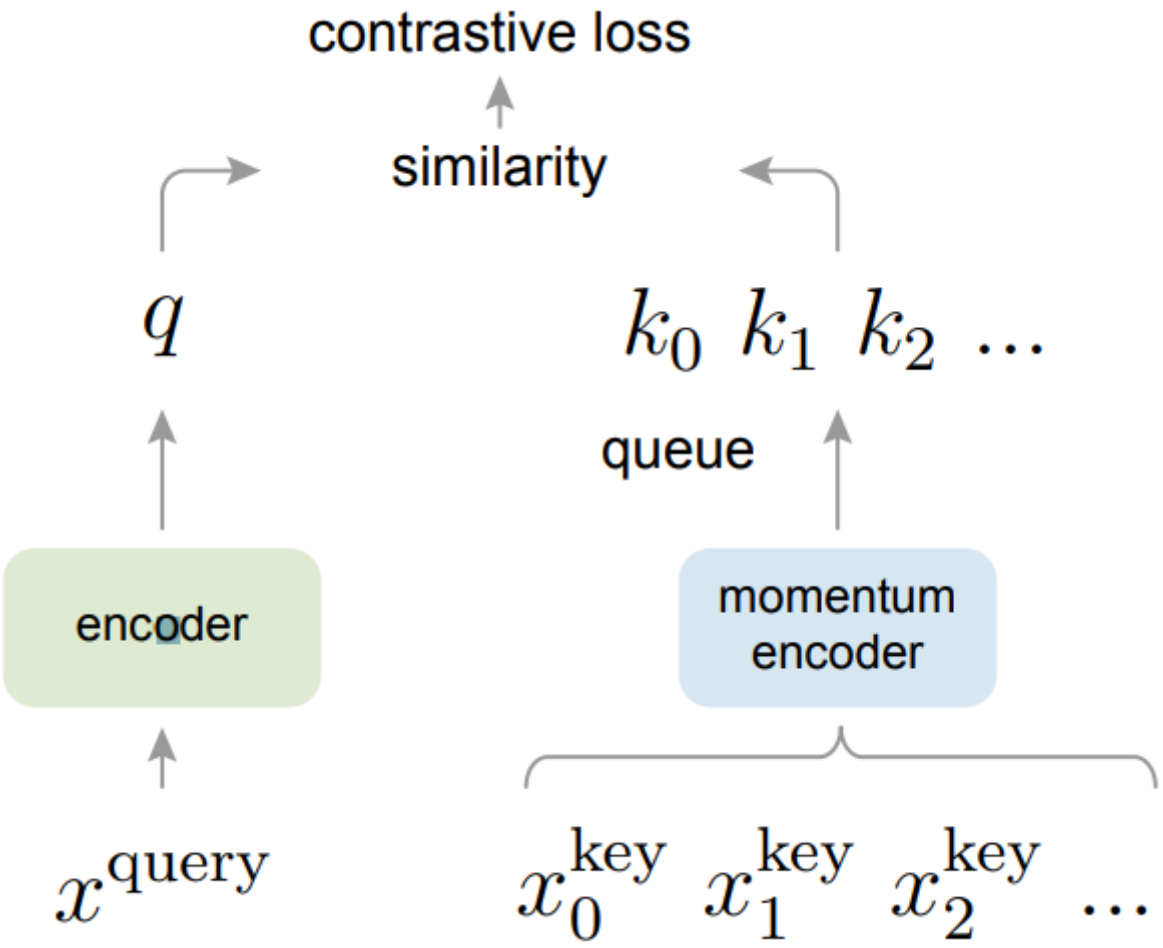
# 7.2. Momentum Contrastive(MoCo), Improved Baselines with Momentum Contrastive Learning

Momentum Contrastive learning (MoCo)

- 이미지는 high-dimensional space에 존재하는 연속적인 데이터로 언어처럼 구조화되있지 않아 tokenized dictionaries를 만들기 힘듦
- 다른 논문 (CPCv2)을 통해 Dictionary는 커야하고, key들은 Consistent할 수록 좋다는 것이 가정됨.
  => 이미지가 연속적이고 high-dimensional space에 존재하므로 dictionary가 클수록 negative sample하기 좋고, encoder에 의해 만들어진 key들은 query와 비교해서 consistent 할수록 좋기 때문
- Key들이 Consistent할려면? loss가 급격히 변하지 않고 Smooth하게 변화해야함




### 3.1 Contrastive Learning as Dictionary Look-up

Contrastive Learning을 위해 InfoNCE라 불리는 loss function을 사용한다.

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i \tau)} \tag{1}$$

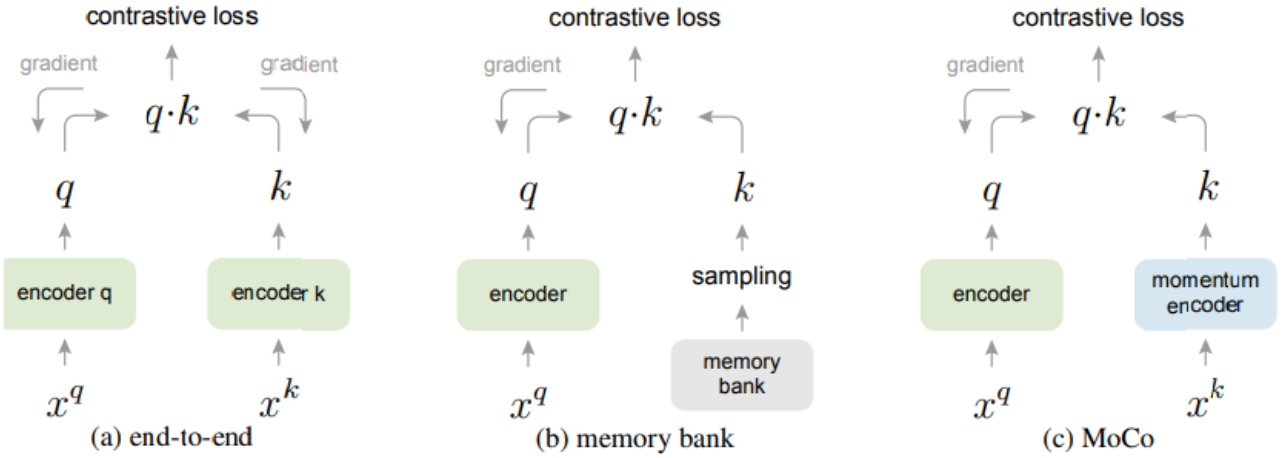
- $q$ : encoded query
- $\{k_0, k_1, k_2, \ldots\}$ : dictionary의 key들로 encoded sample의 집합, $k_+$는 $q$와 일치하는 key,
- $\tau$ : temperature hyper-parameter

> *직관적으로 $q$를 $k_+$로 분류하는 $K+1$개의 softmax-based classifier의 log loss로 볼 수 있다.*

이 loss function은 query와 key를 만들어내는 encoder network를 학습시키기 위한 함수다. 일반적으로 query는 $q = f_q(x^q)$로, key는 $k = f_k(x^k)$로 만들어지는데, 각각의 encoder $f_q$, $f_k$는 방법에 따라 동일하거나 일부 요소만 공유하거나 완전히 다를 수 있다. 입력인 $x^q$, $x^k$또한 마찬가지로 pretext task에 따라 이미지/패치/패치들일 수 있다.

## Momentum update

- Queue를 이용해 dictionary를 크게 만들면 queue안의 모든 샘플에 대해 gradient를 backpropagation 해야 하므로 $f_k$를 학습시키기 어렵다.
    - 단순한 방법은 gradient를 무시하고 $f_q$를 이용해 $f_k$를 구성하는 것인데, 성능이 처참했다. 이는 $f_k$가 급격히 변해 key representation의 consistency를 감소시켰기 때문으로 보인다.
- 이 문제를 Momentum update를 이용해 해결한다.
    - $$\theta_k \leftarrow m\theta_k + (1-m)\theta_q \tag{2}$$
        - $m \in [0, 1)$ : momentum coefficient로 $m = 0.999$가 $m = 0.9$보다 성능이 좋다.
        - $\theta_q$ : $f_q$의 파라미터, $\theta_k$ : $f_k$의 파라미터
        - $k$가 변화하는 encoder $f_k$에 의해 만들어 지더라도 변화의 정도가 느리므로 consistency가 유지된다.




- (a) : 각 학습당 입력되는 mini-batch 하나를 dictionary로 사용하여 key들의 consistency를 유지한다.
    - Dictionary 크기가 mini-batch 크기로 제한되고 dictionary를 크게 하기위해 mini-batch 크기를 키우면 large mini-batch optimization 문제에 직면한다.
    - pretext task에 따라 특별한 네트워크가 필요해 downstream task로 transfer하기 힘들다.
- (b) : 데이터셋의 모든 샘플들의 representation을 memory bank에 저장해 dictionary로 사용해 dictionary의 크기가 커진다.
    - memory bank의 representation은 각 샘플이 $f_q$에 들어갔을때만 업데이트된다. 즉, key들은 여러 epoch에 걸친 다른 $f_k$에 의해 만들어 졌으므로 consistency가 떨어진다.
    - MoCo처럼 Momentum update를 사용하지만 $f_k$가 아닌 memory bank안의 represnetation에 사용한다.

---

2020년 3월본 기준

논문 : https://arxiv.org/abs/2003.04297

논문이 생각하는 문제

- SimCLR은 large training batch를 사용해야함

---

## 논문 주장

- MoCo v2를 제안한다~
  Momentum Contrastive learning에다가 여러 기법을 추가
- MLP head (fine-tuning에서 Linear 레이어 2개 추가)
- SimCLR에서 사용한 Data augmentation
- image blur augmentation 추가

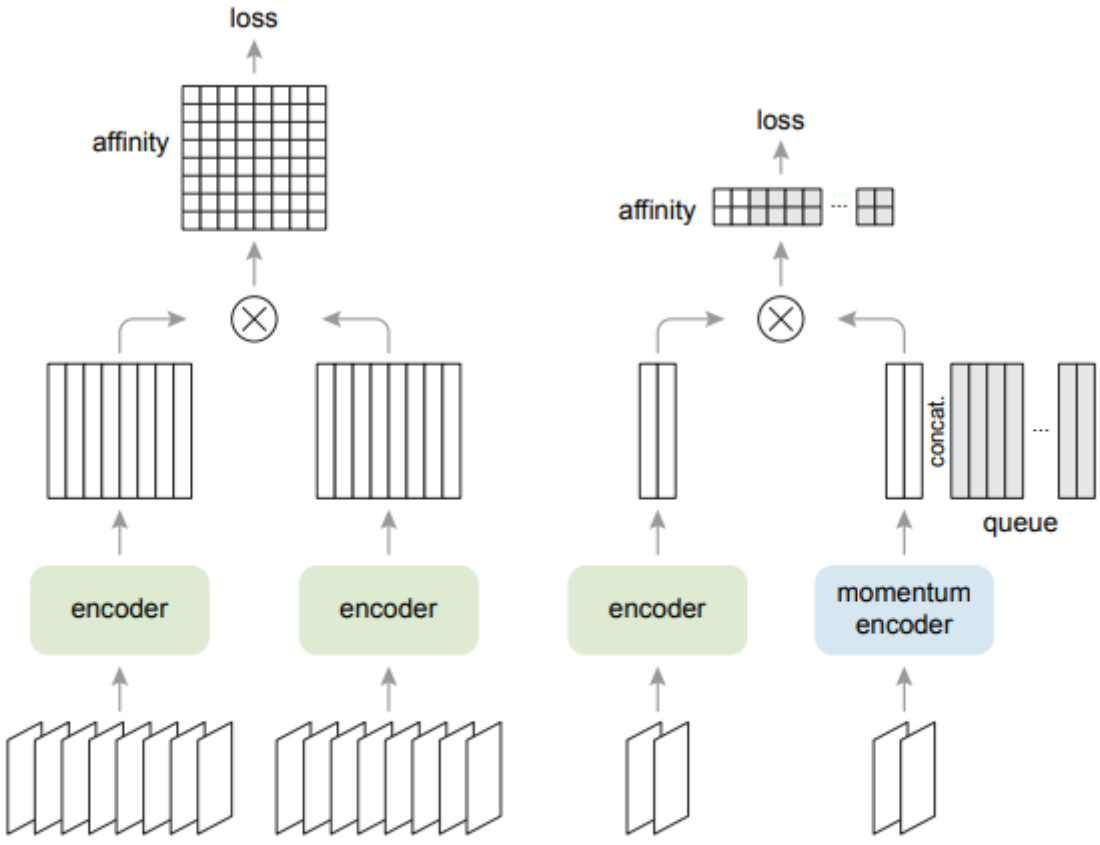



Figure 1. A **batching** perspective of two optimization mechanisms for contrastive learning. Images are encoded into a representation space, in which pairwise affinities are computed.

$$\mathcal{L}_{q,k^+,\{k^-\}} = -\log \frac{\exp(q\cdot k^+/\tau)}{\exp(q\cdot k^+/\tau) + \sum_{k^-} \exp(q\cdot k^-/\tau)}. \quad (1)$$

실험결과

| 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 (i) ImageNet linear classification, and (ii) fine-tuning VOC object detection (mean of 5 trials). "**MLP**": with an MLP head; "**aug+**": with extra blur augmentation; "**cos**": cosine learning rate schedule.

| 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).

| 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.