An Empirical Study of Training SSL Transformers

Tuesday, 17. August 2021 11:34 PM

- Studies current SSL methods and benchmarks in MocoV3
- Instability is a huge issue and can be hidden by good results

Intro :

Unsupervised Pretrainig in Vision vs NLP

Method	Vision	NLP
Paradigm	Siamese Nets	Masked Auto-Encoders
Backbone	Conv	Transformers

=> Differences are important to understand the gap between NLP and vision

- Paper studies methods based on Siamese Nets (MOCO, BYOL, SWAV, SimCLR)
- VIT training has no recipe yet
- Instability is sometimes an issue with VIT, but does not necessarily lead to catastrophi forgetting but rather a drop in accuracy (rare with ConvNets)
  => but : Trick to improve this is given -> based on empirical observation of the gradien patch projection layer is frozen
- Prove that SSL can outperform SL especially as it can benefit from model scaling => fev biases

Related Work

- SSL (contrastive learning) can outperform SL in certain tasks => positive/negative sam
  => recent works remove negative sampling => important to learn invariant features by matching positive samples
- Transformers (VIT allows large scaling with huge data and closes gap to NLP), others a based on it (SWIN, DEIT)
- SSL for vision follow NLP, with masks and pixel reconstruction , but in SSL the loss does use reconstruction of the inputs

MocoV3

MocoV3

- Improvement of MocoV1/V2

As common practice (*e.g.*, [20, 10]), we take two crops for each image under random data augmentation. They are encoded by two encoders, f_q and f_k, with output vectors q and k. Intuitively, q behaves like a "query" [20], and the goal of learning is to retrieve the corresponding "key". This is formulated as minimizing a contrastive loss function [19]. We adopt the form of InfoNCE [34]:

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

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).

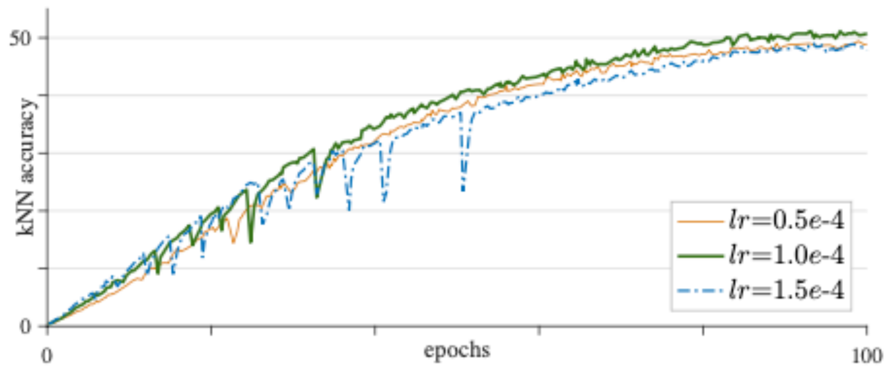
- Use keys which are in the batch, abandon memory queue
- Encoder : resnet backboone, projection and prediction head
- Improvement over MocoV1/2 :

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.

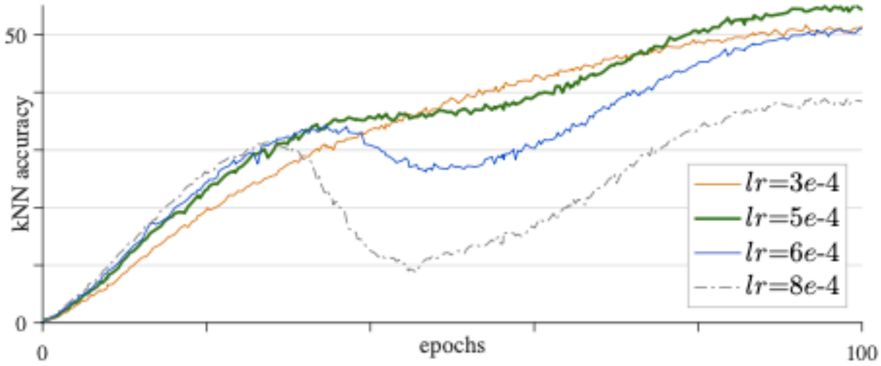
Stability of VIT Training

- In principle easy to replace ResNet with VIT, in practice instabilities
- With some batch sizes, training sort of "partially restarts" and accuracy is based on how good the restart point is, (1k/2k batch are fine, 6k batch suffers from instabilties)
- Problem : only small difference => can be fully hidden (difference in 0.1- 0,3 % range)
- Learning rate : set to base_lr * BatchSize/256 :
  - smaller lr : more stable training, prone to underfitting
- AdamW is used as optimizer



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).



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).

- Also : LARS optim for large batch training, LAMB which is LARS adapted to ADAMW can outperform AdamW but it is more unstable with different learning rates leading to less performance (hyptothesis : it accumulates impact of unreliable gradients)

Trick for better stability

- Gradient change causes dip in the training curve
- Instability happens in shallower layers => freeze the patch projection layer during training
  => patch embedding which is not learned (stop gradient operation)
  => boosts accuracy by 1.7%
- Also helps with Simclr and byol
- Allows larger lr for SwaV

We have also tried BatchNorm (BN) [24], WeightNorm (WN) [40], or gradient clip on patch projection. We observe that BN or WN on the learnable patch projection layer does not improve instability, and produces similar results; gradient clip on this layer is useful if given a sufficiently small threshold, which to the extreme becomes freezing the layer.

- Authors hypothesize that more stability issues relating later layers exist (in the Transformer), but fixing the projection is easier to handle as it is separate

Linear probing. Following common practice, we evaluate the representation quality by linear probing. After self-supervised pre-training, we remove the MLP heads and train a supervised linear classifier on frozen features. We use the SGD optimizer, with a batch size of 4096, wd of 0, and sweep lr for each case. We train this supervised classifier for 90 epochs in the ImageNet training set, using only random resized cropping and flipping augmentation. We evaluate single-crop top-1 accuracy in the validation set.