# RegionCL: Exploring Contrastive Region Pairs for Self-supervised Representation Learning Supplementary Material

Yufei Xu[1⋆], Qiming Zhang[1*], Jing Zhang[1], and Dacheng Tao[2,1]

[1] University of Sydney, Australia
[2] JD Explore Academy, China

## A Appendix

In the appendix, we demonstrate the benefits of leveraging both cropped and remained regions for training with longer training epochs (A.1) and larger models (A.2). We also demonstrate the training efficiency of RegionCL in section A.1. Pose estimation for both human and animals are also employed to evaluate the effectiveness of region-level contrastive learning on downstream tasks (A.3). Besides, we also independently feed the paste and canvas views into the networks for training to further validate the performance of regional contrastive pairs without swapping (A.4). The analysis of parameters, architecture details, and implementation details are provided in A.5, A.6, and A.7, respectively.

### A.1 Training efficiency with longer training epochs

| | ImageNet Top-1 | | | MS COCO $AP^{bb}$ | | | MS COCO $AP^{mk}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| Epochs | 200 | 400 | 800 | 200 | 400 | 800 | 200 | 400 | 800 |
| MoCo v2 | 67.5 | 69.6 | 71.1 | 40.9 | 41.3 | 41.5 | 37.0 | 37.5 | 37.6 |
| RegionCL-M | 69.4 | 72.1 | 73.1 | 41.6 | 41.9 | 42.1 | 37.7 | 38.0 | 38.2 |
| RegionCL-M* | 76.8 | 77.8 | 78.1 | - | - | - | - | - | - |

Table S1: Results of MoCo v2 [3] and RegionCL-M trained for 200, 400, and 800 epochs. '*' denotes that we end-to-end finetune RegionCL-M pretrained models for 50 epochs [8,1].

We investigate effectiveness of leveraging remained regions for longer pre-training by extending the epochs to 200, 400, and 800 respectively. We train MoCov2 [3] and the corresponding RegionCL-M respectively and present their results in Table S1. We evaluate these models' image classification performance on the ImageNet [7] dataset with linear probing. Their object detection and instance segmentation performance are also evaluated on the MS COCO [4]

⋆ Equal contribution.

dataset with ResNet50-FPN and Mask-RCNN [9]. The detection and segmentation models are trained following the 2× schedule, *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 MoCov2 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 can obtain better performance with less training cost, *i.e.*, RegionCL-M with 400 epochs pretraining has significantly outperformed MoCov2 trained with 800 epochs, confirming the good property of training efficiency brought by simply training with the regional contrastive pairs. Although RegionCL costs more forwards each iterations than the baseline method, it improves the overall training efficiency, *i.e.*, RegionCL with 400 epochs training (1200 forwards) beats the baseline methods with 800 epochs training (1600 forwards).

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 remained regions can not only improves the model's convergence but also effectively enhance the model's representation capacity with more training epochs.

### A.2 Generalization ability for models with variant sizes

| | ImageNet Top-1 | | | MS COCO $AP^{bb}$ | | | MS COCO $AP^{mk}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| Models | R50 | w2 | w4 | R50 | w2 | w4 | R50 | w2 | w4 |
| 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 | - | - | - | - | - | - |

Table S2: Results of MoCo v2 [3] and RegionCL-M with R50 [10], R50w2 [10], and R50w4 [10] as backbones. '*' denotes that we end-to-end finetune RegionCL-M pretrained models for 50 epochs [8,1].

To investigate the benefits of regional contrastive pairs on models with variant sizes, we adopt ResNet50 [10], ResNet50-w2 (2×parameters), and ResNet50-w4 (4×parameters) as backbone networks and train them for 200 epochs with MoCov2 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× schedule. It can be observed that RegionCL-M with ResNet50-w2 outperforms MoCov2 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 Results on pose estimation

| | MS COCO [4] AP | AP-10K [15] AP |
|---|---|---|
| Supervised | 71.8 | 69.9 |
| MoCo v2 [3] | 72.0 | 70.1 |
| RegionCL-M | 72.3 | 70.6 |
| DenseCL [13] | 72.4 | 71.1 |
| RegionCL-D | 72.6 | 72.1 |
| SimSiam [5] | 71.9 | 70.5 |
| RegionCL-S | 72.2 | 71.6 |

Table S3: Results of RegionCL compatible models on human (MS COCO [4]) and animal (AP-10K [15]) 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 [4] and AP-10K [15] datasets. We adopt SimpleBaseline [14] as the base pose estimation framework and utilizes backbone models pretrained by MoCov2 [3], DenseCL [13], SimSiam [5], 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 mmpose [6]. The results are available in Table S3. 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.4 Influence of the region swapping operation

To further understand the performance gains brought by the abundant contrastive pairs, we adopt a simple RegionCL variant without the swapping operation, *i.e.*, simply cropping a region from candidate images as paste views and filling zeros into the cropped regions to formulate the canvas views. Using

| Configuration | ImageNet Top-1 | MS COCO $AP^{bb}$ |
|---|---|---|
| MoCov2 | 67.5 | 40.9 |
| w/o swapping | 69.2 | 41.4 |
| w/ swapping | 69.4 | 41.6 |

Table S4: Influence of the region swapping strategy.

RegionCL-M as the base, we train RegionCL-M and its variant without the region swapping operation for 200 epochs, and evaluate their performance on the ImageNet [7] dataset and on the MS COCO [4] dataset with ResNet50-FPN and Mask-RCNN [9] following the 2× schedule. The results are available in Table S4. 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 improve the performance on both tasks, *e.g.*, 0.2% accuracy gains for ImageNet Top-1 accuracy and 0.2 mAP gains for object detection. It indicates that although taking both regions into consideration without swapping can facilitate the models learning, composing the hard negative samples, *i.e.*, the paste and canvas views via swapping, in the same images can further help the model learn better and discriminative feature representations from both instance- and region-level pairs, since features of these two kinds of views share some context from each other during network forward calculation.

### A.5 Influence of different batch size

| | | | ImageNet | | MS COCO | |
|---|---|---|---|---|---|---|
| | Batch Size | LR | Top-1 | Top-5 | $AP^{bb}$ | $AP^{mk}$ |
| MoCo v2 | 256 | 0.03 | 67.5 | - | 40.9 | 37.0 |
| MoCo v2 | 1024 | 0.15 | 67.5 | 88.2 | 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 |

Table S5: The influence of batch size of MoCo v2 and RegionCL.

As the number of negative pairs plays an important role in the InfoNCE [11] loss and affects the pretrained model's performance as pointed in [2], MoCov2 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 MoCov2 [3] and RegionCL-M by increasing the batch size from 256 to 1,024. 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

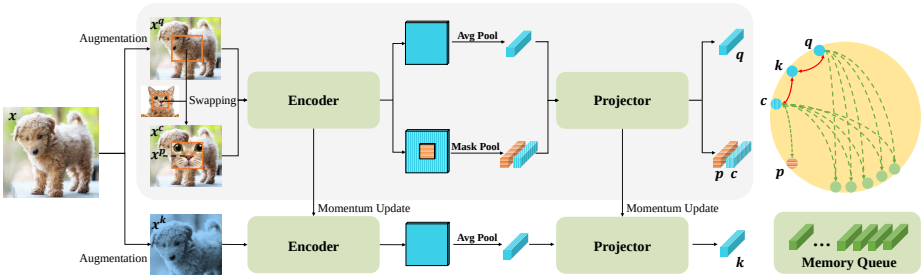


Fig. S1: Illustration of the proposed RegionCL with the MoCov2 framework, *i.e.*, **RegionCL-M**. Taking the two augmented views $x^q$, $x^k$ as inputs, RegionCL employs region swapping among the batch of $x^q$ to generate the composite images with paste views $x^p$ and canvas views $x^c$. Then, for 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.

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 S5. It can be observed that MoCov2'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 Table S5. 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.6 Architecture details

We present the details of the proposed RegionCL-M (MoCov2), 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 [13] adopts both instance-level and pixel-level losses during pretraining. The modifications

**Algorithm 1:** Example code of RegionCL-M.

**Input:** Two augmented views $x^q$, $x^k$
**Input:** The negative Queue $Q$
**Output:** The contrastive loss $L$
`/* Feature extraction */`
`// Online Branch`
$x_c^p = \text{RegionSwapping}(x^q)$
$q = \text{Projector}(\text{AvgPool}(\text{Encoder}(x^q)))$
$p,c = \text{Projector}(\text{MaskPool}(\text{Encoder}(x_c^p)))$
`// Momentum Branch`
$k = \text{Projector}_M(\text{AvgPool}(\text{Encoder}_M(x^k)))$
`/* Loss computation */`
`// Eq. 1`
$L_{ins} = Loss(q, k|Q)$
`// Eq. 2`
$L_{reg} = (Loss(p, k|Q, sg(c)) + Loss(c, k|Q, sg(p)))/2$
$L = L_{ins} + L_{reg}$

**Algorithm 2:** Example code of RegionCL-D.

**Input:** Two augmented views $x^q$, $x^k$
**Input:** The instance negative Queue $Q_{ins}$
**Input:** The dense negative Queue $Q_{dense}$
**Output:** The contrastive loss $L$
`/* Feature extraction */`
`// Online Branch`
$x_c^p = \text{RegionSwapping}(x^q)$
$q = \text{Projector}(\text{AvgPool}(\text{Encoder}(x^q)))$
$p,c = \text{Projector}(\text{MaskPool}(\text{Encoder}(x_c^p)))$
`// Extract dense feature`
$q_d = \text{Projector}_d(\text{Encoder}(x^q))$
`// Momentum Branch`
$k = \text{Projector}_M(\text{AvgPool}(\text{Encoder}_M(x^k)))$
$k_d = \text{Projector}_{dM}(\text{Encoder}_M(x^k))$
`/* Loss computation */`
`// Eq. 1`
$L_{ins} = Loss(q, k|Q_{ins})$
$L_{dense} = DenseLoss(q_d, k_d|Q_{dense})$
`// Eq. 2`
$L_{reg} = (Loss(p, k|Q, sg(c)) + Loss(c, k|Q, sg(p)))/2$
$L = L_{ins} + L_{reg} + L_{dense}$

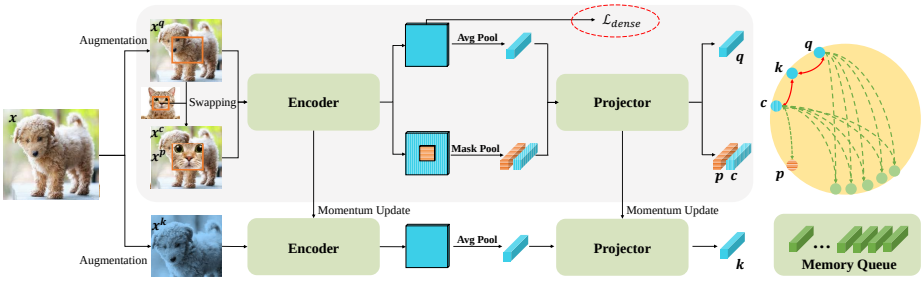


Fig. S2: Illustration of the proposed RegionCL with the DenseCL framework, *i.e.*, **RegionCL-D**. Taken the instance-level views $x^q$ and $x^k$, and the region-level views $x^c$ and $x^p$ as inputs, RegionCL-D firstly extract the instance- and region-level features using the same way as RegionCL-M. The extracted and projected features $q$, $p$, $c$, and $k$ 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.

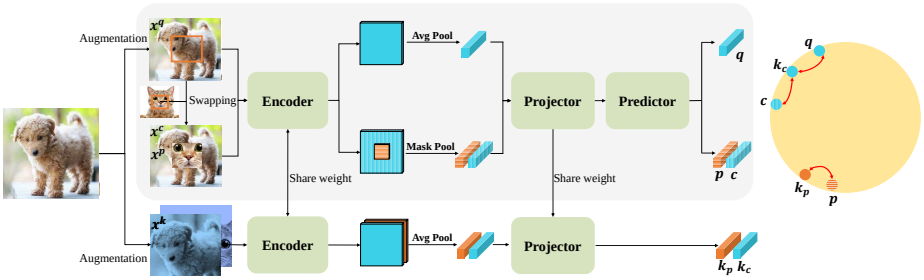


Fig. S3: Illustration of the proposed RegionCL with the SimSiam framework, *i.e.*, **RegionCL-S**. Taking the two augmented views as inputs, RegionCL-S extracts the region-level features in the same way as RegionCL-M. The pooled region-level features are then batched with the instance-level features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

from DenseCL to RegionCL-D appears at the instance-level branch in a same way as the modifications from MoCov2 [3] to RegionCL-M, *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 [5] does not require the negative pairs during pretraining and focuses on attracting the features among positive pairs. The modifications from SimSiam [5] to RegionCL-S are simply providing abundant positive pairs from both instance and region levels. Specifically, given the augmented views $x^q$, $x^k$ and the composite images

**Algorithm 3:** Example code of RegionCL-S.

**Input:** Two augmented views $x^q$, $x^k$
**Output:** The contrastive loss $L$
/* Feature extraction */
// 1st Branch
$x_c^p$ = RegionSwapping($x^q$)
$q$ = Predictor(Projector(AvgPool(Encoder($x^q$))))
$p$,$c$ = Predictor(Projector(MaskPool(Encoder($x_c^p$))))
// 2nd Branch, weight sharing with the 1st Branch
$k$ = Projector(AvgPool(Encoder($x^k$)))
/* Loss computation */
// Eq. 1
$L_{ins} = Loss(q, sg(k))$
// Eq. 2
$L_{reg} = (Loss(p, sg(k)) + Loss(c, sg(k))/2$
$L = L_{ins} + L_{reg}$

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 $q$ ($p$ and $c$). The other view $x^k$ are 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, *i.e.*, the instance-level pairs $q$ and $k$ as in origin SimSiam, the region-level pairs $c$ and $k_c$, 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, *i.e.*, RegionCL-M (MoCov2 [3]), RegionCL-D (DenseCL [13]), and RegionCL-S (SimSiam [5]), respectively.

**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 Sec A.5. The other settings are the same as in MoCov2 [3] and DenseCL [13]. For example, the input images are first randomly cropped and resized to 224 × 224 by remaining 20% → 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 [12] 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 *Linear(2048, 2048)* $\rightarrow$ *ReLU* $\rightarrow$ *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 *Conv2d(2048, 2048)* $\rightarrow$ *ReLU* $\rightarrow$ *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.

**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 MoCov2 [3] and DenseCL [13], *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 $Linear(2048, 2048) \rightarrow BN(2048) \rightarrow ReLU \rightarrow Linear(2048, 2048) \rightarrow BN(2048) \rightarrow ReLU \rightarrow Linear(2048, 2048) \rightarrow BN(2048)$. 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 $Linear(2048, 512) \rightarrow BN(512) \rightarrow ReLU \rightarrow Linear(512, 2048)$. These structures are the same as the original settings in SimSiam.

## References


1. Cai, Z., Ravichandran, A., Maji, S., Fowlkes, C., Tu, Z., Soatto, S.: Exponential moving average normalization for self-supervised and semi-supervised learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 194–203 (2021) 1, 2
2. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. vol. 1, pp. 1597–1607. PMLR (2020) 4
3. Chen, X., Fan, H., Girshick, R.B., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020) 1, 2, 3, 4, 7, 8, 9
4. Chen, X., Fang, H., Lin, T.Y., Vedantam, R., Gupta, S., Dollar, P., Zitnick, C.L.: Microsoft coco captions: Data collection and evaluation server. arXiv preprint arXiv:1504.00325 (2015) 1, 3, 4
5. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 15750–15758 (2021) 3, 7, 8
6. Contributors, M.: Openmmlab pose estimation toolbox and benchmark. `https://github.com/open-mmlab/mmpose` (2020) 3
7. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255. Ieee (2009) 1, 4
8. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. arXiv preprint arXiv:2111.06377 (2021) 1, 2
9. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 2961–2969 (2017) 2, 4
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016) 2
11. Oord, A.v.d., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748 (2018) 4
12. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International conference on machine learning. pp. 1139–1147. PMLR (2013) 8
13. Wang, X., Zhang, R., Shen, C., Kong, T., Li, L.: Dense contrastive learning for self-supervised visual pre-training. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3024–3033 (2021) 3, 5, 8, 9
14. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: Proceedings of the European conference on computer vision (ECCV). pp. 466–481 (2018) 3
15. Yu, H., Xu, Y., Zhang, J., Zhao, W., Guan, Z., Tao, D.: Ap-10k: A benchmark for animal pose estimation in the wild. In: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2) (2021) 3