Exploring Simple Siamese Representation Learning (SimSiam)

CVPR2021 Under review

21.01.11 Leeminsoo

DAVIAN

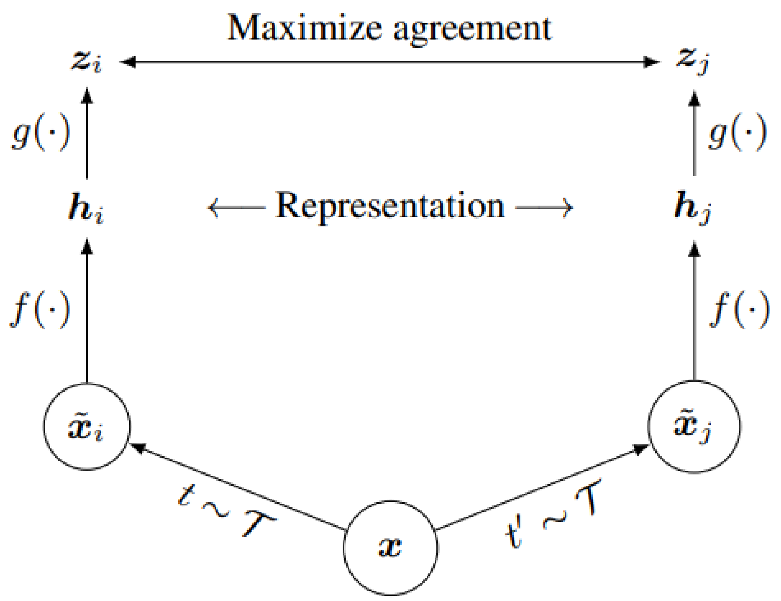
Data and Visual Analytics Lab

Self-Supervised Learning

- Supervised Learning is powerful, but need a large amount of labeled data
  - Many research for tackling this problem is in progress.
  - Transfer learning, Domain adaptation, Semi-supervised, Weakly-supervised and Unsupervised Learning

- Self-Supervised Visual Representation Learning
  - Sub-class of Unsupervised learning where the data provides the "self-supervision"
  - Define pretext tasks(including contrastive learning) which can be formulated using only unlabeled data, but do require higher-level semantic understanding in order to solved
  - The features obtained with pretext tasks can be successfully transferred to down-stream tasks.

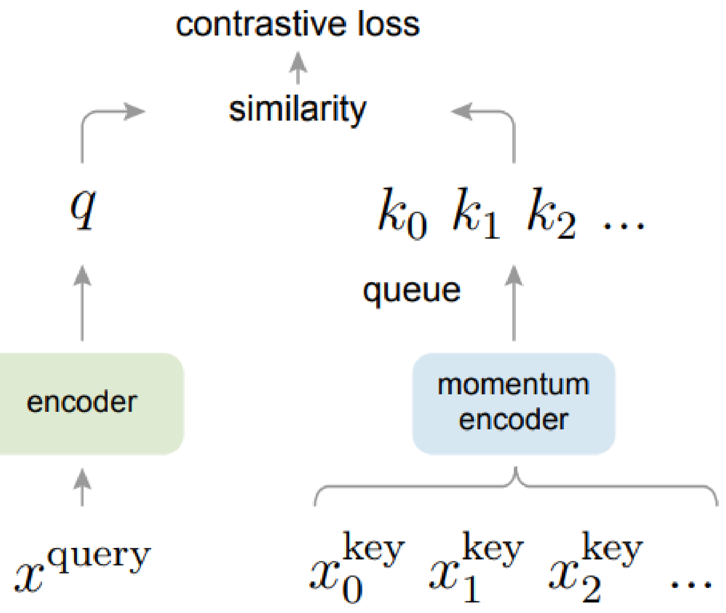
SimCLR



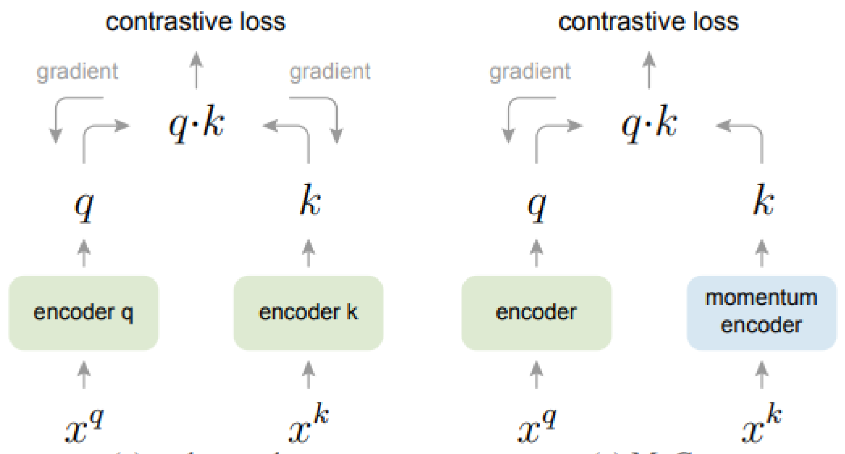
$$\ell_{i,j} = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\mathrm{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

- SimCLR learns representations by contrasting positive and negative examples(Contrastive Learning).
- The projection head g is thrown away, and the encoder f and the representation h are used for downstream tasks.
- SimCLR needs large batch size, because a large number of negatives is required to obtain good representations.

Momentum Contrastive Encoder



[MoCo]

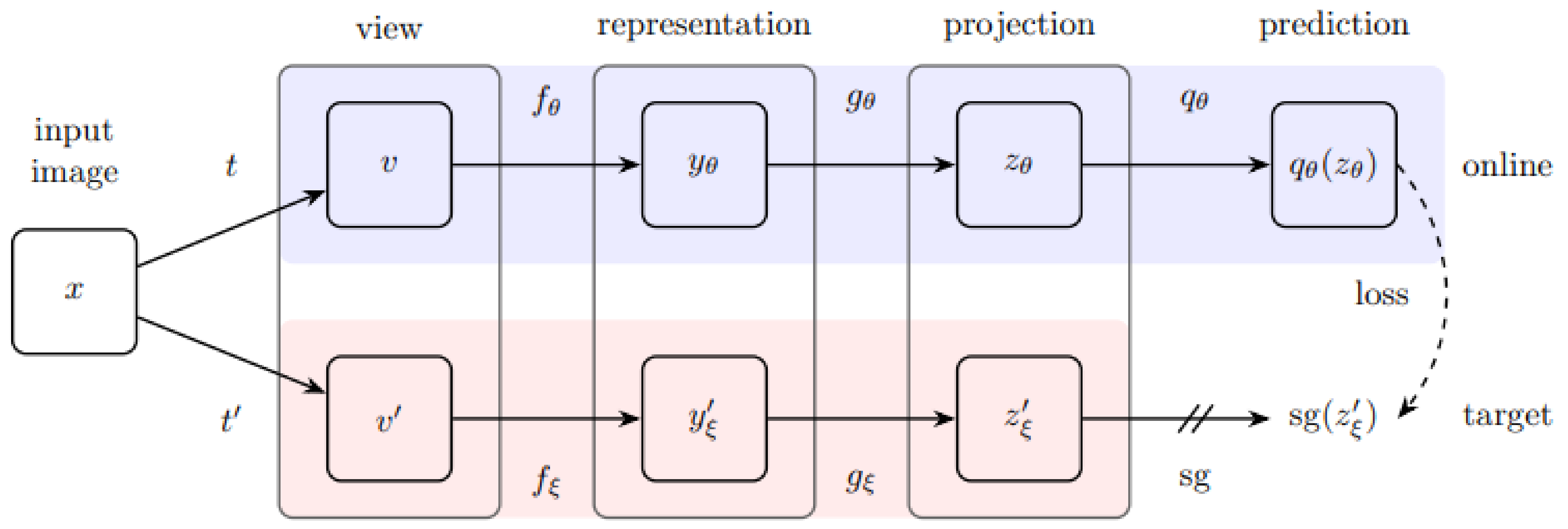


(a) end-to-end (c) MoCo

$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1-m)\theta_{\mathrm{q}}.$$

- A more efficient way of storing negative batches is proposed where you just have queue of vector representations rather than passing the entire high dimensional images through the network.
- But using queue makes the model difficult to update the encoder. The naïve solution is using the same key encoder as the query encoder, but it leads to bad results due to rapid encoder changes that reduce the consistency of key representation. To solve this problem, the momentum encoder is proposed.

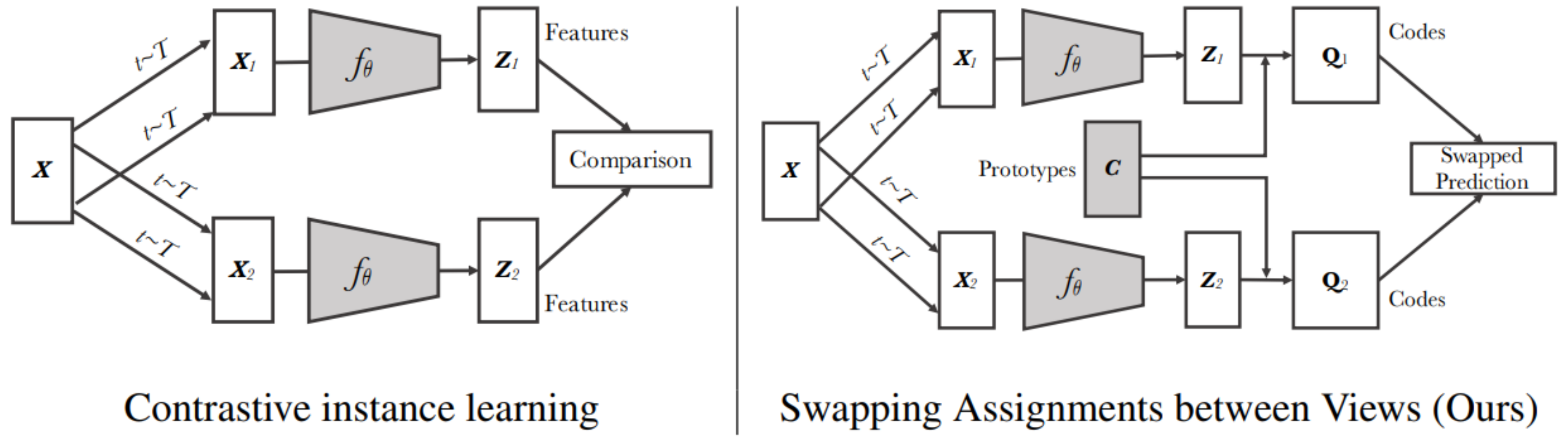
Bootstrap Your Own Latent



[BYOL]

- The online network is predicting the output of the target network.
- Any negative samples are not used to train the model.
- Similar to MoCo, the model use a slowly moving exponential average of the online network as the target network.

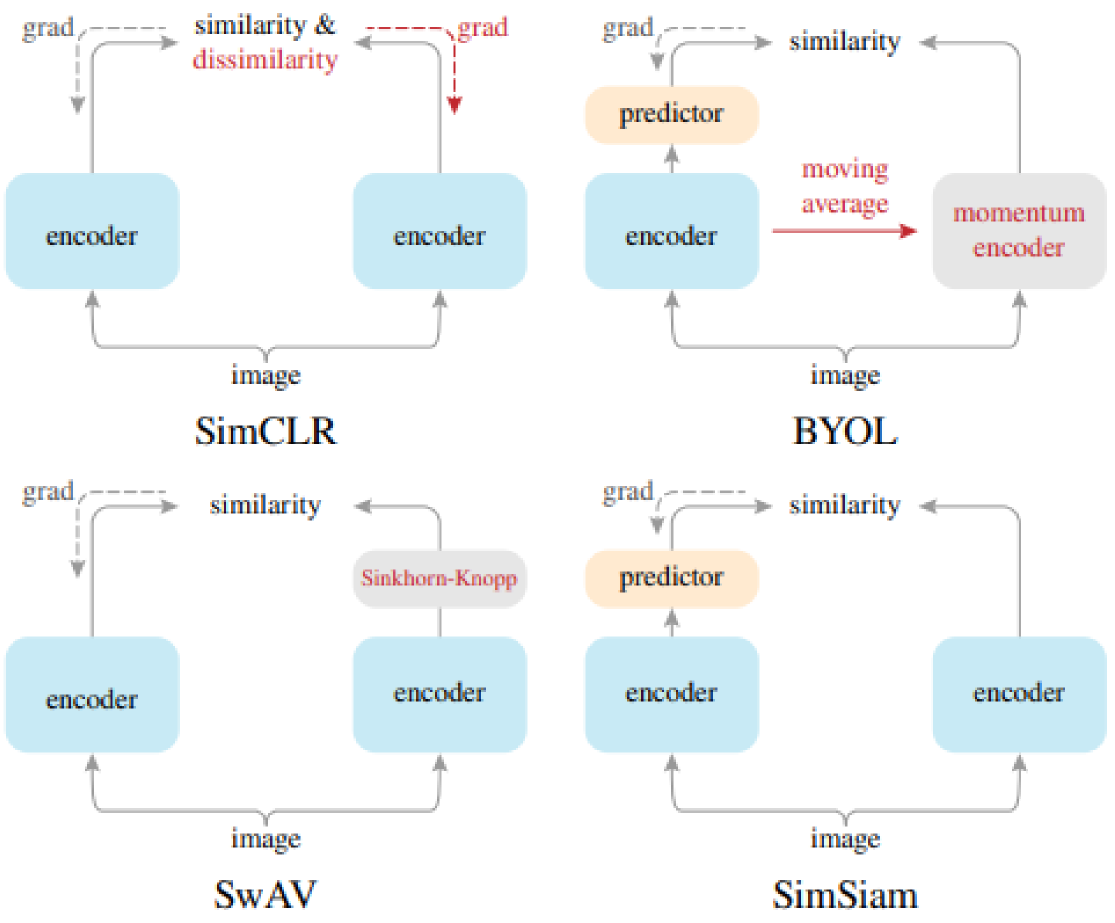
Swapping Assignments between Views



[SwAV]

- The model uses one view of the image to predict which cluster assignment from the other view of the image, rather than trying to maximize the similarity of the two views directly

SimSiam



[Unifying View with SimSiam]

SimSiam

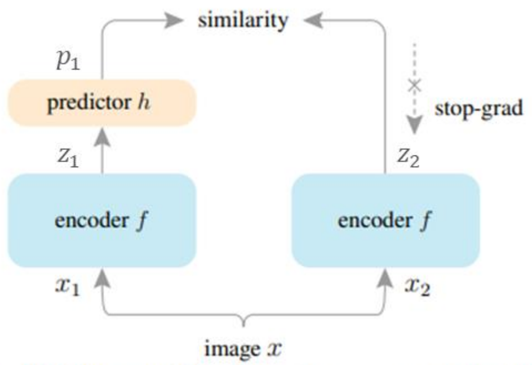


Figure 1. **SimSiam architecture**. Two augmented views of one image are processed by the same encoder network f (a backbone plus a projection MLP). Then a prediction MLP h is applied on one side, and a stop-gradient operation is applied on the other side. The model maximizes the similarity between both sides. It uses neither negative pairs nor a momentum encoder.

$$\mathcal{D}(p_1, z_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{z_2}{\|z_2\|_2},$$

$$\mathcal{L}=\frac{1}{2}\mathcal{D}(p_1, \texttt{stopgrad}(z_2))+\frac{1}{2}\mathcal{D}(p_2, \texttt{stopgrad}(z_1)).$$

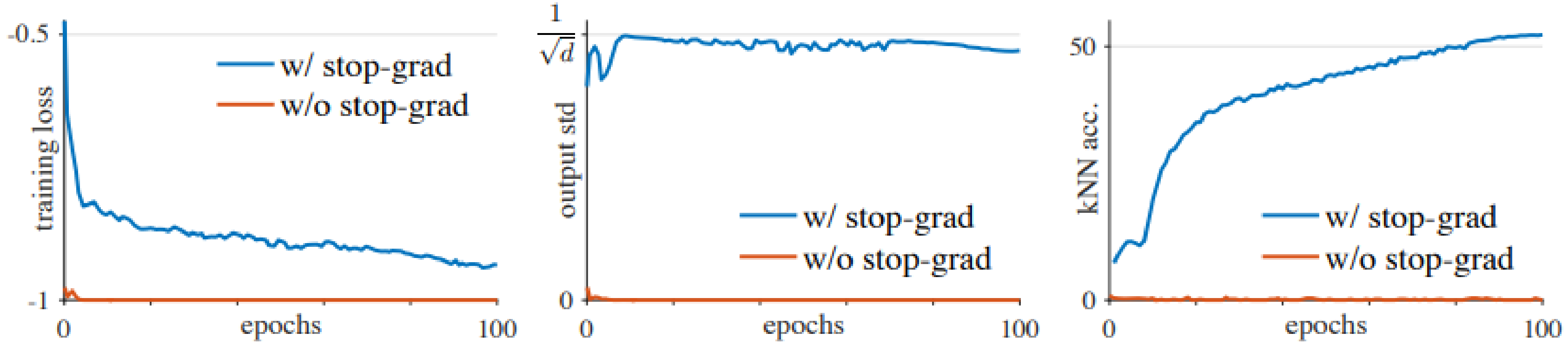
Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# f: backbone + projection mlp
# h: prediction mlp

for x in loader: # load a minibatch x with n samples
    x1, x2 = aug(x), aug(x) # random augmentation
    z1, z2 = f(x1), f(x2) # projections, n-by-d
    p1, p2 = h(z1), h(z2) # predictions, n-by-d

    L = D(p1, z2)/2 + D(p2, z1)/2 # loss

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

def D(p, z): # negative cosine similarity
    z = z.detach() # stop gradient

    p = normalize(p, dim=1) # l2-normalize
    z = normalize(z, dim=1) # l2-normalize
    return -(p*z).sum(dim=1).mean()
```

- Simple Siames networks can learn meaningful representations even using none of the following:

  1) negative sample pairs, 2) large batches, 3) momentum encoders
- A stop-gradient operation is critical to prevent the collapsing solutions.

Experiment

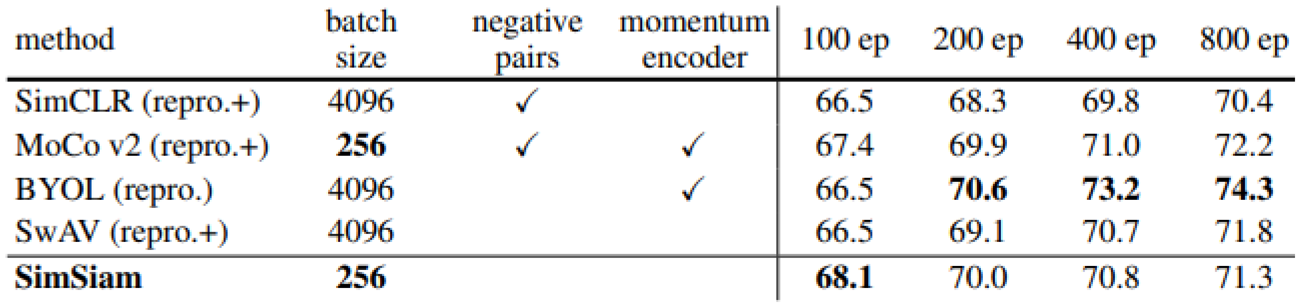


	acc. (%)
w/ stop-grad	67.7±0.1
w/o stop-grad	0.1

Figure 2. **SimSiam with *vs.* without stop-gradient**. **Left plot**: training loss. Without stop-gradient it degenerates immediately. **Middle plot**: the per-channel std of the ℓ_2-normalized output, plotted as the averaged std over all channels. **Right plot**: validation accuracy of a kNN classifier [36] as a monitor of progress. **Table**: ImageNet linear evaluation ("w/ stop-grad" is mean±std over 5 trials).

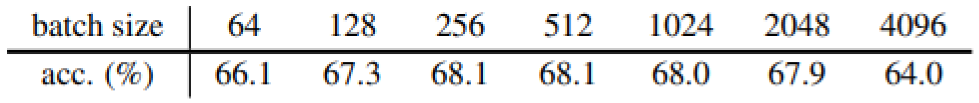
- It presents a comparison on "with vs without stop-gradient".
- Without stop-gradient, the optimizer quickly finds degenerated solution and reaches the minimum possible loss of -1.
- When studying the standard deviation of the l2-normalized output $\frac{z}{\|z\|_2}$, the outputs of the model w/o stop-grad collapse to a constant vector.
- With stop-gradient, the std value is near $\frac{1}{\sqrt{d}}$. This indicates that the outputs do not collapse, and they are scattered on the unit hypersphere.

Experiment



method	batch size	negative pairs	momentum encoder	100 ep	200 ep	400 ep	800 ep
SimCLR (repro.+)	4096	✓		66.5	68.3	69.8	70.4
MoCo v2 (repro.+)	**256**	✓	✓	67.4	69.9	71.0	72.2
BYOL (repro.)	4096		✓	66.5	**70.6**	**73.2**	**74.3**
SwAV (repro.+)	4096			66.5	69.1	70.7	71.8
SimSiam	**256**			**68.1**	70.0	70.8	71.3

[Comparisons on ImageNet linear classification]

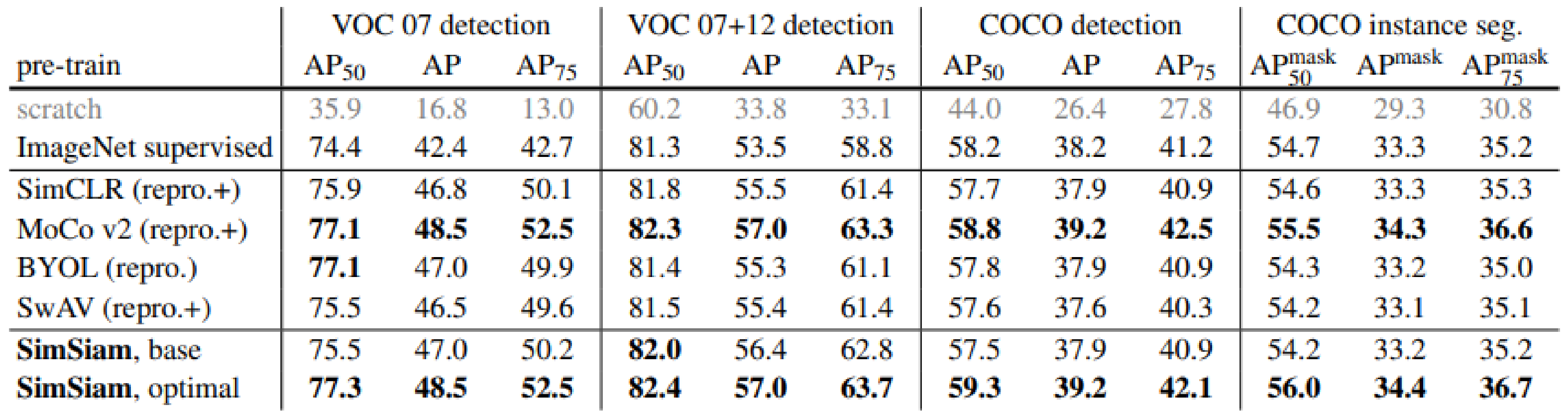


batch size	64	128	256	512	1024	2048	4096
acc. (%)	66.1	67.3	68.1	68.1	68.0	67.9	64.0

[Effect of batch sizes(ImageNet linear evaluation accuracy with 100-epoch pre-training)]

- SimSiam is trained with a batch size of 256, using neither negative samples nor a momentum encoder.
- Despite its simplicity, SimSiam achieves competitive results.
- It has the highest accuracy among all methods under 100-epoch pre-training, though its gain of training longer is smaller.

Experiment



pre-train	VOC 07 detection			VOC 07+12 detection			COCO detection			COCO instance seg.		
	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP^{mask}_{50}	AP^{mask}	AP^{mask}_{75}
scratch	35.9	16.8	13.0	60.2	33.8	33.1	44.0	26.4	27.8	46.9	29.3	30.8
ImageNet supervised	74.4	42.4	42.7	81.3	53.5	58.8	58.2	38.2	41.2	54.7	33.3	35.2
SimCLR (repro.+)	75.9	46.8	50.1	81.8	55.5	61.4	57.7	37.9	40.9	54.6	33.3	35.3
MoCo v2 (repro.+)	**77.1**	**48.5**	**52.5**	**82.3**	**57.0**	**63.3**	**58.8**	**39.2**	**42.5**	**55.5**	**34.3**	**36.6**
BYOL (repro.)	**77.1**	47.0	49.9	81.4	55.3	61.1	57.8	37.9	40.9	54.3	33.2	35.0
SwAV (repro.+)	75.5	46.5	49.6	81.5	55.4	61.4	57.6	37.6	40.3	54.2	33.1	35.1
SimSiam, base	75.5	47.0	50.2	**82.0**	56.4	62.8	57.5	37.9	40.9	54.2	33.2	35.2
SimSiam, optimal	**77.3**	**48.5**	**52.5**	**82.4**	**57.0**	**63.7**	**59.3**	**39.2**	**42.1**	**56.0**	**34.4**	**36.7**

Table 5. **Transfer Learning**. All unsupervised methods are based on 200-epoch pre-training in ImageNet. *VOC 07 detection*: Faster R-CNN [32] fine-tuned in VOC 2007 trainval, evaluated in VOC 2007 test; *VOC 07+12 detection*: Faster R-CNN fine-tuned in VOC 2007 trainval + 2012 train, evaluated in VOC 2007 test; *COCO detection* and *COCO instance segmentation*: Mask R-CNN [18] (1× schedule) fine-tuned in COCO 2017 train, evaluated in COCO 2017 val. All Faster/Mask R-CNN models are with the C4-backbone [13]. All VOC results are the average over 5 trials. **Bold entries** are within 0.5 below the best.

- All methods are based on 200-epoch pre-training in ImageNet.
- Despite the simple structure of SimSiam, it is competitive among these leading methods.
- Another hyperparameters can produce better results in all tasks with similar ImageNet accuracy(“SimSiam, optimal”)

EOD