Momentum Contrast for Unsupervised Visual Representation Learning

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

FAIR

Digital signal processing Lab
Presenter: KIM JONGHYUN

Content

001 Concept

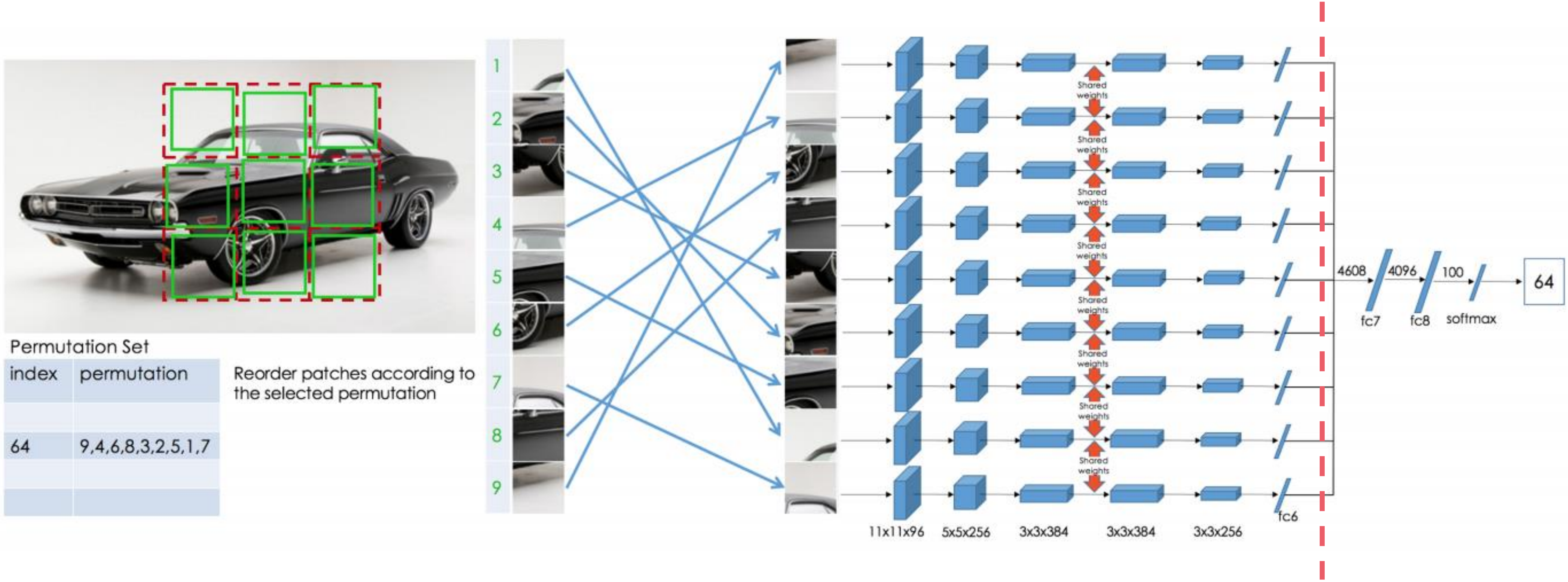
Unsupervised Learning

A main purpose of unsupervised learning is to pre-train representations (i.e., features) that can be transferred to downstream tasks by fine-tuning.

1

001 Concept

Examples(Jigsaw)

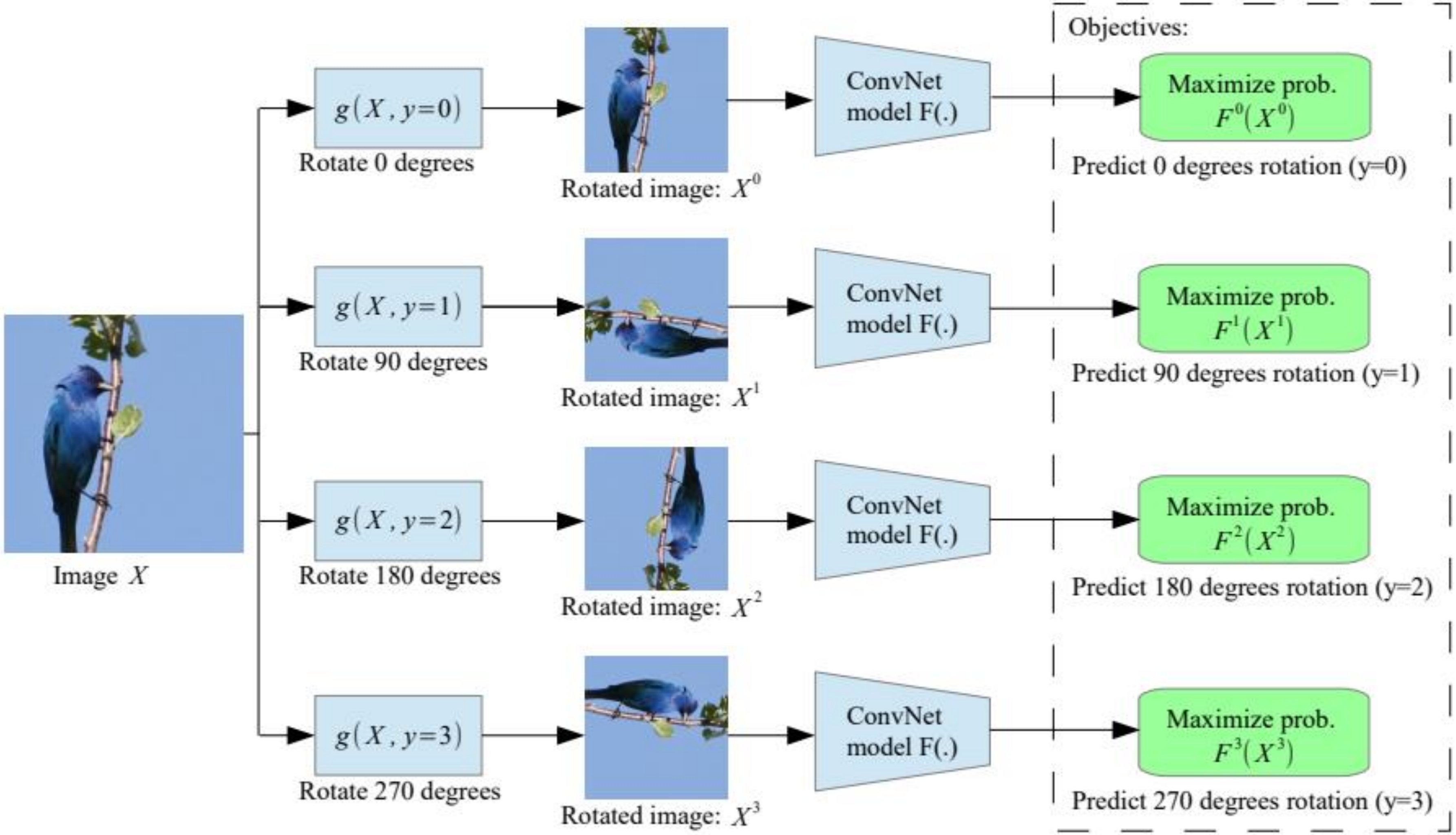


Predict the permutation index

1

001 Concept

Examples(Rotation)



Predict the rotation

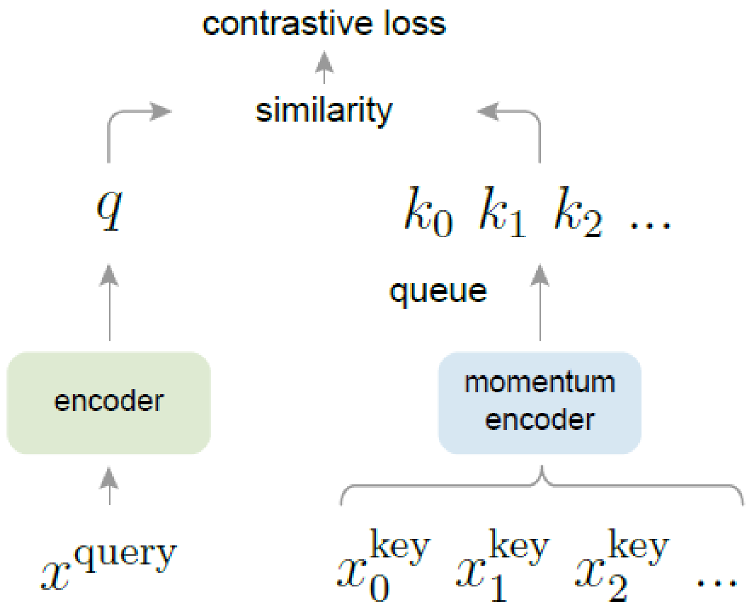
002 MOCO

MOCO

✓Not specific domains

✓Extract good features

Idea



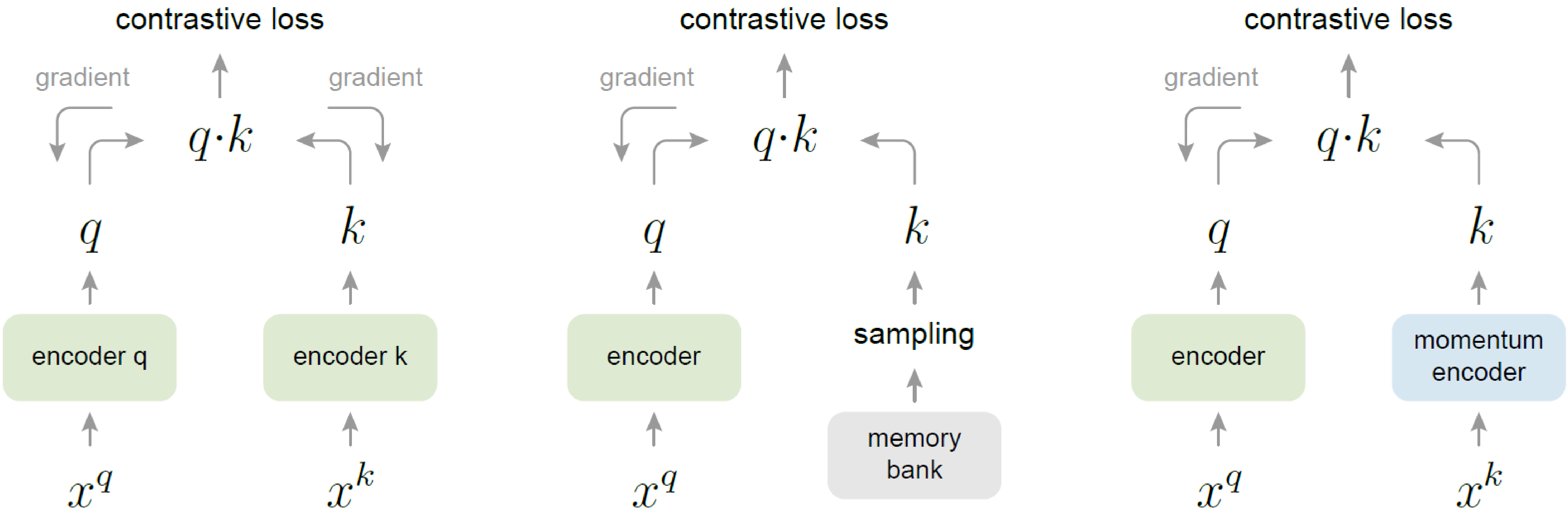
x^{query}: $inputs\ (i.e.\ images, patches, context)$

x^{key}: $samples$

k_i: $output\ of\ the\ encoder$

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

Difference

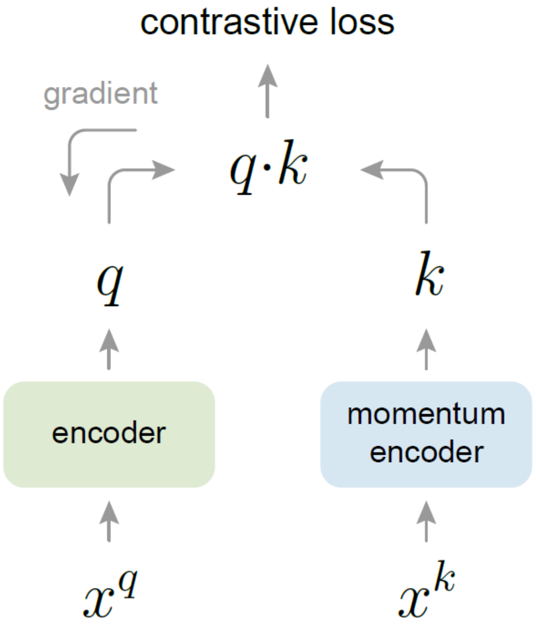


(a) end-to-end (b) memory bank (c) MoCo

✓ Limitation of size of dictionary

✓ Random sample from memory bank

Loss



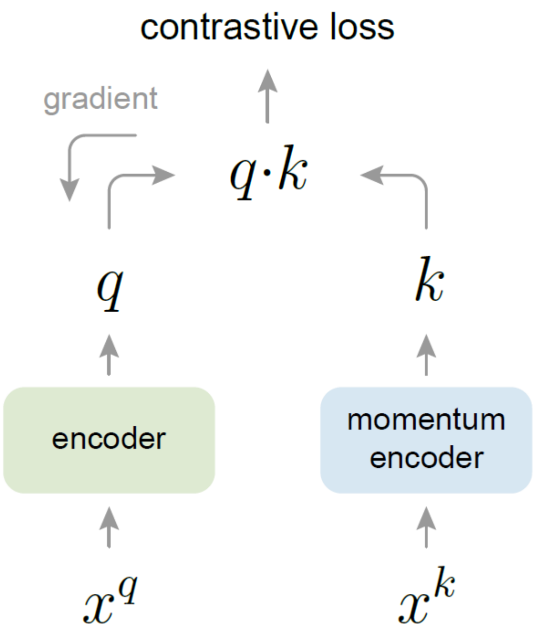
k_+: *Unique key similar to query*

k_i: *A set of encoded keys*

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

(K+1)-way softmax-based classifier that tries to classify q as k+

Loss



θ_k & θ_q: *parameters of each encoder*

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

, *where* m *is a momentum coefficient*

$m \in [0,1)$

✓ **Large momentum works much better than a small value**

Update

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
```

Encode input

```
   # 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))
```

Dot product

```
   # 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)
```

Cross entropy

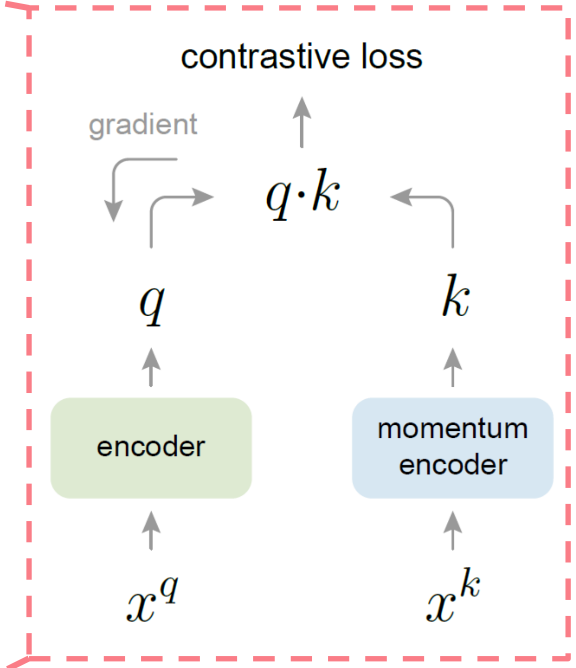
```
   # SGD update: query network
   loss.backward()
   update(f_q.params)
```

Update encoder

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

Update m_encoder

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



Experiment

✓ ImageNet-1M

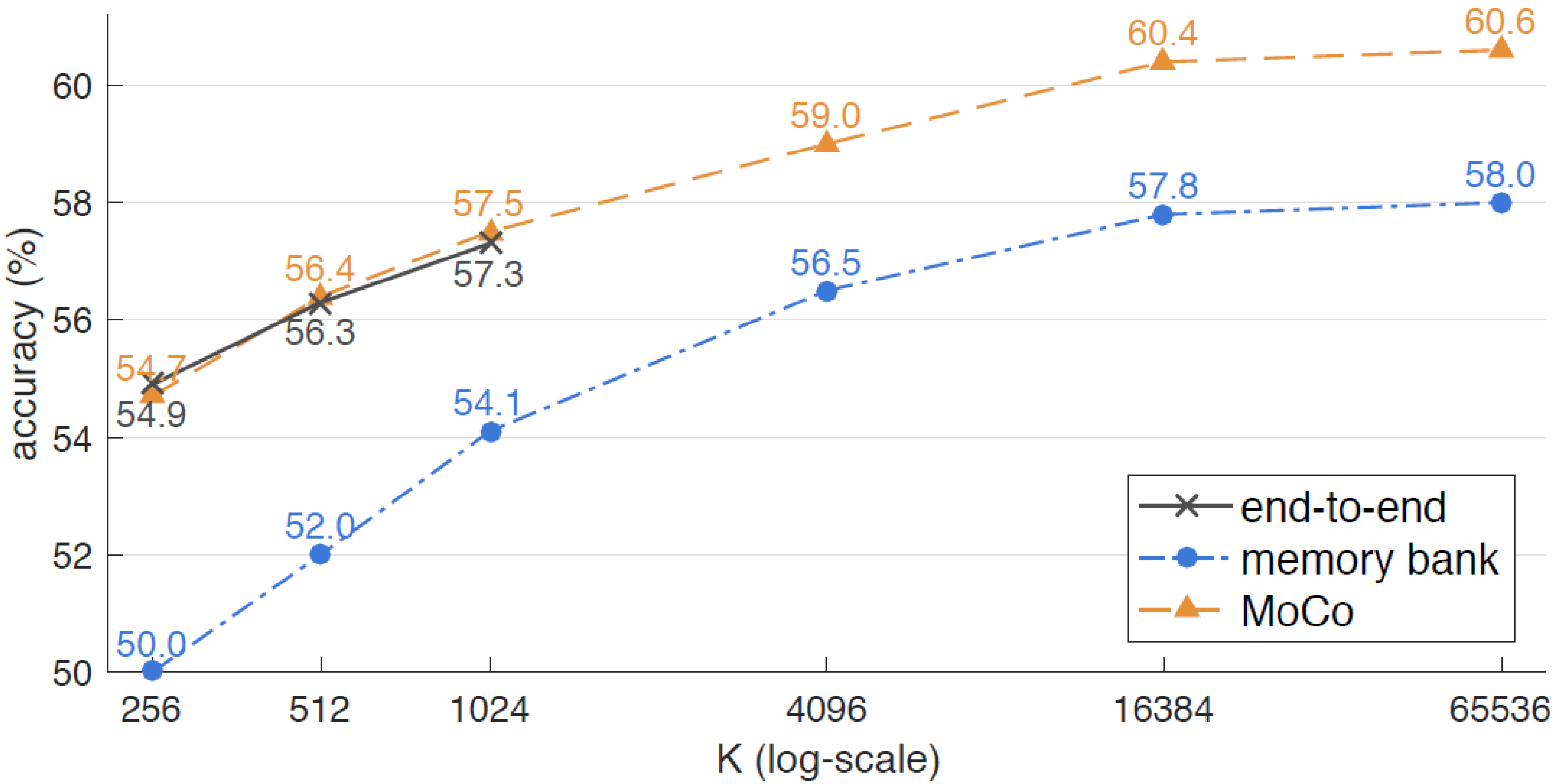
- ✓ 1.28 million images in 1000 classes
- ✓ SGD optimizer with weight decay 0.0001 and momentum 0.9
- ✓ 256 mini-batch in 8GPUs
- ✓ Initial learning rate 0.03 / multiplied by 0.1 at 120 & 160 epochs
- ✓ Total epoch 200
- ✓ ResNet-50 backbone

✓ Instagram-1B

- ✓ 940 million images from Instagram with 1500 hashtags
- ✓ SGD optimizer with weight decay 0.0001 and momentum 0.9
- ✓ 1024 mini-batch in 64GPUs
- ✓ Initial learning rate 0.12 / exponentially decayed by 0.9 per 62.5k iter
- ✓ Total iter 1.25M
- ✓ ResNet-50 backbone

003 Result

Contrastive



Comparison of three contrastive loss mechanisms under the ImageNet linear classification protocol. K is the number of negatives.

Momentum

Ablation: momentum. The table below shows ResNet-50 accuracy with different MoCo momentum values (m in Eqn.(2)) used in pre-training ($K = 4096$ here) :

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

Larger momentum shows a better performance.

Application

Apply ResNet-50 pre-trained by unsupervised method to Object detection & Segmentation

⇨ ✓ **Outperform supervised method**

THANK

YOU