Momentum **Co**ntrast for Unsupervised Visual Representation Learning (MoCo)

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, Ross Girshick
Facebook AI Research (FAIR)

Sungman, Cho.

Introduction

Introduction : Contrastive Loss

- **A common way of defining a loss function** is to measure the difference between a model's prediction and a fixed target.

    Reconstructing : L1 / L2
    Pre-defined categories : Cross-Entropy / Margin-based Loss

Introduction : Contrastive Loss

- **A common way of defining a loss function** is to measure the difference between a model's prediction and a fixed target.

    Reconstructing : L1 / L2
    Pre-defined categories : Cross-Entropy / Margin-based Loss

- **Contrastive losses** measure the **similarities of sample pairs** in a representation space.

Introduction : Contrastive Loss

- **A common way of defining a loss function** is to measure the difference between a model's prediction and a fixed target.

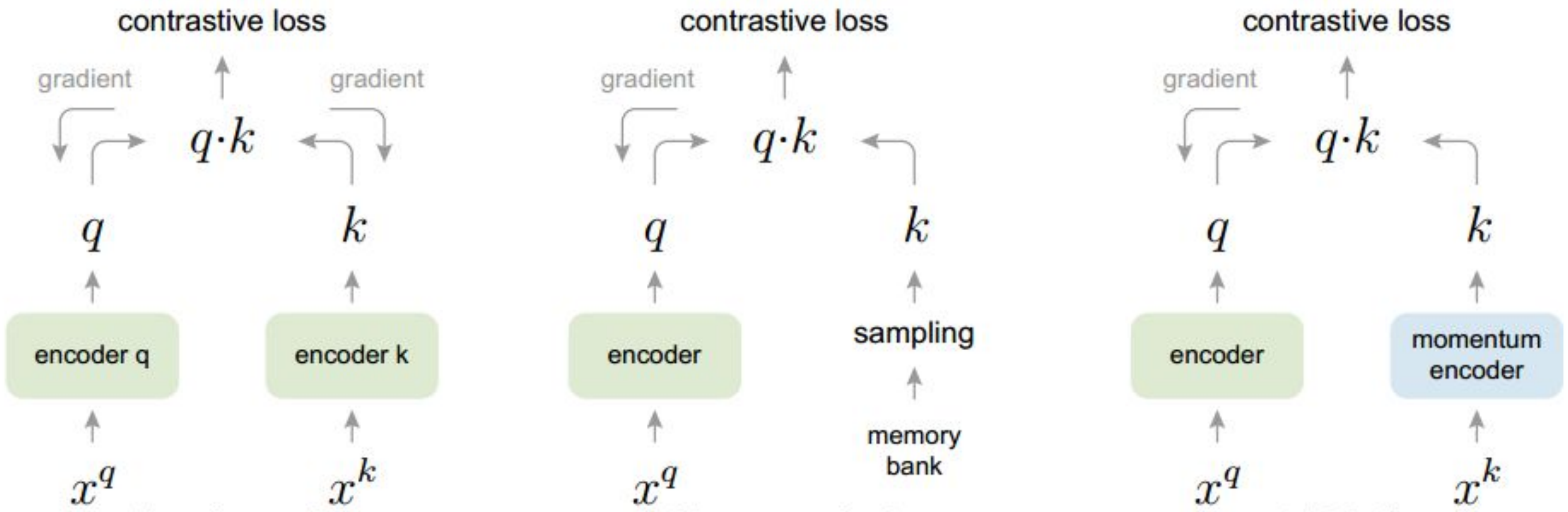
  Reconstructing : L1 / L2
  Pre-defined categories : Cross-Entropy / Margin-based Loss

- **Contrastive losses** measure the **similarities of sample pairs** in a representation space.

- **Adversarial losses** measure the **difference between probability distributions.**

Introduction

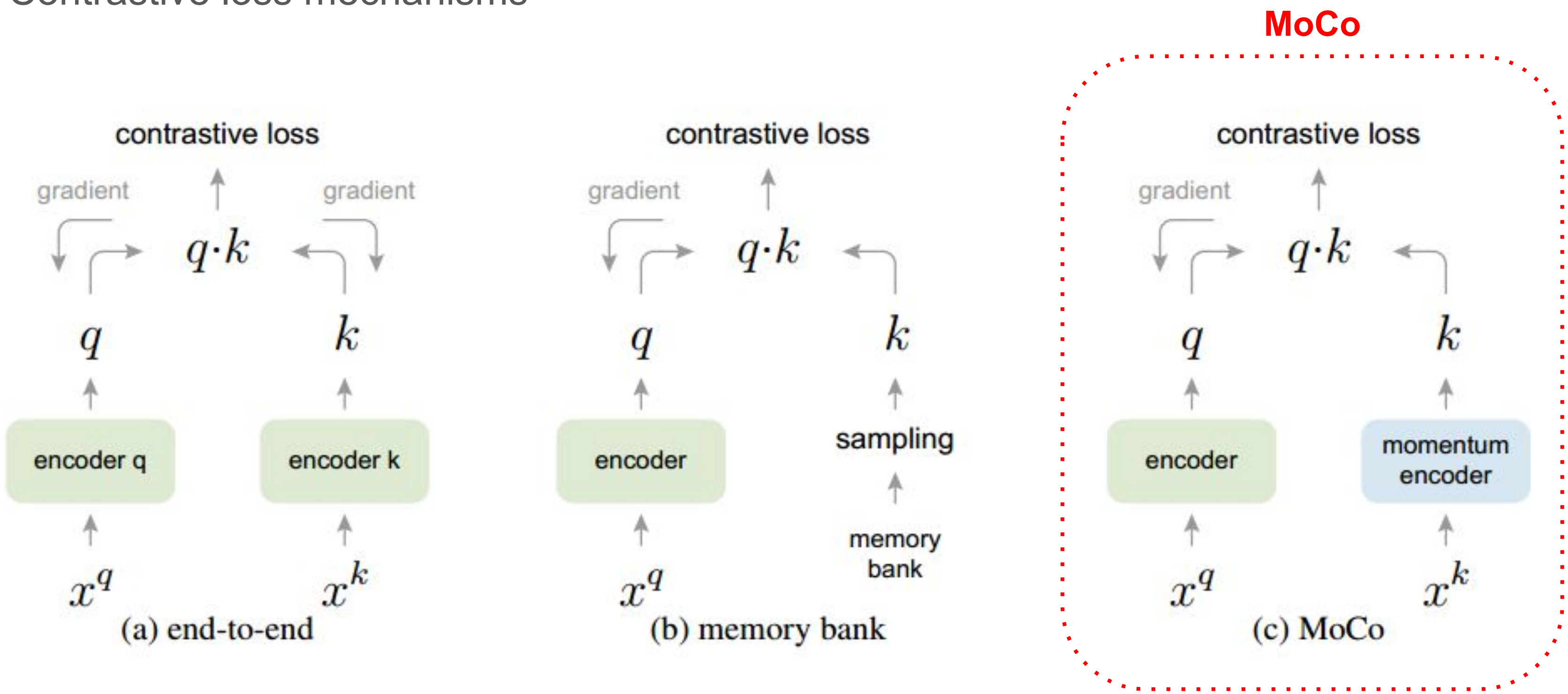
- Contrastive loss mechanisms



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

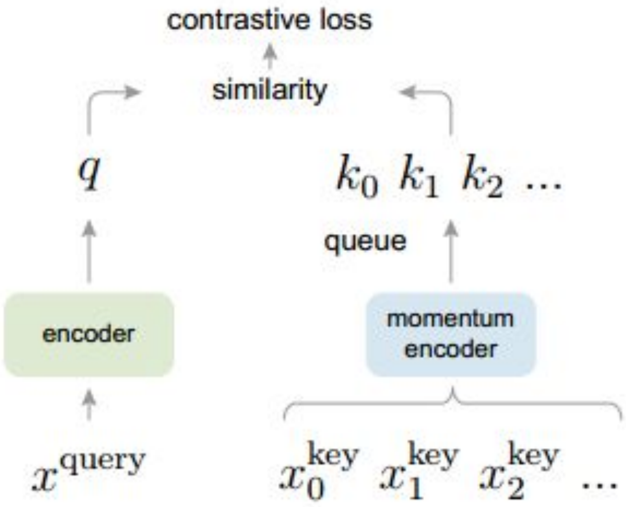
Introduction

- Contrastive loss mechanisms



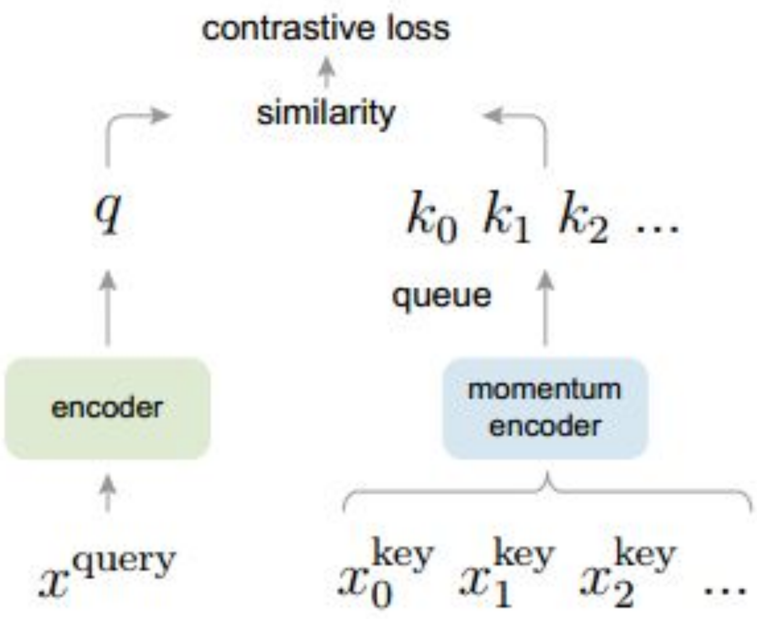
Contribution

- Build a **dynamic dictionary with a queue** and a moving-averaged encoder.
- Competitive results under the common linear protocol on ImageNet classification.



Method

Method : Contrastive as Dictionary Look-up



Matching key

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

- The sum is over *one positive* and *K negative* samples

Method : Momentum Contrast

- Contrastive learning is a way of **building a discrete dictionary on high-dimensional continuous inputs** such as images.

Method : Momentum Contrast

- Contrastive learning is a way of **building a discrete dictionary on high-dimensional continuous inputs** such as images.

- Good features can be **learned by a large dictionary** that covers a **rich set of negative samples**, while the encoder for the **dictionary keys is kept as consistent** as possible despite its evolution.

- **Two subjects**

  - Dictionary as a queue.

  - Momentum update.

Dictionary as a queue

Method : Dictionary as a queue

- At the core of our approach is **maintaining the dictionary as a queue of data samples.**

Method : Dictionary as a queue

- At the core of our approach is maintaining the dictionary as a queue of data samples.

- The introduction of a **queue decouples the dictionary size from the mini-batch size.**

Method : Dictionary as a queue

- At the core of our approach is maintaining the dictionary as a queue of data samples.
- The introduction of a queue decouples the dictionary size from the mini-batch size.
- The samples in the dictionary are progressively replaced. The **current mini-batch is enqueued** to the dictionary, and the **oldest mini-batch in the queue is removed.**

Method : Dictionary as a queue

- At the core of our approach is maintaining the dictionary as a queue of data samples.
- The introduction of a queue decouples the dictionary size from the mini-batch size.
- The samples in the dictionary are progressively replaced. The current mini-batch is enqueued to the dictionary, and the oldest mini-batch in the queue is removed.
- Removing the **oldest mini-batch** can be beneficial, because its encoded keys are the most outdated and thus the **least consistent** with the newest ones.

Momentum update

Method : Momentum update.

- Using a queue can make **dictionary large**, but it also makes it **intractable to update** the key encoder by back-propagation (the gradient should propagate to all samples in the queue).

- **Naive solution** : copy the key encoder from the query encoder, ignoring this gradient.

- Propose a **momentum update** to address this issue.

Method : Momentum update.

Formally, denoting the parameters of f_{k} as θ_{k} and those of f_{q} as θ_{q}, we update θ_{k} by:

$$\theta_{\mathrm{k}} \leftarrow m\theta_{\mathrm{k}} + (1 - m)\theta_{\mathrm{q}}. \tag{2}$$

Here $m \in [0, 1)$ is a momentum coefficient. Only the parameters θ_{q} are updated by back-propagation.

Pseudocode of MoCo

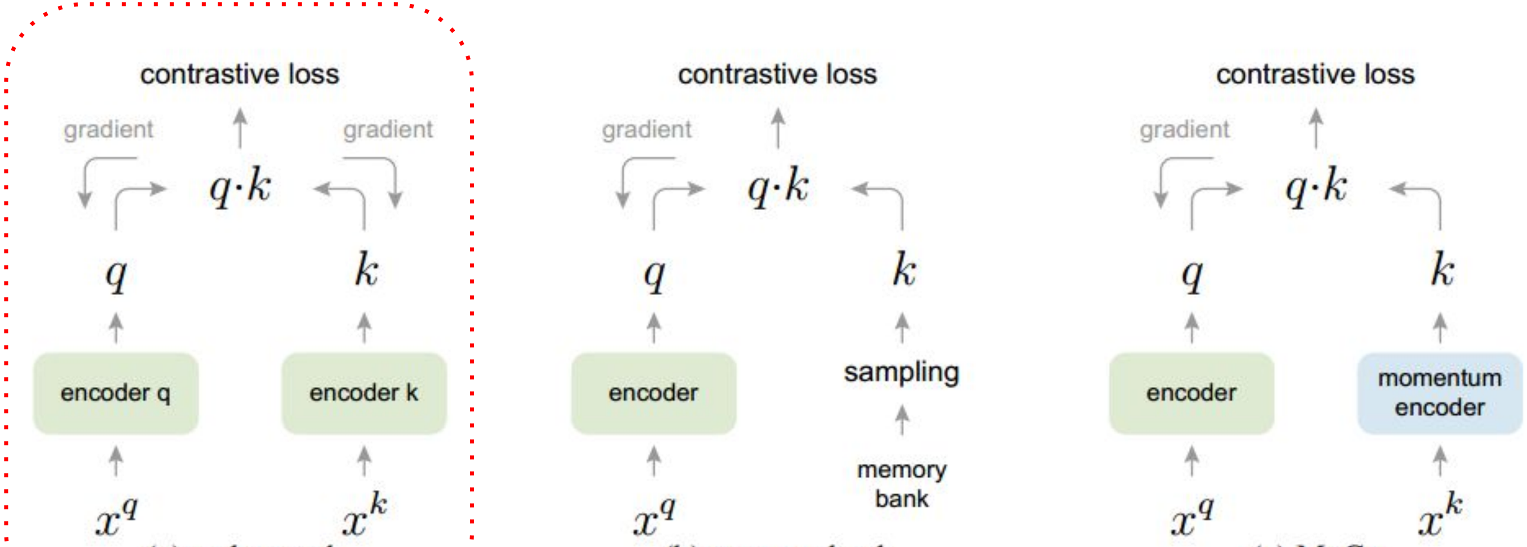
Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

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

   q = f_q.forward(x_q) # queries: NxC
   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))

   # 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)

   # 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

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

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

Relations to previous mechanisms

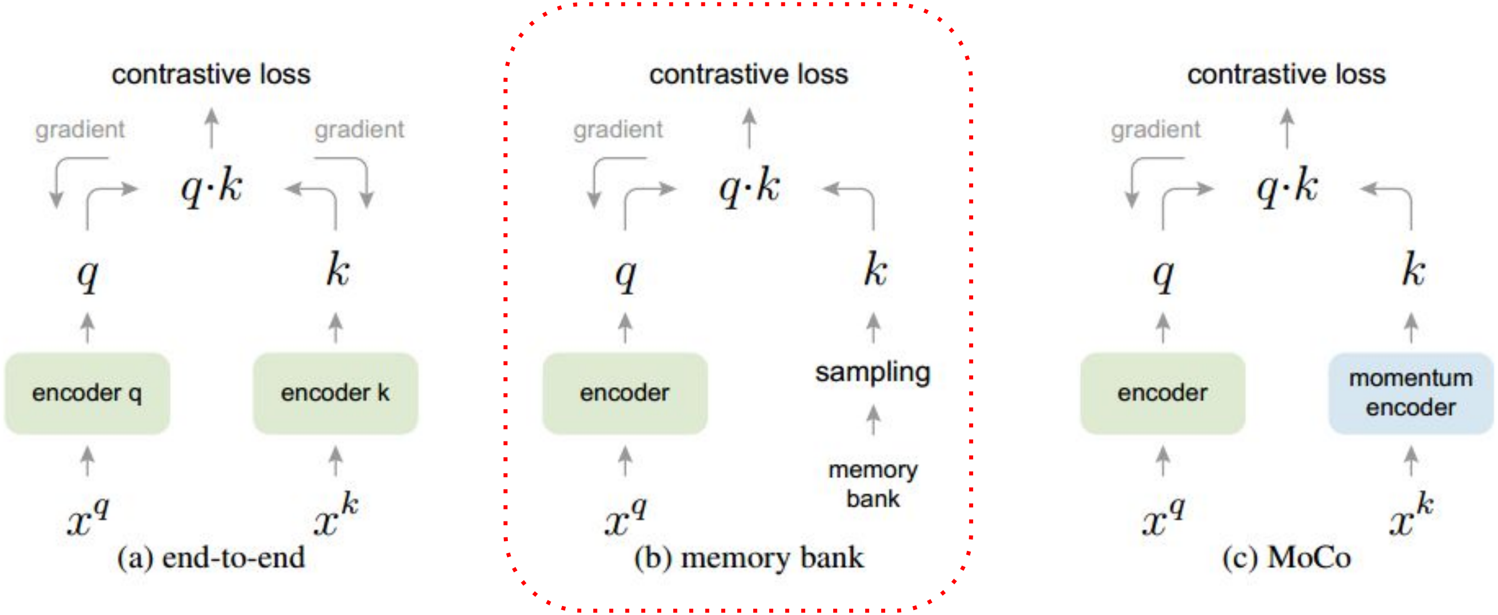
Related Works



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

- Limited by the GPU memory size.

Related Works



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

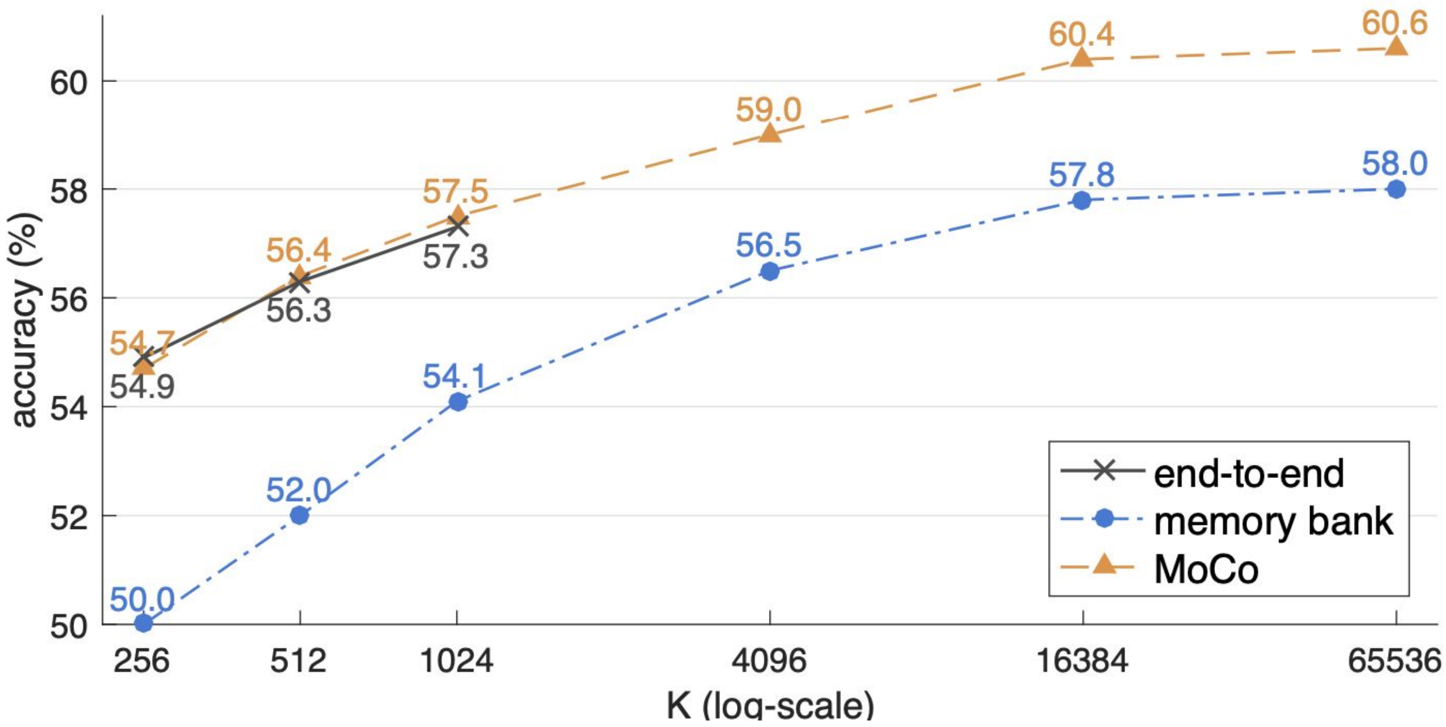
- Can't update memory bank, So less consistent

Shuffling BN

- Using BN prevents the model from learning good representations.
- The model appears to “cheat” the pretext task and easily find a low-loss solution.
- This is possibly because the intra-batch communication among samples leaks information.
- Shuffle the sample order in the current mini-batch before distributing it among GPUs

Experiments

ImageNet



accuracy (%)
60
58
56
54
52
50
60.6
60.4
59.0
57.5
56.4
54.7
58.0
57.8
56.5
54.1
52.0
50.0
57.3
56.3
54.9
end-to-end
memory bank
MoCo
256
512
1024
4096
16384
65536
K (log-scale)

ImageNet

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]	$\text{R170}^*_{\text{wider}}$	303	65.9
CMC [56]	$\text{R50}_{\text{L+ab}}$	47	$64.1^\dagger$
	$\text{R50w2}\times_{\text{L+ab}}$	188	$68.4^\dagger$
AMDIM [2]	$\text{AMDIM}_{\text{small}}$	194	$63.5^\dagger$
	$\text{AMDIM}_{\text{large}}$	626	$68.1^\dagger$
MoCo	R50	24	60.6
	RX50	46	63.9
	R50w2×	94	65.4
	R50w4×	375	**68.6**

Momentum

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

Shuffling BN

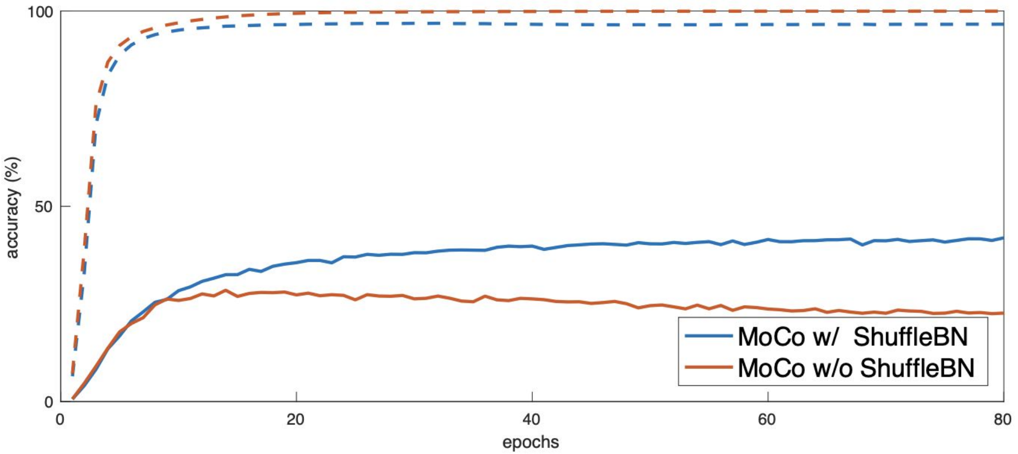


Figure A.1. **Ablation of Shuffling BN**. *Dash*: training curve of the pretext task, plotted as the accuracy of $(K{+}1)$-way dictionary lookup. *Solid*: validation curve of a kNN-based monitor [61] (not a linear classifier) on ImageNet classification accuracy. This plot shows the first 80 epochs of training: training longer without shuffling BN overfits more.

Thank you.