An Empirical Study of Training Self-Supervised Vision Transformers

Xinlei Chen* Saining Xie* Kaiming He
Facebook AI Research (FAIR)

Code: https://github.com/facebookresearch/moco-v3

Abstract

This paper does not describe a novel method. Instead, it studies a straightforward, incremental, yet must-know baseline given the recent progress in computer vision: self-supervised learning for Vision Transformers (ViT). While the training recipes for standard convolutional networks have been highly mature and robust, the recipes for ViT are yet to be built, especially in the self-supervised scenarios where training becomes more challenging. In this work, we go back to basics and investigate the effects of several fundamental components for training self-supervised ViT. We observe that instability is a major issue that degrades accuracy, and it can be hidden by apparently good results. We reveal that these results are indeed partial failure, and they can be improved when training is made more stable. We benchmark ViT results in MoCo v3 and several other self-supervised frameworks, with ablations in various aspects. We discuss the currently positive evidence as well as challenges and open questions. We hope that this work will provide useful data points and experience for future research.

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**

Table 1. **State-of-the-art Self-supervised Transformers** in ImageNet classification, evaluated by linear probing (top panel) or end-to-end fine-tuning (bottom panel). Both iGPT [9] and masked patch prediction [16] belong to the masked auto-encoding paradigm. MoCo v3 is a contrastive learning method that compares two (224×224) crops. ViT-B, -L, -H are the Vision Transformers proposed in [16]. ViT-BN is modified with BatchNorm, and "/7" denotes a patch size of 7×7. †: pre-trained in JFT-300M.

1. Introduction

Unsupervised pre-training has revolutionized natural language processing (NLP) [37, 15, 38, 4]. In computer vision, the un-/self-supervised pre-training paradigms differ from their NLP counterparts in at least two aspects: (i) the learners in NLP are masked auto-encoders, while in vision the recently popular choices are Siamese networks (*e.g.*, [20, 10, 18, 7]); (ii) the backbone architectures in NLP are self-attentional Transformers [43], while in vision the common choice is convolutional [28]—yet non-attentional—deep residual networks (ResNets) [21]. To complete the big picture of self-supervised learning in vision, and towards closing the gap of pre-training methodology between vision and language, it is of scientific merit to investigate these differences.

This work focuses on training Transformers with the leading self-supervised frameworks in vision. This investigation is a straightforward extension given the recent progress on Vision Transformers (ViT) [16]. In contrast to prior works [9, 16] that train self-supervised Transformers with masked auto-encoding, we study the frameworks that are based on Siamese networks, including MoCo [20] and others [10, 18, 7].

Unlike standard convolutional networks whose training practice has been extensively studied thanks to continuous community effort, ViT models are new and their recipes are yet to be established. In this work, we go back to basics and investigate the fundamental components of training deep neural networks: the batch size, learning rate, and optimizer. We find that under various cases, instability is a major issue that impacts self-supervised ViT training.

Interestingly, we observe that unstable ViT training may *not* result in catastrophic failure (*e.g.*, divergence); instead, it can cause *mild* degradation in accuracy (*e.g.*, 1∼3%). Such a degree of degradation may not be noticeable, unless a more stable counterpart is available for comparison. To the best of our knowledge, this phenomena is rare in the literature of training convolutional networks[1], and we believe this problem and its hidden degradation are worth noticing.

To demonstrate the possible harm of instability, we investigate a simple trick that can improve stability in practice. Based on an empirical observation on gradient changes, we freeze the patch projection layer in ViT, *i.e.*, we use fixed random patch projection. We empirically show that this trick alleviates the instability issue in several scenarios and consistently increases accuracy.

*: equal contribution.

[1] See also postscript on a related discussion.

We benchmark and ablate self-supervised ViT in a variety of cases. We provide ViT results in several self-supervised frameworks. We conduct ablations on architecture designs and discuss the implications. Furthermore, we explore scaling up the ViT models, including the non-trivial ViT-Large and ViT-Huge [16] — the latter has 40× more computation than ResNet-50 [21]. Based on these experimental results, we discuss both the currently positive evidence as well as the challenges and open questions.

We report that self-supervised Transformers can achieve strong results using a contrastive learning framework, compared against masked auto-encoding (Table 1). This behavior of Transformers differs from the existing trend in NLP. Moreover, as a promising signal, our bigger self-supervised ViT can achieve better accuracy, unlike the ImageNet-supervised ViT in [16] whose accuracy degrades if getting bigger. For instance, for the very big ViT-Large, our self-supervised pre-training can outperform its supervised pre-training counterpart for transfer learning in certain cases. This presents a proof-of-concept scenario where self-supervised pre-training is needed.

In addition, we report that our self-supervised ViT models have competitive results *vs.* the *big* convolutional ResNets in prior art [11, 18]. On one hand, this comparison shows the potential of ViT, especially considering that it achieves these results using relatively "*fewer inductive biases*" [16]. On the other hand, we suggest that there could be room for self-supervised ViT models to further improve. As one example, we observe that *removing the position embedding* in ViT only degrades accuracy by a small margin. This reveals that self-supervised ViT can learn strong representations *without* the positional inductive bias, but it also implies that the positional information has not been sufficiently exploited.

In summary, we believe that the evidence, challenges, and open questions in this study are worth knowing, if self-supervised Transformers will close the gap in pre-training between vision and language. We hope our data points and experience will be useful to push this frontier.

2. Related Work

Self-supervised visual representation learning. In computer vision, contrastive learning [19] has become increasingly successful for self-supervised learning, *e.g.*, [45, 34, 22, 2, 20, 10]. The methodology is to learn representations that attract similar (positive) samples and dispel different (negative) samples. The representations from contrastive self-supervised pre-training can outperform their supervised counterparts in certain tasks [20, 10].

Contrastive learning is commonly instantiated as some forms of Siamese networks [3]. Recently, a series of works [18, 7, 13] retain the Siamese architectures but eliminate the requirement of negative samples. The success of these methods suggest that it is of central importance to learn invariant features by matching positive samples.

Transformers. Transformers [43] were originally introduced for machine translation and later became a dominant backbone in NLP [37, 15, 38, 4]. The long-range, self-attentional behavior makes Transformers an effective tool given the non-local, relational nature of languages.

There have been continuous efforts on generalizing Transformers to computer vision [44, 5, 39, 49, 6, 16]. The recent work on Vision Transformers (ViT) [16] greatly pushes this frontier. ViT is purely Transformer-based, rather than interlaced with non-degenerated (*i.e.*, non-1×1) convolutions.[2] This largely closes the architectural gap between NLP and vision. ViT achieves compelling accuracy in supervised learning, especially with large-scale data and high-capacity models. Given these properties, we believe ViT is a must-study baseline for self-supervised learning in computer vision.

Self-supervised Transformers for vision. In pioneering works [9, 16], training self-supervised Transformers for vision problems in general follows the *masked auto-encoding* paradigm in NLP [37, 15] (Table 1). iGPT [9] masks and reconstructs pixels, and the self-supervised variant of ViT in [16] masks and reconstructs patches. In this work, we focus on training Transformers in the contrastive/Siamese paradigm, in which the loss is not defined for reconstructing the inputs.

3. MoCo v3

We introduce a "MoCo v3" framework that facilitates our study. MoCo v3 is an incremental improvement of MoCo v1/2 [20, 12], and we strike for a better balance between simplicity, accuracy, and scalability. The pseudocode of MoCo v3 is in Alg. 1, described next.

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]:

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

Here k_+ is f_k's output on the same image as q, known as q's positive sample. The set $\{k^-\}$ consists of f_k's outputs

[2]We argue that it is *imprecise* to simply compare self-attention against "convolutions". Convolutions [28] by definition have several properties: weight-sharing, locally-connected, translation-equivariant. All projection layers in a self-attention block have all these properties of convolutions, and are equivalent to 1×1 convolutions. The counterpart of self-attention is more appropriately the non-degenerated (*e.g.*, 3×3) convolutions.

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).

from other images, known as q's negative samples. τ is a temperature hyper-parameter [45] for ℓ_2-normalized q, k.

Following [46, 22, 2, 10], in MoCo v3 we use the keys that naturally co-exist in the same batch. We abandon the memory queue [20], which we find has diminishing gain if the batch is sufficiently large (*e.g.*, 4096). With this simplification, the contrastive loss in (1) can be implemented by a few lines of code: see `ctr(q, k)` in Alg. 1. We adopt a symmetrized loss [18, 7, 13]: `ctr(q1, k2)+ctr(q2, k1)`.

Our encoder f_q consists of a backbone (*e.g.*, ResNet, ViT), a *projection* head [10], and an extra *prediction* head [18]; the encoder f_k has the backbone and projection head, but not the prediction head. f_k is updated by the moving-average of f_q [20], excluding the prediction head.

As a reference, we examine the MoCo v3 accuracy with ResNet-50 (R50) (detailed in appendix). This table compares the linear probing accuracy in ImageNet:

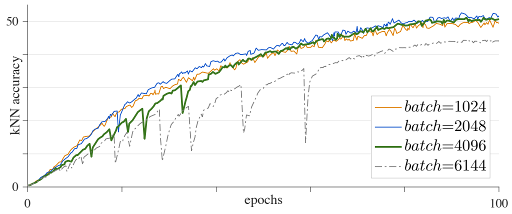
R50, 800-ep	MoCo v2 [12]	MoCo v2+ [13]	MoCo v3
linear acc.	71.1	72.2	**73.8**

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

4. Stability of Self-Supervised ViT Training

In principle, it is straightforward to replace a ResNet backbone with a ViT backbone in the contrastive/Siamese self-supervised frameworks. But in practice, a main challenge we have met is the *instability* of training.

We observe that the instability problem can not be simply reflected by accuracy numbers. In fact, as we will show, the training is "apparently good" and provides decent results, even when it is potentially unstable. To reveal the instability, we monitor the kNN curves [45] (see appendix) during



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).

training. In Sec. 4.1, we study how the basic factors influence stability. The curves suggest that the training can be "partially successful", or in other words, "partially failed". In Sec. 4.2, we explore a simple trick that can improve stability. As a result, the accuracy is improved in various cases.

4.1. Empirical Observations on Basic Factors

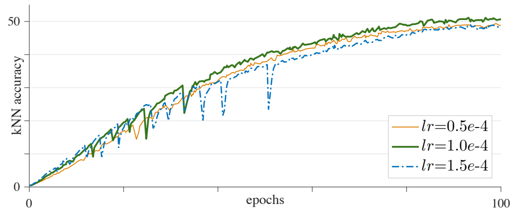
Batch size. ViT models in [16] are by design computationally heavy (see Table 2 and 3), and large-batch training [17, 47, 48] is a desirable solution to big ViT models. A large batch is also beneficial for accuracy in recent self-supervised learning methods [10, 18, 7]. Fig. 1 presents the training curves with different batch sizes.

A batch of 1k and 2k produces reasonably smooth curves, with 71.5% and 72.6% linear probing accuracy. In this regime, the larger batch improves accuracy thanks to more negative samples [20, 10]. The curve of a 4k batch becomes noticeably unstable: see the "dips" in Fig. 1. It has 72.2% linear probing accuracy. Although this seems to be a marginal decrease *vs*. the 2k batch, its accuracy is harmed by the instability, as we will show in the next subsection.

The curve of a 6k batch has worse failure patterns (big dips in Fig. 1). We hypothesize that the training is partially *restarted* and jumps out of the current local optimum, then seeks a new trajectory. As a consequence, the training does not diverge, but the accuracy depends on how good the local restart is. When this partial failure happens, it still provides an apparently decent result (69.7%). This behavior is harmful to explorative research: unlike catastrophic failure that is easily noticeable, the small degradation can be fully hidden.

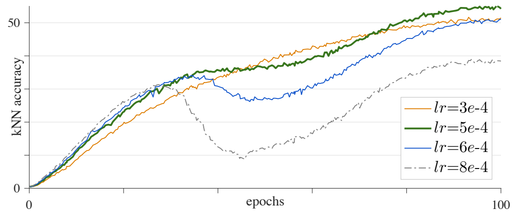
We also find that the mild instability does *not* result in a noticeably large variation. In many of our ablations, running the same configuration for a second trial often results in a small difference of 0.1~0.3%. This also makes it difficult to notice the potential degradation caused by instability.

Learning rate. In practice, the learning rate is often scaled when the batch size increases [27, 17]. In all experiments in this paper, we adopt the linear scaling rule [27, 17]: we set the learning rate as lr×BatchSize/256, where lr is a "base" learning rate. lr is the hyper-parameter being set [20, 10, 18]. In Fig. 2 we study the influence of lr.



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).

When lr is smaller, the training is more stable, but it is prone to under-fitting. In Fig. 2, lr=0.5e-4 has 1.8% worse accuracy than lr=1.0e-4 (70.4 *vs*. 72.2). In this regime, the accuracy is determined by fitting *vs*. under-fitting. Training with a larger lr becomes less stable. Fig. 2 shows that lr=1.5e-4 for this setting has more dips in the curve, and its accuracy is lower. In this regime, the accuracy is determined by stability.

Optimizer. By default, we use AdamW [31] as the optimizer, which is the common choice for training ViT models [16, 42, 36].[3] On the other hand, recent self-supervised methods [10, 18, 7] are based on the LARS optimizer [47] for large-batch training. In Fig. 3, we study the LAMB optimizer [48], which is an AdamW-counterpart of LARS.

Given an appropriate learning rate (lr=5e-4, Fig. 3), LAMB achieves slightly better accuracy (72.5%) than AdamW. But the accuracy drops rapidly when lr is larger than the optimal value. LAMB with lr=6e-4 and 8e-4 has 1.6% and 6.0% lower accuracy. Interestingly, the training curves are still smooth, but they degrade gradually in the middle. We hypothesize that although LAMB can avoid sudden change in the gradients, the negative impact of unreliable gradients is accumulated.

During our exploration, we find that LAMB can achieve comparable accuracy with AdamW, if lr is appropriately chosen. But the sensitivity to lr makes it difficult to ablate different architecture designs without extra lr search. As a result, we opt to use AdamW in other parts of this paper.

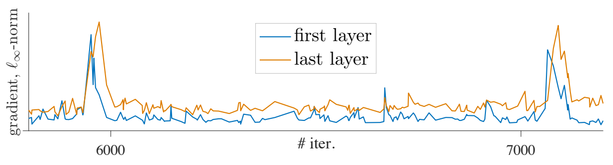
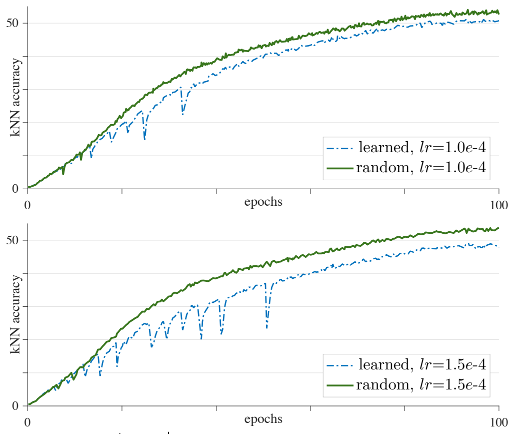


Figure 4. We monitor the gradient magnitude, shown as relative values for the layer. A "spike" in the gradient causes a "dip" in the training curve. We observe that a spike happens earlier in the first layer, and are delayed by tens of iterations in the last layers.



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**

Figure 5. **Random *vs*. learned patch projection** (MoCo v3, ViT-B/16, 100-epoch ImageNet, AdamW, batch 4096). **Top**: lr=1.0e-4. **Bottom**: lr=1.5e-4.

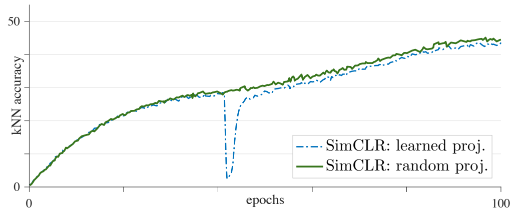
4.2. A Trick for Improving Stability

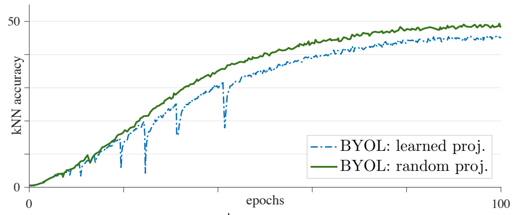
All these experiments suggest that instability is a major issue. Next we describe a simple trick that can improve stability in various cases in our experiments.

During training, we notice that a sudden change of gradients (a "spike" in Fig. 4) causes a "dip" in the training curve, which is as expected. By comparing all layers' gradients, we observe that the gradient spikes happen earlier in the first layer (patch projection), and are delayed by couples of iterations in the last layers (see Fig. 4). Based on this observation, we hypothesize that the instability happens earlier in the shallower layers. Motivated by this, we explore freezing the patch projection layer during training. In other words, we use a fixed ***random patch projection*** layer to embed the patches, which is not learned. This can be easily done by applying a stop-gradient operation right after this layer.

Comparisons. In Fig. 5 we show the MoCo v3 results with learnable *vs*. random patch projection. Random patch projection stabilizes training, with smoother and better training

[3]In original ViT [16] in JAX, the weight decay is "AdamW style": `https://github.com/google/flax/blob/master/flax/optim/adam.py`





	SimCLR	BYOL
learned patch proj.	69.3	69.7
random patch proj.	**70.1**	**71.0**

Figure 6. **Random *vs*. learned patch projection** (ViT-B/16, 100-epoch ImageNet, AdamW, batch 4096). **Top**: SimCLR: lr=2e-4, wd=0.1. **Bottom**: BYOL: lr=1e-4, wd=0.03.

curves. This stability benefits the final accuracy, boosting the accuracy by **1.7%** to 73.4% at lr=1.5e-4. The improvement is bigger for a larger lr (0.4%, 0.6%, 1.7%). This comparison confirms that the training instability is a main issue that impacts accuracy.

Besides MoCo, we find that other related methods [10, 18, 7] can also be unstable. Fig. 6 presents the training curves of ViT in SimCLR [10] and BYOL [18]. Random patch projection improves stability in both SimCLR and BYOL, and increases the accuracy by 0.8% and 1.3%. We also observe the instability issue for SwAV [7], in which, however, the loss diverges (NaN) when it is unstable. Random patch projection helps SwAV by enabling a relatively larger lr without diverging, and improves its accuracy from 65.8% to 66.4% when using the largest stable lr. In sum, this trick is effective in all these self-supervised frameworks.

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.

Discussions. It is an interesting observation that it is not necessary to train the patch projection layer. For the standard ViT patch size, the patch projection matrix is complete (768-d output for a 3-channel 16×16 patch) or over-complete. In this case, random projection should be sufficient to preserve the information of the original patches.

We note that freezing the first layer does not change the architecture, and it actually narrows down the solution space. This indicates that the underlying problem is on optimization. The trick alleviates the issue, but does not solve it. The model can still be unstable if lr is too big. The first layer is unlikely the essential reason for the instability; instead, the issue concerns all layers. The first layer is merely easier to be handled separately, *e.g.*, it is the only non-Transformer layer in the backbone. We hope to see a more fundamental solution in future work.

5. Implementation Details

This section describes the details of ViT+MoCo v3. More subtleties are described in the appendix.

Optimizer. By default we use AdamW [31] and a batch size of 4096 [10, 18, 7]. We search for lr and weight decay wd based on 100-epoch results, and then apply it for longer training. We adopt learning rate warmup [17] for 40 epochs (as per "*warmup of 10k steps*", Table 4 in [16]). This long warmup helps alleviate instability, though all unstable results are already with this warmup. After warmup, lr follows a cosine decay schedule [30].

MLP heads. The projection head [10] is a 3-layer MLP, following [11]. The prediction head [18] is a 2-layer MLP. The hidden layers of both MLPs are 4096-d and are with ReLU [32]; the output layers of both MLPs are 256-d, without ReLU. In MoCo v3, all layers in both MLPs have BN [23], following SimCLR [10]. The MLP heads of BYOL/SwAV have different BN designs (see appendix).

Loss. We scale the contrastive loss in (1) by a constant 2τ (see Alg. 1), following [18]'s appendix. This scale is redundant because it can be absorbed by adjusting lr and wd. But this scale makes it less sensitive to the τ value when lr and wd are fixed. We set τ=0.2 [12] as the default.

ViT architecture. We closely follow the designs in [16]. The input patch size is 16×16 or 14×14 ('/16' or '/14'), and after projection it results in a sequence of length 196 or 256 for a 224×224 input. Position embeddings are added to the sequence, and we use the sine-cosine variant [43] in 2-D. This sequence is concatenated with a learnable class token. The sequence is then encoded by a stack of Transformer blocks [43] following the design in [16]. The class token after the last block (and after the final LayerNorm [1]) is treated as the output of the backbone, and is the input to the MLP heads.

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.

model	blocks	dim	heads	params
ViT-Small	12	384	12	22 M
ViT-Base [16]	12	768	12	86 M
ViT-Large [16]	24	1024	16	304 M
ViT-Huge [16]	32	1280	16	632 M

Table 2. **Configurations of ViT models** in our experiments. Here "blocks" is the number of Transformer blocks, "dim" is the input/output channel dimension of all blocks, and "heads" is the number of heads in multi-head attention. The MLP hidden dimension is 4×dim.

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).

6. Experimental Results

In this section we benchmark and ablate self-supervised ViT. We perform self-supervised training on the 1.28M ImageNet training set [14], and evaluate by linear probing.

Table 2 summarizes the ViT configurations we study. ViT-B/L/H follow [16], and ViT-S is similar to that in [42]. We use ViT-B by default in our ablations.

Training time. We train our models in TPUs (v3) that are publicly available in Google Cloud Platform (GCP). Table 3 summarizes the training time (per 100 epochs). It takes 2.1 hours training ViT-B for 100 epochs, and our ablations typically take 6.3 hours each (300 epochs). This is a competitive performance, as it enables us to ablate many design decisions. The TPU implemenataion also makes it possible to explore the ViT-H model, which takes 9.8 hours per 100 epochs using 512 TPUs. This is a gigantic scale of training: for the 300-epoch ViT-H, this amounts to ∼625 TPU·days, or ∼1.7 TPU·years of training.

We also verify our models in GPUs using PyTorch. It takes 24 hours for ViT-B in 128 GPUs (*vs.* 2.1 hours in 256 TPUs). With an increasing number of devices, we observe that TPUs scale up more favorably than GPUs. While further engineering optimization could speed up our GPU system, we opt to use the TPU system for the ease of research.

6.1. Self-supervised Learning Frameworks

We benchmark self-supervised ViT in four frameworks: MoCo v3, SimCLR [10], BYOL [18], and SwAV [7].We use the same random projection trick in all cases. We sweep lr and wd for each individual framework for fair comparisons.

Table 4 reports the results of ViT-S/16 and ViT-B/16. MoCo v3 has better accuracy on ViT than other frameworks. The *relative* accuracy among these methods is different between ViT-B and R50: see Fig 7. MoCo v3 and SimCLR are more favorable for ViT-B than R50 (above the diagonal line).

model	MoCo v3	SimCLR	BYOL	SwAV
R-50, 800-ep	73.8	70.4	**74.3**	71.8
ViT-S, 300-ep	**72.5**	69.0	71.0	67.1
ViT-B, 300-ep	**76.5**	73.9	73.9	71.6

Table 4. **ViT-S/16 and ViT-B/16 in different self-supervised learning frameworks** (ImageNet, linear probing). R-50 results of other frameworks are from the improved implementation in [13]. For fair comparisons, all are pre-trained with two 224×224 crops for each image (multi-crop training [7] could improve results, which is beyond the focus of this work).

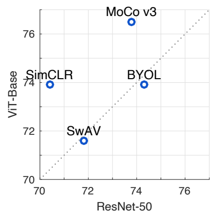


Figure 7. Different self-supervised learning frameworks perform differently between R-50 [21] (x-axis) and ViT-B [16] (y-axis). The numbers are ImageNet linear probing accuracy from Table 4.

6.2. Ablations of ViT + MoCo v3

Next we ablate the designs of the ViT + MoCo v3 system. We use random patch projection in all ablations.

Position embedding. The following table compares the choice of position embedding (our default is sin-cos):

ViT-B, 300-ep	sin-cos	learned	none
linear acc.	76.5	76.1	74.9

The learned version works well, but not better than sin-cos. Surprisingly, the model works decently even with *no* position embedding (74.9%). The capability to encode positions contributes only 1.6%. We believe this data point reveals both strengths and limitations of the current model. On the positive side, it suggests that the model can learn strong representations just by a *set* of patches, which are fully *permutation-invariant*. This is analogous to bag-of-words models [41]. This model has *no* positional inductive bias. On the negative side, it also suggests that the model has not made good use of positions, and the gesture of the object contributes relatively little to the representation. We hope this data point will draw attention to future study.

Class token. The following table ablates the role of the class token [CLS] in ViT:

ViT-B, 300-ep	w/ [CLS]	w/o [CLS]; LN+pool	w/o [CLS]; pool
linear acc.	76.5	69.7	76.3

Global average pooling is used right after the final block if [CLS] is not used. ViT has an extra LayerNorm (LN) after the final block [16], and if we keep this LN and remove [CLS], the result is much worse (69.7%). But if we remove this LN and [CLS], the result is nearly unchanged (76.3%).

This comparison indicates that the class token is not essential for the system to work. It also suggests that the choice of normalization layers can make a difference.

BatchNorm in MLP heads. Unlike the standard ResNets [21], ViT models by default have no BN, and thus all BN layers are in the MLP heads. The following table compares with *vs*. without BN in the heads:

ViT-B, 300-ep	heads w/ BN	heads w/o BN
linear acc.	76.5	74.4

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.

Prediction head. MoCo v3 uses a prediction MLP head as per [18]. The next table ablates this design:

ViT-B, 300-ep	w/ pred. MLP	w/o pred. MLP
linear acc.	76.5	75.5

Removing the prediction MLP head has a decent result of 75.5%. While this extra head boosts accuracy, MoCo as a contrastive method does not need the predictor MLP to work, unlike the negative-free methods in [18, 13].

Momentum encoder. The following table compares the momentum coefficient (m) in the momentum encoder:

ViT-B, 300-ep	m=0	m=0.9	m=0.99	m=0.999
linear acc.	74.3	75.6	76.5	75.0

The optimal value is m=0.99 (our default). The case of m=0 is analogous to SimCLR (plus the prediction head and stop-gradient on the keys), and its accuracy of 74.3% is similar to SimCLR's (73.9%, Table 4). The usage of the momentum encoder leads to 2.2% increase.

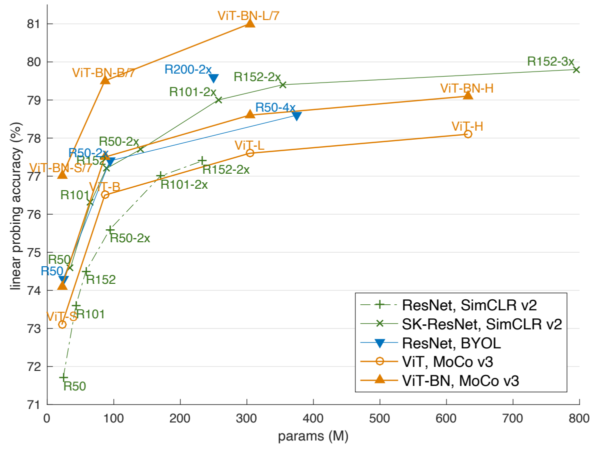
Training length. In the following table we report ViT-S/B + MoCo v3 *vs*. training length:

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

The smaller ViT-S enjoys the benefit of training longer, and improves by 0.9% when extending to 600 epochs. This is similar to the behavior of R50, which was typically trained for 800 epochs [10]. But the gain of training longer is diminishing on ViT-B. Based on this ablation, we train the bigger ViT-L/H for 300 epochs presented next (Table 1).

6.3. Comparisons with Prior Art

Self-supervised Transformers. Table 1 in Sec. 1 presents MoCo v3 results with different ViT models, compared with state-of-the-art *self-supervised Transformers*. Both iGPT [9] and the masked patch prediction in [16] can be categorized as the *masked auto-encoding* paradigm (*e.g.*, GPT [37] and BERT [15]). Our MoCo-based ViT has higher accuracy and smaller models than iGPT, under the same linear probing protocol and training data. The mask patch prediction in [16] is pre-trained on JFT-300M and end-to-end fine-tuned in ImageNet, which we append as a reference.



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**	-

Figure 8. **Comparisons with state-of-the-art big ResNets**, presented as parameters-*vs*.-accuracy trade-off. All entries are pretrained 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].

Our self-supervised ViT models have *higher* accuracy when the models are *bigger*. This is in contrast to the *supervised* results in [16], where ViT-L has lower accuracy than ViT-B when pre-trained in ImageNet-1k/21k. Actually, for ViT-L, our self-supervised pre-training with linear probing (77.6%) is *better* than the supervised counterpart in [16] (76.53%) when trained in ImageNet-1k.[4] These comparisons suggest that self-supervised learning as a tool for generic representation learning is less prone to over-fitting.

Comparisons with big ResNets.[5] In Fig. 8 we compare with the state-of-the-art *big* ResNets reported by SimCLR v2 [11] and BYOL [18]. We note that both SimCLR v2 and BYOL use a momentum encoder. Our baseline ViT MoCo (the curve of "ViT, MoCo v3") is slightly better than ResNet SimCLR v2 in the small-model regime, but the envelopes become just comparable for larger models. SimCLR v2 with SK-ResNet (Selective Kernel [29], a form of attention) has a higher envelope. BYOL also has a higher envelope with wider ResNets (1-4×), and has an outstanding point with a deeper ResNet (R200-2×).

[4]Stronger regularization could reduce over-fitting for supervised ViT [42], though regularizing the very big ViT-L/H is yet to be explored.

[5]Transformers [43] by design consist of residual blocks [21], and thus are a form of residual networks. In the literature on "Transformer *vs*. ResNet", precisely speaking, the term of "ResNet" refers to the specific design that has non-degenerated (*e.g.*, 3×3) convolutions.

pre-train	CIFAR-10 [26]			CIFAR-100 [26]			Oxford Flowers-102 [33]			Oxford-IIIT-Pets [35]		
	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.

case	pre-train	ViT-S	ViT-B	ViT-L
masked patch pred. [16]	JFT-300M	-	79.9	-
DeiT [42]	-	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.

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).

In iGPT [9], accuracy can be improved by using *longer* sequences in the pixel domain. Here we explore longer sequences by reducing the patch size to 7×7 ("/7" in Fig. 8). This keeps the model size unchanged, but increases FLOPs to ∼6×. It can improve the accuracy by ∼2-3%. The gain of using small patches is also observed in [8]. MoCo v3 achieves **81.0%** with ViT-BN-L/7.[7] As a comparison, the previous best results under the linear probing protocol are 79.8% with SimCLR v2 (SK-ResNet152-3×), and 79.6% with BYOL (ResNet200-2×).

Discussion. While the bigger self-supervised ViT can achieve better accuracy, the results are saturated. This is unlike the trend in NLP, where bigger Transformers learn better representations (*e.g*., [4]). A potential solution is to use more data. The saturation can also be caused by the limited power of the existing *instance-based* pretext task [45]. It may be desired to design more difficult pretext tasks.

Our self-supervised ViT models are competitive with the big convolutional ResNets. It suggests that ViT can learn strong representations with "*fewer inductive biases*" [16]. However, we also find that the accuracy only decreases by a bit even if removing the only positional inductive bias (position embedding), suggesting that in our method ViT relies less on positional information than convolutional networks.

End-to-end fine-tuning. Table 5 reports end-to-end fine-tuning results. We use the DeiT codebase [42] and all its default settings unless specified. MoCo v3 achieves **83.2%** with ViT-B under 150-epoch fine-tuning, substantially better than DeiT's 81.8% at 300 epochs.

In addition, MoCo v3 has **84.1%** with ViT-L when fine-tuned for only 100 epochs with a drop path rate of 0.5. This short schedule demonstrates the effectiveness of MoCo pre-training. We have also found DeiT-L diverges under its default settings, and a different solution may be needed.

6.4. Transfer Learning

In Table 6 we evaluate transfer learning. We study the four downstream datasets as in [16]. We fine-tune the models end-to-end, also following [16].

Our self-supervised ViT has better transfer learning accuracy when the model size increases from ViT-B to ViT-L, yet it gets saturated when increased to ViT-H. As a comparison, the ImageNet-supervised ViT in [16] becomes saturated or overfitted starting at ViT-L. Our self-supervised ViT achieves *better* results than its ImageNet-supervised counterparts in three of these four datasets.

The overfitting is more prominent when training the big ViT models from scratch in these small datasets: the accuracy in general decreases with bigger ViT. We also find that the from-scratch ViT results are much worse than their ResNet-counterparts (*c.f*., Table 8 in [10]) in these small datasets. This suggests that if data are not enough, it is *difficult* for ViT to learn representations in the lack of inductive biases. Self-supervised pre-training can close this gap and largely reduce overfitting in small datasets.

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 self-supervised pre-training for big ViT models in bigger data. This is analogous to the trajectory of unsupervised pre-training in NLP in the past years [37, 15, 38, 4], *i.e*., both models and datasets are scaled up.

7. Conclusion

We have explored training ViT in the recently popular self-supervised frameworks. Our comparisons concern several aspects, including ViT *vs*. convolutional networks, supervised *vs*. self-supervised, and contrastive learning *vs*. masked auto-encoding. We report positive evidence as well as challenges, open questions, and opportunities. We hope our empirical study will be useful for the community to close the gap of pre-training between vision and language.

[6]We are trying to replace every LN with BN in the ViT backbone. In preliminary experiments, doing so leads to convergence problems.

[7]ViT-BN-H/7 is out of memory in our unoptimized implementation.

Postscript. After the first version of this manuscript, an author of BiT [25] and ViT [16], Lucas Beyer, echoed that "*the exact same behaviour*" was observed for supervised BiT-ResNet in ImageNet-21k. The instability problem can be more general than the scope of this paper.

A. Additional Implementation Details

Data augmentation. We follow the good practice in existing works [45, 20, 10, 18]. Our augmentation policy includes random resized cropping, horizontal flipping, color jittering [45], grayscale conversion [45], blurring [10], and solarization [18]. We take two 224×224 crops for each image in each iteration.

BatchNorm. We use SyncBN as our default BatchNorm implementation, following [10]. When BN is used, there are two options on batching: **(i)** all samples *and* crops are in the same batch, *i.e.*, BN is over 4096×2 crops for 4096 images; and **(ii)** only different images are in the same batch, *i.e.*, the two crops of the same image are separately forwarded in two 4096 batches. We notice that the code of SimCLR [10] adopts the former option, while the code in BYOL [18] adopts the latter. The pseudo-code in our Alg. 1 implies that we adopt the latter. The BN batching size influences the gradient variance, and the two implementations should lead to different results.

AdamW implementation. We notice that in PyTorch and JAX, the weight decay in AdamW is implemented as "-$lr * wd *$ weight" (consistent with [31]), but in TensorFlow it is implemented as "-$wd *$ weight", and wd needs to be scaled beforehand.[8] In our TPU/TensorFlow code, we follow the version consistent with [31].

MLP heads in BYOL and SwAV. In our BYOL+ViT implementation, the projection/prediction MLP heads have BN in their hidden layers, but not in their output layers, which faithfully follow [18]. In our SwAV+ViT implementation, we use no BN in the MLP heads, which is a configuration that performs the best in our SwAV experiments.

kNN monitor. The kNN monitor [45] is a widely used tool in self-supervised learning research. The kNN evaluation was often performed sparsely, *e.g.*, once per epoch. We notice that this may hide the sudden "dips" in the curves.

To better reveal the sudden changes, we monitor the kNN performance more densely, *e.g.*, every tens of iterations. This is prohibitive even though the kNN classifier does not need training. We adopt a few approximations to make it feasible. We maintain a small memory bank [45] (whose length is 10% of ImageNet) for the purpose of kNN search. This memory bank is updated per iteration by the features from the training samples (which are augmented images). This memory bank is maintained as a queue similar to [20], so it requires no extra feature extraction. We use the features from the class token for kNN monitoring, so the monitor is independent of the choice of the head. Other details follow the kNN implementation in [45]. We find that this approximate kNN monitor is sufficient to reflect the stability of training.

[8] https://www.tensorflow.org/addons/api_docs/python/tfa/optimizers/AdamW

MoCo v3 for ResNet-50. The implementation follows the good practice in recent works [10, 18]. It uses the LARS optimizer [47] with a 4096 batch [10], lr=0.3, wd=1.5e-6. The temperature is τ=1.0. The encoder f_k's momentum coefficient is m=0.996 and increases to 1 with a cosine schedule [18]. The augmentation is the same as described above.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv:1607.06450*, 2016. 5

[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NeurIPS*, 2019. 2, 3

[3] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a "Siamese" time delay neural network. In *NeurIPS*, 1994. 2

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. 1, 2, 8

[5] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In *ICCV Workshops*, 2019. 2

[6] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with Transformers. In *ECCV*, 2020. 2

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 1, 2, 3, 4, 5, 6

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv:2104.14294*, 2021. 8

[9] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020. 1, 2, 7, 8

[10] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 2, 3, 4, 5, 6, 7, 8, 9

[11] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020. 2, 5, 7

[12] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020. 2, 3, 5

[13] Xinlei Chen and Kaiming He. Exploring simple Siamese representation learning. In *CVPR*, 2021. 2, 3, 6, 7

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 6

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 1, 2, 7, 8

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 9

[17] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv:1706.02677*, 2017. 3, 5

[18] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *NeurIPS*, 2020. 1, 2, 3, 4, 5, 6, 7, 9

[19] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 2

[20] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2, 3, 9

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 6, 7, 8

[22] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2019. 2, 3

[23] Sergey Ioffe. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In *NeurIPS*, 2017. 5

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 5

[25] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big Transfer (BiT): General visual representation learning. In *ECCV*, 2020. 9

[26] Alex Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009. 8

[27] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv:1404.5997*, 2014. 3

[28] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 1, 2

[29] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *CVPR*, 2019. 7

[30] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *ICLR*, 2017. 5

[31] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 4, 5, 9

[32] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 5

[33] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 8

[34] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018. 2

[35] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, 2012. 8

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. 2021. 4

[37] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. 1, 2, 7, 8

[38] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 1, 2, 8

[39] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019. 2

[40] Tim Salimans and Diederik P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPS*, 2016. 5

[41] Josef Sivic and Andrew Zisserman. Video Google: a text retrieval approach to object matching in videos. In *ICCV*, 2003. 6

[42] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv:2012.12877*, 2020. 4, 6, 7, 8

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1, 2, 5, 7

[44] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018. 2

[45] Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, 2018. 2, 3, 8, 9

[46] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *CVPR*, 2019. 3

[47] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv:1708.03888*, 2017. 3, 4, 9

[48] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *ICLR*, 2020. 3, 4

[49] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020. 2