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.

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.

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