An Empirical Study of Training Self-Supervised Vision Transformers

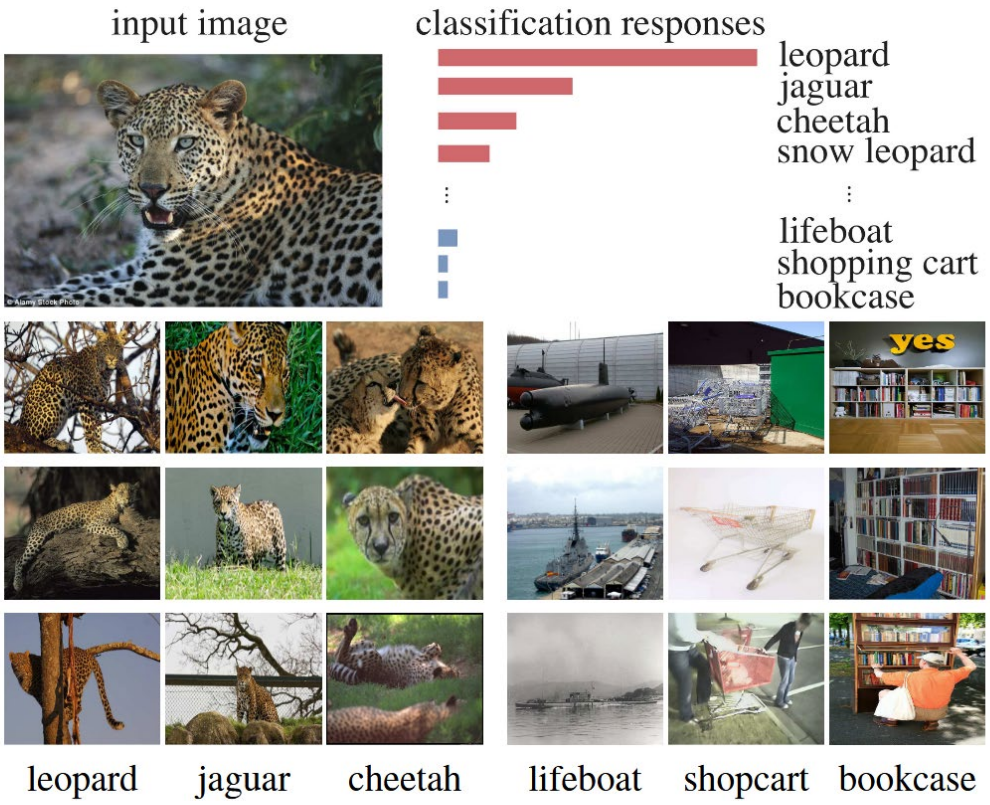
Xinlei Chen* Saining Xie* Kaiming He

CVPR 2021

Background

An Empirical Study of Training Self-Supervised Vision Transformers

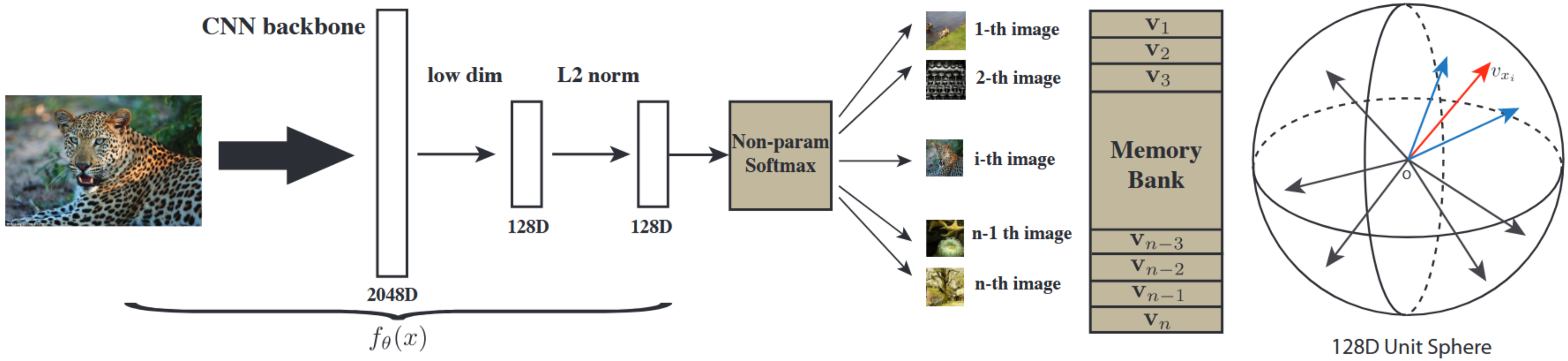
Xinlei Chen* Saining Xie* Kaiming He



- Unsupervised learning
- Large amounts of unlabeled data
- Pretext task
- Flexibility

Related Work

InstDisc CVPR 2018



- Data augmentation
- Memory bank
- NCE loss *details*
- Proximal Regularization

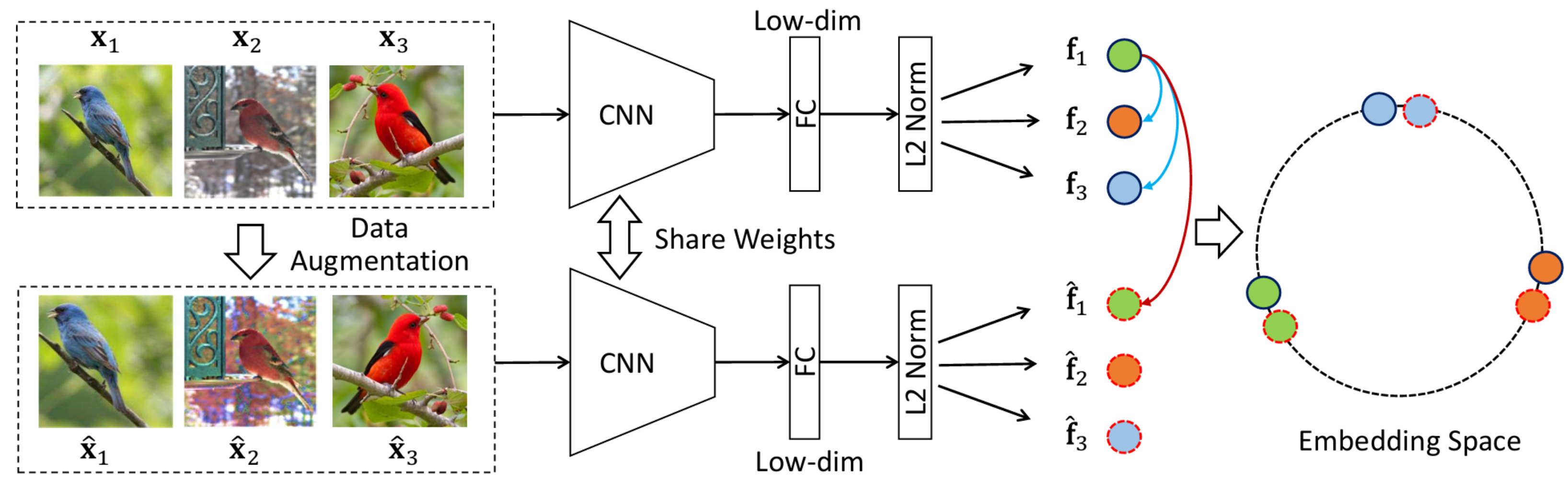
As learning converges, the difference between iterations, *i.e.* $\mathbf{v}_i^{(t)} - \mathbf{v}_i^{(t-1)}$, gradually vanishes, and the augmented loss is reduced to the original one. With proximal regularization, our final objective becomes:

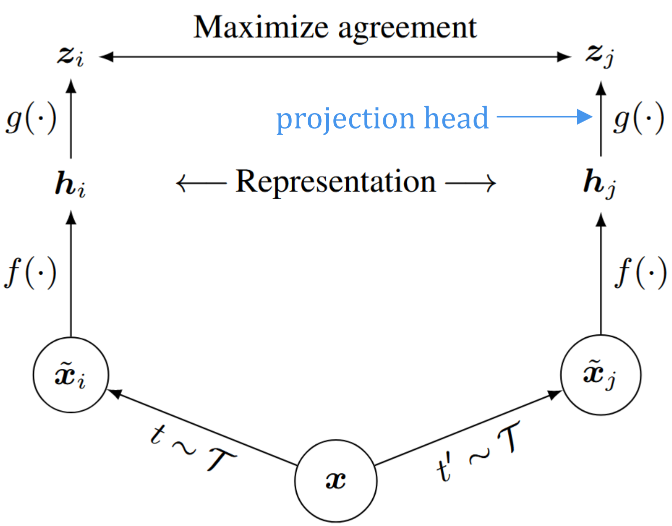
$$J_{NCE}(\boldsymbol{\theta}) = -E_{P_d}\left[\log h(i, \mathbf{v}_i^{(t-1)}) - \lambda \|\mathbf{v}_i^{(t)} - \mathbf{v}_i^{(t-1)}\|_2^2\right] - m \cdot E_{P_n}\left[\log(1 - h(i, \mathbf{v}'^{(t-1)}))\right]. \quad (10)$$

Wu, Zhirong, et al. "Unsupervised feature learning via non-parametric instance discrimination." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

Related Work

SimCLR ICML 2020





1st transformation \ 2nd transformation	Crop	Cutout	Color	Sobel	Noise	Blur	Rotate	Average
Crop	33.1	33.9	56.3	46.0	39.9	35.0	30.2	39.2
Cutout	32.2	25.6	33.9	40.0	26.5	25.2	22.4	29.4
Color	55.8	35.5	18.8	21.0	11.4	16.5	20.8	25.7
Sobel	46.2	40.6	20.9	4.0	9.3	6.2	4.2	18.8
Noise	38.8	25.8	7.5	7.6	9.8	9.8	9.6	15.5
Blur	35.1	25.2	16.6	5.8	9.7	2.6	6.7	14.5
Rotate	30.0	22.5	20.7	4.3	9.7	6.5	2.6	13.8

Limit :
- GPU memory
- convergence

Advantage :
- consistency

Chen, Ting, et al. "A simple framework for contrastive learning of visual representations." *International conference on machine learning*. PMLR, 2020.

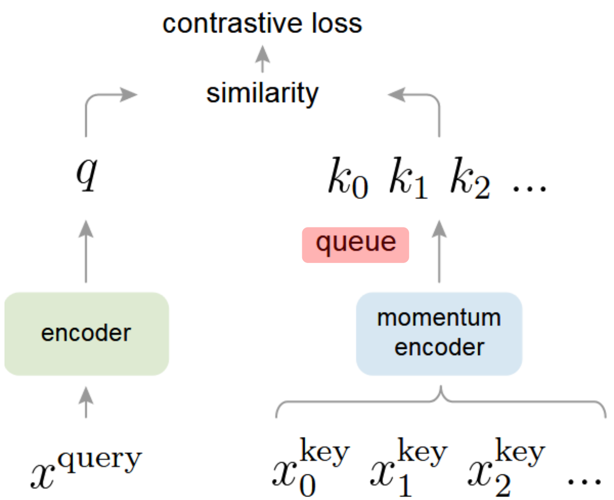
Related Work

MoCo-v1 CVPR 2020

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.



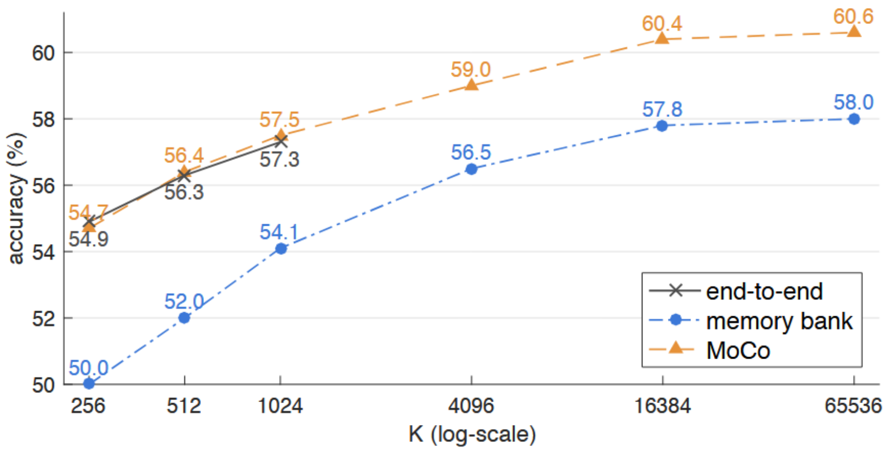
- InfoNCE

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

- Momentum Contrast

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

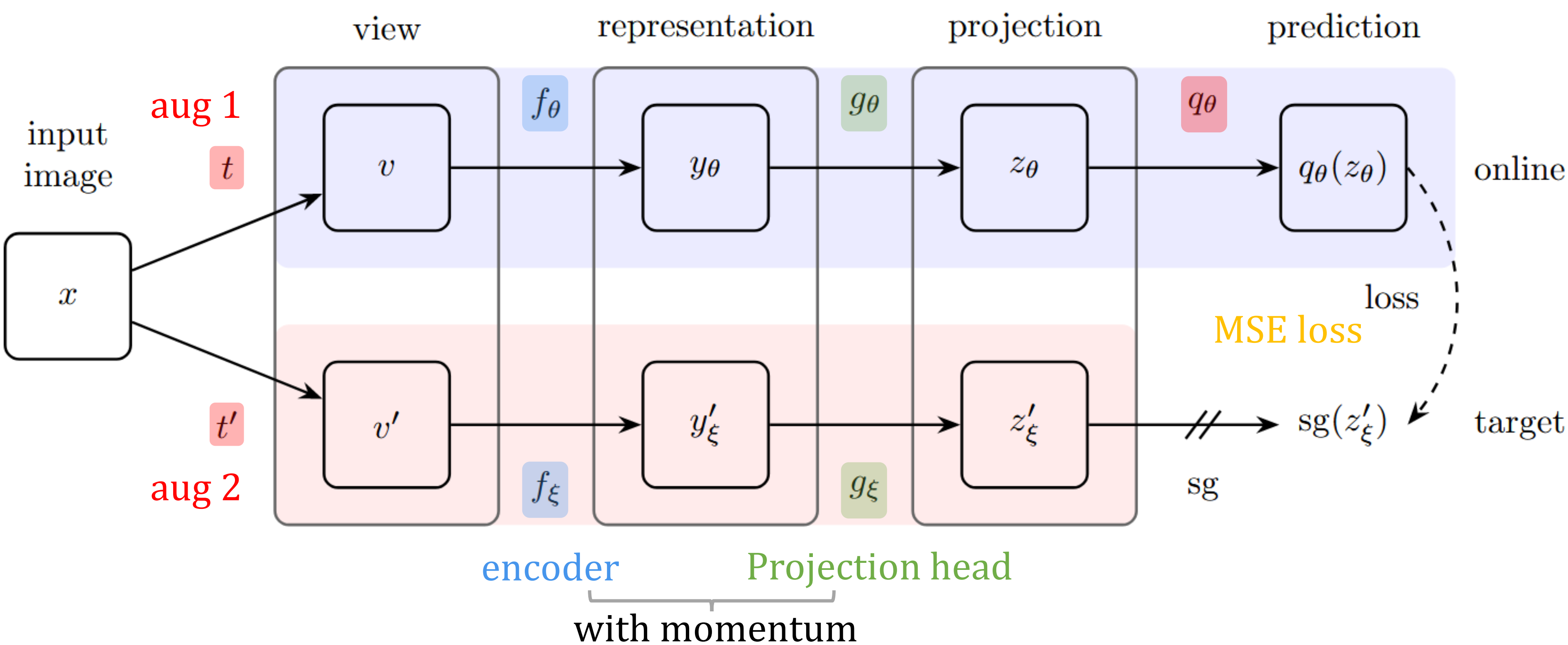
$m = 0.999$ *(0.99 in MoCo-v3)*



He, Kaiming, et al. "Momentum contrast for unsupervised visual representation learning." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

Related Work

BYOL NeurIPS 2020

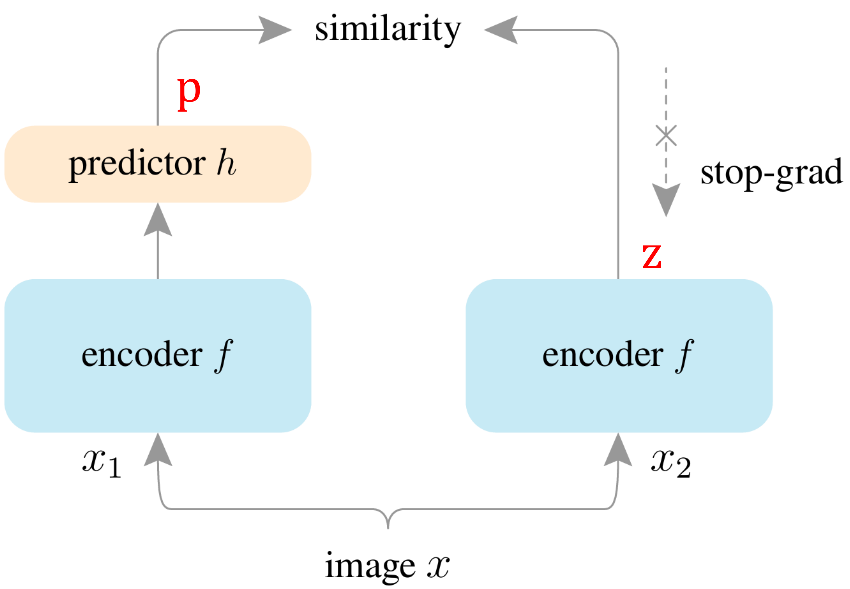


train the online network to predict the target network representation of the same image under a different augmented view

Grill, Jean-Bastien, et al. "Bootstrap your own latent-a new approach to self-supervised learning." *Advances in neural information processing systems* 33 (2020): 21271-21284.

Related Work

SimSam NeurIPS 2020



Algorithm 1 SimSiam Pseudocode, PyTorch-like

```
# f: backbone + projection mlp
# h: prediction mlp

for x in loader: # load a minibatch x with n samples
    x1, x2 = aug(x), aug(x) # random augmentation
    z1, z2 = f(x1), f(x2) # projections, n-by-d
    p1, p2 = h(z1), h(z2) # predictions, n-by-d

    L = D(p1, z2)/2 + D(p2, z1)/2 # loss

    L.backward() # back-propagate
    update(f, h) # SGD update

def D(p, z): # negative cosine similarity
    z = z.detach() # stop gradient

    p = normalize(p, dim=1) # l2-normalize
    z = normalize(z, dim=1) # l2-normalize
    return -(p*z).sum(dim=1).mean()
```

method	batch size	negative pairs	momentum encoder	100 ep	200 ep	400 ep	800 ep
SimCLR (repro.+)	4096	✓		66.5	68.3	69.8	70.4
MoCo v2 (repro.+)	**256**	✓	✓	67.4	69.9	71.0	72.2
BYOL (repro.)	4096		✓	66.5	**70.6**	**73.2**	**74.3**
SwAV (repro.+)	4096			66.5	69.1	70.7	71.8
SimSiam	**256**			**68.1**	70.0	70.8	71.3

$$\mathcal{D}(p_1, z_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{z_2}{\|z_2\|_2}$$

$$\mathcal{L} = \frac{1}{2}\mathcal{D}(p_1, \texttt{stopgrad}(z_2)) + \frac{1}{2}\mathcal{D}(p_2, \texttt{stopgrad}(z_1)).$$

Chen, Xinlei, and Kaiming He. "Exploring simple siamese representation learning." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021.

Method

technique 1 : Network Architecture

Algorithm 1 MoCo v3: PyTorch-like Pseudocode

```
# f_q: encoder: backbone + proj mlp + pred mlp
# f_k: momentum encoder: backbone + proj mlp
# m: momentum coefficient
# tau: temperature

for x in loader: # load a minibatch x with N samples
   x1, x2 = aug(x), aug(x) # augmentation
   q1, q2 = f_q(x1), f_q(x2) # queries: [N, C] each
   k1, k2 = f_k(x1), f_k(x2) # keys: [N, C] each

   loss = ctr(q1, k2) + ctr(q2, k1) # symmetrized
   loss.backward()

   update(f_q) # optimizer update: f_q
   f_k = m*f_k + (1-m)*f_q # momentum update: f_k

# contrastive loss
def ctr(q, k):
   logits = mm(q, k.t()) # [N, N] pairs
   labels = range(N) # positives are in diagonal
   loss = CrossEntropyLoss(logits/tau, labels)
   return 2 * tau * loss
```

Notes: `mm` is matrix multiplication. `k.t()` is `k`'s transpose. The prediction head is excluded from `f_k` (and thus the momentum update).

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

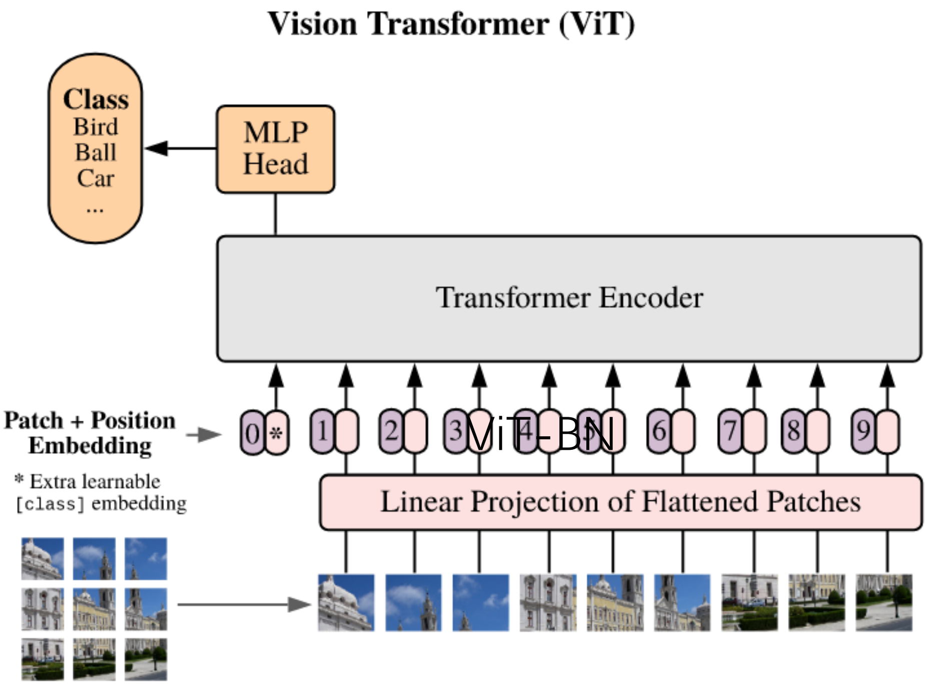
0.99	0.06	0.28	0.55	0.50	0.36	0.22	0.22	0.70	0.10
0.55	0.96	0.73	0.41	0.37	0.77	0.02	0.43	0.24	0.65
0.56	0.46	1.00	0.13	0.28	0.68	0.72	0.64	0.49	0.66
0.31	0.42	0.27	0.91	0.19	0.49	0.02	0.30	0.02	0.18
0.37	0.32	0.12	0.06	0.97	0.26	0.62	0.33	0.14	0.55
0.29	0.01	0.44	0.05	0.19	0.99	0.78	0.12	0.25	0.54
0.40	0.12	0.61	0.20	0.63	0.74	0.98	0.61	0.23	0.17
0.01	0.44	0.26	0.65	0.79	0.05	0.18	0.90	0.11	0.09
0.33	0.75	0.67	0.12	0.02	0.40	0.46	0.03	0.98	0.05
0.48	0.77	0.23	0.61	0.46	0.73	0.60	0.29	0.13	0.96

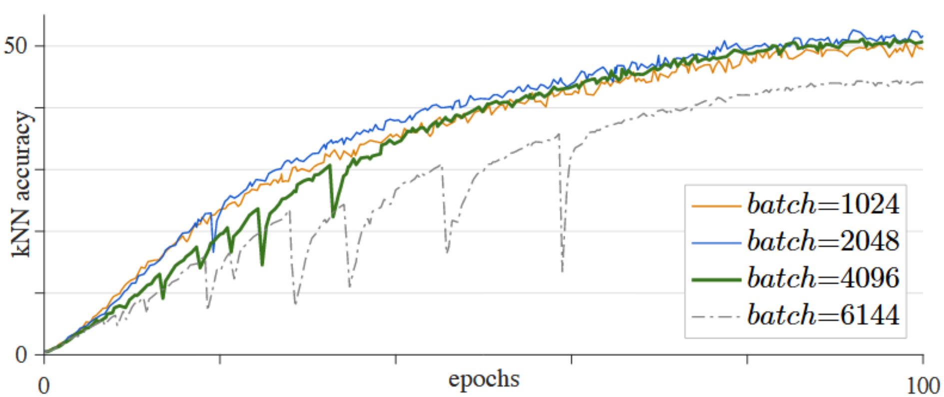
R50, 800-ep	MoCo v2 [12]	MoCo v2+ [13]	MoCo v3
linear acc.	71.1	72.2	**73.8**

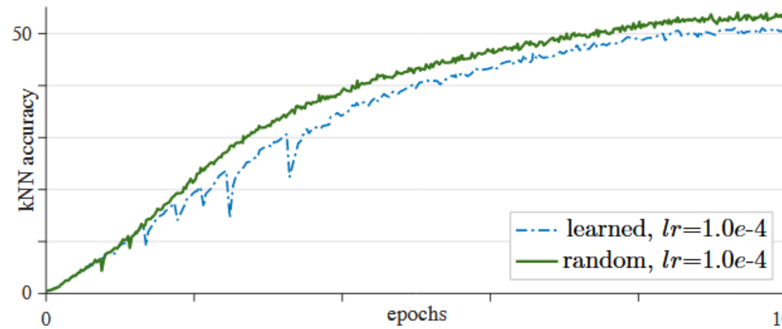
The improvement here is mainly due to the extra prediction head and large-batch (4096) training.

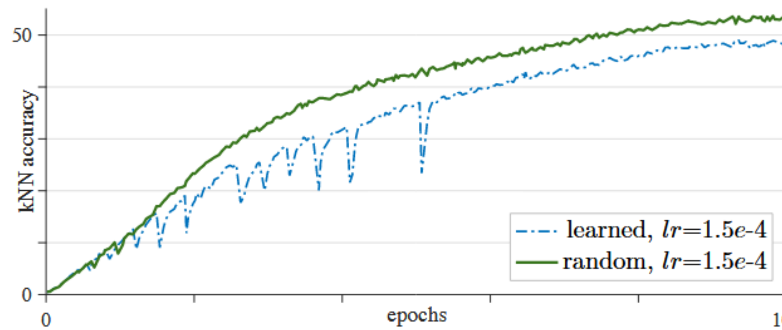
Method

technique 2 : Fixed Random Patch Projection









lr, $\times 10^{-4}$	0.5	1.0	1.5
learned patch proj.	70.4	72.2	71.7
random patch proj.	**70.8**	**72.8**	**73.4**

For the standard ViT patch size, the patch projection matrix is complete (768-d output for a 3-channel 16×16 patch) or overcomplete.

Method

technique 3 : BatchNorm in ViT

framework	model	params	acc. (%)
linear probing:			
iGPT [9]	iGPT-L	1362M	69.0
iGPT [9]	iGPT-XL	6801M	72.0
MoCo v3	ViT-B	86M	76.7
MoCo v3	ViT-L	304M	77.6
MoCo v3	ViT-H	632M	78.1
MoCo v3	ViT-BN-H	632M	79.1
MoCo v3	ViT-BN-L/7	304M	**81.0**
end-to-end fine-tuning:			
masked patch pred. [16]	ViT-B	86M	79.9†
MoCo v3	ViT-B	86M	83.2
MoCo v3	ViT-L	304M	**84.1**

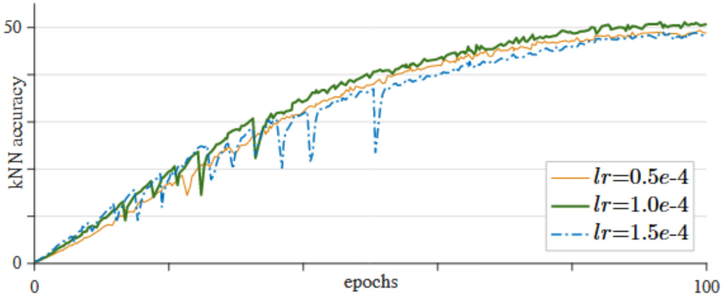
We notice that this comparison concerns a composition of many choices. As one example, the default ViT backbone in [16] uses LayerNorm (LN), while the default ResNet [21] uses BatchNorm (BN). These design choices can lead to a systematic gap. In our preliminary experiments, we explore replacing LN with BN in the ViT backbone's MLP blocks (*i.e.*, excluding self-attention blocks).[6] We simply refer to this as a "ViT-BN" backbone. It leads to ~1% improvement consistently (see Fig. 8).

We have to set the batch size as 2048 when removing BN, otherwise it does not converge. Removing BN reduces accuracy by 2.1%. Despite the decrease, this is a completely *BN-free* system. This data point suggests that BN is not necessary for contrastive learning to work, yet appropriate usage of BN can improve accuracy.

Experiment

Learning Rate:

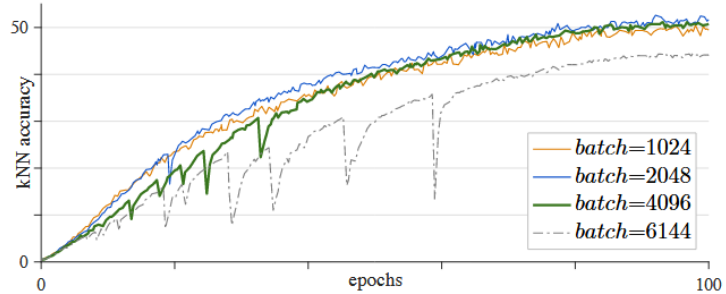
lr×BatchSize/256



lr, ×1e-4	0.5	1.0	1.5
linear acc.	70.4	72.2	71.7

Figure 2. **Training curves of different learning rates** (MoCo v3, ViT-B/16, 100-epoch ImageNet, AdamW, batch 4096).

Batch Size:

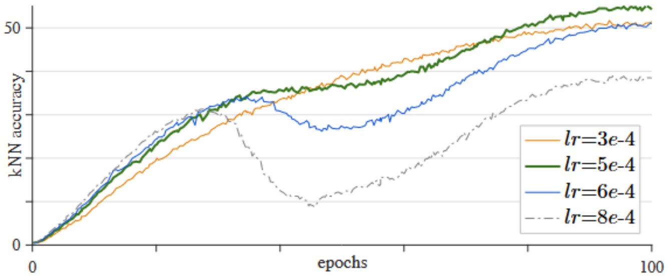


batch	1024	2048	4096	6144
linear acc.	71.5	72.6	72.2	69.7

Figure 1. **Training curves of different batch sizes** (MoCo v3, ViT-B/16, 100-epoch ImageNet, AdamW, lr=1.0e-4).

Optimizer:

AdamW, LARS, LAMB (sensitive)



lr, ×1e-4	3.0	5.0	6.0	8.0
linear acc.	71.6	72.5	70.9	66.5

Figure 3. **Training curves of LAMB optimizer** (MoCo v3, ViT-B/16, 100-epoch ImageNet, wd=1e-3, batch 4096).

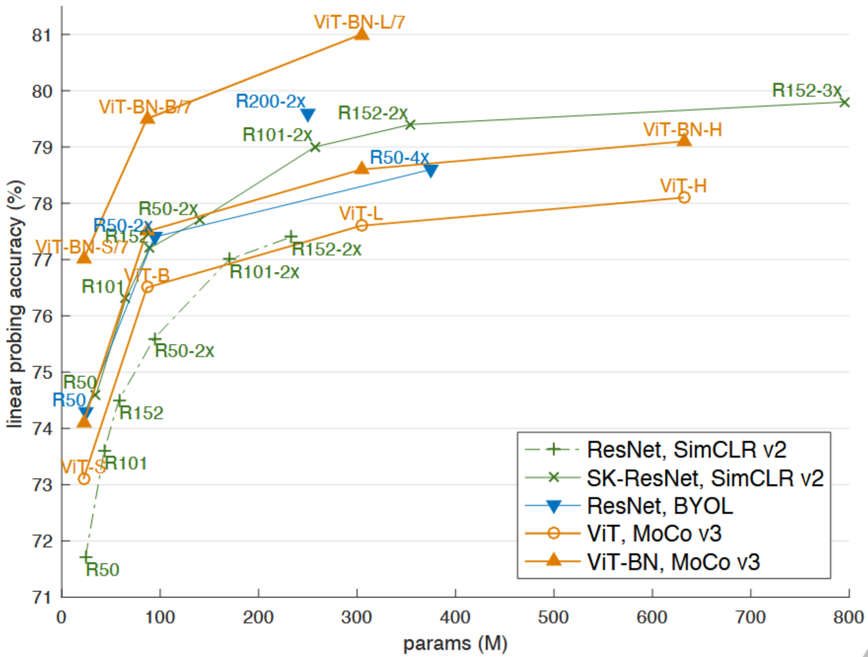
Training Time:

model	FLOPs	*vs.* R50	TPUs	hours
ViT-S/16	4.6 G	1.1×	256	1.2
ViT-B/16	17.5 G	4.3×	256	2.1
ViT-L/16	61.3 G	15.0×	256	6.1
ViT-H/14	166.7 G	40.7×	512	9.8

Table 3. **Training time of ViT + MoCo v3**, per 100 ImageNet-epochs, in our TensorFlow implementation. The FLOPs number (in multiply-adds) is per 224×224 crop, and "*vs.* R50" is the relative FLOPs *vs.* ResNet-50 (4.1G).

	300-ep	600-ep
ViT-S/16	72.5	73.4
ViT-B/16	76.5	76.7

Result



MoCo v3 w/	ViT-S	ViT-B	ViT-L	ViT-H
ViT baseline	73.4	76.7	77.6	78.1
ViT-BN	74.1	77.5	78.6	79.1
ViT-BN/7	77.0	79.5	**81.0**	-

case	pre-train	ViT-S	ViT-B	ViT-L
masked patch pred. [16]	JFT-300M	-	79.9	-
DeiT [41]	-	79.9	81.8	n/a
MoCo v3	ImageNet-1k	**81.4**	**83.2**	**84.1**

Table 5. **End-to-end fine-tuning** accuracy (%) in ImageNet-1k.

Figure 8. **Comparisons with state-of-the-art big ResNets**, presented as parameters-*vs*.-accuracy trade-off. All entries are pre-trained with two 224×224 crops, and are evaluated by linear probing. SimCLR v2 results are from Table 1 in [11], and BYOL results are from Table 1 in [18].

Finally, we note that with supervised pre-training in bigger datasets (ImageNet-21k or JFT-300M), the ViT results in [16] can be better than ours when transferring to these small datasets. A potential future work is to perform

	CIFAR-10 [25]			CIFAR-100 [25]			Oxford Flowers-102 [32]			Oxford-IIIT-Pets [34]		
pre-train	ViT-B	ViT-L	ViT-H	ViT-B	ViT-L	ViT-H	ViT-B	ViT-L	ViT-H	ViT-B	ViT-L	ViT-H
random init.	77.8	77.1	75.9	48.5	48.3	48.0	54.4	54.3	52.8	40.1	42.8	40.4
ImNet supervised [16]	98.1	97.9	n/a	87.1	86.4	n/a	89.5	89.7	n/a	93.8	93.6	n/a
ImNet self-sup., MoCo v3	98.9 ↑0.8	99.1 ↑1.2	99.1	90.5 ↑3.4	91.1 ↑4.7	91.2	97.7 ↑8.2	98.6 ↑8.9	98.8	93.2 ↓0.6	93.7 ↑0.1	94.2

Table 6. **Transfer learning** accuracy (%) in four datasets. All entries are end-to-end fine-tuned [16]. Pre-training are performed in the ImageNet-1k training set. The models are ViT-B/16, ViT-L/16, and ViT-H/14. Results of ImageNet-supervised pre-training are from Table 3 in [16]. The arrows indicate the changes w.r.t. the ImageNet-supervised counterparts.

Thanks!