Fast-MoCo: Boost Momentum-based Contrastive Learning with Combinatorial Patches **Supplementary Material**

A Algorithm

Algorithm 1 Pytorch-style Pseudocode for Fast-MoCo

```
# f_o: online branch networks [encoder, projector, predictor]
# f_t: target branch networks [encoder, projector]
# a: exponential moving average momentum \alpha, t: temperature \tau
# combine: generate all possible 2-combinations between patch embeddings

for x in loader: # load a minibatch
    x1, x2 = aug(x), aug(x) # augemtation, NxCxHxW

    x1_d, x2_d = divide(x1), divide(x2) # Divide step, 4NxCx(H/2)x(W/2)
    v1, v2 = f_o[0](x1_d), f_o[0](x2_d) # online branch encode
    c1, c2 = combine(v1), combine(v2) # Combine step

    z1_c, z2_c = f_o[1:](c1), f_o[1:](c2) # project & predict
    z1, z2 = f_t(x1), f_t(x2) # target branch encode & project

    loss = ctr(z1_c, z2) + ctr(z2_c, z1)
    loss.backward()

    # weight update
    update(f_o.params)
    f_t.params = a * f_t.params + (1-a) * f_o[:2].params

def ctr(z_c, z)
    z_c, z = normalize(z_c, dim=1), normalize(z, dim=1) # l2-normalize
    z_c = z_c.split(z.size(0))

    # calculate loss for each of the 6 combined samples
    loss = 0
    for _z in z_c:
        logits = mm(_z, z.t())
        loss += CorssEntropyLoss(logits/t, labels)
        # positive pairs are sourced from the same instance
    return loss /=  len(z_c)
```

B Additional Implementation Details

B.1 Self-Supervised Pretraining

Here in Table 1, we show the detailed configurations for self-supervised pretraining. For ablation study experiments that are only trained for 20 epochs, we multiply the learning rate by 1.5 compared to default setting.

Config	**Values**	
Epochs	$\{100, 200, 400\}$	20
Optimizer	SGD	
Optimizer momentum	0.9	
Weight decay	1e-4	
Gradient clipping	1.0	
Learning rate schedule	Cosine	
Initial Learning rate	0.1	0.15
Final Learning rate	0.0	
Warmup epochs	1	
Warmup initial learning rate	0.025	0.0375
Batch size	512	
Temperature τ	1.0	
Exponential moving average momentum α	0.99	
Augmentation	As in [4]	

Table 1: **Self-supervised pretraining setup.**

B.2 Linear Evaluation

Here in Table 2, we show the detailed configurations for linear evaluation. For ablation study experiments that only trained for 20 epochs, we have a shorter training schedule with doubled learning rate.

Config	**Values**	
Pre-trainig epochs	$\{100, 200, 400\}$	20
Fine-tuning epochs	90	10
Optimizer	LARS	
Optimizer momentum	0.9	
Learning rate schedule	Cosine	
Initial Learning rate	0.8	1.6
Final Learning rate	0.0	
Batch size	4096	
Augmentation	As in [5]	

Table 2: **Linear evaluation setup.**

B.3 Semi-Supervised Training

We follow the evaluation protocol as in [2,9,15] and apply the same augmentations as used in the linear evaluation. For both 1% and 10% settings, we adopt the same dataset split as in [2]. We use a SGD optimizer with Nesterov momentum of 0.9 and a batch size of 256. We do not apply any regularization such as weight decay and gradient clipping. The learning rate is scaled by a factor of 0.2 at 60% and 80% of the training schedule. For the 1% setting, we fine-tune for 60 epochs with a learning rate of 5e2 and 0 for the linear layer and backbone, respectively. For the 10% setting, we fine-tune for 20 epochs with a learning rate of 5e2 and 1e-6 for the linear layer and backbone, respectively.

B.4 Transfer Learning

For object detection and segmentation, we follow the evaluation protocol as in [10,3,4,15] and conduct experiments on detectron2 [14] codebase with the R50-C4 backbone variant [8]. The detailed configurations are as follows:

PASCAL-VOC For object detection on PASCAL-VOC [7] with Faster R-CNN [13], we have all weights finetuned on the `trainval07+12` dataset and evaluated on the `test07` dataset. We fine-tune for $24K$ iterations with batchsize 16. The learning rate is 1e-2 and 1e-3 for the heads and backbone, respectively, and is scaled by 0.1 at $18K$ and $22K$ iterations.

COCO For detection and instance segmentation on COCO [12], we finetune our weights with Mask R-CNN [11] on the `train` set and report results on the `val` split. We use the $1\times$ schedule in detectron2 [14]. The learning rate is 1e-2 and 1e-3 for the heads and backbone, respectively.

B.5 Fast-MoCo w/ `mulit-crop`

`Multi-crop` is a technique proposed in SwAV [1]. In addition to two 224×224 crops, `multi-crop` additionally adds six 96×96 patches as samples so that the encoder is trained with samples that have multiple resolutions and hard samples. For fair comparison, we add an additional 224×224 crop as sample (which adds approximately the same computational cost as six 96×96 patch samples) for Fast-MoCo w/ `mulit-crop`.

C Additional Results

C.1 Combinatorial Patches on Other SSL Frameworks.

We further evaluate our method by directly applying it to NNCLR (neighborhood-replaced target embedding), BYOL (non-contrastive), and SimSiam (momentum-free); results in Table 3 show that the proposed combinatorial patches can generalize well to different SSL frameworks. Please note, while the proposed combinatorial patches can be applied to MoCoV3, NNCLR, and BYOL seamlessly,

integrating it to SimSiam will increase the computation cost since we need to forward the same view twice to get the symmetrized loss while SimSiam only needs one forward pass. Nevertheless, combinatorial patches can still boost SimSiam significantly, e.g., achieving 71.7% top-1 linear evaluation accuracy with only 100 epochs, which is higher than the original SimSiam running 800 epochs (71.3%).

Method	Top-1 (e100)
NNCLR [6]	69.4
NNCLR + combinatorial patches	**72.5** (+3.1)
BYOL [9]	66.5
BYOL + combinatorial patches	**73.6** (+7.1)
SimSiam [4]	68.1
SimSiam + combinatorial patches	**71.7** (+3.6)

Table 3: ImageNet linear evaluation performance of our method on other frameworks with ResNet-50. Results of compared methods w/o combinatorial patches are from their original paper.

C.2 Downstream Results with Different Pretrain Epochs

The faster convergence of our method also stand for the low-data regime and transfer learning as shown in Figure 1. Fast-MoCo achieves better or similar performance with only 100 or 200 epochs than other methods trained with 800 or 1000 epochs on semi-supervised classification with 1% labeled data (Figure 1(a)) and 10% labeled data (Figure 1(b)), COCO detection (Figure 1(c)), and COCO segmentation (Figure 1(d)).

C.3 Robustness to the Selection of Batch Size

In Table 4, we show the comparison of robustness to batch size varying among $\{1024, 512, 256\}$ with the reported values in SimSiam, BYOL and Barlow Twins. As observed, Fast-MoCo has similar robustness as BYOL and Barlow Twins.

C.4 AdamW Optimizer

When using AdamW as optimizer and pretrain for 100 epochs based on MoCo v3 (69.3% w/ AdamW, result run by us) under the same setting, our Fast-MoCo can achieve 71.7% (+2.4%) Top-1 accuracy for ImageNet linear evaluation, which demonstrates that our approach also works well with AdamW.

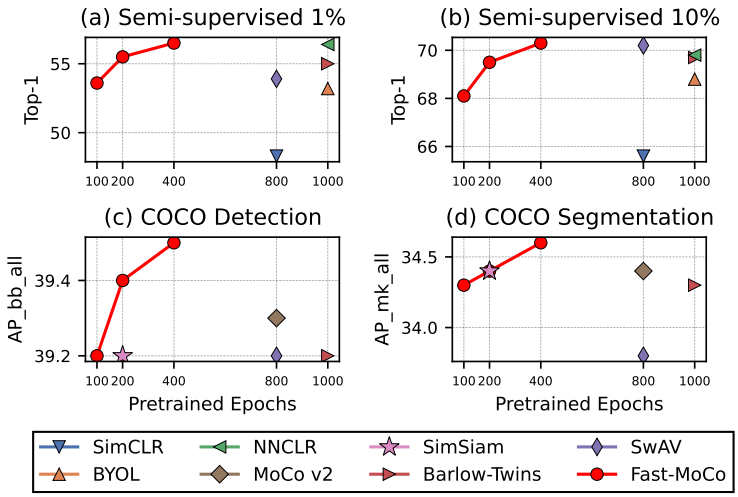

Fig. 1: Downstream task results in different epochs.

Method	Top-1		
	b1024	b512	b256
SimSiam [4] (100 ep.)	68.0	68.1	68.1
BYOL [9] (300 ep.)	72.3	72.2	71.9
Barlow Twins [15] (300 ep.)	71.7	71.4	70.7
Fast-MoCo (100 ep.)	73.6	73.5	72.5

Table 4: Comparison of robustness to batch size from 1024 to 256. Result of the compared methods are from their original paper.

C.5 Combine Stage Selection

In Figure 2 of this supplementary, we provide additional results on different choices of Combine stage. We have two target-sample pairs per image when number of patch combined $n = 2$. When the combine stage is before "final", we stitch the feature map (for stage $1 \sim 3$) or patch image (for stage "input") vertically and horizontally with respect to their original position. Here we can see the performance drops significantly when the Combine step is conducted after the projection layer or prediction layer.

D Patch Encode Approaches

In Figure 3, we illustrate every patch encoding approach compared in Section 5.2.

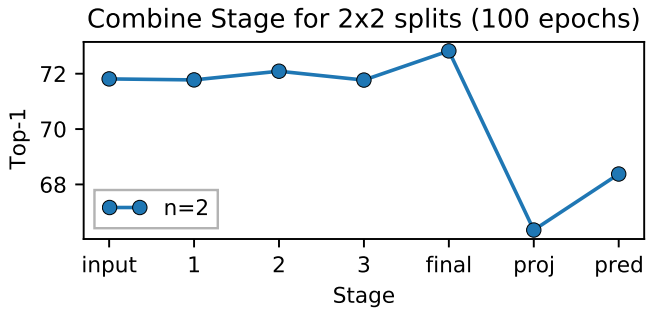

Fig. 2: ImageNet linear evaluation accuracy (Y-axis) when Combine step is conducted at different stages (X-axis) for combining $n = 2$ divided patches. Here "input" denotes the patches are combined at image level, $1 \sim 3$ denotes patches are combined at feature map level after ResNet stage $1 \sim 3$. "final", "proj" and "pred" denotes patches are combined at embedding level after encoder, projector, and predictor, respectively.

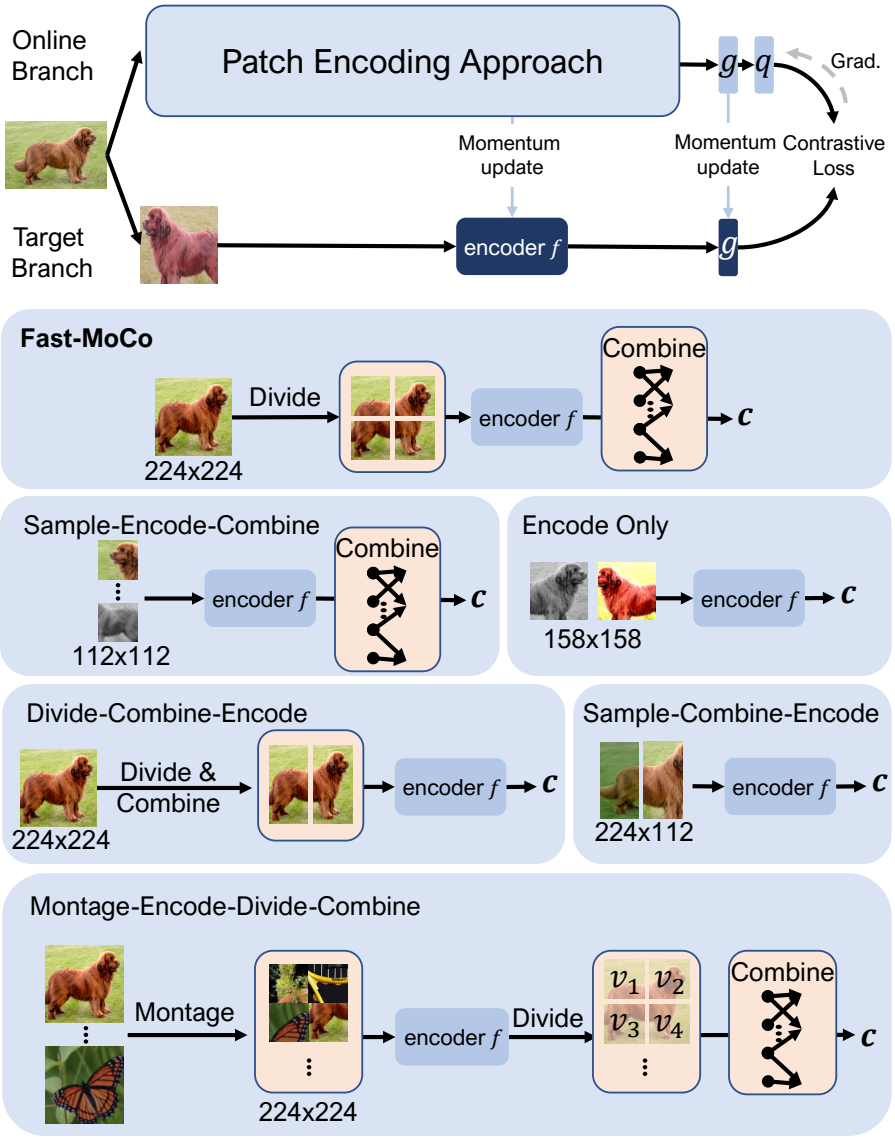

Fig. 3: Illustration of patch encoding approaches compared in Section 5.2.

References

1. Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., Joulin, A.: Unsupervised learning of visual features by contrasting cluster assignments. Advances in Neural Information Processing Systems **33**, 9912–9924 (2020)
2. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
3. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020)
4. Chen, X., He, K.: Exploring simple siamese representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15750–15758 (2021)
5. Chen, X., Xie, S., He, K.: An empirical study of training self-supervised vision transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9640–9649 (2021)
6. Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., Zisserman, A.: With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9588–9597 (2021)
7. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010)
8. Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K.: Detectron (2018)
9. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al.: Bootstrap your own latent-a new approach to self-supervised learning. Advances in Neural Information Processing Systems **33**, 21271–21284 (2020)
10. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020)
11. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017)
12. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)
13. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
14. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. `https://github.com/facebookresearch/detectron2` (2019)
15. Zbontar, J., Jing, L., Misra, I., LeCun, Y., Deny, S.: Barlow twins: Self-supervised learning via redundancy reduction. In: International Conference on Machine Learning. pp. 12310–12320. PMLR (2021)