A. Implementation details

A.1. Standard augmentations

First, an 224×224 image patch is randomly *cropped* from the image of *scale* in [0,2, 1.0] with random *horizontal flip*, followed by a *color distortion* with brightness, contrast, saturation, hue strength of $\{0.4, 0.4, 0.4, 0.1\}$ with an applying probability of 0.8, and an optional *grayscale* conversion with applying probability of 0.2. At last, *Gaussian blur* with kernel std in [0.1, 2.0] processes the image patch with applying probability of 0.5. Each of above image augmentations has been proved to be effective in at least one typical instance-wise self-supervised learning method. The composition of all above image augmentations are treated as standard augmentation $\mathcal{T}$.

A.2. Heavy augmentations

RandAugment covers various of image transformations and has been demonstrated with significant performance improvement on supervised representation learning. It has two tunable hyperparameters m, n, where n is the number of augmentation policies randomly selected from 14 predefined augmentations, m is the magnitude for all the transformation. Larger m and n results in stronger image transformation. In this paper, we denote *RandAugment* as *RA*(n, m). Similarity, *Jigsaw* has also been used in pretext task based self-supervised representation learning and fine-grained visual patterns. The hyperparameter n for *Jigsaw* is the number of grid $n \times n$ for each images. Our heavy augmentations $\widehat{\mathcal{T}}$ consists of *RA*(2,5) and *Jigsaw*(4) with probability of 0.9 and 0.1 respectively. Except these two augmentation policies, we applied *RA*(8,16) and *UniformAugment* as the additional heavy augmentation ($+\widehat{\mathcal{T}}$) for robustness analysis experiments in Table 3. There is no hyperparameter for *UniformAugment*. It is a search-free DA method consists of 13 different image transformations by assuming that the augmentation space is approximately distribution invariant.

We show some example images augmented by the standard augmentations and various heavy augmentations in Figure A1. All heavy augmented images are derived from the standard augmentation results.

A.3. ImageNet linear evaluation

For linear evaluation on ImageNet, we follow a similar procedure of SimCLR, BYOL, and SimSiam. The frozen ResNet's global average pooling layer outputs are used to train a supervised linear classifier. For SimCLR, we use the SGD optimizer as the original paper recommended. For SimSiam and BYOL, we used a LARS optimizer. We trained all linear classifier with base $lr = 1.6$ (following lr = 0.1 × BatchSize/256) and batch size of 4096 for 90 epochs. For the SGD optimizer, we decay the learning rate with the linear decay schedule. For the LARS optimizer, we decay the learning rate with the cosine decay schedule. After training the linear classifiers, we evaluate them on the center cropped 224×224 resolution inputs in validation set.

Algorithm A1 The Pseudocode of SimSiam baseline with 2 view-pairs

```
# f: feature encoder; g: prediction head;
# aug: data augmentations;

for I in dataloader:
    # random augmentation
    v1, v1', v2, v2' = aug(I), aug(I), aug(I), aug(I)
    # projections
    y1, y1', y2, y2' = f(v1), f(v1'), f(v2), f(v2')
    # predictions
    z1, z1', z2, z2' = g(y1), g(y1'), g(y2), g(y2')
    # loss
    L = D(z1,y1')/4 + D(z1',y1)/4 + D(z2,y2')/4 + D(z2',y2)/4

    L.backward()  # back-propagate
    update(f, h)  # SGD update

def D(z, y):  # negative cosine similarity
    y = y.detach()  # stop gradient
    # l2-normalize
    z = normalize(z, dim=1)
    y = normalize(y, dim=1)
    return -(z * y).sum(dim=1).mean()
```

Baseline Implementation	CIFAR-10	ImageNet
1 standard view-pair	92.14	67.7
2 standard view-pairs	91.68	67.4
2 heavily augmented view-pairs	*collapse*	*collapse*

Table A1. Linear evaluation Top-1 accuracy of SimSiam on ImageNet and CIFAR-10 with same training epochs but different training pair settings.

B. Experimental results

B.1. Compared methods

Algorithm A1 shows the Pseudocode of our re-implemented SimSiam with four views. We list the SimSiam results of its original version (with two standard views), our re-implemented version with four standard views, and re-implemented versions with four heavily augmented views in Table A1. Although previous work SwAV [1] shows that contrasting more image crops with smaller size (e.g., 96×96) with regular image crops (224×224) results in performance improvement owing to more visual details in low-resolution crops are highlighted, introducing more views with the same resolution in each min-batch has little impact on performance. Even though the number of views is increased, the frequency of model updating during training and the number of raw samples in min-batch keep the same and results in similar results (one view-pair vs. two view-pairs).

For SimCLR that considers negative samples in the whole min-batch (batch size of n), DSSL only add two

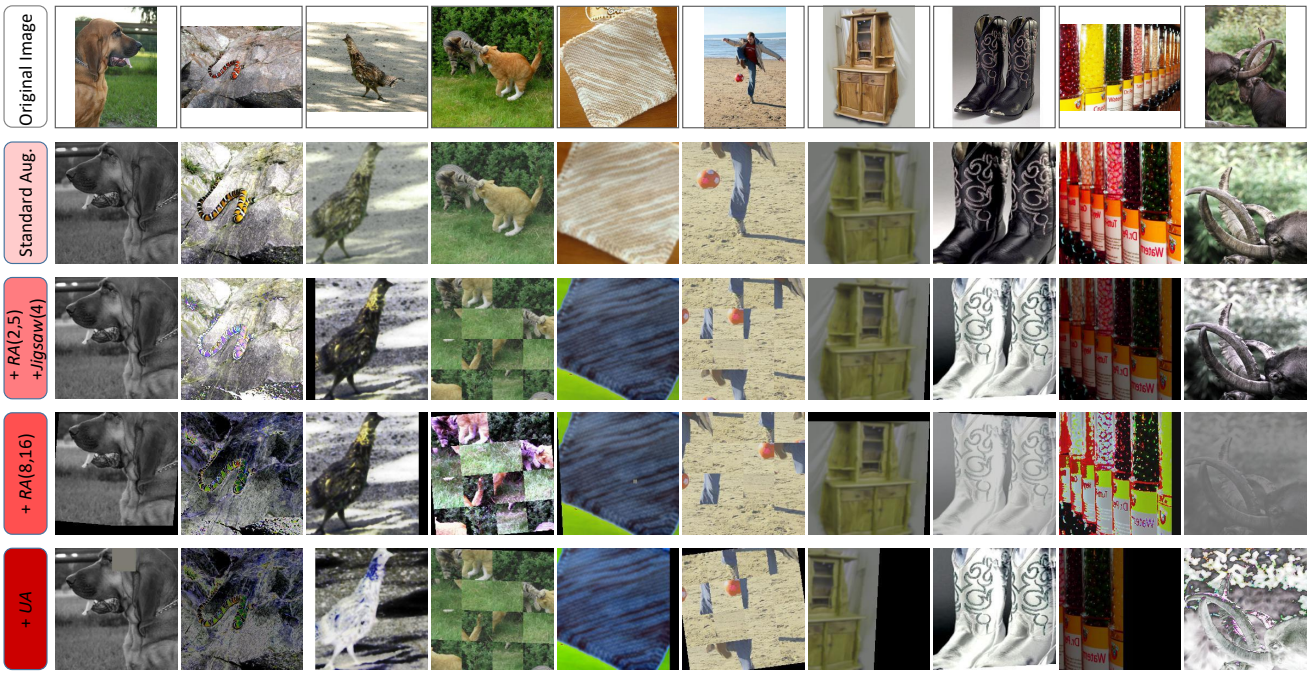

Figure A1. Example views augmented by various data augmentation on ImageNet dataset. For each original image (in the first row), we show their standard views (in the second row), the heavily augmented views derived from the standard views by applying $\hbar$ as *RA*(2,5) and *Jigsaw*(4) (in the third row), and the results of the additional heavy augmentation *RA*(8,16) and *UA* (in the last two rows).

	$m = 1$	$m = 2$	$m = 5$	$m = 9$
$n = 1$	88.0 (92.3)	85.4 (92.8)	85.7 (94.0)	86.0 (92.6)
$n = 2$	84.3 (93.1)	82.8 (92.56)	83.5 (94.2)	*collapse* (92.8)
$n = 5$	*collapse* (92.6)	*collapse* (92.9)	*collapse* (93.0)	*collapse* (90.9)
$n = 9$	*collapse* (91.8)	*collapse* (91.2)	*collapse* (88.2)	*collapse* (87.7)

Table A2. SimSiam and (SimSiam w/DSSL) linear evaluation accuracy (%) on CIFAR-10. $\hbar = RA(n,m)$

heavily augmented views and two $t(I) \leftarrow \widehat{t}(I)$ asymmetric loss items for each image. The computation complexity slightly increases from $2n^2$ to $2n^2+2n$ for each min-batch after applying DSSL.

As shown in Algorithm 1 and Eq. (5), DSSL has no additional hyperparameters comparing to the standard instance-wise self-supervised learning methods.

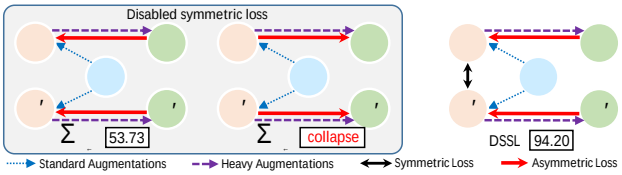

Figure A2. Results of models by removing symmetric loss.

B.2. Ablation Studies

Impact of distortion magnitudes of $\widehat{\mathcal{T}}$. In Tab. A2, we compared the results from the standard SimSiam and its DSSL version by introducing only *RA* as heavy augmentation but with various distortion magnitudes. We use the hyperparameter m, n of RandAugment to control the distortion magnitudes and amount of the randomly selected policies respectively.

It can be found that, after increasing m and n, the self-supervised models result in divergence. In contrast, SimSiam *w/ DSSL* can converge stably in a wide range of m and n. It further demonstrates the robustness of our proposed directional self-supervised learning on partially ordered views.

Impact of the original symmetric loss Fig. A2 (Left) show the result of model by removing the symmetric loss $\mathcal{L}_S$ in Eq. (5). The symmetric loss is still essential to learn high-quality representation by clustering the slightly augmented views. The asymmetric loss in DSSL is designed for merely handing the missing information heavily augmented views, it can not completely substitute the effect of symmetric loss among slightly augmented views.