Are Large-scale Datasets Necessary for Self-Supervised Pre-training?

Appendix

A. Implementation Details

Tokenizers. Similarly to the tokenizer used in [21], all tokenizers presented in Table 2 have a vocabulary of size 8192. For the random tokenizer, we sample 8192 vectors with uniform component-wise distribution. For the random patches tokenizer we sample 8192 patches from different images. For the K-means tokenizer, the 8192 elements of the vocabulary are obtained by applying the K-means algorithm to 3 millions patches sampled from the dataset.

Pre-training. We use the original ViT formulation as proposed by Dosovitskiy et al. [2] and we follow the pre-training hyperparameters of Bao et al. [21]. All baselines reported use the same backbone implementation and trained in similar settings. For SplitMask, by default, we use random block masking [21] of 50% masking ratio to obtain a mask and its complement to extract the two subsets. The maximum and minimum number of patches per block is 75 and 16 respectively. We use the standard random cropping and horizontal flipping as data augmentations. We use 2 transformer layers for the decoder with embedding dimension matching that of the encoder.

However, for the smallest datasets (i.e. Stanford-Cars, ClipArt, Sketch and Paintings), we found that stronger data augmentation and more aggressive masking prevents early overfitting. In particular, we use a uniform masking of 75% (like in the work by He et al. [45]), as well as using random greyscale, solarization, Gaussian blur and color jittering as additional forms of data augmentation.

The BEiT baselines pre-trained on ImageNet and reported in Table 3 and 5 use the DALL-E tokenizer. Other BEiT and SplitMask models have been pre-trained using our random projection tokenizer. For the InfoNCE loss we use $\tau = 0.2$ following Chen et al. [42].

Object detection and Instance segmentation. We use the Mask R-CNN detection method [8] with ViT backbone as our detection method. In order to obtain features compatible with the Feature Pyramid Network (FPN) design [46], we use max pooling and transposed convolution operations similar to El-Nouby et al. [47]. To accommodate for the variable resolution we replace the absolute positional encoding for our models and the baselines with sinusoidal positional encoding [48]. All models are trained using the 3x schedule (36 epochs) unless mentioned otherwise. We use the training hyper-parameters used by Liu et al. [3].

Image classification finetuning. Hyperparameters used for finetuning each of the specific image classification datasets reported in Table 5 is provided in Appendix E.

B. SplitMask vs BEiT

We ablate our proposed components in SplitMask compared to a BEiT baseline in Table 6. All models use a ViT-B backbone and pre-trained for 300 epochs. First, we observe that the ImageNet finetuning performance improves with a margin (+0.5) by simply adopting the encoder-decoder architecture and processing two disjoint subsets per iteration. Second, the global contrastive loss on its own, without the MIM objective, provides a very weak performance. This is expected since there is no training signal for the local patch representations, and a global matching objective with 50% masking of patches may be too hard, providing a noisy training signal and hindering the model's ability to learn informative features.

Our full SplitMask model that uses both the MIM and contrastive objectives obtains the best performance and outperforms BEiT by a large margin of +0.8. The Linear probing performance of SplitMask is stronger than BEiT. However, both models provide a relatively weak performance on this benchmark compared to instance discrimination methods, whose final layers are more aligned to the classification task. Note, SplitMask adds a negligible computing overhead compared to the BEiT baseline: its wall-clock training time is marginally higher as detailed in Table 6. All models are trained using 16 GPUs and batch size of 2048.

Table 6. Ablations of different components in our SplitMask model in comparison with a BEiT baseline. All models including the baseline have been trained for 300 epochs using a ViT-B backbone.

Method	Split	Inpaint	Match	Finetune	Lin.	Hours
BEiT [21]	✗	✓	✗	82.8	41.0	32.5
SplitMask	✓	✓	✗	83.3	46.4	**31.0**
	✓	✗	✓	79.3	4.0	32.5
	✓	✓	✓	**83.6**	**46.5**	34.0

C. Encoder-Decoder vs BEiT

An advantage of the encoder-decoder design we propose in **??** is that it encourages decoupling of general-purpose encoding of image features, which is required for the downstream tasks, and features specific to solving the pretext task of MIM. In particular, compared to BEiT the encoder is not

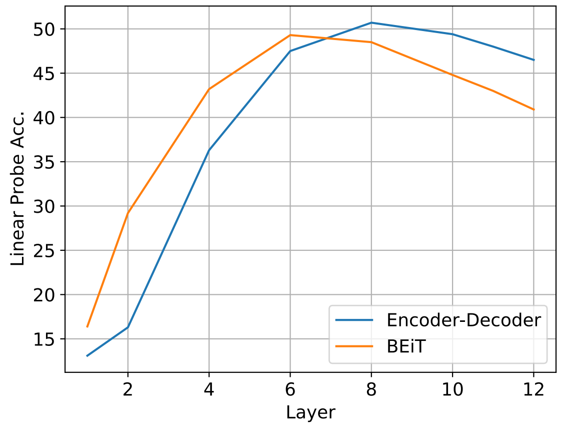

Figure 2. Linear probing accuracy on ImageNet for SplitMask and BEiT using features extracted from different layers.

capable of solving the pretext task on its own since it does not have access to the mask token. Therefore, it can only help solve the task by providing informative representation to the decoder which is the component responsible of solving the pretext task. We can see in Figure 2 that this property improves the transferability of later layers representation to downstream tasks compared to BEiT which has a stronger drop in linear probing performance in later layers.

D. Overfitting during pre-training

We observed that for pre-training of very small datasets (e.g. Stanford-Cars), longer pre-training schedules can be counterproductive. For example, if we follow the assumption we need to pre-training for the same number of updates of ImageNet pre-training for 300 epochs, the Stanford-Cars equivilant schedule would be 45k epochs. However, as we see in Figure 3, pre-training longer than 5k epochs leads to a severe drop in finetuning performance.

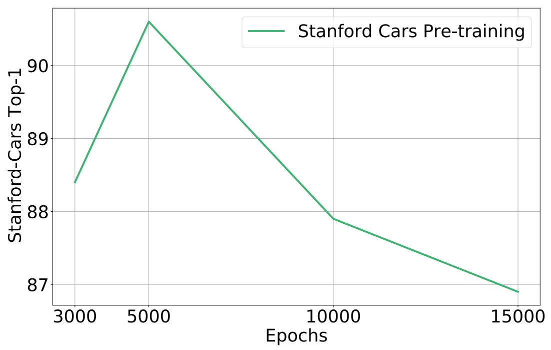

Figure 3. Finetuning performance for the Stanford Cars datasets as a function of number of pre-training epochs using the same datasets images.

E. Image Classification Finetuning

We detail the hyperparameters used to finetune each of the classification datasets in Table 7.

Table 7. Hyperparameters used for finetuning on the different classification datasets

Dataset	iNat18	iNat19	Food 101	Cars	Clipart	Painting	Sketch
Train Res	224	224	224	224	224	224	224
Test Res	224	224	224	224	224	224	224
Epochs	300	300	300	300	300	300	300
Batch size	1024	1024	1024	1024	1024	1024	1024
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate (LR)	1.4e-4	1.4e-4	1.4e-4	4e-3	4e-3	4e-3	4e-3
LR schedule	cosine	cosine	cosine	cosine	cosine	cosine	cosine
LR layer decay small models	✗	✗	✗	0.65	0.65	0.65	0.65
LR layer decay base models	✗	✗	✗	0.65	0.65	0.65	0.65
Weight decay	0.05	0.05	0.05	0.05	0.05	0.05	0.05
Warmup epochs	5	5	5	60	60	60	60
Label smoothing	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Dropout	✗	✗	✗	✗	✗	✗	✗
Stoch. Depth	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Repeated Aug	✓	✓	✓	✗	✗	✗	✗
Gradient Clip.	✗	✗	✗	✗	✗	✗	✗
H. flip	✓	✓	✓	✓	✓	✓	✓
Random Resize Crop	✓	✓	✓	✓	✓	✓	✓
Rand Augment (magnitude/std)	7/0.5	7/0.5	7/0.5	9/0.5	9/0.5	9/0.5 9/0.5	9/0.5
Auto Augment	✓	✓	✓	✓	✓	✓	✓
Mixup alpha	0.8	0.8	0.8	0.8	0.8	0.8	0.8
Cutmix alpha	1.0	1.0	1.0	1.0	1.0	1.0	1.0
ColorJitter	0.4	0.4	0.4	0.4	0.4	0.4	0.4
Test crop ratio	0.875	0.875	0.875	0.875	0.875	0.875	0.875