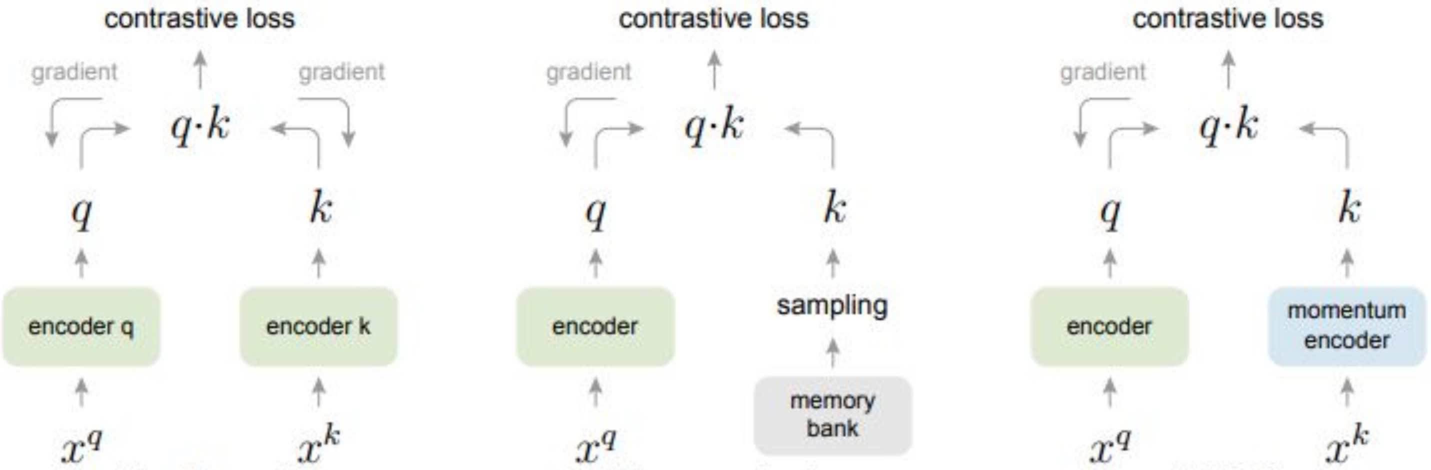
BYOL:

Bootstrap Your Own Latent

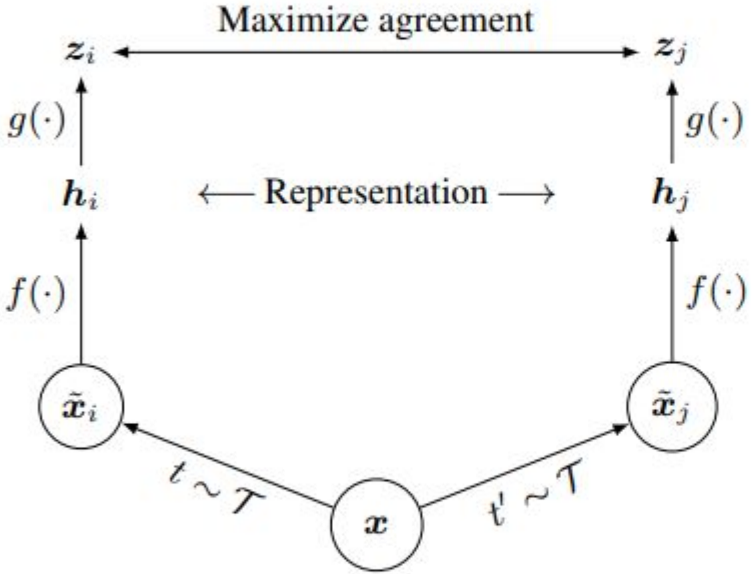
https://arxiv.org/abs/2006.07733

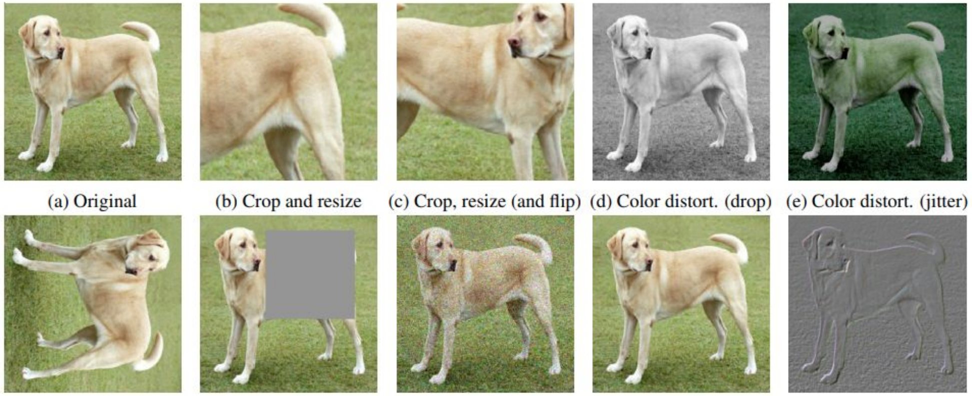
MoCo v2, 2020 CVPR



(a) end-to-end (b) memory bank (c) MoCo

SimCLR, arXiv:2002.05709





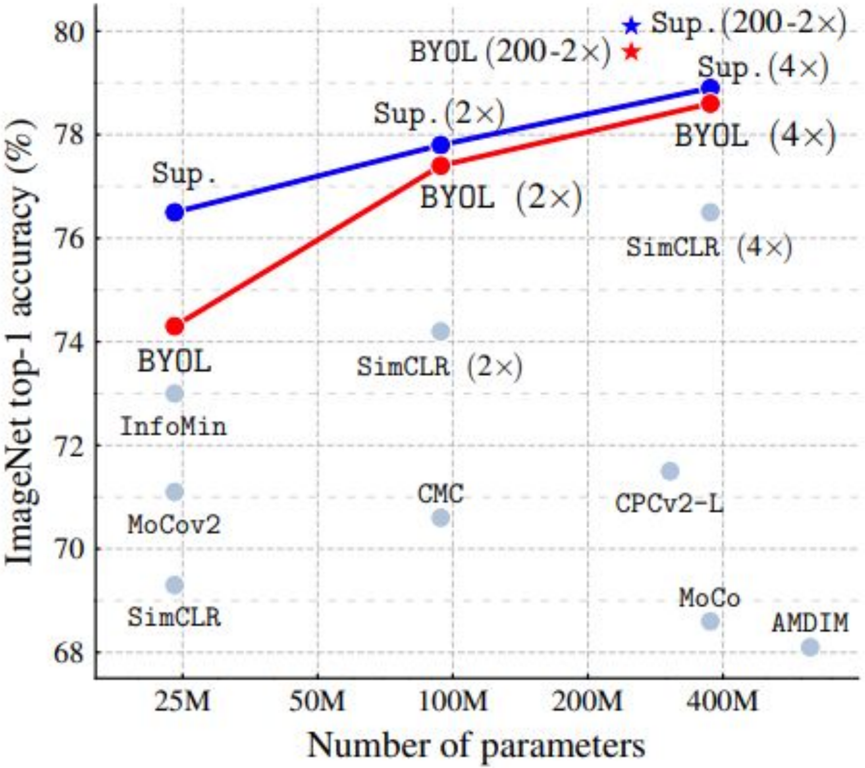
(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate {90°, 180°, 270°} (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Introduction

- State-of-the-art contrastive methods
  - reducing the distance between positive pairs
  - increasing the distance between negative pairs
- Careful treatment of negative pairs
  - **Large batch size** : SimCLR(20.02) / Googlebrain
  - **Memory bank** : MoCo(19.11), MoCo v2(20.03) / FAIR
  - **Customized mining strategies**

Introduction



Contributions

- Achieves higher performance than state-of-the-art contrastive methods **without using negative pairs**.

- More robust to the choice of image augmentations than contrastive methods.

- Uses two neural networks, referred to as online and target networks, that interact and learn from each other.

- Trains its online network to predict the target network's representation of another augmented view of the same image.

Related work

- Most unsupervised methods for representation learning:

    - **Generative :**

      build a distribution over data and latent embedding and use the learned embeddings as image representations.

      Operate directly in pixel space.

      → computationally expensive
      → high level of detail required for image generation may not be necessary for representation learning.

Related work

- Most unsupervised methods for representation learning:
    - **Discriminative :**

      → Contrastive approaches avoid a costly generation step in pixel space.

      → Deep Cluster uses bootstrapping on previous versions of its representation to produce targets for the next representation

      → Relative patch prediction, Colorizing gray-scale image, image inpainting, ...

Related work - Discriminative

Relative Position

Jigsaw

Colorization

Inpainting

Rotation Prediction

Counting

Clustering

Motion Segmentation

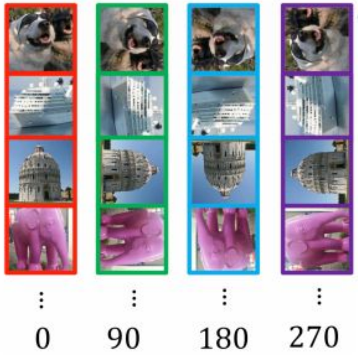
Auto-Encoder (Reconstruction)

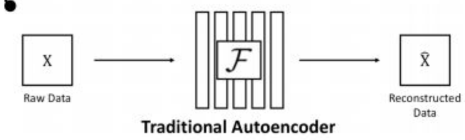
Split-Channel Auto-Encoder

Transformation Auto-Encoder

Future Frame Prediciton

Trajectory Prediciton





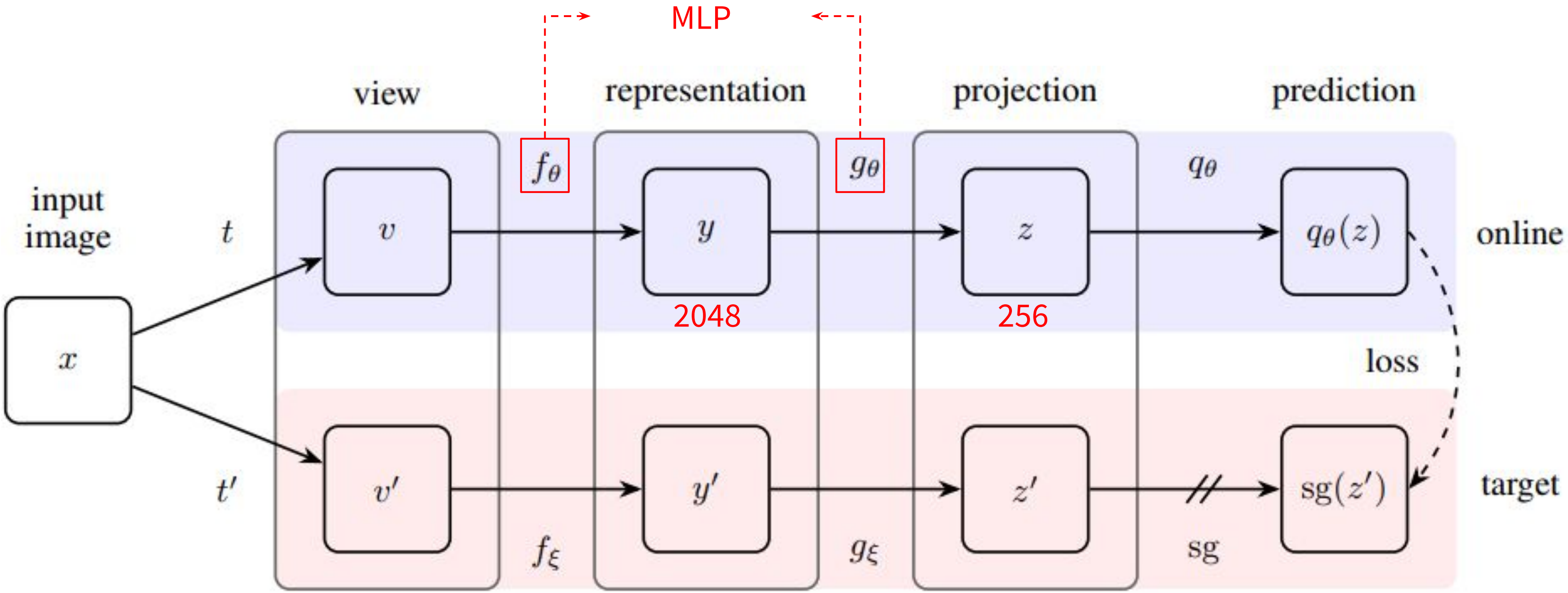
Method

- Many approaches cast the prediction problem directly in representation space: augmented views.

- Predicting directly in representation space can lead to collapsed representation: for instance, a representation that is constant across view is always **fully predictive of it self.**

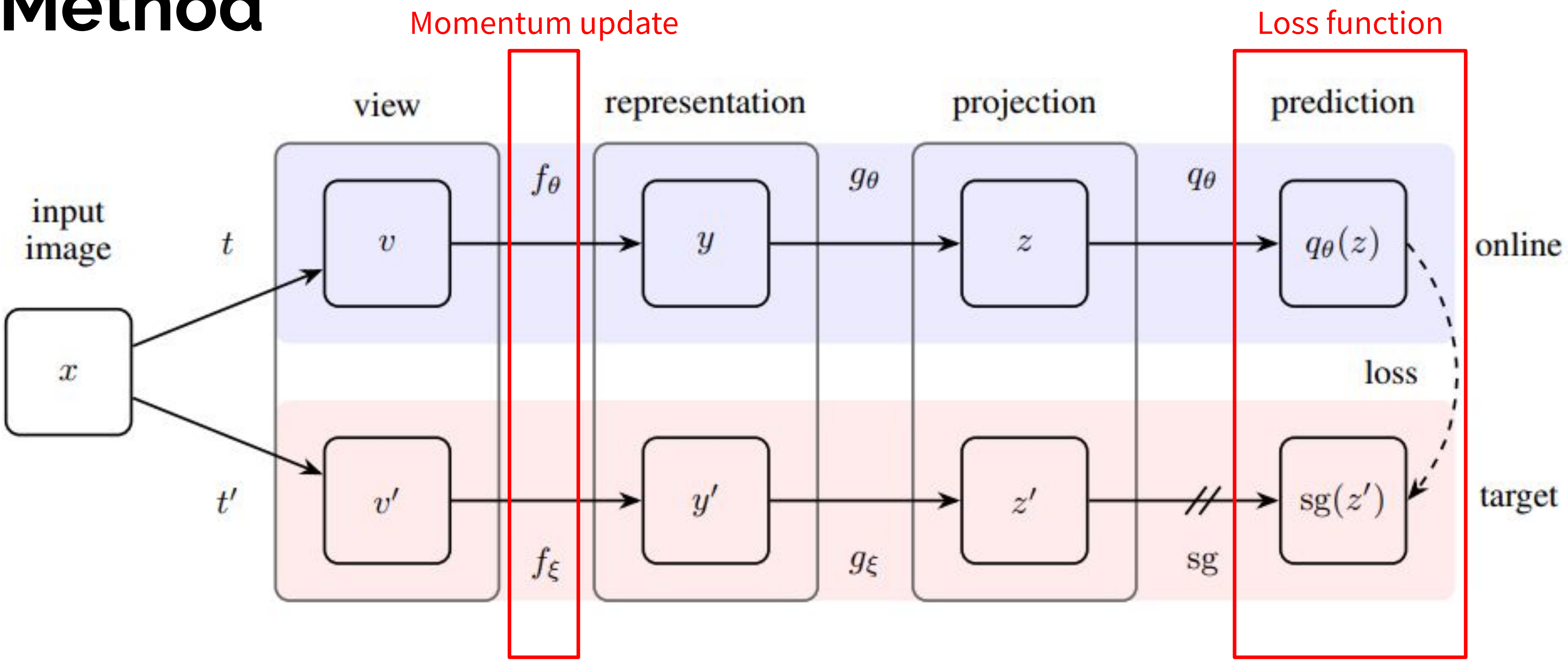
- Discriminative approach typically requires comparing each representation of an augmented view with many negative examples.

Method



MLP
view
representation
projection
prediction
input image
t
t'
x
v
v'
f_θ
f_ξ
y
y'
2048
g_θ
g_ξ
z
z'
256
q_θ
$q_\theta(z)$
sg
$\mathrm{sg}(z')$
online
loss
target

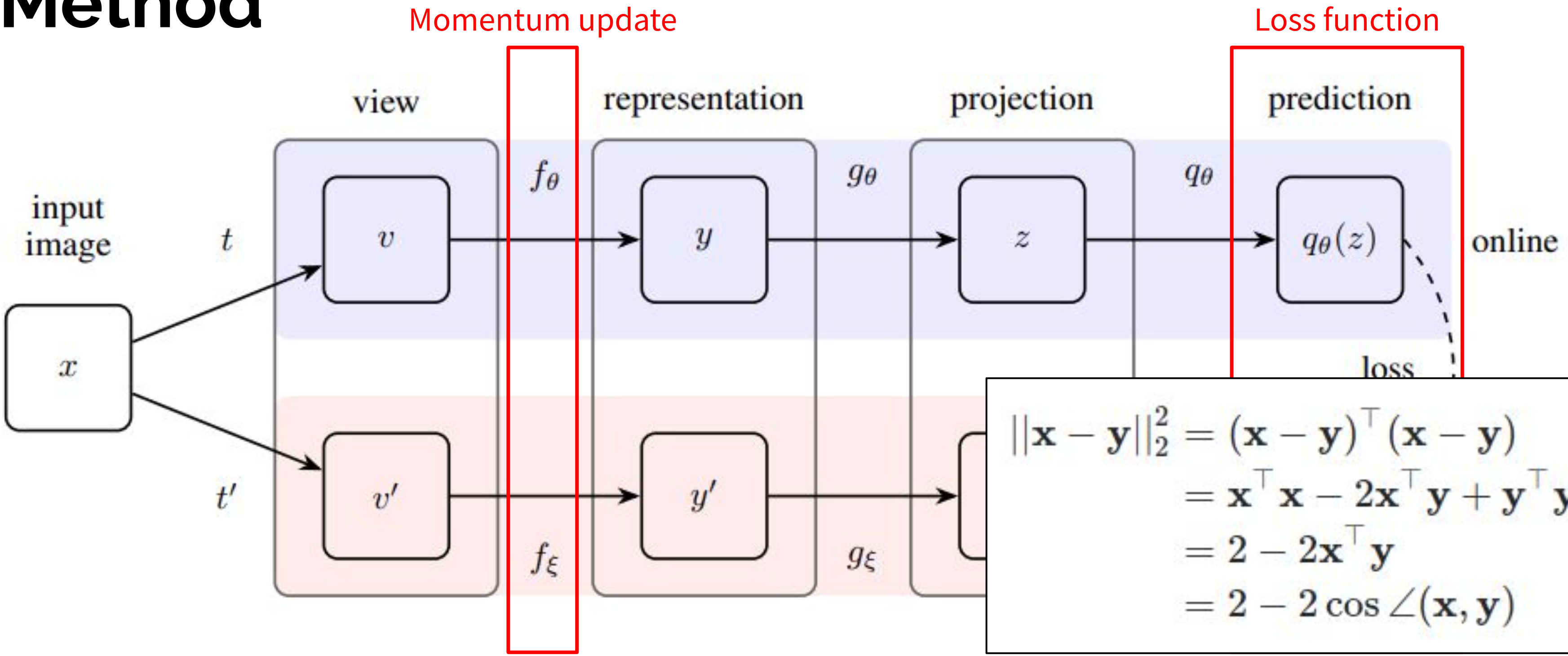
Method



$$\xi \leftarrow \tau\xi + (1-\tau)\theta.$$

$$\mathcal{L}_\theta^{\mathrm{BYOL}} \triangleq \left\|\overline{q_\theta}(z_\theta) - \bar{z}'_\xi\right\|_2^2 = 2 - 2 \cdot \frac{\langle q_\theta(z_\theta), z'_\xi\rangle}{\left\|q_\theta(z_\theta)\right\|_2 \cdot \left\|z'_\xi\right\|_2}.$$

Method

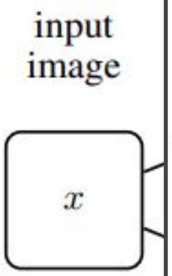


$$
\begin{aligned}
\|\mathbf{x}-\mathbf{y}\|_2^2 &= (\mathbf{x}-\mathbf{y})^\top(\mathbf{x}-\mathbf{y}) \\
&= \mathbf{x}^\top\mathbf{x} - 2\mathbf{x}^\top\mathbf{y} + \mathbf{y}^\top\mathbf{y} \\
&= 2 - 2\mathbf{x}^\top\mathbf{y} \\
&= 2 - 2\cos\angle(\mathbf{x},\mathbf{y})
\end{aligned}
$$

$$\xi \leftarrow \tau\xi + (1-\tau)\theta.$$

$$\mathcal{L}_\theta^{\text{BYOL}} \triangleq \left\|\overline{q_\theta}(z_\theta) - \overline{z}'_\xi\right\|_2^2 = 2 - 2\cdot\frac{\langle q_\theta(z_\theta), z'_\xi\rangle}{\left\|q_\theta(z_\theta)\right\|_2 \cdot \left\|z'_\xi\right\|_2}.$$

Method (detail)



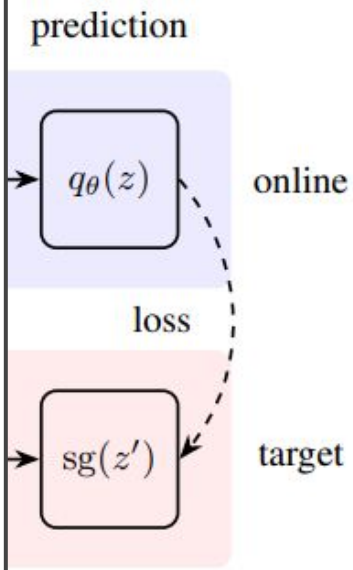
MLP

```python
def network(inputs):
  """Build the encoder, projector and predictor."""
  embedding = ResNet(name='encoder', configuration='ResNetV1_50x1')(inputs)
  proj_out = MLP(name='projector')(embedding)
  pred_out = MLP(name='predictor')(proj_out)
  return dict(projection=proj_out, prediction=pred_out)


class MLP(hk.Module):
  """Multi Layer Perceptron, with normalization."""

  def __init(self, name):
    super().__init__(name=name)

  def __call__(self, inputs):
    out = hk.Linear(output_size=HPS['mlp_hidden_size'])(inputs)
    out = hk.BatchNorm(**HPS['batchnorm_kwargs'])(out)
    out = jax.nn.relu(out)
    out = hk.Linear(output_size=HPS['projection_size'])(out)
    return out
```



Details

Proj. g_θ depth	Pred. q_θ depth	Top-1	Top-5
1	1	61.9	86.0
	2	65.0	86.8
	3	65.7	86.8
2	1	71.5	90.7
	2	**72.5**	**90.8**
	3	71.4	90.4
3	1	71.4	90.4
	2	72.1	90.5
	3	72.1	90.5

(a) Projector and predictor depth (i.e. the number of Linear layers).

Projector g_θ output dim	Top-1	Top-5
16	69.9$\pm$0.3	89.9
32	71.3	90.6
64	72.2	90.9
128	72.5	91.0
256	72.5	90.8
512	**72.6**	**91.0**

(b) Projection dimension.

Details

Learning rate	Top-1	Top-5
0.01	34.8±3.0	60.8±3.2
0.1	65.0	87.0
0.2	71.7	90.6
0.3	**72.5**	**90.8**
0.4	72.3	90.6
0.5	71.5	90.1
1	69.4	89.2

(a) Base learning rate.

Weight decay coefficient	Top-1	Top-5
$1 \cdot 10^{-7}$	72.1	90.4
$5 \cdot 10^{-7}$	**72.6**	**91.0**
$1 \cdot 10^{-6}$	72.5	90.8
$5 \cdot 10^{-6}$	71.0±0.3	90.0
$1 \cdot 10^{-5}$	69.6±0.4	89.3

(b) Weight decay.

Table 15: Effect of learning rate and weight decay. We note that BYOL's performance is quite robust within a range of hyperparameters.

Batch size	Top-1		Top-5	
	BYOL (ours)	SimCLR (repro)	BYOL (ours)	SimCLR (repro)
4096	**72.5**	67.9	**90.8**	88.5
2048	72.4	67.8	90.7	88.5
1024	72.2	67.4	90.7	88.1
512	72.2	66.5	90.8	87.6
256	71.8	64.3±2.1	90.7	86.3±1.0
128	69.6±0.5	63.6	89.6	85.9
64	59.7±1.5	59.2±2.9	83.2±1.2	83.0±1.9

Table 16: Influence of the batch size.

Pseudo Code

Algorithm 1: BYOL: **B**ootstrap **Y**our **O**wn **L**atent

Inputs :

$\mathcal{D}$, $\mathcal{T}$, and $\mathcal{T}'$	set of images and distributions of transformations
θ, f_θ, g_θ, and q_θ	initial online parameters, encoder, projector, and predictor
ξ, f_ξ, g_ξ	initial target parameters, target encoder, and target projector
optimizer	optimizer, updates online parameters using the loss gradient
K and N	total number of optimization steps and batch size
$\{\tau_k\}_{k=1}^K$ and $\{\eta_k\}_{k=1}^K$	target network update schedule and learning rate schedule

for $k = 1$ **to** K **do**

  $\mathcal{B} \leftarrow \{x_i \sim \mathcal{D}\}_{i=1}^N$ `// sample a batch of` N `images`

  for $x_i \in \mathcal{B}$ **do**

    $t \sim \mathcal{T}$ and $t' \sim \mathcal{T}'$ `// sample image transformations`

    $z_1 \leftarrow g_\theta(f_\theta(t(x_i)))$ and $z_2 \leftarrow g_\theta(f_\theta(t'(x_i)))$ `// compute projections`

    $z_1' \leftarrow g_\xi(f_\xi(t'(x_i)))$ and $z_2' \leftarrow g_\xi(f_\xi(t(x_i)))$ `// compute target projections`

    $l_i \leftarrow -2 \cdot \left(\frac{\langle q_\theta(z_1), z_1' \rangle}{\|q_\theta(z_1)\|_2 \cdot \|z_1'\|_2} + \frac{\langle q_\theta(z_2), z_2' \rangle}{\|q_\theta(z_2)\|_2 \cdot \|z_2'\|_2} \right)$ `// compute the loss for` x_i

  end

  $\delta\theta \leftarrow \frac{1}{N} \sum_{i=1}^{N} \partial_\theta l_i$ `// compute the total loss gradient w.r.t.` θ

  $\theta \leftarrow \text{optimizer}(\theta, \delta\theta, \eta_k)$ `// update online parameters`

  $\xi \leftarrow \tau_k \xi + (1 - \tau_k)\theta$ `// update target parameters`

end

Output : encoder f_θ

Experiments - Linear Evaluation

Method	Top-1	Top-5
Local Agg.	60.2	-
PIRL [32]	63.6	-
CPC v2 [29]	63.8	85.3
CMC [11]	66.2	87.0
SimCLR [8]	69.3	89.0
MoCo v2 [34]	71.1	-
InfoMin Aug. [12]	73.0	91.1
BYOL (ours)	**74.3**	**91.6**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1	Top-5
SimCLR [8]	ResNet-50 (2×)	94M	74.2	92.0
CMC [11]	ResNet-50 (2×)	94M	70.6	89.7
BYOL (ours)	ResNet-50 (2×)	94M	77.4	93.6
CPC v2 [29]	ResNet-161	305M	71.5	90.1
MoCo [9]	ResNet-50 (4×)	375M	68.6	-
SimCLR [8]	ResNet-50 (4×)	375M	76.5	93.2
BYOL (ours)	ResNet-50 (4×)	375M	78.6	94.2
BYOL (ours)	ResNet-200 (2×)	250M	**79.6**	**94.8**

(b) Other ResNet encoder architectures.

Table 1: Top-1 and top-5 accuracies (in %) under linear evaluation on ImageNet.

Experiments - Semi-supervised

Method	Top-1		Top-5	
	1%	10%	1%	10%
Supervised [64]	25.4	56.4	48.4	80.4
InstDisc	-	-	39.2	77.4
PIRL [32]	-	-	57.2	83.8
SimCLR [8]	48.3	65.6	75.5	87.8
BYOL (ours)	**53.2**	**68.8**	**78.4**	**89.0**

(a) ResNet-50 encoder.

Method	Architecture	Param.	Top-1		Top-5	
			1%	10%	1%	10%
CPC v2 [29]	ResNet-161	305M	-	-	77.9	91.2
SimCLR [8]	ResNet-50 (2×)	94M	58.5	71.7	83.0	91.2
BYOL (ours)	ResNet-50 (2×)	94M	62.2	73.5	84.1	91.7
SimCLR [8]	ResNet-50 (4×)	375M	63.0	74.4	85.8	92.6
BYOL (ours)	ResNet-50 (4×)	375M	69.1	75.7	87.9	92.5
BYOL (ours)	ResNet-200 (2×)	250M	**71.2**	**77.7**	**89.5**	**93.7**

(b) Other ResNet encoder architectures.

Table 2: Semi-supervised training with a fraction of ImageNet labels.

Experiments - Transfer, Classification

Method	Food101	CIFAR10	CIFAR100	Birdsnap	SUN397	Cars	Aircraft	VOC2007	DTD	Pets	Caltech-101	Flowers
Linear evaluation:												
BYOL (ours)	**75.3**	91.3	**78.4**	**57.2**	**62.2**	**67.8**	60.6	82.5	75.5	90.4	94.2	**96.1**
SimCLR (repro)	72.8	90.5	74.4	42.4	60.6	49.3	49.8	81.4	**75.7**	84.6	89.3	92.6
SimCLR [8]	68.4	90.6	71.6	37.4	58.8	50.3	50.3	80.5	74.5	83.6	90.3	91.2
Supervised-IN [8]	72.3	**93.6**	78.3	53.7	61.9	66.7	**61.0**	**82.8**	74.9	**91.5**	**94.5**	94.7
Fine-tuned:												
BYOL (ours)	**88.5**	**97.8**	86.1	**76.3**	63.7	91.6	**88.1**	**85.4**	**76.2**	91.7	**93.8**	97.0
SimCLR (repro)	87.5	97.4	85.3	75.0	63.9	91.4	87.6	84.5	75.4	89.4	91.7	96.6
SimCLR [8]	88.2	97.7	85.9	75.9	63.5	91.3	88.1	84.1	73.2	89.2	92.1	97.0
Supervised-IN [8]	88.3	97.5	**86.4**	75.8	**64.3**	**92.1**	86.0	85.0	74.6	**92.1**	93.3	**97.6**
Random init [8]	86.9	95.9	80.2	76.1	53.6	91.4	85.9	67.3	64.8	81.5	72.6	92.0

Table 3: Transfer learning results from ImageNet (IN) with the standard ResNet-50 architecture.

Experiments - Transfer, Others

Method	AP_{50}	mIoU
Supervised-IN [9]	74.4	74.4
MoCo [9]	74.9	72.5
SimCLR (repro)	75.2	75.2
BYOL (ours)	**77.5**	**76.3**

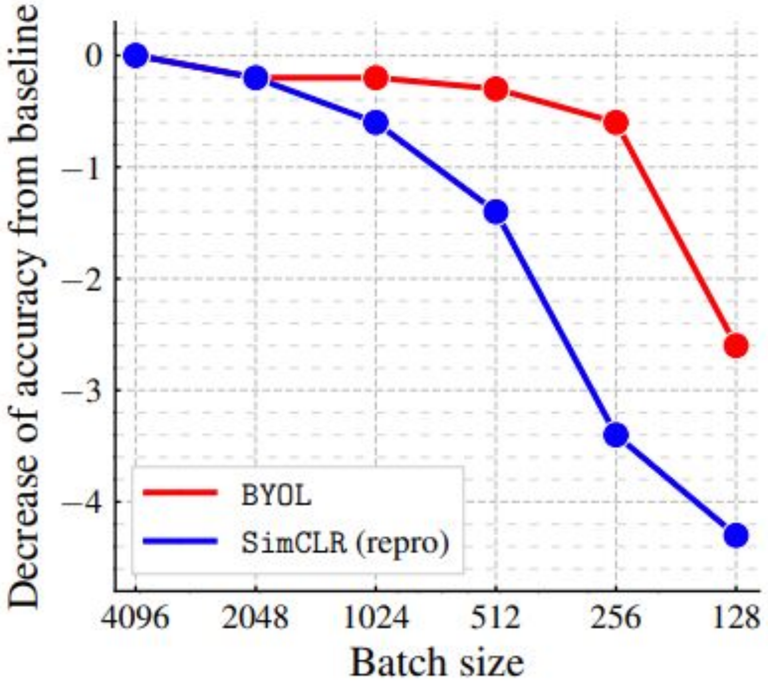
(a) Transfer results in semantic segmentation and object detection.

	Higher better			Lower better	
Method	pct.<1.25	pct.$<1.25^2$	pct.$<1.25^3$	rms	rel
Supervised-IN [70]	81.1	95.3	98.8	0.573	**0.127**
SimCLR (repro)	83.3	96.5	99.1	0.557	0.134
BYOL (ours)	**84.6**	**96.7**	**99.1**	**0.541**	0.129

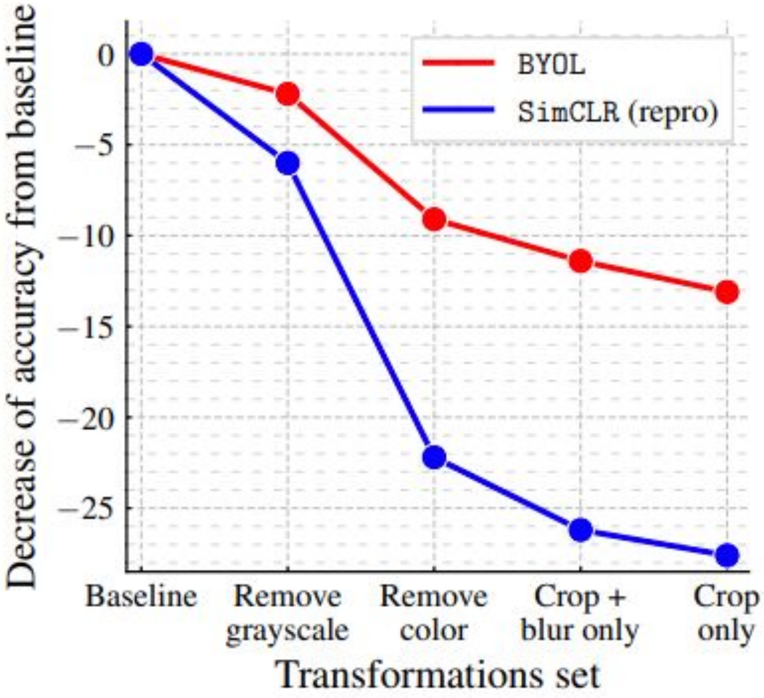
(b) Transfer results on NYU v2 depth estimation.

Table 4: Results on transferring BYOL's representation to other vision tasks.

Experiments - Batch, Transformation



(a) Impact of batch size



(b) Impact of progressively removing transformations

Experiments - Hyperparameters

Target	τ_{base}	Top-1
Constant random network	1	$18.8{\pm}0.7$
Moving average of online	0.999	69.8
Moving average of online	0.99	**72.5**
Moving average of online	0.9	68.4
Stop gradient of online[†]	0	0.3

(a) Results for different target modes. [†]In the *stop gradient of online*, $\tau = \tau_{\text{base}} = 0$ is kept constant throughout training.

Method	Predictor	Target network	β	Top-1
`BYOL`	✓	✓	0	**72.5**
	✓	✓	1	70.9
		✓	1	70.7
`SimCLR`			1	69.4
	✓		1	69.1
	✓		0	0.3
		✓	0	0.2
			0	0.1

(b) Intermediate variants between `BYOL` and `SimCLR`.

Table 5: Ablations with top-1 accuracy (in %) at 300 epochs under linear evaluation on ImageNet.

Appendix. InfoNCE

$$S_\theta(u_1, u_2) \triangleq \frac{\langle \phi(u_1), \psi(u_2) \rangle}{\|\phi(u_1)\|_2 \cdot \|\psi(u_2)\|_2}.$$

$$\text{InfoNCE}_\theta \triangleq \frac{2}{B} \sum_{i=1}^{B} S_\theta(v_i, v_i') - \beta \cdot \frac{2\alpha}{B} \sum_{i=1}^{B} \ln \left(\sum_{j \neq i} \exp \frac{S_\theta(v_i, v_j)}{\alpha} + \sum_{j} \exp \frac{S_\theta(v_i, v_j')}{\alpha} \right),$$

- NCE : Noise-contrastive estimation: A new estimation principle for unnormalized statistical models (jmlr 2010)
- Learning word embeddings efficiently with noise-contrastive estimation (NIPS 2013)

Experiments - Normalization

Normalization	Top-1	Top-5
ℓ_2-norm	**72.5**	**90.8**
LAYERNORM	72.5±0.4	90.1
No normalization	67.4	87.1
BATCHNORM	65.3	85.3

$$n^j_{\mathrm{BN}_i} : x \to \frac{x^j_i - \mu^j_{\mathrm{BN}}(x)}{\sigma^j_{\mathrm{BN}}(x)\cdot\sqrt{d}}, \quad n^j_{\mathrm{LN}_i} : x \to \frac{x^j_i - \mu_{\mathrm{LN}_i}(x)}{\sigma_{\mathrm{LN}_i}(x)\cdot\sqrt{d}}, \quad n_{\mathrm{ID}} : x \to x,$$

$$\mu^j_{\mathrm{BN}} : x \to \frac{1}{B}\sum_{i=1}^{B} x^j_i, \quad \sigma^j_{\mathrm{BN}} : x \to \sqrt{\frac{1}{B}\sum_{i=1}^{B}\left(x^j_i\right)^2 - \mu^j_{\mathrm{BN}}(x)^2},$$

$$\mu_{\mathrm{LN}_i} : x \to \frac{1}{d}\sum_{j=1}^{d} x^j_i, \quad \sigma_{\mathrm{LN}_i} : x \to \frac{\|x_i - \mu_{\mathrm{LN}_i}(x)\|_2}{\sqrt{d}}$$

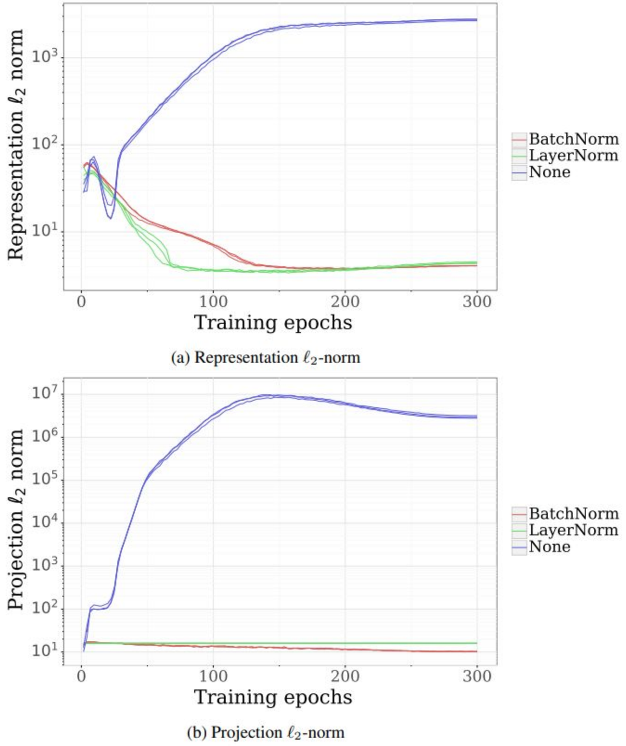


Figure 6: Effect of normalization on the ℓ_2 norm of network outputs.

SimSiam:
Exploring Simple Siamese Representation Learning

https://arxiv.org/pdf/2011.10566.pdf

Idea

General idea behind this paper is to prove that all contrastive learning method are result of siamese network in one way or other and all other techniques used in MoCo, BYOL, SimCLR or SwAV are just design choices.

SimSam just uses stop gradient in order to train a good enough contrastive model with far less batch size.

Focus of paper

This paper focuses on employing simple Siamese networks to learn meaningful representation even in the absence of

1) negative sample pairs (SimCLR)
2) large batches
3) momentum encoders

They show collapsing solutions do exist for the loss and structure, but a stop-gradient operation plays an essential role in preventing collapsing.

Dissimilarities with other CL methods

SimSam can be thought of as

1) BYOL without the momentum encoder

2) SimCLR without negative pairs

3) SwAV without online clustering

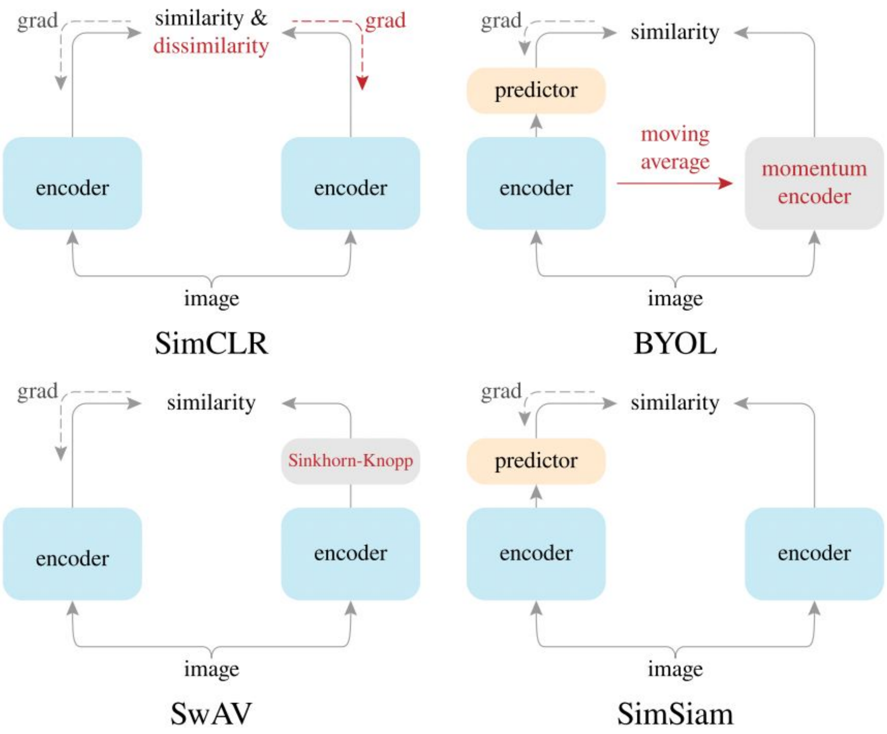


Figure 3. **Comparison on Siamese architectures**. The encoder includes all layers that can be shared between both branches. The dash lines indicate the gradient propagation flow. In BYOL, SwAV, and SimSiam, the lack of a dash line implies stop-gradient, and their symmetrization is not illustrated for simplicity. The components in red are those missing in SimSiam.

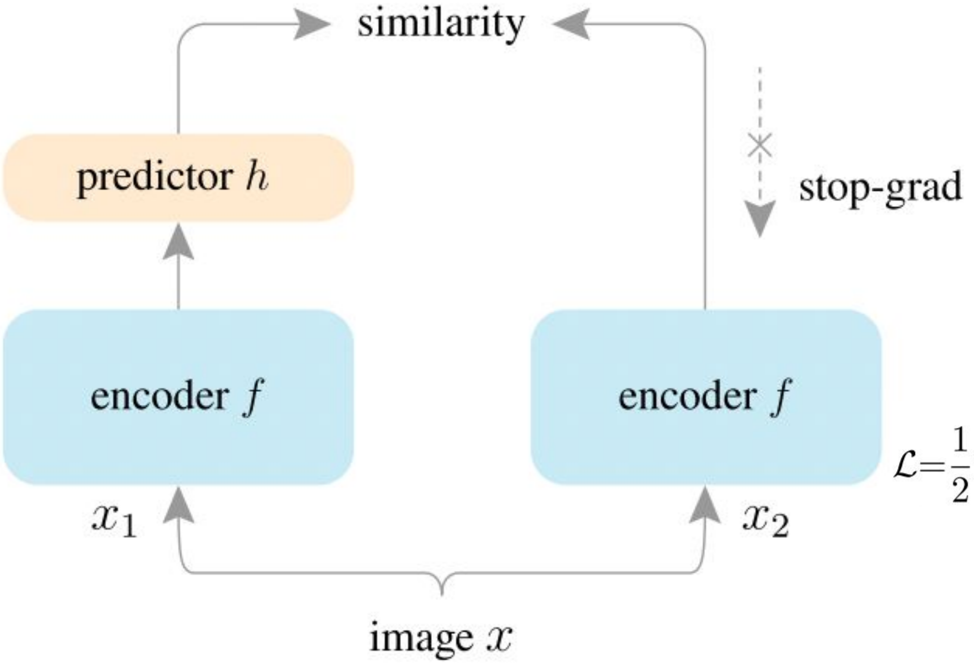
Finding

stop-gradient operation is critical.

This finding can be obscured with the usage of a momentum encoder, which is always accompanied with stop-gradient (as it is not updated by its parameters' gradients).

While the moving-average behavior may improve accuracy with an appropriate momentum coefficient, our experiments show that it is not directly related to preventing collapsing.

Architecture



$$p_1 \triangleq h(f(x_1)) \text{ and } z_2 \triangleq f(x_2),$$

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

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

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

The encoder on x_2 receives no gradient from z_2 in the first term, but it receives gradients from p_2 in the second term (and vice versa for x_1).

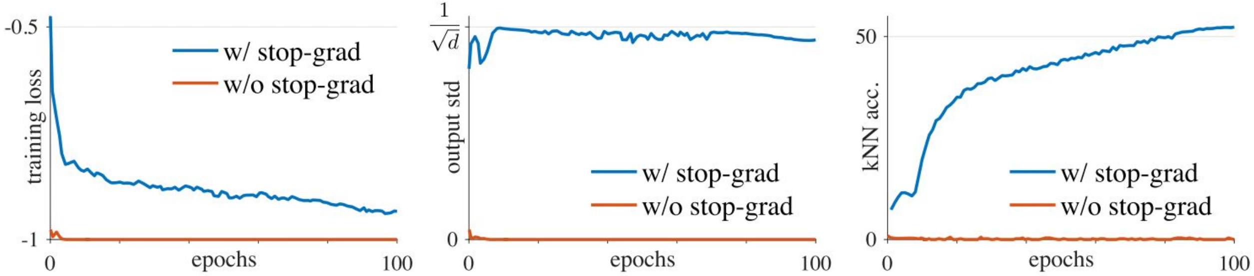
x_1 is firstly fed to trainable encoder and then x_2 is fed to it in one training step.

They also show doing the training way boosts the accuracy. They also trying using asymmetric loss by sampling two pairs for each image in the asymmetric version ("2×"). It makes the gap smaller.

	sym.	asym.	asym. 2×
acc. (%)	68.1	64.8	67.3

Training with and without stop gradient

Without stop-gradient, the optimizer quickly finds a degenerated solution and reaches the minimum possible loss of−1.



	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 [34] as a monitor of progress. **Table**: ImageNet linear evaluation ("w/ stop-grad" is mean±std over 5 trials).

Clustering way of looking at SimSam

F is a network parameterized by θ. T is the augmentation. x is an image. The expectation E[·] is over the distribution of images and augmentations.

$$\mathcal{L}(\theta, \eta) = \mathbb{E}_{x,\mathcal{T}}\left[\left\|\mathcal{F}_\theta(\mathcal{T}(x)) - \eta_x\right\|_2^2\right].$$

ηx is the representation of the image x, η is not necessarily the output of a network; it is the argument of an optimization problem

$$\min_{\theta,\eta} \mathcal{L}(\theta, \eta).$$

The variable θ is analogous to the clustering centers: it is the learnable parameters of an encoder. The variable ηx is analogous to the assignment vector of the sample x (a one-hot vector in k- means): it is the representation of x.

they alternate between these sub-problems:

$$\begin{aligned} \theta^t &\leftarrow \arg\min_{\theta} \mathcal{L}(\theta, \eta^{t-1}) \\ \eta^t &\leftarrow \arg\min_{\eta} \mathcal{L}(\theta^t, \eta) \end{aligned}$$

Solving for θ. use SGD to solve the sub-problem, η_{t-1} which is a constant in this subproblem.

Solving for η. The sub-problem can be solved independently for each ηx.

$$\mathbb{E}_{\mathcal{T}}\Big[\|\mathcal{F}_{\theta^t}(\mathcal{T}(x)) - \eta_x\|_2^2\Big]$$

$$\eta_x^t \leftarrow \mathbb{E}_{\mathcal{T}}\Big[\mathcal{F}_{\theta^t}(\mathcal{T}(x))\Big].$$

Alteration:

$$\theta^{t+1} \leftarrow \arg\min_{\theta} \mathbb{E}_{x,\mathcal{T}}\Big[\big\|\mathcal{F}_{\theta}(\mathcal{T}(x)) - \mathcal{F}_{\theta^t}(\mathcal{T}'(x))\big\|_2^2\Big]$$

	1-step	10-step	100-step	1-epoch
acc. (%)	68.1	68.7	68.9	67.0

Results

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

Table 4. **Comparisons on ImageNet linear classification**. All are based on **ResNet-50** pre-trained with **two 224×224 views**. Evaluation is on a single crop. All competitors are from our reproduction, and "+" denotes *improved* reproduction *vs*. original papers (see supplement).

	VOC 07 detection			VOC 07+12 detection			COCO detection			COCO instance seg.		
pre-train	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 [30] 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.