Broader impacts. The proposed method predicts content based on learned statistics of the training dataset and as such will reflect biases in those data, including ones with negative societal impacts. The model may generate inexistent content. These issues warrant further research and consideration when building upon this work to generate images.

A. Implementation Details

A.1. ImageNet Experiments

ViT architecture. We follow the standard ViT architecture [16]. It has a stack of Transformer blocks [57], and each block consists of a multi-head self-attention block and an MLP block, both having LayerNorm (LN) [1]. The encoder ends with LN. As the MAE encoder and decoder have different width, we adopt a linear projection layer after the encoder to match it. Our MAE adds positional embeddings [57] (the sine-cosine version) to both the encoder and decoder inputs. Our MAE does *not* use relative position or layer scaling (which are used in the code of [2]).

We extract features from the encoder output for fine-tuning and linear probing. As ViT has a class token [16], to adapt to this design, in our MAE pre-training we append an auxiliary dummy token to the encoder input. This token will be treated as the class token for training the classifier in linear probing and fine-tuning. Our MAE works similarly well without this token (with average pooling).

Pre-training. The default setting is in Table 8. We do *not* use color jittering, drop path, or gradient clip. We use xavier_uniform [18] to initialize all Transformer blocks, following ViT's official code [16]. We use the linear *lr* scaling rule [20]: *lr* = *base_lr*×batchsize / 256.

End-to-end fine-tuning. Our fine-tuning follows common practice of supervised ViT training. The default setting is in Table 9. We use layer-wise *lr* decay [10] following [2].

Linear probing. Our linear classifier training follows [9]. See Table 10. We observe that linear probing requires a very different recipe than end-to-end fine-tuning. In particular, regularization is in general harmful for linear probing. Following [9], we disable many common regularization strategies: we do *not* use mixup [69], cutmix [68], drop path [30], or color jittering, and we set weight decay as zero.

It is a common practice to normalize the classifier input when training a classical linear classifier (*e.g.*, SVM [11]). Similarly, it is beneficial to normalize the pre-trained features when training the linear probing classifier. Following [15], we adopt an extra BatchNorm layer [31] without affine transformation (`affine=False`). This layer is applied on the pre-trained features produced by the encoder, and is before the linear classifier. We note that the layer does *not* break the linear property, and it can be absorbed into the linear classifier after training: it is essentially a re-parameterized linear classifier.[3] Introducing this layer helps calibrate the feature magnitudes across different variants in our ablations, so that they can use the same setting without further *lr* search.

config	value
optimizer	AdamW [39]
base learning rate	1.5e-4
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.95 [6]
batch size	4096
learning rate schedule	cosine decay [38]
warmup epochs [20]	40
augmentation	RandomResizedCrop

Table 8. **Pre-training setting.**

config	value
optimizer	AdamW
base learning rate	1e-3
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
layer-wise lr decay [10, 2]	0.75
batch size	1024
learning rate schedule	cosine decay
warmup epochs	5
training epochs	100 (B), 50 (L/H)
augmentation	RandAug (9, 0.5) [12]
label smoothing [52]	0.1
mixup [69]	0.8
cutmix [68]	1.0
drop path [30]	0.1 (B/L) 0.2 (H)

Table 9. **End-to-end fine-tuning setting.**

config	value
optimizer	LARS [66]
base learning rate	0.1
weight decay	0
optimizer momentum	0.9
batch size	16384
learning rate schedule	cosine decay
warmup epochs	10
training epochs	90
augmentation	RandomResizedCrop

Table 10. **Linear probing setting.** We use LARS with a large batch for faster training; SGD works similarly with a 4096 batch.

Partial fine-tuning. Our MAE partial fine-tuning (§4.3) follows the setting in Table 9, except that we adjust the number of fine-tuning epochs. We observe that tuning fewer blocks requires a longer schedule. We set the numbers of fine-tuning epochs as {50, 100, 200} and use the optimal one for each number of blocks tuned.

A.2. Supervised Training ViT-L/H from Scratch

We find that it is nontrivial to train *supervised* ViT-L/H *from scratch* on ImageNet-1K. The training is unstable. While there have been strong baselines with publicly available implementations [53] for smaller models, the recipes for the larger ViT-L/H are unexplored. Directly applying the previous recipes to these larger models does not work. A NaN loss is frequently observed during training.

[3] Alternatively, we can pre-compute the mean and std of the features and use the normalized features to train linear classifiers.

config	value
optimizer	AdamW
base learning rate	1e-4
weight decay	0.3
optimizer momentum	β_1, β_2=0.9, 0.95
batch size	4096
learning rate schedule	cosine decay
warmup epochs	20
training epochs	300 (B), 200 (L/H)
augmentation	`RandAug` (9, 0.5) [12]
label smoothing [52]	0.1
mixup [69]	0.8
cutmix [68]	1.0
drop path [30]	0.1 (B), 0.2 (L/H)
exp. moving average (EMA)	0.9999

Table 11. **Supervised training ViT from scratch.**

method	model	params	acc
iGPT [6]	iGPT-L	1362 M	69.0
iGPT [6]	iGPT-XL	6801 M	72.0
BEiT [2]	ViT-L	304 M	52.1†
MAE	ViT-B	86 M	68.0
MAE	ViT-L	304 M	75.8
MAE	ViT-H	632 M	76.6

Table 12. **Linear probing results of masked encoding methods**. Our fine-tuning results are in Table 3. †: our implementation.

We provide our recipe in Table 11. We use a *wd* of 0.3, a large batch size of 4096, and a long warmup, following the original ViT [16]. We use β_2=0.95 following [6]. We use the regularizations listed in Table 11 and disable others, following [64]. All these choices are for improving training stability. Our recipe can finish training with no NaN loss. The accuracy is 82.6% for ViT-L (81.5% w/o EMA), and 83.1% for ViT-H (80.9% w/o EMA). Both ViT-L and ViT-H show an overfitting trend if not using EMA. As a by-product, our recipe for ViT-B has 82.3% accuracy (82.1% w/o EMA), *vs*. 81.8% in [53].

A.3. Object Detection and Segmentation in COCO

We adapt the vanilla ViT for the use of an FPN backbone [36] in Mask R-CNN [24]. ViT has a stack of Transformer blocks that all produce feature maps at a single scale (*e.g.*, stride 16). We equally divide this stack into 4 subsets and apply convolutions to upsample or downsample the intermediate feature maps for producing different scales (stride 4, 8, 16, or 32, the same as a standard ResNet [25]). FPN is built on these multi-scale maps.

For fair comparisons among different methods, we search for hyper-parameters for each entry in Table 4 (including all competitors). The hyper-parameters we search for are the learning rate, weight decay, drop path rate, and fine-tuning epochs. We will release code along with the specific configurations. For full model and training details, plus additional experiments, see [35].

A.4. Semantic Segmentation in ADE20K

We use UperNet [63] following the semantic segmentation code of [2]. We fine-tune end-to-end for 100 epochs with a batch size of 16. We search for the optimal *lr* for each entry in Table 5 (including all competitors).

dataset	ViT-B	ViT-L	ViT-H	ViT-H$_{448}$	prev best
IN-Corruption ↓ [27]	51.7	41.8	**33.8**	36.8	42.5 [32]
IN-Adversarial [28]	35.9	57.1	68.2	**76.7**	35.8 [41]
IN-Rendition [26]	48.3	59.9	64.4	**66.5**	48.7 [41]
IN-Sketch [60]	34.5	45.3	49.6	**50.9**	36.0 [41]
our supervised training baselines:					
IN-Corruption ↓	45.8	42.3	**41.3**		
IN-Adversarial	27.2	29.6	**33.1**		
IN-Rendition	49.4	**50.9**	50.3		
IN-Sketch	35.6	37.5	**38.0**		

Table 13. **Robustness evaluation on ImageNet variants** (top-1 accuracy, except for IN-C [27] which evaluates mean corruption error). We test the same MAE models (Table 3) on different ImageNet validation sets, *without* any specialized fine-tuning. We provide system-level comparisons with the previous best results.

The semantic segmentation code of [2] uses relative position bias [49]. Our MAE pre-training does *not* use it. For fair comparison, we turn on relative position bias *only* during transfer learning, initialized as zero. We note that our BEiT reproduction uses relative position bias in *both* pre-training and fine-tuning, following their code.

A.5. Additional Classification Tasks

We follow the setting in Table 9 for iNaturalist and Places fine-tuning (Table 6). We adjust the *lr* and fine-tuning epochs for each individual dataset.

B. Comparison on Linear Probing Results

In §4.3 we have shown that linear probing accuracy and fine-tuning accuracy are largely *uncorrelated* and they have different focuses about linear separability. We notice that existing masked image encoding methods are generally less competitive in linear probing (*e.g.*, than contrastive learning). For completeness, in Table 12 we compare on linear probing accuracy with masking-based methods.

Our MAE with ViT-L has 75.8% linear probing accuracy. This is substantially better than previous masking-based methods. On the other hand, it still lags behind contrastive methods under this protocol: *e.g.*, MoCo v3 [9] has 77.6% linear probing accuracy for the ViT-L (Figure 9).

C. Robustness Evaluation on ImageNet

In Table 13 we evaluate the robustness of our models on different variants of ImageNet validation sets. We use the same models fine-tuned on *original* ImageNet (Table 3) and only run inference on the different validation sets, *without* any specialized fine-tuning. Table 13 shows that our method has strong scaling behavior: increasing the model sizes has significant gains. Increasing the image size helps in all sets but IN-C. Our results outperform the previous best results (of specialized systems) by large margins.

In contrast, *supervised* training performs much worse (Table 13 bottom; models described in A.2). For example, with ViT-H, our MAE pre-training is 35% better on IN-A (68.2% vs 33.1%) than the supervised counterpart.

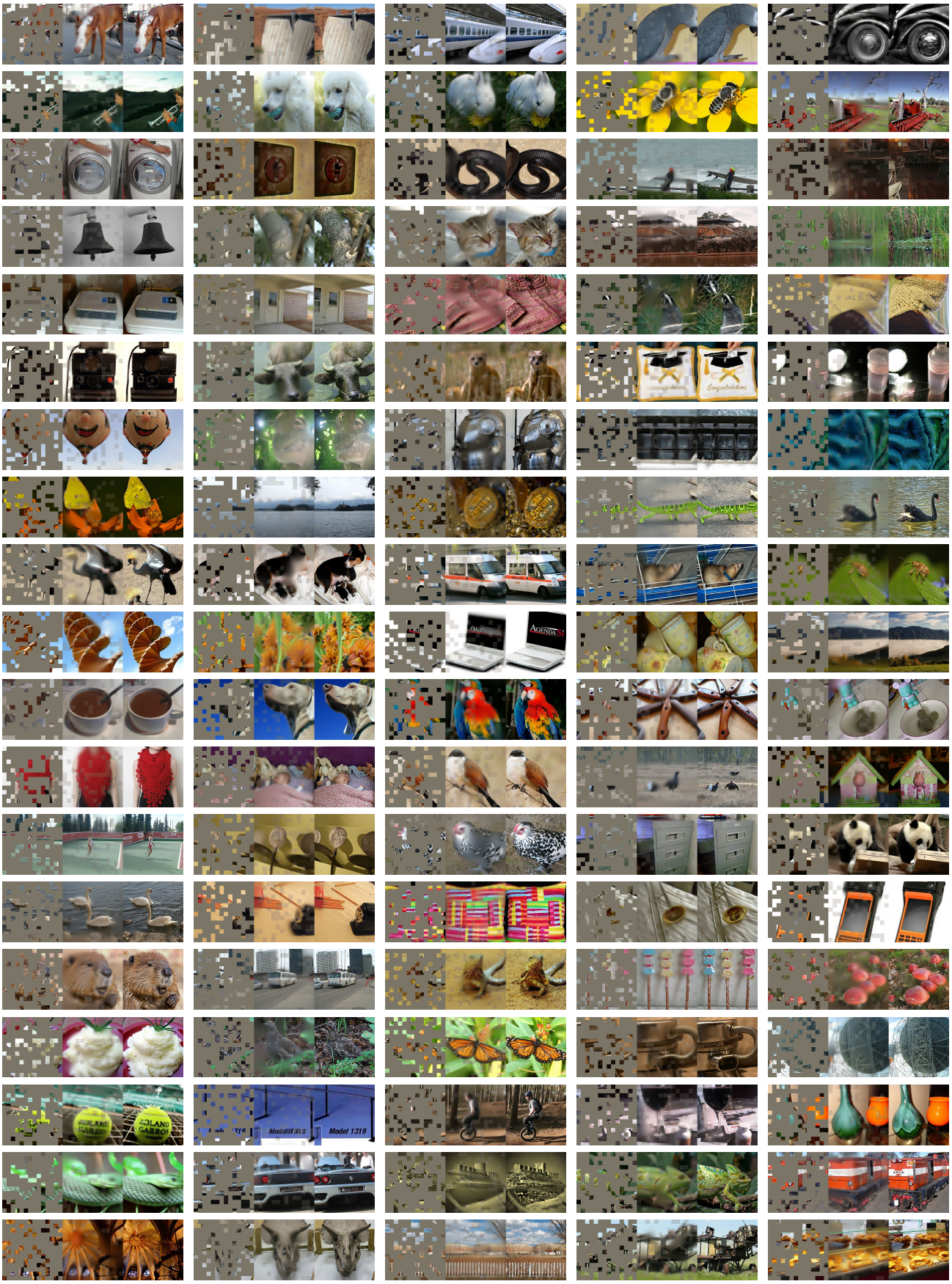

Figure 10. **Uncurated random samples** on ImageNet *validation* images. For each triplet, we show the masked image (left), our MAE reconstruction (middle), and the ground-truth (right). The masking ratio is 75%.

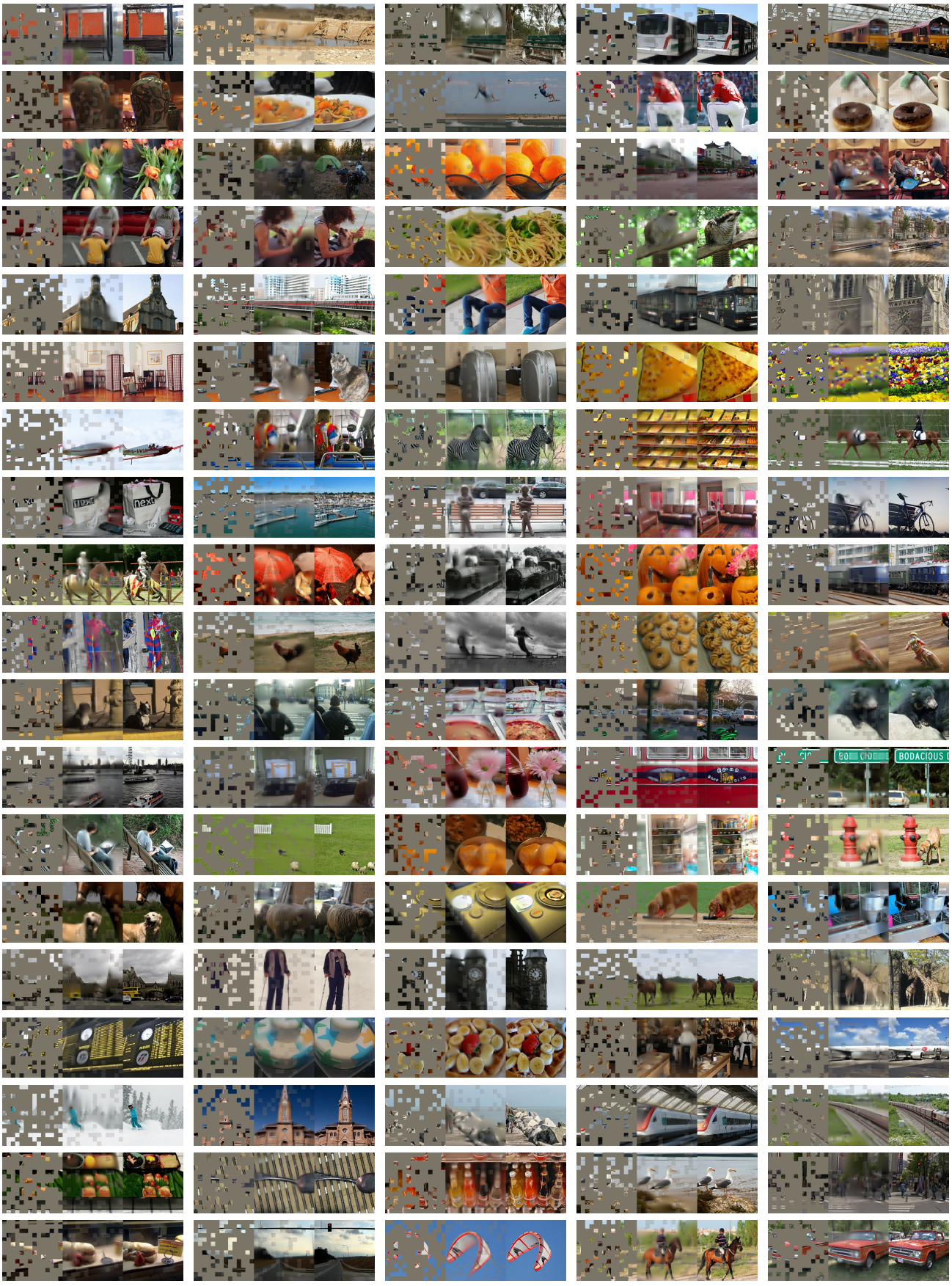

Figure 11. **Uncurated random samples** on COCO validation images, using an MAE trained on ImageNet. For each triplet, we show the masked image (left), our MAE reconstruction (middle), and the ground-truth (right). The masking ratio is 75%.