Momentum Contrast for Unsupervised Visual Representation Learning

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, Ross Girshick

Facebook AI Research (FAIR)

Published in 2020 CVPR

발제자: 윤예준

https://arxiv.org/abs/1911.05722

목차

MoCo (Momentum Contrast) outline

- Dynamic Dictionary 구축
  - "This enables building a large and consistent dictionary on-the-fly that facilitates contrastive unsupervised learning."

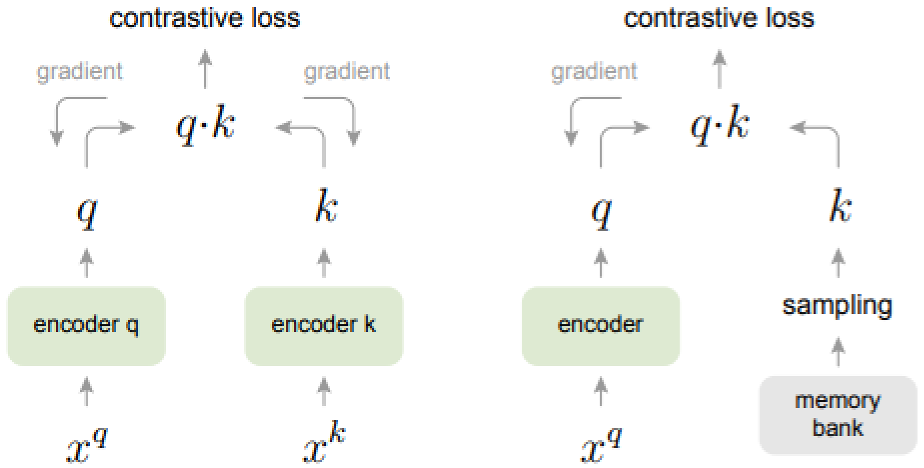
- Hypothesis
  1. "Good features can be learned by a large dictionary that covers a rich set of negative samples." => Dictionary as a queue
  2. "The encoder for the dictionary keys is kept as consistent as possible despite its evolution." => Momentum update

- InfoNCE loss 사용

- Pretext task and downstream vision tasks results

MoCo Motivate

이전 학습 방식의 문제점



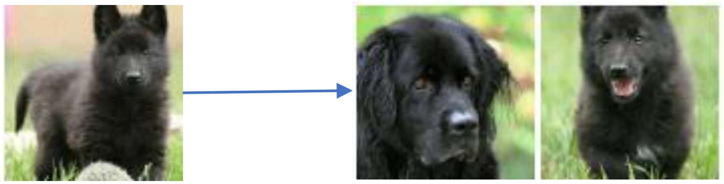
(a) end-to-end (b) memory bank

Pretext Task

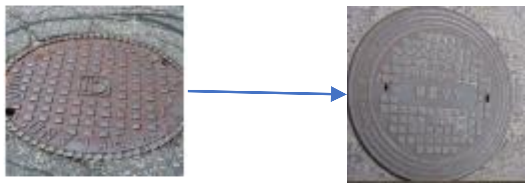
- 연구자가 직접 정의한 task

- Instance Discrimination Task[61]
  - Positive pair: same image
  - Negative pair: Otherwise

x^{query}, x^{key+}



x^{key-}



[61] https://arxiv.org/abs/1805.01978v1

InfoNCE

- NCE (Noise-Contrastive Estimation)
  - Target data를 target이 아닌 data와 비교하여 정답 확률 추정하는 방법

- InfoNCE

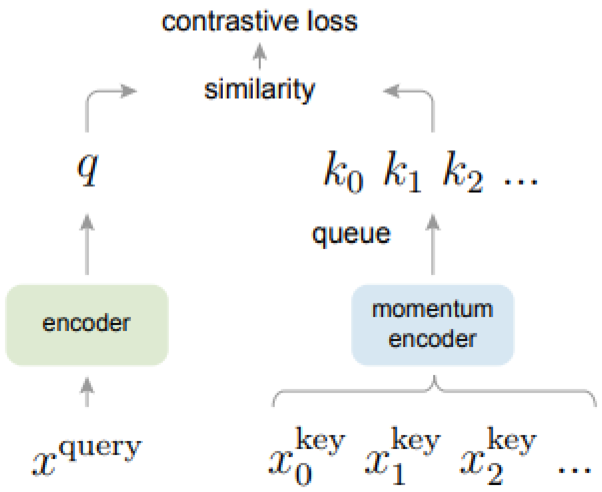
$$\mathcal{L}_q = -\log \frac{\exp(q{\cdot}k_+/\tau)}{\sum_{i=0}^{K} \exp(q{\cdot}k_i/\tau)}$$

- q = query sample
- k = Key sample
- τ = hyperparameter 0.07

https://arxiv.org/abs/1805.01978v1

Momentum Contrast – Dictionary as a queue

- Maintaining the dictionary as a queue of data samples
- Reusing the encoded keys from the immediate preceding mini-batches
- Decoupling the dictionary size from the mini-batch size
- Removing the oldest mini-batch in queue



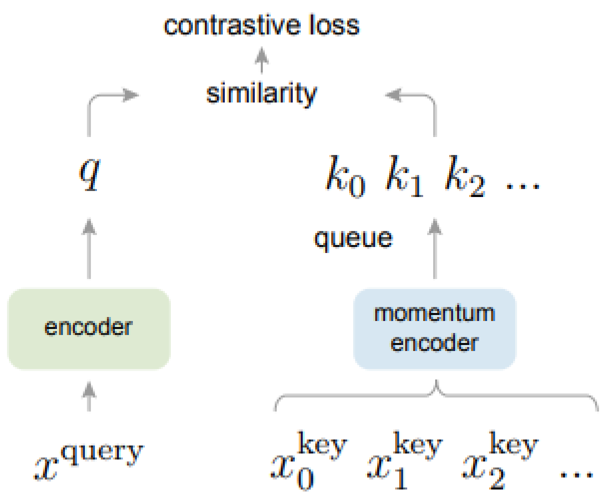
Momentum Contrast – Momentum update

Momentum update

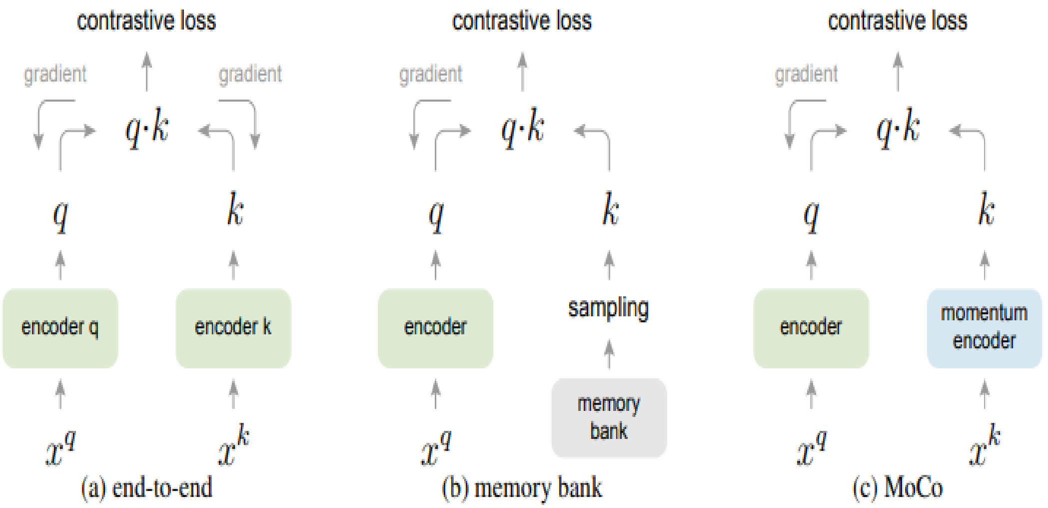
$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1-m)\theta_{\mathrm{q}}.$$

Momentum update experiment

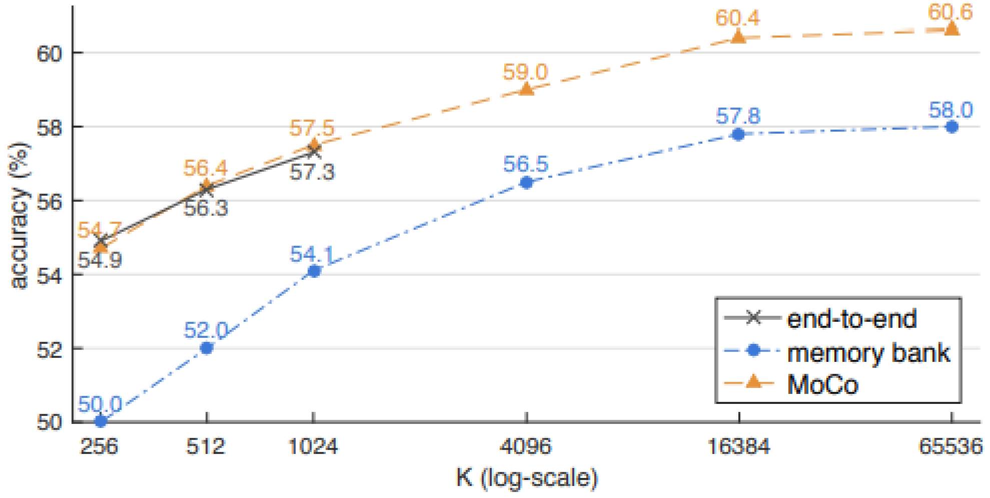
momentum m	0	0.9	0.99	0.999	0.9999
accuracy (%)	*fail*	55.2	57.8	59.0	58.9



Momentum Contrast – Relations to previous mechanisms



Comparison of three contrastive loss mechanisms under the ImageNet linear classification protocol



pre-train	R50-dilated-C5 AP_{50}	R50-dilated-C5 AP	R50-dilated-C5 AP_{75}	R50-C4 AP_{50}	R50-C4 AP	R50-C4 AP_{75}
end-to-end	79.2	52.0	56.6	80.4	54.6	60.3
memory bank	79.8	52.9	57.9	80.6	54.9	60.6
MoCo	**81.1**	**54.6**	**59.9**	**81.5**	**55.9**	**62.6**

Comparison of three contrastive loss mechanisms on PASCAL VOC object detection

https://arxiv.org/abs/1805.01978v1

Technical Details

- Encoder와 data augmentation은 NPID[61] 셋팅 사용

- ResNet
  - last fully-connected layer (after global average pooling) has a fixed-dimensional output (128-D)
  - Output vector is normalized by its L2-norm

- Data Augmentation
  - 224x224-pixel crop is taken from a randomly resize image
  - Random color jittering
  - Random horizontal flip
  - Random grayscale conversion

[61]https://arxiv.org/abs/1805.01978v1

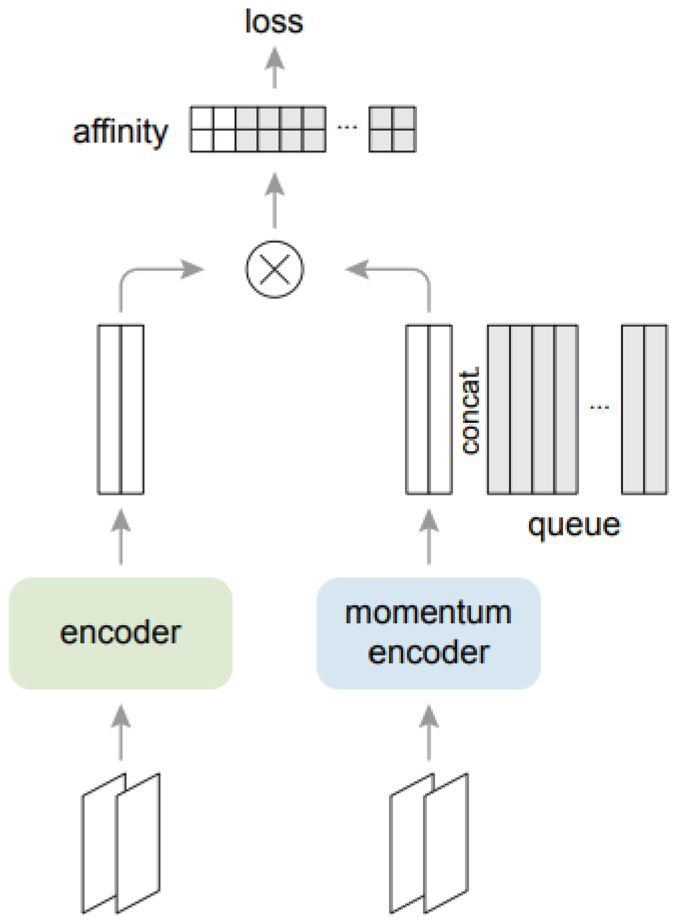
Shuffling BN

- Standard ResNet BN 사용 문제 => 좋은 표현 학습 방해[35]
  - BN이 적용된 배치 내에서 sample간의 Information leak이 생기면서 모델이 Pretext task에서 cheating을 하면서 low-loss solution을 쉽게 찾는 것 같다고 주장.

- 해결책으로 Shuffling BN 사용
  - Train with multiple GPUs and perform BN on the samples independently for each GPU.
  - For the key encoder f_k, we shuffle the sample order in the current mini-batch before distributing it among GPUs (and shuffle back after encoding)
  - the sample order of the mini-batch for the query encoder f_q is not altered.
  - This ensures the batch statistics used to compute a query and its positive key come from two different subsets.

[35] https://openreview.net/ pdf?id=rJerHlrYwH

MoCo 요약



Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

```
   # f_q, f_k: encoder networks for query and key
   # queue: dictionary as a queue of K keys (CxK)
   # m: momentum
   # t: temperature

   f_k.params = f_q.params # initialize
   for x in loader: # load a minibatch x with N samples
      x_q = aug(x) # a randomly augmented version
1.    x_k = aug(x) # another randomly augmented version

      q = f_q.forward(x_q) # queries: NxC
2.    k = f_k.forward(x_k) # keys: NxC
      k = k.detach() # no gradient to keys

      # positive logits: Nx1
      l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

3.    # negative logits: NxK
      l_neg = mm(q.view(N,C), queue.view(C,K))

      # logits: Nx(1+K)
      logits = cat([l_pos, l_neg], dim=1)

      # contrastive loss, Eqn.(1)
4.    labels = zeros(N) # positives are the 0-th
      loss = CrossEntropyLoss(logits/t, labels)

      # SGD update: query network
      loss.backward()
      update(f_q.params)

5.    # momentum update: key network
      f_k.params = m*f_k.params+(1-m)*f_q.params

      # update dictionary
6.    enqueue(queue, k) # enqueue the current minibatch
      dequeue(queue) # dequeue the earliest minibatch
```

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

Dataset

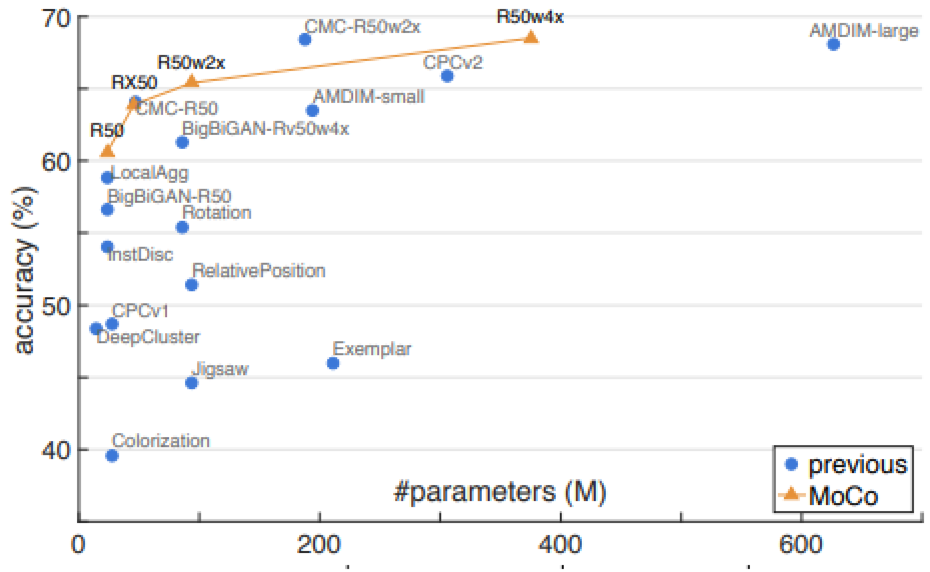
- ImageNet-1M (IN-1M)
    - 1.28 million images in 1000 classes
    - Well-balanced in class distribution
    - Images generally contain iconic view of objects

- Instagram-1B (IG-1B)
    - 1 billion (940M) public images from Instagram
    - Images from ~1500 hashtags that are related to the ImageNet categories
    - Relatively uncurated comparing to IN-1M
    - Long-tailed, unbalanced distribution of real-world data
    - Contains bot Iconic objects and scene-level images

Training

- Optimizer
  - SGD
    - 0.0001 decay and 0.9 SGD momentum
- IN-1M
  - 256 mini-batch
  - initial learning rate 0.03
  - 200 epoch
  - Learning rate multiplied by 0.1 at 120 and 160 epochs
  - 8 GPU
- IG-1B
  - 1024 mini-batch
  - Initial learning rate 0.12
  - Exponentially decayed by 0.9 x after every 62.5k iterations (64M images)
  - 1.25 iterations(~.14 epochs of IG-1B)
  - 64 GPU

Linear Classification Protocol



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	$64.1^{\dagger}$
	$R50w2\times_{L+ab}$	188	$68.4^{\dagger}$
AMDIM [2]	$AMDIM_{small}$	194	$63.5^{\dagger}$
	$AMDIM_{large}$	626	$68.1^{\dagger}$
MoCo	R50	24	60.6
	RX50	46	63.9
	R50w2×	94	65.4
	R50w4×	375	**68.6**

Transferring Features

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**

	AP_{50}					AP	AP_{75}	
pre-train	RelPos, by [14]	Multi-task [14]	Jigsaw, by [26]	LocalAgg [66]	**MoCo**	**MoCo**	Multi-task [14]	**MoCo**
super. IN-1M	74.2	74.2	70.5	74.6	74.4	42.4	44.3	42.7
unsup. IN-1M	66.8 (−7.4)	70.5 (−3.7)	61.4 (−9.1)	69.1 (−5.5)	74.9 (+0.5)	46.6 (+4.2)	43.9 (−0.4)	50.1 (+7.4)
unsup. IN-14M	-	-	69.2 (−1.3)	-	75.2 (+0.8)	46.9 (+4.5)	-	50.2 (+7.5)
unsup. YFCC-100M	-	-	66.6 (−3.9)	-	74.7 (+0.3)	45.9 (+3.5)	-	49.0 (+6.3)
unsup. IG-1B	-	-	-	-	75.6 (+1.2)	47.6 (+5.2)	-	51.7 (+9.0)

Table 4. **Comparison with previous methods on object detection fine-tuned on PASCAL VOC** `trainval2007`. Evaluation is on

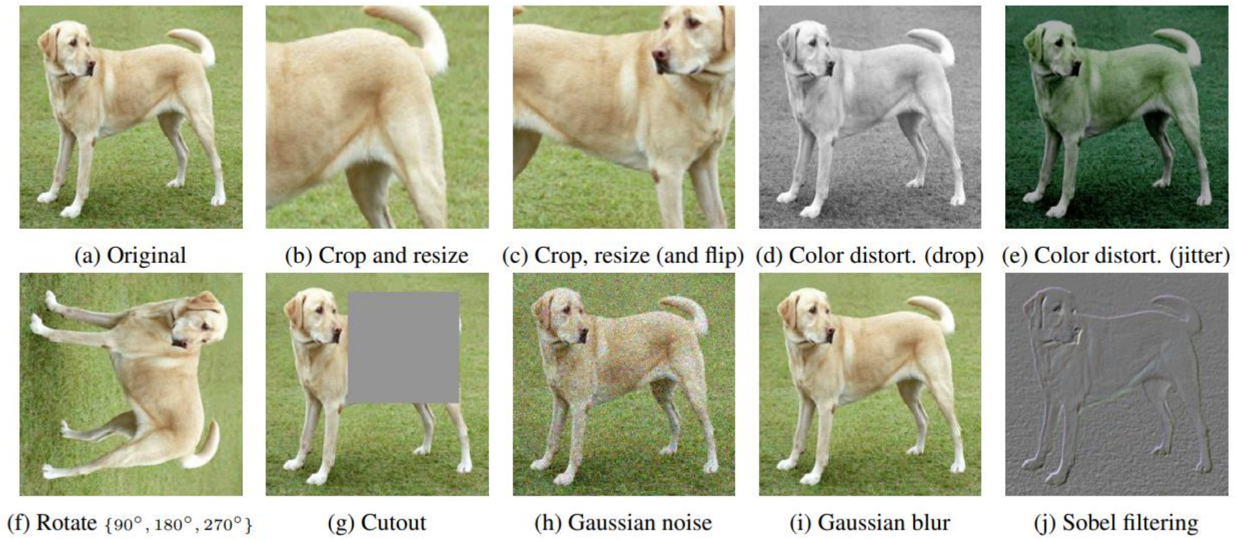
Improved Baselines with Momentum Contrastive Learning

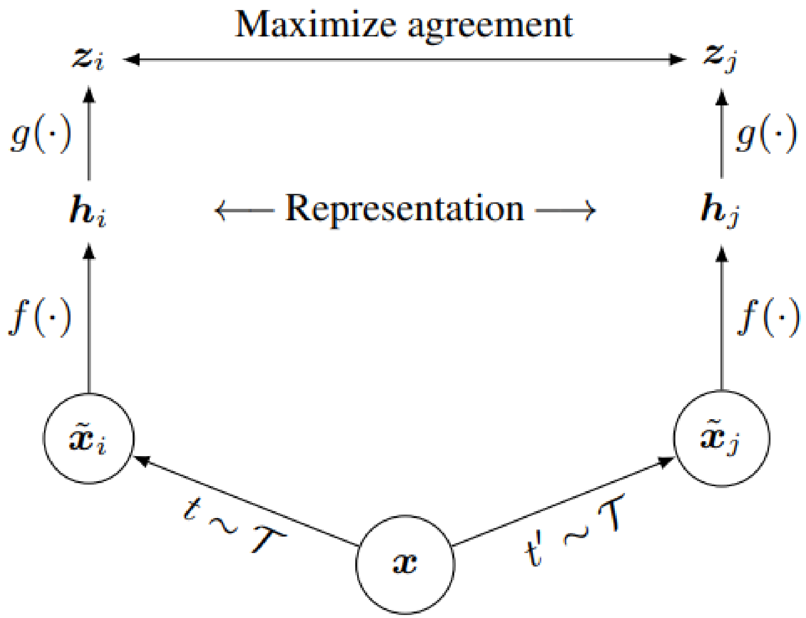
저자: Xinlei Chen, Haoqi Fan, Ross Girshick, Kaiming He

소속: Facebook AI Research (FAIR)

SimCLR 핵심 요소 3가지

1. Large batch, longer training
2. Stronger augmentation(color distortion, random resize crop, blur 등)
3. MLP projection head





MoCo 성능 향상 방법

- SimCLR 핵심 요소 2, 3 이용
  - Stronger augmentation
  - MLP projection head
- Cosine learning rate scheduler 적용

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**

Table 1. **Ablation of MoCo baselines**, evaluated by ResNet-50 for

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

mechanism	batch	memory / GPU	time / 200-ep.
MoCo	256	**5.0G**	**53 hrs**
end-to-end	256	7.4G	65 hrs
end-to-end	4096	93.0G†	n/a

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. †: based on our estimation.