A. Implementation Details

Unsupervised pre-training. Our implementation follows the practice of existing works [36, 17, 8, 9, 15].

Data augmentation. We describe data augmentation using the PyTorch [31] notations. Geometric augmentation is `RandomResizedCrop` with scale in $[0.2, 1.0]$ [36] and `RandomHorizontalFlip`. Color augmentation is `ColorJitter` with {brightness, contrast, saturation, hue} strength of {0.4, 0.4, 0.4, 0.1} with an applying probability of 0.8, and `RandomGrayscale` with an applying probability of 0.2. Blurring augmentation [8] has a Gaussian kernel with std in $[0.1, 2.0]$.

Initialization. The convolution and fc layers follow the default PyTorch initializers. Note that by default PyTorch initializes fc layers' weight and bias by a uniform distribution $\mathcal{U}(-\sqrt{k}, \sqrt{k})$ where $k=\frac{1}{\text{in\_channels}}$. Models with substantially different fc initializers (*e.g.*, a fixed std of 0.01) may not converge. Moreover, similar to the implementation of [8], we initialize the scale parameters as 0 [14] in the last BN layer for every residual block.

Weight decay. We use a weight decay of 0.0001 for all parameter layers, including the BN scales and biases, in the SGD optimizer. This is in contrast to the implementation of [8, 15] that excludes BN scales and biases from weight decay in their LARS optimizer.

Linear evaluation. Given the pre-trained network, we train a supervised linear classifier on frozen features, which are from ResNet's global average pooling layer (pool_5). The linear classifier training uses base $lr = 0.02$ with a cosine decay schedule for 90 epochs, weight decay $= 0$, momentum $= 0.9$, batch size $= 4096$ with a LARS optimizer [38]. We have also tried the SGD optimizer following [17] with base $lr = 30.0$, weight decay $= 0$, momentum $= 0.9$, and batch size $= 256$, which gives $\sim$1% lower accuracy. After training the linear classifier, we evaluate it on the center 224×224 crop in the validation set.

B. Additional Ablations on ImageNet

The following table reports the SimSiam results *vs*. the output dimension d:

output d	256	512	1024	2048
acc. (%)	65.3	67.2	67.5	68.1

It benefits from a larger d and gets saturated at $d = 2048$. This is unlike existing methods [36, 17, 8, 15] whose accuracy is saturated when d is 256 or 512.

In this table, the prediction MLP's hidden layer dimension is always 1/4 of the output dimension. We find that this bottleneck structure is more robust. If we set the hidden dimension to be equal to the output dimension, the training can be less stable or fail in some variants of our exploration. We hypothesize that this bottleneck structure, which behaves like an auto-encoder, can force the predictor to digest the information. We recommend to use this bottleneck structure for our method.

	SimCLR			MoCo v2		BYOL			SwAV
epoch	200	800	1000	200	800	300	800	1000	400
origin	66.6	68.3	69.3	67.5	71.1	72.5	-	74.3	70.1
repro.	68.3	70.4	-	69.9	72.2	72.4	74.3	-	70.7

Table C.1. **Our reproduction *vs*. original papers' results**. All are based on ResNet-50 pre-trained with two 224×224 crops.

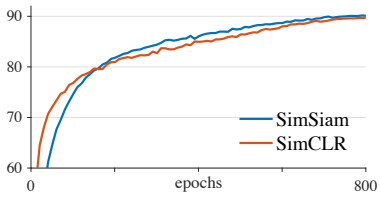
C. Reproducing Related Methods

Our comparison in Table 4 is based on our reproduction of the related methods. We re-implement the related methods as faithfully as possible following each individual paper. In addition, we are able to improve SimCLR, MoCo v2, and SwAV by small and straightforward modifications: specifically, we use 3 layers in the projection MLP in SimCLR and SwAV (*vs*. originally 2), and use symmetrized loss for MoCo v2 (*vs*. originally asymmetric). Table C.1 compares our reproduction of these methods with the original papers' results (if available). Our reproduction has *better* results for SimCLR, MoCo v2, and SwAV (denoted as "+" in Table 4), and has at least comparable results for BYOL.

D. CIFAR Experiments

We have observed similar behaviors of SimSiam in the CIFAR-10 dataset [24]. The implementation is similar to that in ImageNet. We use SGD with base $lr = 0.03$ and a cosine decay schedule for 800 epochs, weight decay $= 0.0005$, momentum $= 0.9$, and batch size $= 512$. The input image size is 32×32. We do not use blur augmentation. The backbone is the CIFAR variant of ResNet-18 [19], followed by a 2-layer projection MLP. The outputs are 2048-d.

Figure D.1 shows the kNN classification accuracy (left) and the linear evaluation (right). Similar to the ImageNet observations, SimSiam achieves a reasonable result and does not collapse. We compare with SimCLR [8] trained with the same setting. Interestingly, the training curves are similar between SimSiam and SimCLR. SimSiam is slightly better by 0.7% under this setting.



	acc. (%)
SimCLR	91.1
SimSiam	91.8

Figure D.1. **CIFAR-10 experiments**. Left: validation accuracy of kNN classification as a monitor during pre-training. Right: linear evaluation accuracy. The backbone is ResNet-18.