Self-Supervised Learning

韩坤洋

Self-Supervised Learning

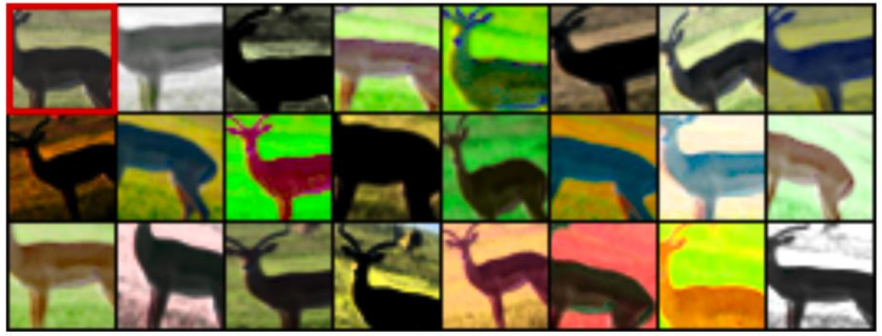
Segmentation

- Task: predict logits mask (supervised)
  - Image -> Semantic information -> logits mask prediction

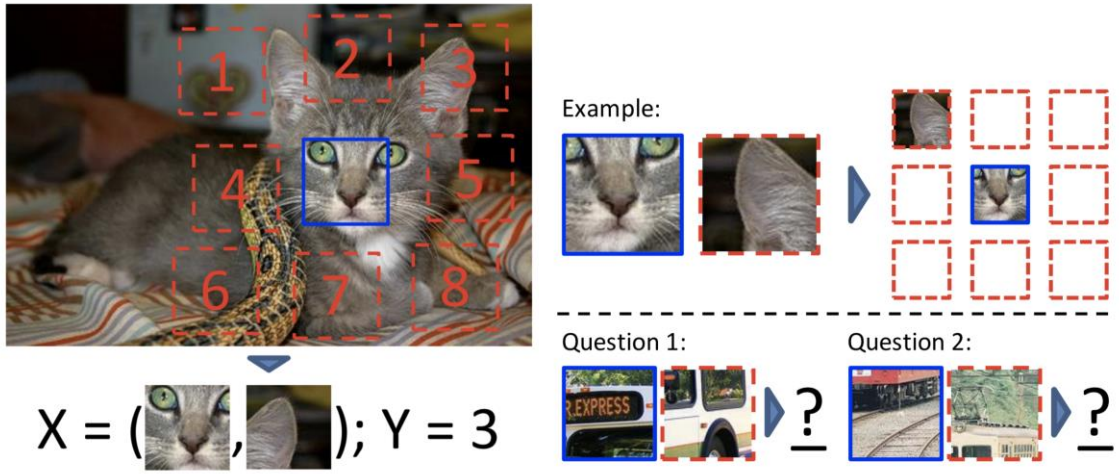
- **Self-supervised task**, in unlabeled dataset
  - Image -> Semantic information
- Predict logits mask, use conv layers
  - Semantic information -> logits mask prediction

Self-Supervised Learning

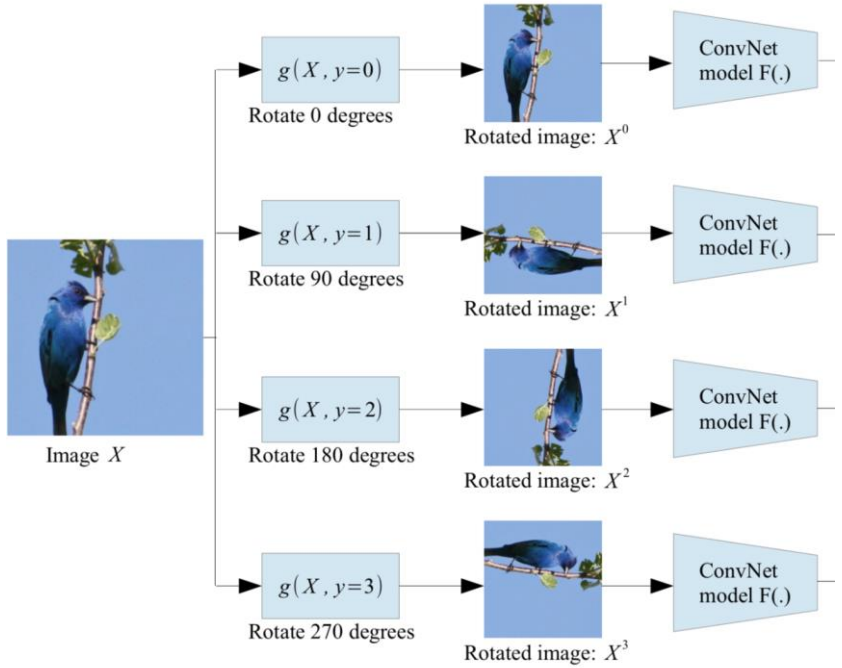
- Distortion



- Patches



- Rotation

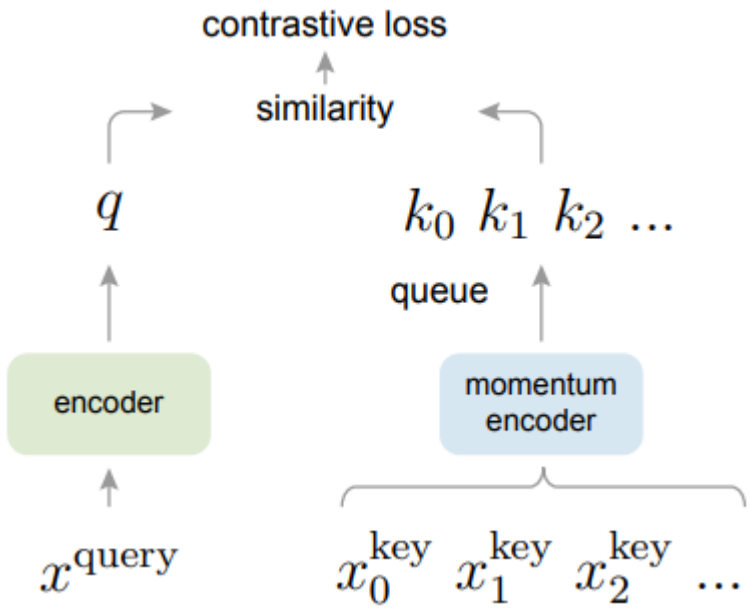


Paper list

- Momentum Contrast for Unsupervised Visual Representation Learning
- Improved Baselines with Momentum Contrastive Learning
- Propagate Yourself: Exploring Pixel-Level Consistency for Unsupervised Visual Representation Learning
- Exploring Simple Siamese Representation Learning

Momentum Contrast for Unsupervised Visual Representation Learning

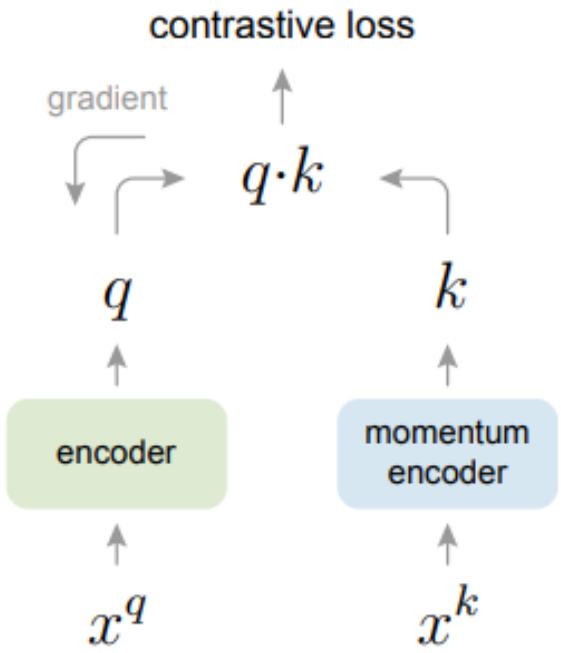
- Mechanism for building dynamic dictionaries for contrastive learning



MoCo - Dictionary Look-up

- Keys in the dictionaries
  - Sample from data, images or patches
  - Represented by encoder network
- Encoded ‘query’
  - Similar to its matching ‘key’
  - Dissimilar to others
- Loss

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$

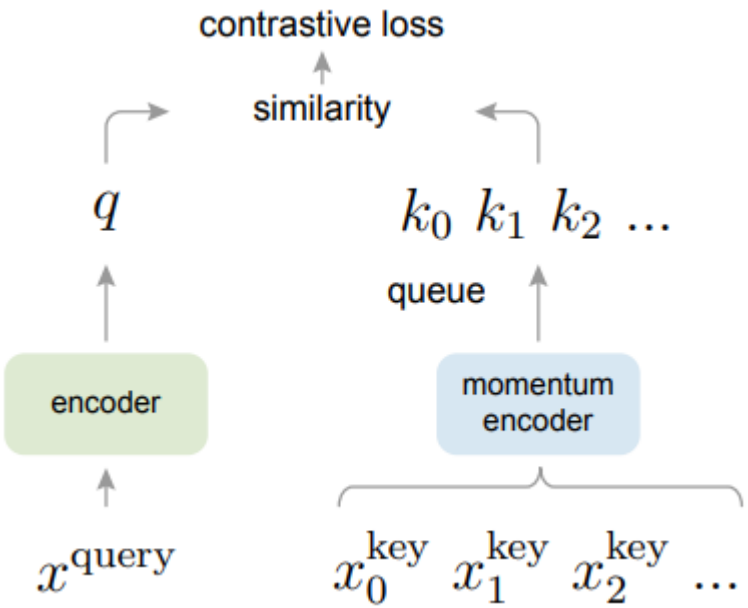


MoCo - Momentum Contrast

- Dictionary as a queue
  - Dictionary can be much larger than batch size
  - Current enqueued, oldest dequeued

- Momentum update
  - Hard to train key encoder, because large dict
  - Update key encoder param by:

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



MoCo - Details

- ResNet as backbone
  - 128-D vector as output, L2-norm
- Transform
  - Resize and 224 x 224-pixel crop
  - Color jitter, horizontal flip, grayscale conversion
- Shuffling BN, BN performs bad
  - Intra-batch communication leaks information
  - Shuffle the sample order before key encoder

MoCo - Details

- Dataset
  - ImageNet-1M, 1.28 million images, 1000 classes
  - Instagram-1B, 940 million images, 1500 hashtags
- Train
  - SGD
  - IN-1M, 256 batch size in 8 GPUs, 53 hours
  - IG-1B, 1024 batch size in 64 GPUs, 6 days

MoCo - Experiment

method	architecture	#params (M)	accuracy (%)
Exemplar [17]	R50w3×	211	46.0 [38]
RelativePosition [13]	R50w2×	94	51.4 [38]
Jigsaw [45]	R50w2×	94	44.6 [38]
Rotation [19]	Rv50w4×	86	55.4 [38]
Colorization [64]	R101*	28	39.6 [14]
DeepCluster [3]	VGG [53]	15	48.4 [4]
BigBiGAN [16]	R50	24	56.6
	Rv50w4×	86	61.3
methods based on contrastive learning follow:			
InstDisc [61]	R50	24	54.0
LocalAgg [66]	R50	24	58.8
CPC v1 [46]	R101*	28	48.7
CPC v2 [35]	$R170^*_{wider}$	303	65.9
CMC [56]	$R50_{L+ab}$	47	64.1†
	$R50w2\times_{L+ab}$	188	68.4†
AMDIM [2]	$AMDIM_{small}$	194	63.5†
	$AMDIM_{large}$	626	68.1†
MoCo	R50	24	60.6
	RX50	46	63.9
	R50w2×	94	65.4
	R50w4×	375	**68.6**

Table 1. **Comparison under the linear classification protocol on ImageNet**. The figure visualizes the table. All are reported as unsupervised pre-training on the ImageNet-1M training set, followed by supervised linear classification trained on frozen features, evaluated on the validation set. The parameter counts are those of the feature extractors. We compare with improved re-implementations if available (referenced after the numbers).

Notations: R101*/R170* is ResNet-101/170 with the last residual stage removed [14, 46, 35], and R170 is made wider [35]; Rv50 is a reversible net [23], RX50 is ResNeXt-50-32×8d [62].

†: *Pre-training uses FastAutoAugment [40] that is supervised by ImageNet labels.*

MoCo - Experiment

pre-train	AP_{50}	AP	AP_{75}
random init.	64.4	37.9	38.6
super. IN-1M	81.4	54.0	59.1
MoCo IN-1M	81.1 (−0.3)	54.6 (+**0.6**)	59.9 (+**0.8**)
MoCo IG-1B	81.6 (+0.2)	55.5 (+**1.5**)	61.2 (+**2.1**)

(a) Faster R-CNN, R50-**dilated-C5**

pre-train	AP_{50}	AP	AP_{75}
random init.	60.2	33.8	33.1
super. IN-1M	81.3	53.5	58.8
MoCo IN-1M	81.5 (+0.2)	55.9 (+**2.4**)	62.6 (+**3.8**)
MoCo IG-1B	82.2 (+**0.9**)	57.2 (+**3.7**)	63.7 (+**4.9**)

(b) Faster R-CNN, R50-**C4**

Table 2. **Object detection fine-tuned on PASCAL VOC** `trainval07+12`. Evaluation is on `test2007`: AP_{50} (default VOC metric), AP (COCO-style), and AP_{75}, averaged over 5 trials. All are fine-tuned for 24k iterations (~23 epochs). In the brackets are the gaps to the ImageNet supervised pre-training counterpart. In green are the gaps of at least +**0.5** point.

Improved Baselines with Momentum Contrastive Learning

- Implement SimCLR’s design in the MoCo framework
  - MLP head
  - Data augmentation
  - Cosine lr schedule

MoCo v2 - MLP head

- Replace fc head with a 2-layer MLP head
- 2048-d hidden layer
- Only influences unsupervised training stage

τ	0.07	0.1	0.2	0.3	0.4	0.5
w/o MLP	60.6	**60.7**	59.0	58.2	57.2	56.4
w/ MLP	62.9	64.9	**66.2**	65.7	65.0	64.3

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$

MoCo v2 - Augmentation

- Blur augmentation
- Color jitter (strengthened -> random)

case	unsup. pre-train MLP	aug+	cos	epochs	ImageNet acc.	VOC detection AP_{50}	AP	AP_{75}
supervised					76.5	81.3	53.5	58.8
MoCo v1				200	60.6	81.5	55.9	62.6
(a)	✓			200	66.2	82.0	56.4	62.6
(b)		✓		200	63.4	82.2	56.8	63.2
(c)	✓	✓		200	67.3	**82.5**	57.2	63.9
(d)	✓	✓	✓	200	67.5	82.4	57.0	63.6
(e)	✓	✓	✓	**800**	**71.1**	**82.5**	**57.4**	**64.0**

MoCo v2 - Comparison

case	unsup. pre-train MLP	aug+	cos	epochs	batch	ImageNet acc.
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	**67.5**
*results of **longer** unsupervised training follow:*						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	**71.1**

Table 2. **MoCo *vs.* SimCLR**: ImageNet linear classifier accuracy (**ResNet-50, 1-crop 224×224**), trained on features from unsupervised pre-training. "aug+" in SimCLR includes blur and stronger color distortion. SimCLR ablations are from Fig. 9 in [2] (we thank the authors for providing the numerical results).

Propagate Yourself: Exploring Pixel-Level Consistency for Unsupervised Visual Representation Learning

- The first task directly applies contrastive learning at the **pixel level**.

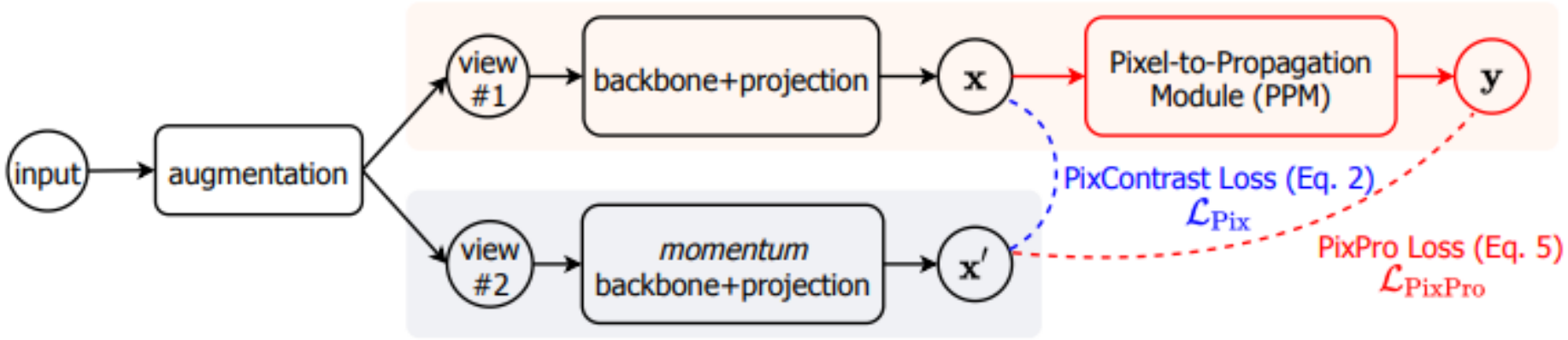
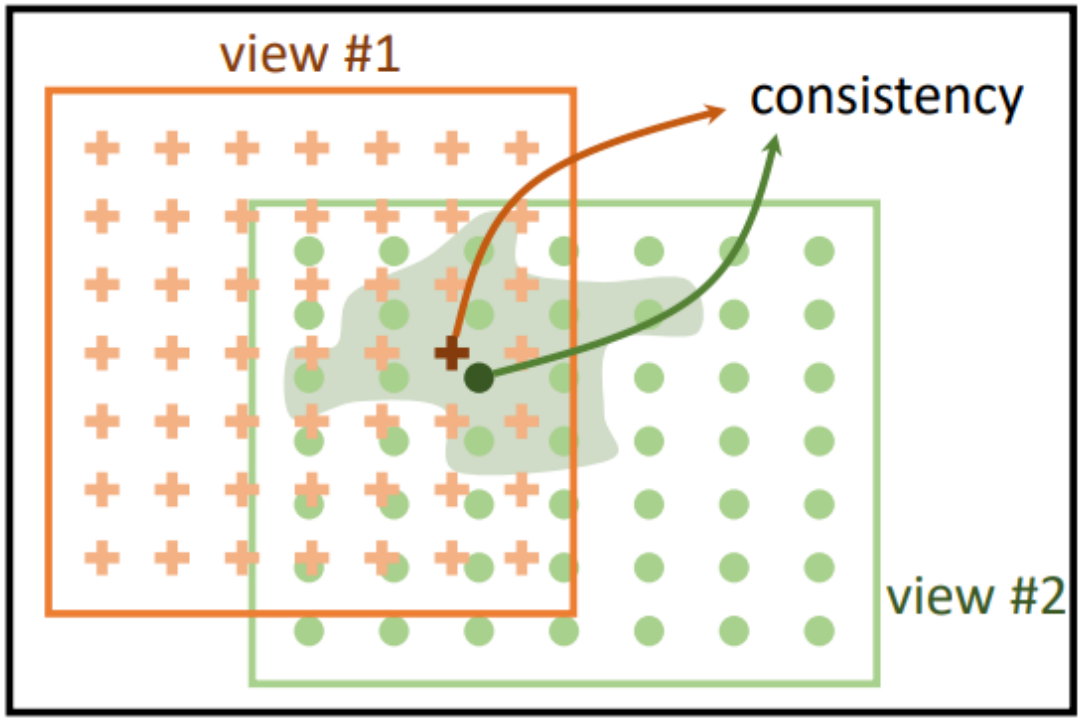


Figure 2. Architecture of the *PixContrast* and *PixPro* methods.

PixPro – Pixel Contrast

- Compute pairs of pixels, from two views
- Contrastive loss

$$\mathcal{L}_{\text{Pix}}(i) = -\log \frac{\sum_{j \in \Omega_p^i} e^{\cos\left(\mathbf{x}_i, \mathbf{x}'_j\right)/\tau}}{\sum_{j \in \Omega_p^i} e^{\cos\left(\mathbf{x}_i, \mathbf{x}'_j\right)/\tau} + \sum_{k \in \Omega_n^i} e^{\cos\left(\mathbf{x}_i, \mathbf{x}'_k\right)/\tau}},$$



PixPro – Pixel-to-Propagation Consistency

- Spatial sensitivity
  - Discriminate spatially close pixels
  - PixContrast
- Spatial smoothness
  - Encourages spatially close pixels to be similar

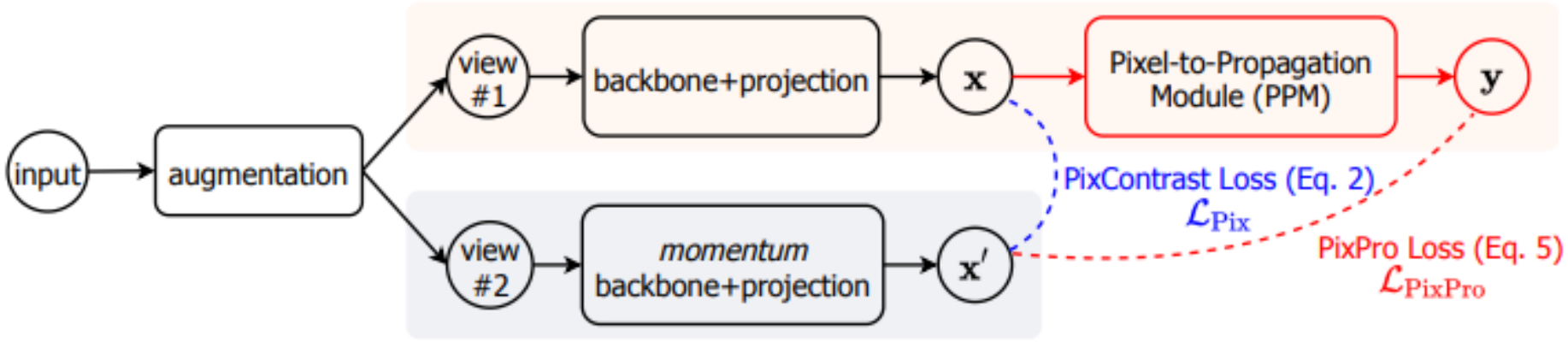


Figure 2. Architecture of the *PixContrast* and *PixPro* methods.

ProPix - PPM

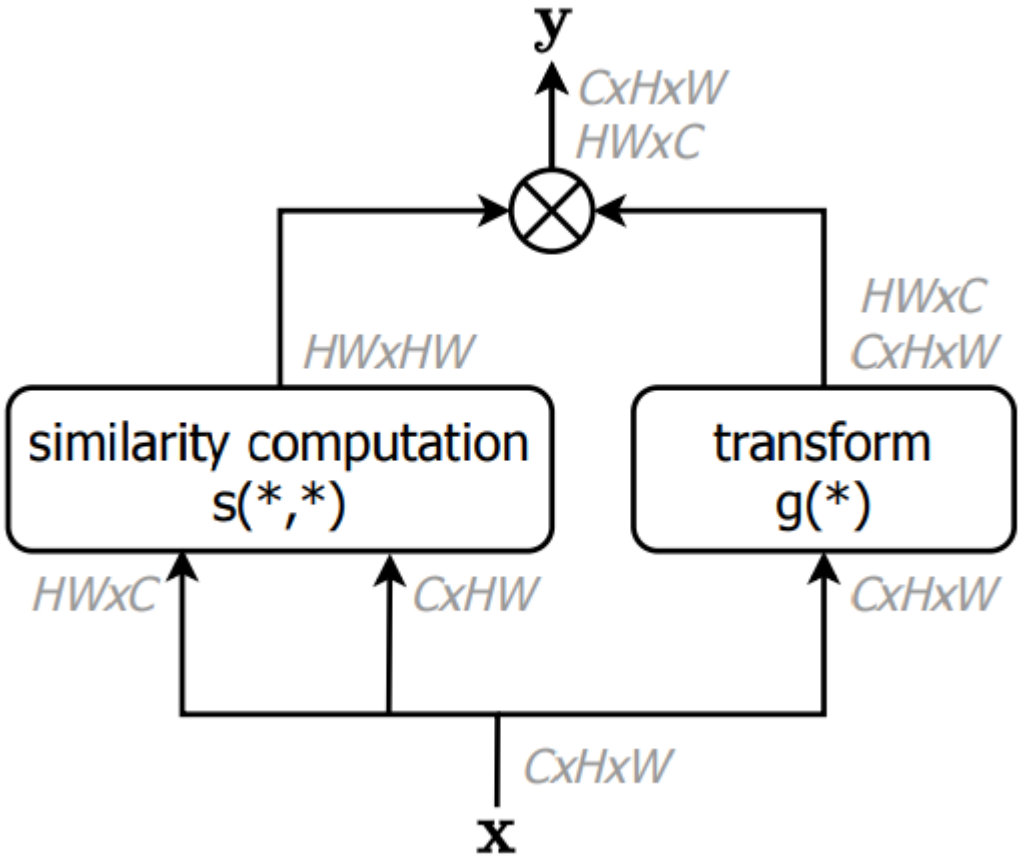
- PPM function

$$\mathbf{y}_i = \Sigma_{j \in \Omega} s(\mathbf{x}_i, \mathbf{x}_j) \cdot g(\mathbf{x}_j),$$

where $s(\cdot, \cdot)$ is a similarity function defined as

$$s(\mathbf{x}_i, \mathbf{x}_j) = (\max(\cos(\mathbf{x}_i, \mathbf{x}_j), 0))^{\gamma},$$

- g(*)
  - L conv layers with BN and ReLU



PixPro - PPC Loss

$$\mathcal{L}_{\text{PixPro}} = -\cos(\mathbf{y}_i, \mathbf{x}'_j) - \cos(\mathbf{y}_j, \mathbf{x}'_i),$$

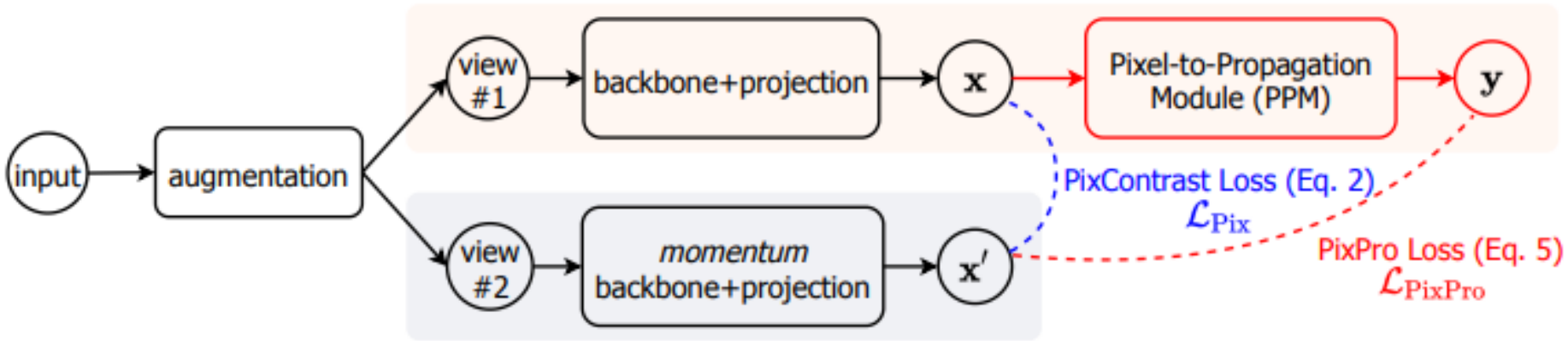


Figure 2. Architecture of the *PixContrast* and *PixPro* methods.

PixPro - Experiment

PixPro (pixel)	SimCLR* (instance)	VOC	COCO	ImageNet
		AP	mAP	top-1 acc
✓		58.8	40.8	55.1
	✓	53.4	40.5	65.4
✓	✓	58.7	40.9	66.3

Table 4. Transfer performance of combining a pixel-level and an instance-level method. "SimCLR*" denotes a variant of SimCLR with the same encoders as our pixel-level approach. 100 epoch pre-training is adopted for all experiments.

+FPN	+head	+instance	COCO (FCOS)		
			mAP	AP_{50}	AP_{75}
			37.8	56.2	40.6
✓			38.1	56.7	41.2
✓	✓		38.6	57.3	41.5
✓	✓	✓	**39.8**	**58.4**	**42.7**

Table 5. FPN and head pre-training with transfer to COCO using an FCOS detector [36]. 100 epoch pre-training is adopted for all experiments.

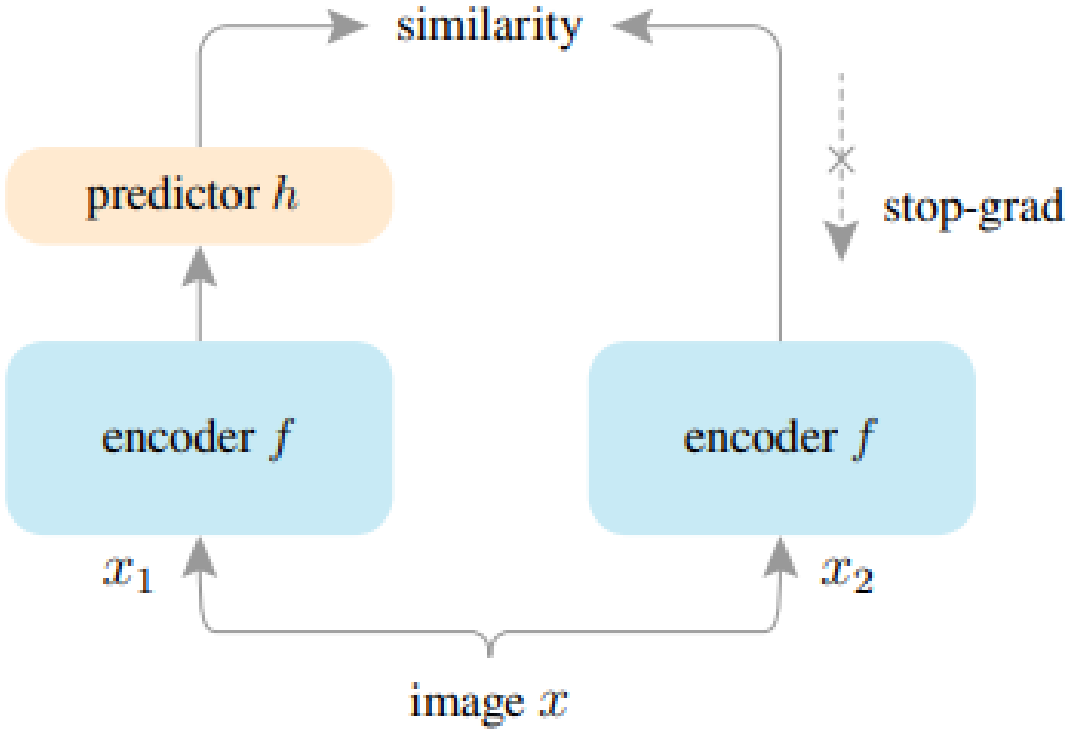
PixPro - Experiment

Method	#. Epoch	Pascal VOC (R50-C4)			COCO (R50-FPN)			COCO (R50-C4)			Cityscapes (R50)
		AP	AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}	mAP	AP_{50}	AP_{75}	mIoU
scratch	-	33.8	60.2	33.1	32.8	51.0	35.3	26.4	44.0	27.8	65.3
supervised	100	53.5	81.3	58.8	39.7	59.5	43.3	38.2	58.2	41.2	74.6
MoCo [19]	200	55.9	81.5	62.6	39.4	59.1	43.0	38.5	58.3	41.6	75.3
SimCLR [9]	1000	56.3	81.9	62.5	39.8	59.5	43.6	38.4	58.3	41.6	75.8
MoCo v2 [10]	800	57.6	82.7	64.4	40.4	60.1	44.3	39.5	59.0	42.6	76.2
InfoMin [35]	200	57.6	82.7	64.6	40.6	60.6	44.6	39.0	58.5	42.0	75.6
InfoMin [35]	800	57.5	82.5	64.0	40.4	60.4	44.3	38.8	58.2	41.7	75.6
PixPro (ours)	100	58.8	83.0	66.5	41.3	61.3	45.4	40.0	59.3	43.4	76.8
PixPro (ours)	400	**60.2**	**83.8**	**67.7**	**41.4**	**61.6**	**45.4**	**40.5**	**59.8**	**44.0**	**77.2**

Exploring Simple Siamese Representation Learning

SimSiam can learn meaningful representations without following:

- Negative sample pairs
- Large batches
- Momentum encoders



SimSiam

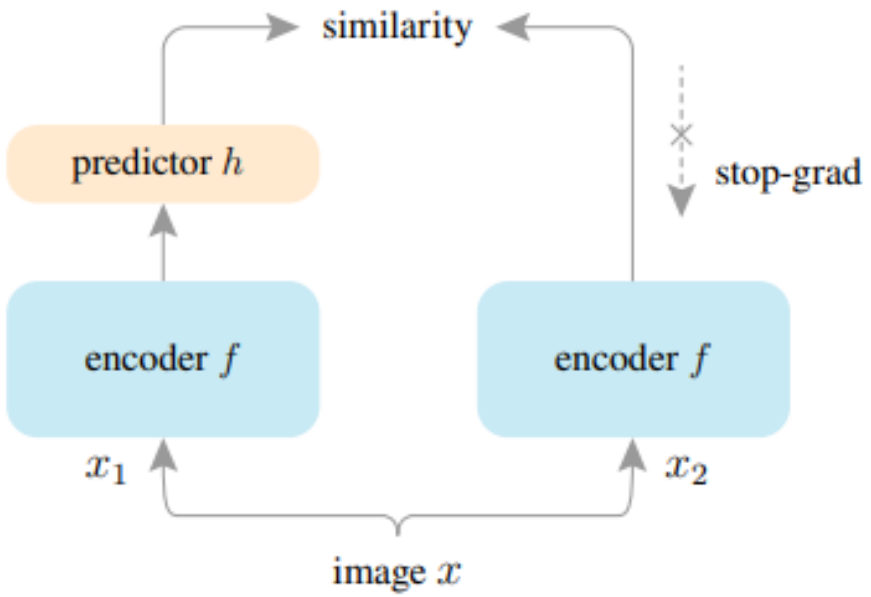
- Encoder f
  - ResNet, projection MLP head
- h
  - Prediction MLP head
- D, cosine similarity

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

Algorithm 1 SimSiam Pseudocode, PyTorch-like

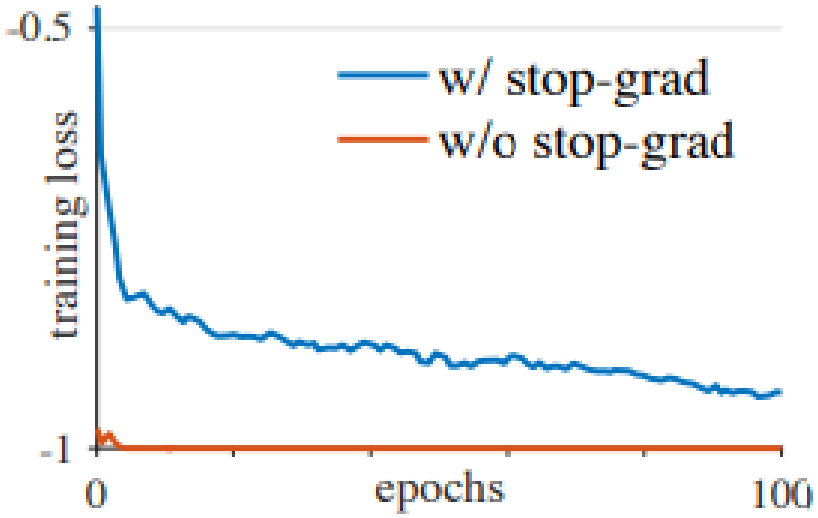
```
# f: backbone + projection mlp
# h: prediction mlp

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()
```

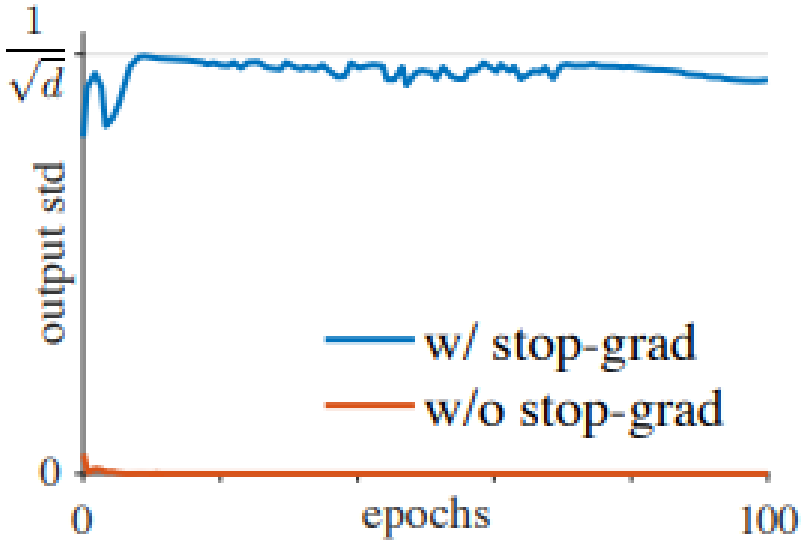
SimSiam - Stop Gradient

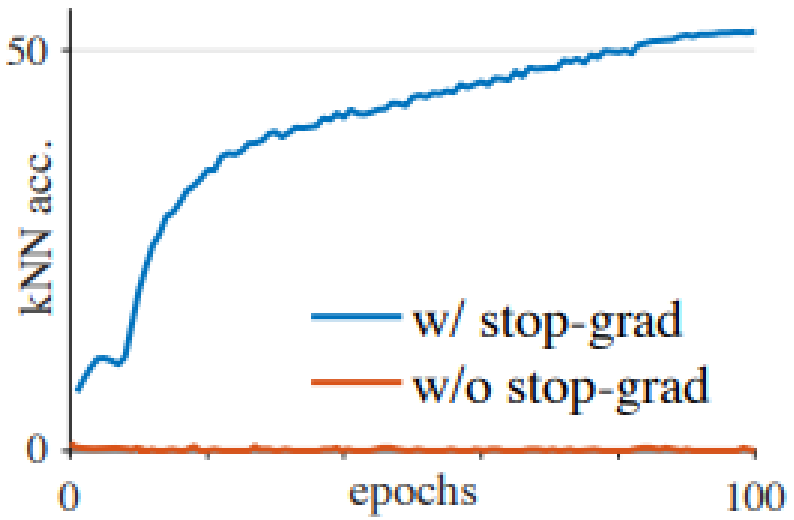


$$\mathcal{L} = \frac{1}{2}\mathcal{D}(p_1, \texttt{stopgrad}(z_2)) + \frac{1}{2}\mathcal{D}(p_2, \texttt{stopgrad}(z_1)).$$

SimSiam - Experiment







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

SimSiam - Experiment

- If h removed

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

$\mathcal{D}(z_1, z_2)$ with magnitude scaled by 1/2

  - Using stop-grad == removing stop-grad and scaling loss by 1/2

	pred. MLP h	acc. (%)
baseline	lr with cosine decay	67.7
(a)	no pred. MLP	0.1
(b)	fixed random init.	1.5
(c)	lr not decayed	68.1

SimSiam - Experiment

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

Table 2. **Effect of batch sizes** (ImageNet linear evaluation accuracy with 100-epoch pre-training).

	case	proj. MLP's BN hidden	proj. MLP's BN output	pred. MLP's BN hidden	pred. MLP's BN output	acc. (%)
(a)	none	-	-	-	-	34.6
(b)	hidden-only	✓	-	✓	-	67.4
(c)	default	✓	✓	✓	-	68.1
(d)	all	✓	✓	✓	✓	unstable

Table 3. **Effect of batch normalization on MLP heads** (ImageNet linear evaluation accuracy with 100-epoch pre-training).

SimSiam - 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

SimSiam - Experiment

	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**