \rightarrow \text{ReLU} \rightarrow \text{Conv2d}(2048, 128) \), where the kernel size for the convolutions is 1\(\times\)1, and 128 is the output dimension. The length of the memory queue is 65,536.

**A.7.2 RegionCL-S**

- **Training settings.** As there is no component like memory queues in SimSiam’s architecture, we follow exactly the same training settings as in origin SimSiam to train the RegionCL model RegionCL-S. Specifically, a total batch size of 512 is employed during the pretraining process. The SGD optimizer with learning rate 0.1, weight decay 1e-4, and momentum 0.9 is adopted to train the model. The data augmentation strategy is the same as the in MoCo v2 [7] and DenseCL [33], i.e., the input images are first randomly cropped and resized to 224 \(\times\) 224 by remaining 20\% \(\rightarrow\) 100\% regions, which is followed by color jitter with a probability of 0.8, grayscale with a probability of 0.2, Gaussian blur with a probability of 0.5, and random horizontal flips. The models are trained for 100 epochs with cosine learning rate scheduler.

- **Architecture settings.** There is no modification on the model’s architectures settings comparing the SimSiam and RegionCL-S. The projection head follows average pooling or mask pooling and has 3 layers to project the features, which takes the form of \( \text{Linear}(2048, 2048) \rightarrow \text{BN}(2048) \rightarrow \text{ReLU} \rightarrow \text{Linear}(2048, 2048) \rightarrow \text{BN}(2048) \rightarrow \text{ReLU} \rightarrow \text{Linear}(2048, 2048) \rightarrow \text{BN}(2048) \). The predictor head has 2 layers to further process the features and align them with the features extracted from the key views. The structure of the predictor head is \( \text{Linear}(2048, 512) \rightarrow \text{BN}(512) \rightarrow \text{ReLU} \rightarrow \text{Linear}(512, 2048) \). These structures are the same as the original settings in SimSiam.
References

[1] Lucas Beyer, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, and Aäron van den Oord. Are we done with imagenet? arXiv preprint arXiv:2006.07159, 2020.

[2] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charles Fowlkes, Zhouwen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 194–203, 2021.

[3] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. arXiv preprint arXiv:2006.09882, 2020.

[4] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. arXiv preprint arXiv:2104.14294, 2021.

[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, volume 1, pages 1597–1607. PMLR, 2020.

[6] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. Advances in Neural Information Processing Systems, 33:22243–22255, 2020.

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

[8] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325, 2015.

[9] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 15750–15758, 2021.

[10] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmseg, 2020.

[11] MMPose Contributors. Openmmlab pose estimation toolbox and benchmark. https://github.com/open-mmlab/mmpose, 2020.

[12] 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 Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 248–255. Ieee, 2009.

[14] 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.

[15] 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.

[16] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 1735–1742. IEEE, 2006.

[17] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377, 2021.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 9729–9738, 2020.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 2961–2969, 2017.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[21] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In Proceedings of the European Conference on Computer Vision (ECCV), pages 577–593. Springer, 2016.

[22] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 2980–2988, 2017.

[23] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6707–6717, 2020.

[24] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In Proceedings of the European Conference on Computer Vision (ECCV), pages 69–84. Springer, 2016.

[25] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018.

[26] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2536–2544, 2016.
[27] Pedro O Pinheiro, Amjad Almahairi, Ryan Y Benmalek, Florian Golemo, and Aaron Courville. Unsupervised learning of dense visual representations. arXiv preprint arXiv:2011.05499, 2020.

[28] Byungseok Roh, Wuhyun Shin, Ildoo Kim, and Sungwoong Kim. Spatially consistent representation learning. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1144–1153, 2021. 1, 2, 3, 4

[29] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 618–626, 2017. 8

[30] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In International conference on machine learning, pages 1139–1147. PMLR, 2013. 5, 13

[31] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In Proceedings of the European Conference on Computer Vision (ECCV), pages 776–794. Springer, 2020. 2

[32] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(11), 2008. 8

[33] Xinlong Wang, Rufeng Zhang, Chunhua Shen, Tao Kong, and Lei Li. Dense contrastive learning for self-supervised visual pre-training. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3024–3033, 2021. 1, 2, 3, 5, 6, 7, 8, 10, 12, 13

[34] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 3733–3742, 2018. 4, 7

[35] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In Proceedings of the European conference on computer vision (ECCV), pages 466–481, 2018. 10

[36] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In Proceedings of the European Conference on Computer Vision (ECCV), pages 418–434, 2018. 6

[37] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. Detco: Unsupervised contrastive learning for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 8392–8401, 2021. 1, 3, 4, 6, 7

[38] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 16684–16693, 2021. 1, 2

[39] Ceyuan Yang, Zhirong Wu, Bolei Zhou, and Stephen Lin. Instance localization for self-supervised detection pretraining.