Momentum Contrast for Unsupervised Visual Representation Learning

School of Industrial and Management Engineering, Korea University

Lee Kyung Yoo

KOREA UNIVERSITY

DMQA

Contents

Research Purpose

❖ Momentum Contrast for Unsupervised Visual Representation Learning (CVPR, 2020)

- Facebook AI Research에서 연구하였으며 2022년 5월 20일 기준으로 3,312회 인용

Momentum Contrast for Unsupervised Visual Representation Learning

Kaiming He Haoqi Fan Yuxin Wu Saining Xie Ross Girshick

Facebook AI Research (FAIR)

Code: https://github.com/facebookresearch/moco

Abstract

We present Momentum Contrast (MoCo) for unsupervised visual representation learning. From a perspective on contrastive learning [29] as dictionary look-up, we build a dynamic dictionary with a queue and a moving-averaged encoder. This enables building a large and consistent dictionary on-the-fly that facilitates contrastive unsupervised learning. MoCo provides competitive results under the common linear protocol on ImageNet classification. More importantly, the representations learned by MoCo transfer well to downstream tasks. MoCo can ***outperform*** *its supervised pre-training counterpart in* ***7*** *detection/segmentation tasks on PASCAL VOC, COCO, and other datasets, sometimes surpassing it by large margins. This suggests that the gap between unsupervised and supervised representation learning has been largely closed in many vision tasks.*

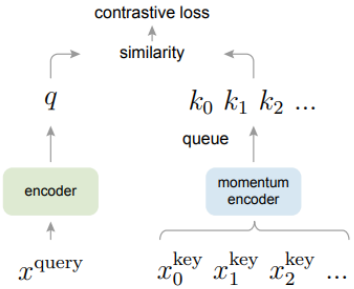


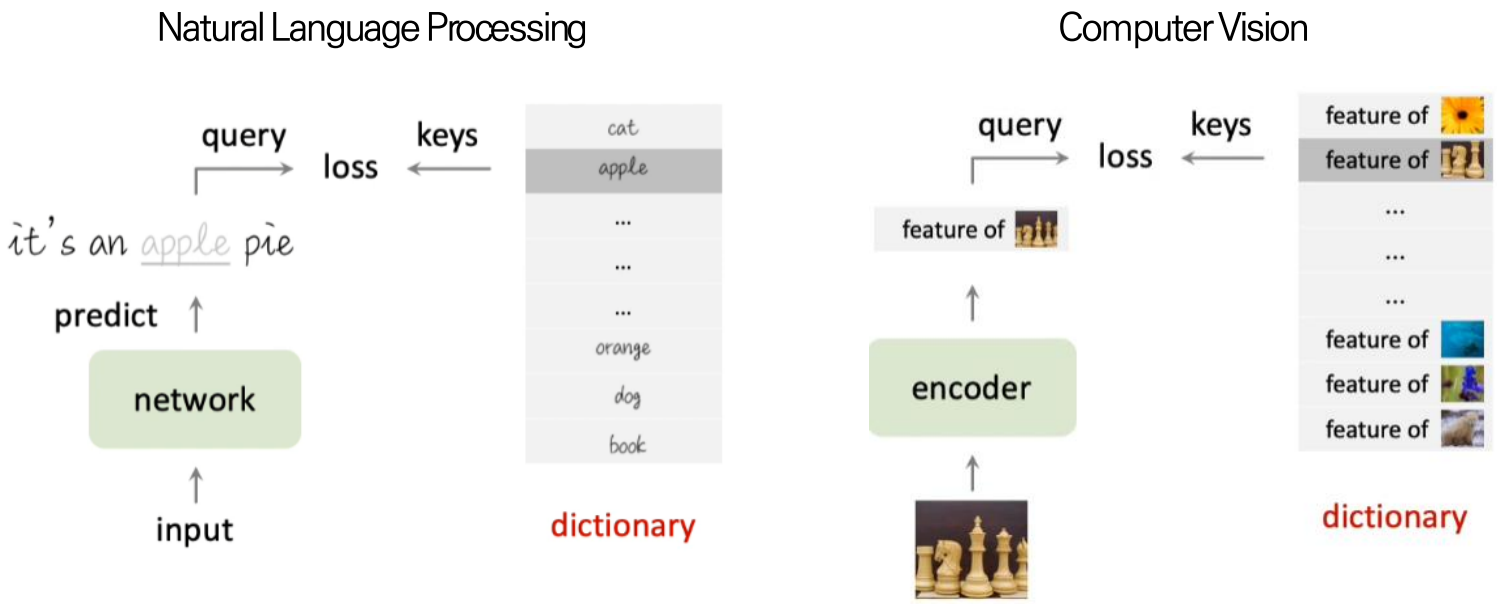
Figure 1. Momentum Contrast (MoCo) trains a visual representation encoder by matching an encoded query q to a dictionary of encoded keys using a contrastive loss. The dictionary keys $\{k_0, k_1, k_2, ...\}$ are defined on-the-fly by a set of data samples. The dictionary is built as a queue, with the current mini-batch enqueued and the oldest mini-batch dequeued, decoupling it from

DMQA

Research Purpose

❖ Unsupervised learning in NLP vs. CV

- NLP에서는 dictionary look-up task를 기반으로 unsupervised learning이 우수한 성능을 보임
- CV 역시 하나의 이미지에 대한 feature를 query, 다량의 이미지에 대한 feature를 dictionary로 하여 dictionary look-up task를 기반으로 unsupervised learning을 수행할 수 있음

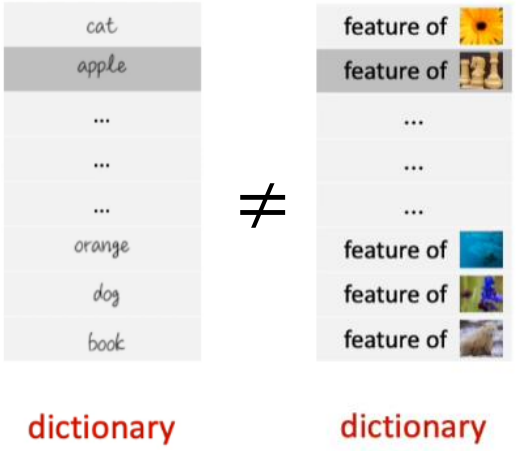


DMQA

Research Purpose

❖ Unsupervised learning in NLP vs. CV

- 그러나, CV는 NLP와 같은 dictionary를 구축하는데 어려움이 따름
- 이는 NLP와 CV 간 각각의 signal space 차이에 기인함
  - ➢ NLP에서는 토큰화된 dictionary를 위한 discrete signal space 존재 (words, sub-word units, …)
  - ➢ CV에서는 이와 달리 high-dimensional + continuous signal space 존재



다량의 이미지에 대한

∴ 고차원의 연속적인 signal에 맞는 dictionary를 구축해야 함

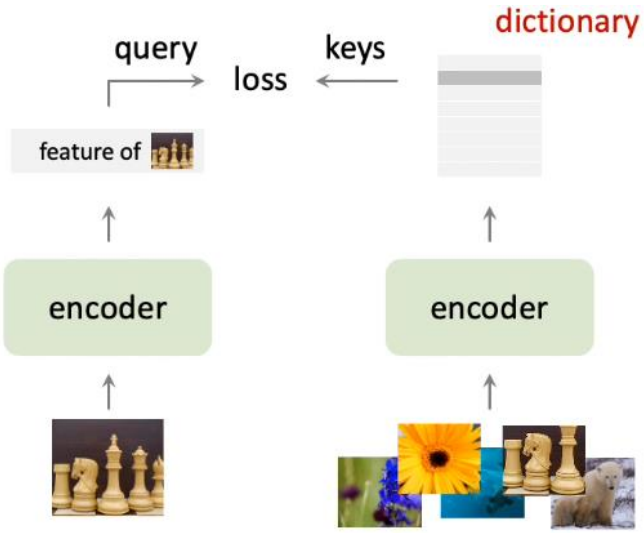
DMQA

Research Purpose

❖ Contrastive learning as a dictionary look-up

- Contrastive learning은 고차원의 연속적인 input에 대해 discrete한 dictionary을 구축하는 방법
  - ➢ Dictionary의 key는 이미지 데이터로부터 샘플링 후 encoder에 의해 즉석에서 생성
  - ➢ Dictionary look-up task를 수행하도록 encoder를 학습 (contrastive loss 기반)
  - ➢ 랜덤 샘플링 및 key encoder가 학습 동안 계속해서 업데이트 된다는 점에서 **"dynamic dictionary"**

Contrastive learning



Research Purpose

❖ Two challenges in contrastive learning

- Large dictionary
  - ➢ 좋은 성능을 위해 어떻게 dictionary를 가능한 크게 생성할 수 있는가
- Consistent dictionary
  - ➢ Key encoder가 지속적으로 업데이트됨에도 어떻게 dictionary를 가능한 일관되게 만들 것인가

Assumptions

1) 좋은 feature들이 풍부한 negative 샘플들을 포함해 **dictionary가 클 경우**
2) 학습이 안정되도록 **dictionary가 일관적일 경우**

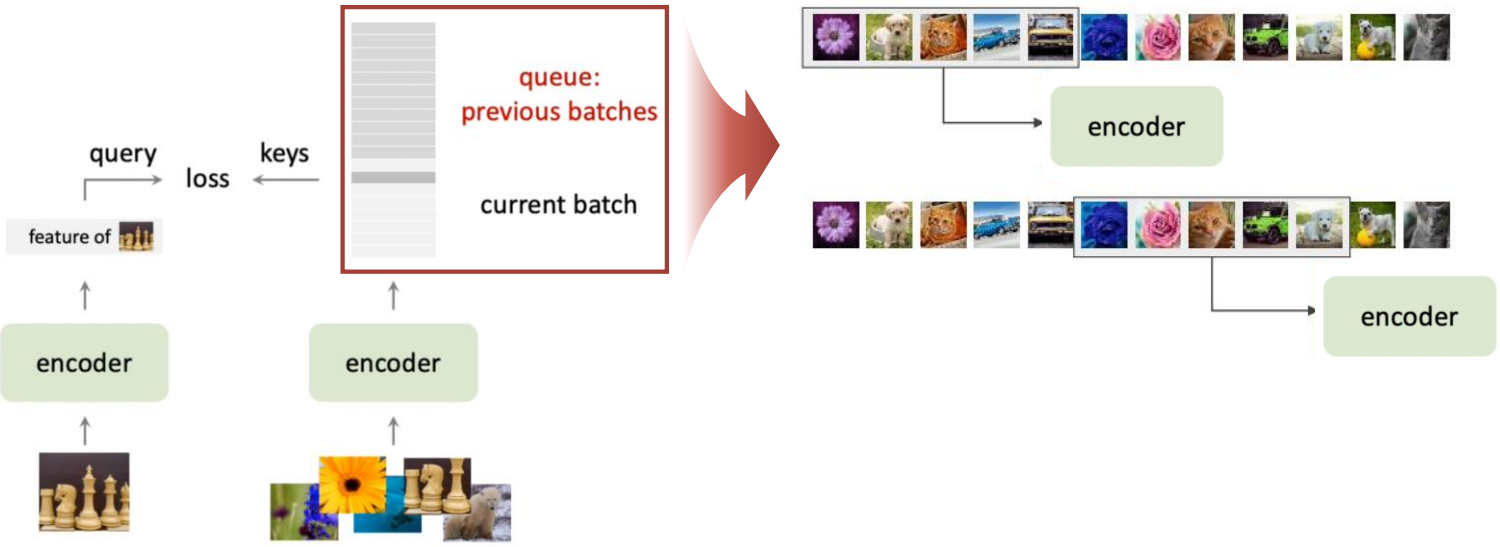
향상된 성능을 이끌어낼 수 있다고 접근

Momentum Contrast (MoCo)

Method (1)

❖ Dictionary as a queue (Memory queue)

- 충분히 큰 크기의 dictionary를 "queue" 형태로 유지, 이로부터 negative sample들을 추출
  - ➢ 지정된 dictionary 사이즈에 따라 현재 mini-batch는 dictionary에 enqueue 됨
  - ➢ 이와 동시에 dictionary에 있는 가장 오래된 mini-batch는 dequeue 됨
- Dictionary 사이즈는 mini-batch 사이즈보다 크게 유지하면서 최근 mini-batch로 업데이트 가능



DMQA

Momentum Contrast (MoCo)

Method (2)

❖ Momentum update

- Queue 형태의 큰 dictionary를 이용하게 되면, key encoder를 학습하는 과정에서 back-propagation을 통해 수많은 negative sample들에 대해 gradient를 전파해야하기 때문에 학습이 어려움
- 나이브한 해결책으로 key encoder를 따로 학습하지 않고(=stop gradient), 학습된 query encoder를 그대로 가져와서 이용하는 것 고려
- 그러나, 이 방법은 실험에서 더 좋지 않은 결과를 보임
  - ➢ 연구진은 해당 실패가 key representation의 일관성을 줄이는 급격한 encoder 변화에서 왔다고 가정함

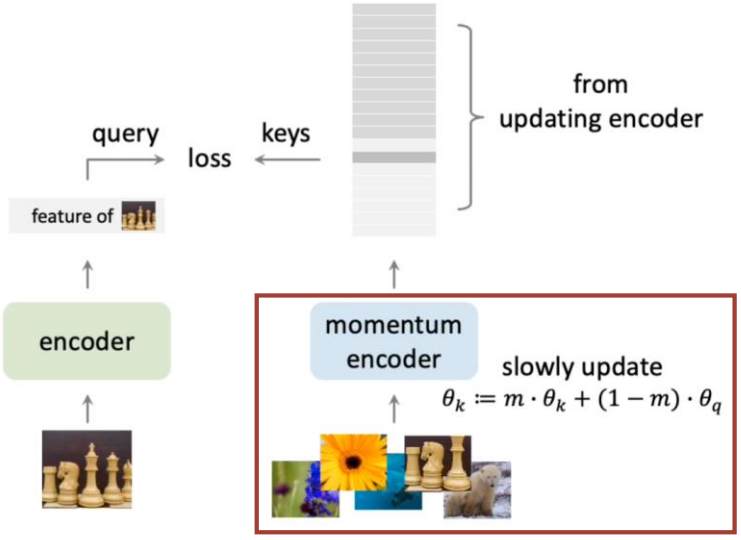
momentum m	0	0.9	0.99	0.999	0.9999
accuracy (%)	*fail*	55.2	57.8	59.0	58.9

Momentum Contrast (MoCo)

Method (2)

❖ Momentum update

- Momentum update를 적용시켜 key encoder를 "천천히" 업데이트시키는 방법을 이용
- 그 결과, queue 안의 key들이 다른 mini-batch에 의한 encoder에 encoding 되더라도 dictionary 내의 key 구성의 차이를 작게(=consistent) 만들 수 있음
- 상대적으로 큰 모멘텀 ($m = 0.999$)이 작은 모멘텀 ($m = 0.9$) 보다 더 안정적인 학습 가능



Moving average

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q$$

- $\theta_k = key\ encoder\ f_k$ 의 파라미터
- $\theta_q = query\ encoder\ f_q$ 의 파라미터

Momentum Contrast (MoCo)

Overall Algorithm

❖ Pseudocode

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
```

Momentum encoder를 **encoder로 initialize**

```
f_k.params = f_q.params # initialize
```

```
for x in loader: # load a minibatch x with N samples
```

Augmented 이미지로 **query와 key에 해당하는 이미지를 정의**

```
x_q = aug(x) # a randomly augmented version
x_k = aug(x) # another randomly augmented version
```

앞서 구분된 input이 각각 **encoder와 momentum encoder를 통과**하고 momentum encoder에 대해서는 gradient가 계산되지 않도록 함

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

Positive, negative 각각에 대해 logit 및 **InfoNCE를 계산**

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

# 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)
labels = zeros(N) # positives are the 0-th
loss = CrossEntropyLoss(logits/t, labels)
```

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

Backpropagation하여 encoder를 update하고 기존에 대한 momentum m 을 가지도록 **momentum encoder를 재정의**

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

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

Queue에 해당 mini-batch의 **key를 추가하고 가장 오래된 key를 제외**

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

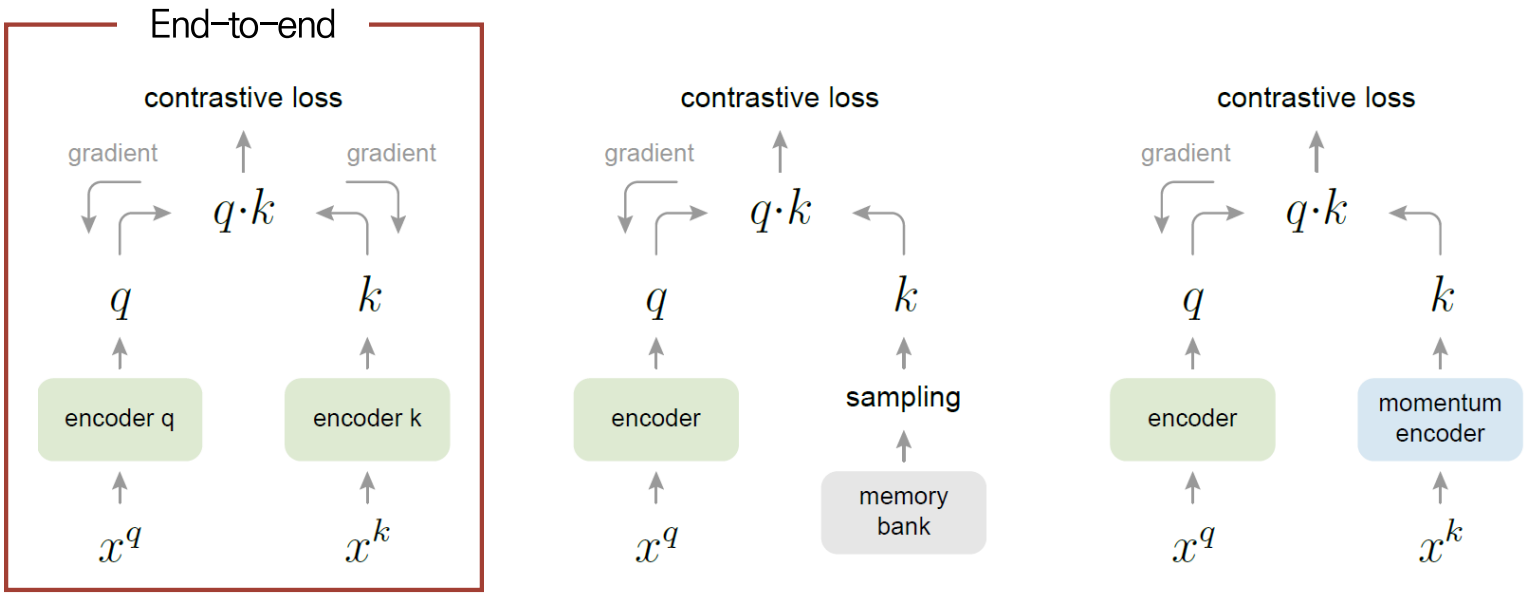
bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

Momentum Contrast (MoCo)

Comparison to previous mechanisms

❖ End-to-end vs. Memory bank vs. MoCo

- End-to-end 방식
  - ➢ Mini-batch의 모든 이미지에 대해 loss를 계산하여 encoder를 업데이트하는 방식
  - ➢ 같은 encoder로부터 나오는 feature를 통해 loss를 계산하고 업데이트 되므로 일관된 업데이트 가능
  - ➢ Dictionary 사이즈가 mini-batch 사이즈에 종속되며 GPU 메모리에도 한계 존재



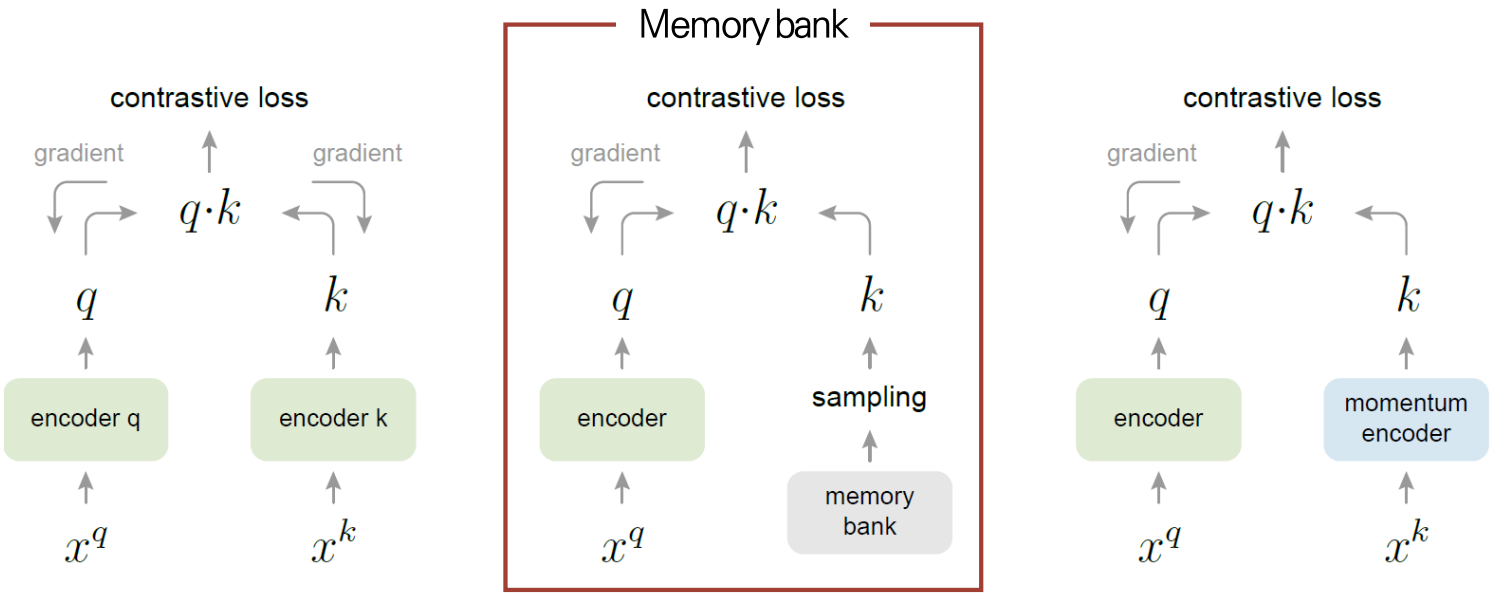
DMQA

Momentum Contrast (MoCo)

Comparison to previous mechanisms

❖ End-to-end vs. Memory bank vs. MoCo

- Memory bank 방식
  - ➢ 모든 이미지의 key값이 저장된 memory bank로부터 랜덤하게 샘플링하는 방식
  - ➢ 대량의 샘플을 사용할 수 있어 풍부한 negative sample 포함 가능
  - ➢ 샘플링을 사용하기 때문에 일관된 업데이트를 하기 어려움



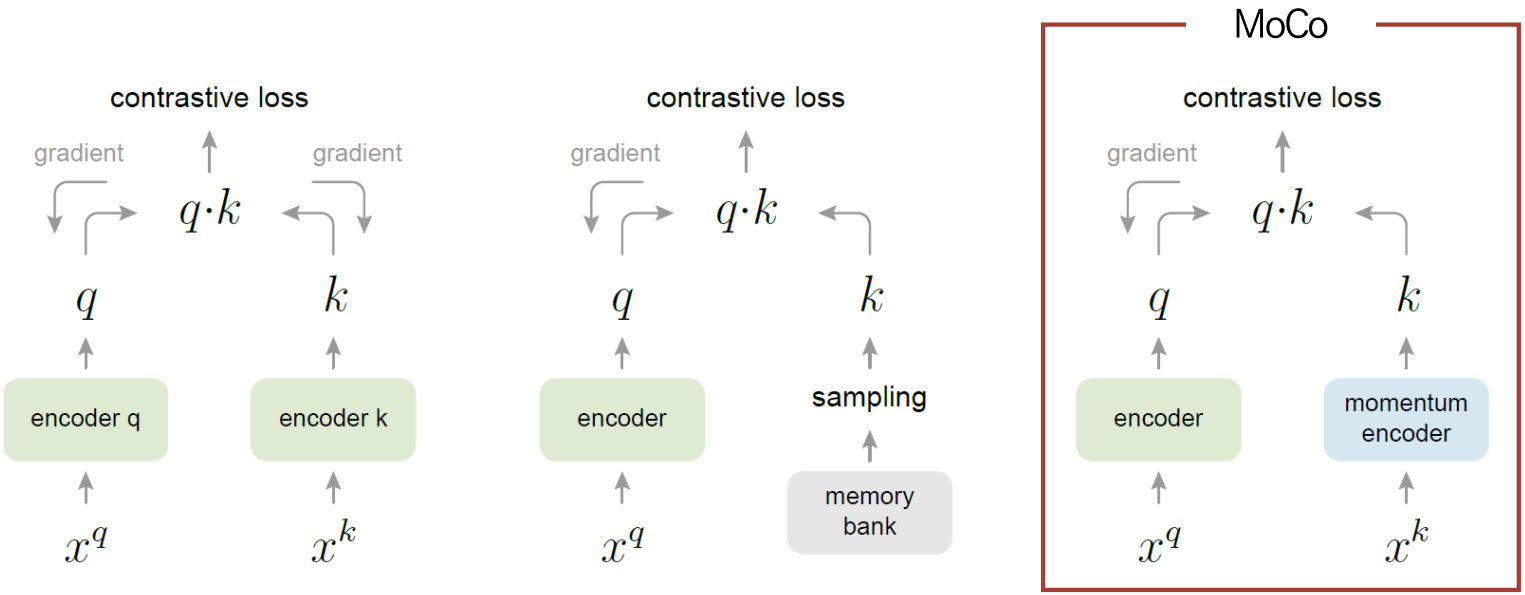
DMQA

Momentum Contrast (MoCo)

Comparison to previous mechanisms

❖ End-to-end vs. Memory bank vs. MoCo

- MoCo 방식
  - ➢ 앞선 End-to-end 방식과 Memory bank 방식의 단점을 개선한 방식
  - ➢ 대량의 negative sample을 사용 가능
  - ➢ Encoder와 dictionary key를 consistent하게 유지 가능



DMQA

Experiments

Settings

❖ Experiment Settings

- Dataset
  - ImageNet-1M (IN-1M)
  - Instagram-1B (IG-1B)

    : 이미지들은 ImageNet 카테고리와 연관 있는 1500개의 해시태그로부터 추출함

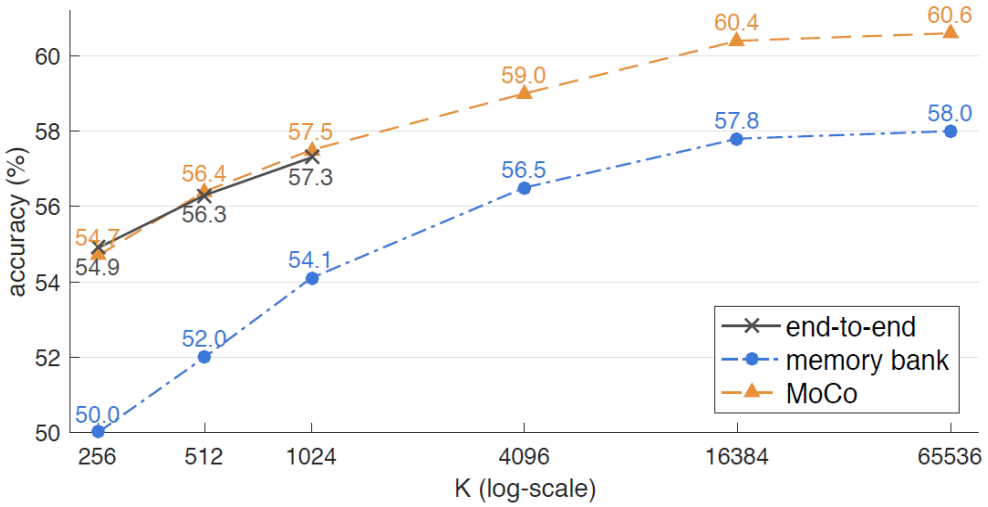
    : ImageNet 데이터에 비해 상대적으로 덜 정렬되었으며 long-tailed, unbalanced 분포를 가짐
- Training
  - Optimizer = SGD
  - SGD weight decay = 0.0001
  - SGD momentum = 0.9
  - IN-1M에서는 8개의 GPU에 256 사이즈의 mini-batch 사용, 초기 learning rate = 0.03
  - IG-1B에서는 64개의 GPU에 1024 사이즈의 mini-batch 를 사용, 초기 learning rate = 0.12

Experiments

Results

❖ End-to-end vs. Memory bank vs. MoCo

- 각각의 구조에 대해 ImageNet 데이터를 통한 Linear classification protocol로 비교한 결과
  - ➢ Linear classification protocol : 학습된 모델은 freeze하고 linear layer 하나만 추가해서 supervised learning으로 학습하여 평가하는 방식
- End-to-end와 MoCo가 더 나은 성능을 비슷하게 보이지만 MoCo 구조가 end-to-end 구조 대비 negative sample을 많이 보게 하는데 유리함



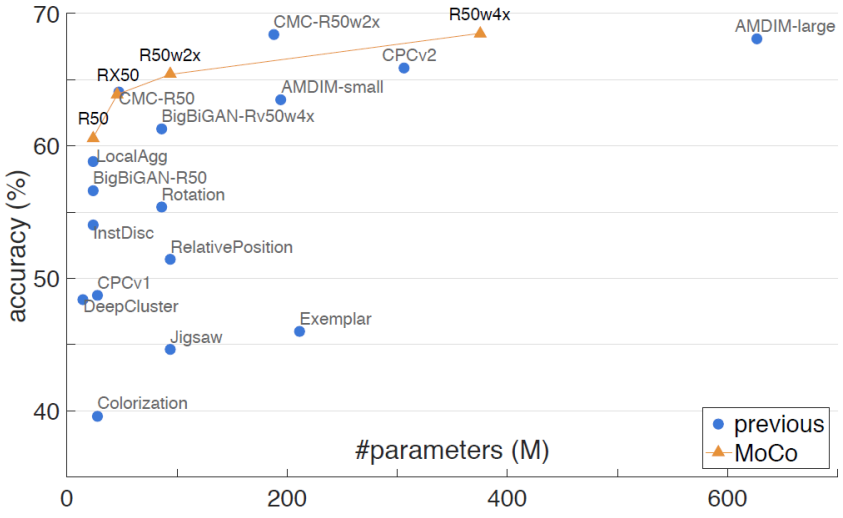
DMQA

Experiments

Results

❖ Other previous SSL methods

- 앞선 실험과 동일하게 ImageNet 데이터를 통한 Linear classification protocol로 비교한 결과
- MoCo 등장 이전의 다른 self-supervised learning 기법들보다 높은 성능임을 확인



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**

DMQA

Experiments

Results

❖ Fine-tuned on various tasks

- Object detection과 instance segmentation task에 대해 COCO dataset으로 fine-tuning한 결과
- ImageNet supervised pre-training counterpart보다 좋은 성능을 보이는 결과 다수 존재

pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	31.0	49.5	33.2	28.5	46.8	30.4
super. IN-1M	38.9	59.6	42.7	35.4	56.5	38.1
MoCo IN-1M	38.5 (−0.4)	58.9 (−0.7)	42.0 (−0.7)	35.1 (−0.3)	55.9 (−0.6)	37.7 (−0.4)
MoCo IG-1B	38.9 (0.0)	59.4 (−0.2)	42.3 (−0.4)	35.4 (0.0)	56.5 (0.0)	37.9 (−0.2)

(a) Mask R-CNN, R50-FPN, 1× schedule

pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	36.7	56.7	40.0	33.7	53.8	35.9
super. IN-1M	40.6	61.3	44.4	36.8	58.1	39.5
MoCo IN-1M	40.8 (+0.2)	61.6 (+0.3)	44.7 (+0.3)	36.9 (+0.1)	58.4 (+0.3)	39.7 (+0.2)
MoCo IG-1B	41.1 (+0.5)	61.8 (+0.5)	45.1 (+0.7)	37.4 (+0.6)	59.1 (+1.0)	40.2 (+0.7)

(b) Mask R-CNN, R50-FPN, 2× schedule

pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	26.4	44.0	27.8	29.3	46.9	30.8
super. IN-1M	38.2	58.2	41.2	33.3	54.7	35.2
MoCo IN-1M	38.5 (+0.3)	58.3 (+0.1)	41.6 (+0.4)	33.6 (+0.3)	54.8 (+0.1)	35.6 (+0.4)
MoCo IG-1B	39.1 (+0.9)	58.7 (+0.5)	42.2 (+1.0)	34.1 (+0.8)	55.4 (+0.7)	36.4 (+1.2)

pre-train	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}	AP^{mk}	AP^{mk}_{50}	AP^{mk}_{75}
random init.	35.6	54.6	38.2	31.4	51.5	33.5
super. IN-1M	40.0	59.9	43.1	34.7	56.5	36.9
MoCo IN-1M	40.7 (+0.7)	60.5 (+0.6)	44.1 (+1.0)	35.4 (+0.7)	57.3 (+0.8)	37.6 (+0.7)
MoCo IG-1B	41.1 (+1.1)	60.7 (+0.8)	44.8 (+1.7)	35.6 (+0.9)	57.4 (+0.9)	38.1 (+1.2)

* AP^{bb} : Bounding-box Average Precision, AP^{mk} : Mask Average Precision

* 괄호 안의 표기값 = ImageNet supervised pre-training과의 성능 차

Conclusion

❖ Conclusion

- MoCo에서 강조하는 것은 크고 일관성 있는 dictionary의 구성
- 이를 위해 queue 형태의 dictionary와 momentum update를 이용한 key encoder를 적용
  - ➢ Queue형태의 Dictionary를 사용하여 batch size에 상관없이 negative sample을 많이 볼 수 있게 됨
  - ➢ Momentum encoder를 사용해 encoder와 dictionary key를 consistent하게 유지됨
- 여러 downstream tasks에서 unsupervised pre-training이 supervised counterparts를 능가할 수 있음을 증명함

Reference

❖ Reference

- Wu, Z., Xiong, Y., Yu, S. X., & Lin, D. (2018). Unsupervised feature learning via non-parametric instance discrimination. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3733-3742).
- He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9729-9738).
- Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In International conference on machine learning (pp. 1597-1607). PMLR.
- https://animilux.github.io/paper_review/2021/01/25/moco.html
- http://dmqm.korea.ac.kr/activity/seminar/308


Thank You