A. Appendix

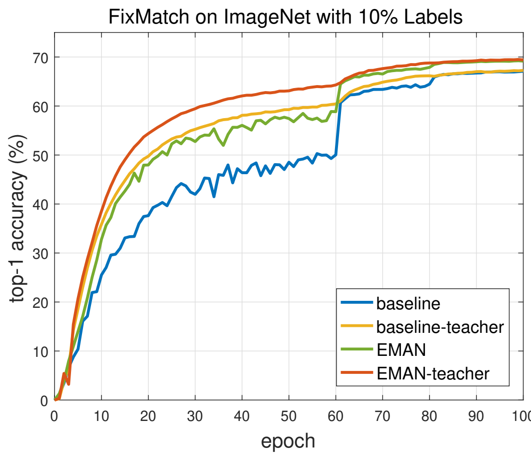


Figure 4. The FixMatch accuracy with 10% labels

A.1. FixMatch

We re-implemented FixMatch in PyTorch, and followed the exactly same hyperparameter settings as in the official FixMatch [39] [1]. The number of labeled (unlabeled) images in a batch is 64 (320). The loss weight for the supervised (unsupervised) loss is 1.0 (10.0). The EMA momentum $m = 0.999$. The number of epoch is counted on unlabeled images.

As discussed in [1, 24, 26], the EMA updated teacher model can present more reliable results. We show the accuracy curves of the teachers (with EMAN) for both baseline FixMatch and FixMatch-EMAN on ImageNet validation in Figure 4. Although the baseline FixMatch did not use the EMA-teacher framework, we can collect its EMA updated model (both model parameters and BN statistics) during training for the purpose of inference only. It can be found that the EMA teacher indeed has higher and more stable accuracy, especially during the former epochs. This is the reason why we reformulate the baseline FixMatch to EMA-teacher framework.

A.2. MoCo

We used the code and followed the exactly same settings as in the official MoCo-v2 [10][2]. When using EMAN, the training will become less stable at the beginning, because the whole network, including the normalization statistics, is slowly updated, with momentum $m = 0.999$. In this case, the warn-up learning rate schedule is more important. Without it, MoCo-EMAN will have worse performance in our experiments. In addition, as seen in Figure 3 (b), the kNN accuracy of MoCo-EMAN will have a slight decrease in the last training epochs, because the learning rate is too small. This behavior is quite consistent in our MoCo-EMAN experiments, and the best model is usually at around 90% training epochs. To avoid the decrease, we simply used the 90%-th epoch checkpoint for the evaluation of other downstream tasks. Another solution is to set the minimum learning rate to a not-too-small value, e.g. 0.001, in the cosine learning rate schedule. These two strategies achieved very close performances in our experiments.

In Figure 5 (a) and (b), we show the curves of loss and instance discrimination accuracy of MoCo during training. Although they do not directly relate to the actual representations power, they can help to understand what is happening during training. It can be found that the training behaviors are quite different between MoCo-ShuffleBN and MoCo-EMAN. EMAN will make the self-supervised learning task of MoCo more difficult, and lead to higher training loss and lower instance discrimination accuracy. It suggests that EMAN can better prevent the MoCo model from cheating, and could potentially improve the representation power.

[1]https://github.com/google-research/fixmatch
[2]https://github.com/facebookresearch/moco

A.3. BYOL

We re-implemented BYOL in PyTorch, and followed some hyperparameter settings as in the official BYOL [17][3]. The official implementation chooses different hyperparameters for experiments of different numbers of epochs. To be consistent, we set weight decay as 0.000001, and initial EMA momentum as 0.98, for experiments of both 50 and 200 epochs. The initial learning rate was set as 0.9 (1.8) for 50 (200) epochs. In all BYOL experiments, the batch size was 512 on a machine with 8 GPUs.

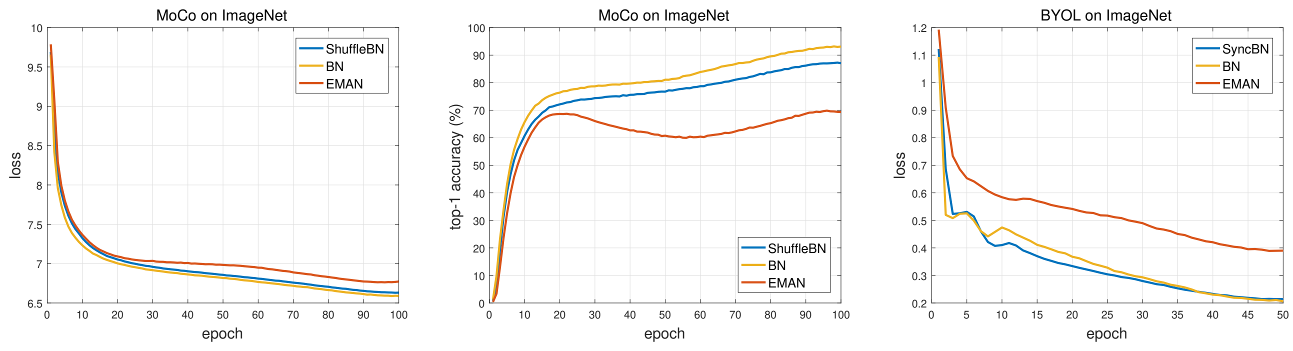
The training loss curves of BYOL are also shown in Figure 5 (c). Similar to the observations from MoCo curves, EMAN will make the self-supervised learning task of BYOL more difficult, and result in higher training loss. It suggests that EMAN can better prevent the BYOL model from cheating, and could potentially improve the representation power.

A.4. Other Settings

ImageNet Sampling We sample 1% (10%) images per class for the semi-supervised learning experiments of 1% (10%) labels, which are 12,820 (128,118) images in total.

Linear and Finetuning Evaluation In addition to the details in Section 5.2, weight decay = 0 and momentum = 0.9 for linear evaluation, and weight decay = 0.0001, and momentum = 0.9 for finetuning. The standard data augmentation (`RandomResizedCrop` and `RandomHorizontalFlip`) was used during training. At inference, the center 224×224 crop was used.

[3]https://github.com/deepmind/deepmind-research/tree/master/byol



(a) (b) (c)

Figure 5. (a) and (b) are the training loss and instance discrimination top-1 accuracy of MoCo, and (c) the training loss of BYOL.

kNN Evaluation Different from [45, 52], we did not use the weighting mechanism. Instead, it was just a standard kNN classifier with top $k = 20$ neighbors, where a query will be classified to the majority class of neighbor samples. The center 224×224 crop was used, also in the experiments of Image Retrieval and Low-shot Classification.