Appendix: Masked Self-Supervised Transformer for Visual Representation

Anonymous Author(s)
Affiliation
Address
email

1 The setting of computation resources

In ablation studies, the MST with 1024 images is trained in 128 AMD DCUs that are publicly available in Sugon Cloud. For verifying the generality of the results, the pre-trained model is used to validate downstream experiments for 32 Nvidia Tesla V100 GPUs. Meanwhile, the same random seed is set for fair comparison. Also, we report the average result after running multiple experiments. For 100 epochs, the standard error of linear probing is 0.2236%. For 300 epochs, the standard error of linear probing is 0.1581%.

2 Data augmentation

The image augmentation pipeline consists of the following transformations: random resized cropping, horizontal flipping, color jittering, grayscale conversion, Gaussian blurring, solarization, and multi-crop. The random resized cropping and multi-crop transformations are always applied, while the rest of transformations are applied randomly, with some probability. This probability is different for the two distorted views in the blurring and solarization transformations. We use the same augmentation parameters as BYOL besides multi-crop. The multi-crop follows SwAV [1] and DINO [2]. Each input image with 224×224 is transformed twice to produce the two distorted views.

3 BatchNorm

Following [3, 4, 5], we adopt SyncBN as our default BatchNorm. The running mean and running variance of BN of MST only are updated from different images in the same batch while SimCLR [6] is updated from total images in teacher and student batches. The two kinds of BN influence the gradient variance. Hence, the two implementations should lead to different results. Meanwhile, the running mean and running variance of BN are only updated from the global views when our method adopts masked self-supervised Transformer.

4 k-NN classification

According to Wu *et al*. [7], we evaluate the quality of features with a simple weighted k Nearest Neighbor classifier. We freeze the parameters of pre-trained model and extract the features of class embedding for the train and validation dataset. As shown in Table 1, we evaluate different values for k and find that the setting of 10 is consistently leading to the best accuracy across our runs. More importantly, we evaluate Top-1 accuracy in the validation dataset.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Table 1: **The setting of k.** We report Top-1 accuracy on ImageNet validation dataset by using 300-epoch pre-trained DeiT-S model.

Method	Architecture	epoch	k	k-NN Top-1 (%)
Ours	DeiT-S	300	10	**75.0**
			20	74.8
			100	72.9
			200	71.8

Algorithm 1 Pseudo code of MST in a PyTorch-like style.

```
# f_s: backbone + projection head
# f_t: backbone + projection head
# g: decoder
# m: momentum coefficient
# temp: temperature coefficient
# O_i: output class tokens
# Atten_i: output self-attention map
# Res_i: the input tokens of decoder
# v_1: the coefficient of restoration task
# v_2: the coefficient of basic instance discrimination task

for x in loader: # load a batch x with B samples
    x1, x2 = augment(x), augment(x) # random data augmentation

    with torch.no_grad():
        O_t1, _, Atten_1, O_t2, _, Atten_2 = f_t(x1), f_t(x2)
    O_s1, R_1, _, O_s2, R_2, _ = f_s(x1, Atten_1), f_s(x2,Atten_2)
    Re_1, Re_2 = g(R_1), g(R_2)
    loss1 = 0.5 * (L1_loss(Re_1, x1) + L1_loss(Re_2,x2))
    loss2 = 0.5 * (Loss(O_t1, O_s2) + Loss(O_t2, O_s1))
    loss = v_1 * loss1 + v_2 * loss2
    loss.backward()

    update(f_s)
    f_t = m * f_t + (1 - m) * f_s
    update(m) # update momentum coefficient

def L1_Loss(p,q):
    loss = abs(p-q).sum().mean()

    return loss

def Loss(O_t,O_s):
    O_t = softmax(O_t/temp,dim = 1)
    O_s = softmax(O_s/temp,dim = 1)
    loss = -(O_t * log(O_s)).sum(dim=1).mean()

    return loss
```

5 Linear probing

Following the popular setting of self-supervised learning, 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 SGD optimizer, with a batch size of 1024, weight decay of 0 and learning rate of 0.00024 during 100 epochs on ImageNet training dataset, using only random resized cropping and flipping augmentation. Meanwhile, we evaluate single-crop Top-1 accuracy in the validation dataset. For the linear probing of DeiT-S, we adopt the class tokens of last layer as the input, following the common practice. However, DINO [2] concatenates the late few blocks as the input to the linear classifier. For fair comparison, we adopt the linear probing of DINO as the final result while reporting common linear probing on ablation studies. The results can be observed by Table 2.

Table 2: **Comparison of different strategies of linear probing.** We report Top-1 accuracy on ImageNet validation dataset by using 100-epoch pre-trained DeiT-S model. † adopts the linear probing of DINO.

Method	Architecture	epoch	Linear Top-1 (%)	k-NN Top-1 (%)
Ours	DeiT-S	100	**75.0**	72.1
Ours †			73.9	72.1

6 Impact of longer training

From Table 3, we observe that longer training improves the performance of our method with DeiT-S regardless of the kind of linear probing. This phenomenon is consistent with previous self-supervised learning methods.

Table 3: **Impact of longer training.** † adopts the linear probing of DINO.

Method	Architecture	epoch	Linear (%)	k-NN (%)
Ours	DeiT-S	100	73.9	72.1
Ours		300	**76.3**	**75.0**
Ours †	DeiT-S	100	75.0	72.1
Ours †		300	**76.9**	**75.0**

7 Implementation pseudo code

The complete algorithm of our method is shown as Alg. 1. Our model is optimized by AdamW [8] with learning rate 2×10^{-3} and batch size 1024. The initial weight decay is set to be 0.04. After warmup [9] in the first 10 epochs, the learning rate follows a cosine decay schedule [10]. The model uses multi-crop similar to [1] and data augmentations similar to [4]. The setting of momentum, temperature coefficient, and weight decay follows [2]. The coefficient λ_1 of basic instance discrimination task is set as 1.0 while the restoration task λ_2 is set as 0.6.

Table 4: **Differences with BERT.**

Mask strategy	Mask replacement style	Reconstructing	DINO loss	Linear (%)
Random	BERT	Masked tokens	No	61.0
Random	BERT	Masked tokens	Yes	71.9
Our	BERT	Original image	Yes	73.5
Our	Our	Original image	Yes	**73.9**

8 Differences with BERT

In Table 4, we conduct the experiment by using pure MLM with DeiT-S under 100 epochs, the result is about 40% with the same experimental configuration. Then we further adjust its learning rate and other hyperparameters, the best result is only 61%, which is far lower than that of the DINO by 10.6% (71.6% in Table 6 of our paper) and also lower than the vanilla supervised result by 7.7% (the vanilla supervised result is 68.7%). It shows the pure MLM method may be not suitable for computer vision tasks. Moreover, We experiment with the contrastive loss + BERT solution (that's DINO+pure MLM), the linear result is 71.9%. Our method outperforms its result by 2.0% (73.9%). The result proves our method is better than the original MLM method. Meanwhile, we further conduct the experiment by only replacing the [mask] token with the strategy of pure MLM for our method, the linear result is 73.5%, which also behinds our result. These results fully demonstrate the better setting of MLM for computer vision and further highlight the technical contributions of our paper.

9 The impact of the random mask strategy with different sampling ratios

In the first line of Table 5 of our paper, we already show the results of the random mask strategy with different sampling ratios. We also have tried a small p (0.01) with random masking, the result without BN is 71.1%. When the p is smaller, the performance will be better. The result without BN is best (72.6%, contrastive loss + restore loss) when p is set to 0.

10 The impact of loss weight

Empirically, we set the restoration coefficient λ_2 to 0.6, which makes the contrastive loss and the restoration loss roughly equally weighted. We have also tried several different settings of λ_2 (e.g., 0.2, 0.4, 0.6, 0.8), the result is 73.7%, 73.5%, 73.9% and 73.6% respectively. It can be observed that the results are also insensitive to λ_2, and the best performance is achieved when the two losses are equally weighted.

11 Masking is done after the linear projection

The goal of linear projection is to map the image patches into tokens/embeddings. Following the setting of MLM, the masking (token) strategy should be done after linear projection.

12 Visualization of the attention maps

As shown in Figure 1, we provide the attention maps of supervised and our method. These images consist of original images, attention maps of supervised method, and attention maps of our method. We observe that the visualization of attention maps of our method is clearer than the supervised.

13 Video of attention maps

Similarly, we provide the video of attention maps in supplementary material. The video is from `https://www.youtube.com/watch?v=TZn7oWMHD90&t=12s`.

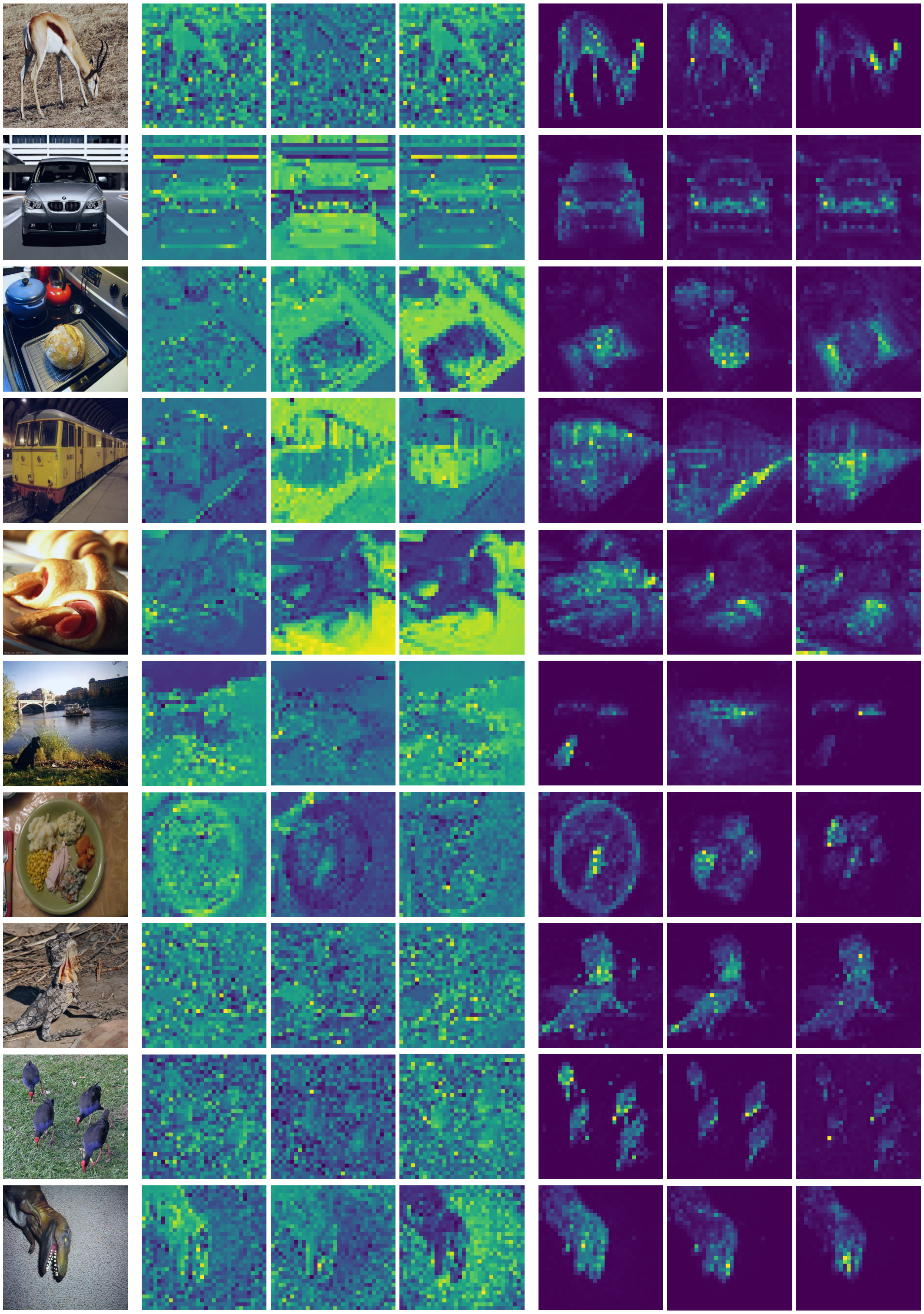

Figure 1: Comparison of the attention map of supervised and our method. Description of images from left to right: (a) the input image, (b) attention map obtained by supervised method, (c) attention map obtained by our method.

References

[1] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924, 2020.

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

[3] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 22243–22255, 2020.

[4] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 21271–21284, 2020.

[5] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021.

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[7] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

[9] 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 preprint arXiv:1706.02677*, 2017.

[10] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.