Hard Patches Mining for Masked Image Modeling
– *Supplementary Material* –

Haochen Wang[1,3] Kaiyou Song[2] Junsong Fan[1,4] Yuxi Wang[1,4] Jin Xie[2] Zhaoxiang Zhang[1,3,4]

[1]Center for Research on Intelligent Perception and Computing,
National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[2]Megvii Technology [3]University of Chinese Academy of Sciences
[4]Centre for Artificial Intelligence and Robotics,
Hong Kong Institute of Science & Innovation, Chinese Academy of Science

{wanghaochen2022, junsong.fan, zhaoxiang.zhang}@ia.ac.cn
{songkaiyou, xiejin}@megvii.com yuxiwang93@gmail.com

Supplementary Material

In this supplementary material, we first provide mode implementation details for reproducibility in Sec. A. Next, in Sec. B, we ablate baselines (*i.e.*, BEiT [1] and iBOT [29]) and decoder designs. The pseudo-code of the easy-to-hard mask generation in a Pytorch-like style is provided in Sec. C. Finally, in Sec. D, we provide both visual and quantitative evidence of our key assumption: *discriminative patches are usually hard to reconstruct.*

A. Implementation Details

ViT Architecture. We follow the standard vanilla ViT [9] architecture used in MAE [11] as the backbone, which is a stack of Transformer blocks [23]. Following MAE [11] and UM-MAE [15], we use the sine-cosine positional embedding. For the downstream classification task, we use features globally averaged from the encoder output for both end-to-end fine-tuning, linear probing, and k-NN classification.

Decoder Design. Our HPM contains two decoders, *i.e.*, the image reconstructor and the loss predictor. These two decoders share the architecture, and each decoder is a stack of Transformer blocks [23] followed by a linear projector.

Effective Training Epochs. Following iBOT [29], we take the effective training epochs as the metric of the training schedule, due to extra computation costs brought by multi-crop [2] augmentation, which is a widely used technique for contrastive methods. Specifically, the effective training epochs are defined as the actual pre-training epochs multiplied with a scaling factor r. For instance, DINO [3] is trained with 2 global 224×224 crops and 10 local 96×96 crops, and thus $r = 2 + (96/224)^2 \times 10 \approx 4$. More details and examples can be found in [29].

Table S1. **Pre-training settings**. By default, we use ViT-B/16 [9] as the backbone and apply 200 epochs pre-training.

config	value
optimizer	AdamW [19]
base learning rate	1.5e-4
weight decay	0.05
momentum	β_1, β_2 = 0.9, 0.95 [4]
layer-wise lr decay [6]	1.0
batch size	4096
learning rate schedule	cosine decay [20]
warmup epochs	10 (ViT-B), 40 (ViT-L)
training epochs	200
augmentation	RandomResizedCrop

A.1. ImageNet Classification

For all experiments in this paper, we take ImageNet-1K [21], which contains 1.3M images for 1K categories, as the pre-trained dataset. By default, we take ViT-B/16 [9] as the backbone and it is pre-trained 200 epochs followed by 100 epochs of end-to-end fine-tuning. Implementation details can be found in Tab. S1, Tab. S2, and Tab. S3. Most of the configurations are borrowed from MAE [11]. The linear learning rate scaling rule [10] is adopted: $lr = lr_{\text{base}} \times \text{batch\_size} \ / \ 256$. For supervised training from scratch, we simply follow the fine-tuning setting without another tuning.

We follow the linear probing setting of MoCo v3 [5]. We do not use mixup [27], cutmix [26], drop path [14], and color jitter. The k-NN classification settings are borrowed from DINO [3]. All images are first resized to 256×256 and then center-cropped to 224×224. We report the best result among $k = 10, 20, 100, 200$.

Table S2. **Fine-tuning settings**. By default, we use ViT-B/16 [9] as the backbone and apply 100 epochs fine-tuning on ImageNet-1K [21] after pre-training.

config	value
optimizer	AdamW [19]
base learning rate	5e-4
weight decay	0.05
momentum	$\beta_1, \beta_2 = 0.9, 0.999$
layer-wise lr decay [6]	0.8
batch size	1024
learning rate schedule	cosine decay [20]
warmup epochs	5
training epochs	100 (ViT-B/16), 50 (ViT-L/16)
augmentation	RandAug (9, 0.5) [8]
label smoothing [22]	0.1
mixup [27]	0.8
cutmix [26]	1.0
drop path [14]	0.1

Table S3. **Linear probing settings**. By default, we use ViT-B/16 [9] as the backbone and apply 100 epochs linear probing on ImageNet-1K [21] after pre-training.

config	value
optimizer	SGD
base learning rate	1e-3
weight decay	0
momentum	$\beta_1 = 0.9$
batch size	4096
learning rate schedule	cosine decay [20]
warmup epochs	10
training epochs	100
augmentation	RandomResizedCrop

Table S4. Ablation study on different **decoder designs**. The speedup is evaluated under 8 Telsa V100 GPUs with 32 images with resolution 224×224 per GPU. The default settings of our proposed HPM are highlighted in color.

# blocks	speedup	fine-tune	linear	k-NN
1	1.94×	82.67	39.83	16.83
2	1.68×	82.50	46.74	22.63
4	1.37×	82.75	53.95	33.60
8	1.00×	**82.95**	54.92	36.09
12	0.76×	82.84	54.83	35.93

# dim	speedup	fine-tune	linear	k-NN
128	1.31×	82.74	42.51	17.67
256	1.18×	82.80	52.39	29.46
512	1.00×	**82.95**	54.92	36.09
1024	0.61×	82.81	54.01	36.54

A.2. COCO Object Detection and Segmentation

Network Architecture. We take Mask R-CNN [12] with FPN [17] as the object detector. Following [11] and [15], to obtain pyramid feature maps for matching the requirements of FPN [17], whose feature maps are all with a stride of 16, we equally divide the backbone into 4 subsets, each consisting of a last global-window block and several local-window blocks otherwise, and then apply convolutions to get the intermediate feature maps at different scales (stride 4, 8, 16, or 32), which is the same as ResNet [13].

Training. We perform end-to-end fine-tuning on COCO [18] for 1× schedule, *i.e.*, 12 epochs, for ablations (*i.e.*, Tab. 6) with 1024×1024 resolution. We simply follow the configuration of ViTDet [16] in detectron2 [24]. Experiments are conducted on 8 Telsa V100 GPUs with a batch size of 16.

A.3. ADE20k Semantic Segmentation

Network Architecture. We take UperNet [25] as the segmentation decoder following the code of [1, 7, 15].

Training. Fine-tuning on ADE20k [28] for 80k iterations is performed for ablations. When compared with previous methods, 160k iterations of fine-tuning are performed. We adopt the exact same setting in mmsegmentation [7]. Specifically, each iteration consists of 16 images with 512×512 resolution. The AdamW [19] optimizer is adopted with an initial learning rate of 1e-4 and a weight decay of 0.05 with ViT-B. For ViT-L, the learning rate is 2e-5. We apply a polynomial learning rate schedule with the first warmup of 1500 iterations following common practice [1, 7, 15]. Experiments are conducted on 8 Telsa V100 GPUs.

B. More Experiments

HPM over other baselines. We study the effectiveness of HPM over BEiT [1] and iBOT [29] in the right table. We perform 200 and 50 epochs pre-training for BEiT [1] and iBOT [29], respectively. Note that iBOT [29] utilizes 2 global crops (224^2) and 10 local crops (96^2). Therefore, the effective pre-training epoch of iBOT-based experiments is $50 \times (2 + \frac{10\times96^2}{224^2}) \approx 200$. From the table, we can tell that HPM brings consistent improvements.

method	fine-tune
BEiT [1]	80.9
HPM (w/ BEiT)	**81.5** ↑ **0.6**
iBOT [29]	82.9
HPM (w/ iBOT)	**83.4** ↑ **0.5**

Ablations on decoder design. Our decoder is a stack of Transformer blocks [23] with a fixed width following [11]. We study its depth and width in Tab. S4. 8 blocks with 512-d features is the best choice, which is exactly the same with MAE [11].

C. Implementation of Easy-to-Hard Masking

Algorithm S1 shows the implementation of easy-to-hard mask generation introduced in Sec. 3.4. Specifically, at training epoch t, we want to generate a binary mask $\mathbf{M}$ with γN patches to be masked. Under the easy-to-hard manner, there are $\alpha_t \gamma N$ patches masked by predicted loss $\hat{\mathcal{L}}^t$ and the remaining $(1 - \alpha_t)\gamma N$ are randomly selected.

Algorithm S1 Pseudo-Code of Easy-to-Hard Masking.

```
# pred_t: predicted reconstruction loss
# t: current epoch
# T: total training epochs

# easy-to-hard mask generation
def mask_generation(pred_t, t, T, mask_ratio):
    L = len(pred_t)
    # total number of visible patches
    len_keep = int(L * (1 - mask_ratio))

    # number of patches masked by predicted loss
    alpha_t = alpha_0 + t/T * (alpha_T - alpha_0)
    len_pred = int(L * mask_ratio * alpha_t)
    ids_shuffle = argsort(pred_t)

    # compute remaining patches
    remain = delete(arange(L) - ids_shuffle[-len_pred:])

    # random masking for remained patches
    ids_shuffle[:(L-len_pred)] = shuffle(remain)

    # generate mask: 0 is remove, 1 is keep
    mask = ones([L,]).bool()
    mask[:len_keep] = 1

    # restore the mask
    ids_restore = argsort(ids_shuffle)
    return gather(mask, ids_restore)
```

D. Hard to Reconstruct *v.s.* Discrimination

Visual evidence. We provide qualitative results on ImageNet-1K [21] *validation* set in Fig. S1 and COCO [18] *validation* set in Fig. S2, respectively. As illustrated in Figs. S1 and S2, patches with higher *predicted* reconstruction loss usually are more discriminative (*i.e.*, object or forehead).

Quantitative evidence. Here, we present a toy experiment to explore the relationship between *hard to reconstruct* and *discrimination for classification*. In the right table, three ViT-B/16 [9] models are trained from scratch on ImageNet-1K for 100 epochs under image-level supervision. *Only 50% patches are input*, and "bottom" and "top" indicates patches with lower and higher $\mathcal{L}_{\text{pred}}$ are visible, respectively. We load HPM pre-trained with 200 epochs for computing $\mathcal{L}_{\text{pred}}$. Empirically, patches with higher $\mathcal{L}_{\text{pred}}$ contribute more to classification. We hope this will inspire future work.

input	accuracy
random 50%	79.1
bottom 50%	78.7 ↓ 0.4
top 50%	**79.8 ↑ 0.7**
all 100%	80.9

References

[1] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. In *International Conference on Learning Representations (ICLR)*, 2022. 1, 2

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1

[4] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning (ICML)*, 2020. 1

[5] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1

[6] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations (ICLR)*, 2020. 1, 2

[7] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 2

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2020. 2

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 2, 3

[10] 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. 1

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[14] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision (ECCV)*, 2016. 1, 2

[15] Xiang Li, Wenhai Wang, Lingfeng Yang, and Jian Yang. Uniform masking: Enabling mae pre-training for pyramid-based vision transformers with locality. *arXiv preprint arXiv:2205.10063*, 2022. 1, 2

[16] Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers. *arXiv preprint arXiv:2111.11429*, 2021. 2

[17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014. 2, 3

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1, 2

[20] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017. 1, 2

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 2015. 1, 2, 3

[22] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 1, 2

[24] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 2

[25] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *European Conference on Computer Vision (ECCV)*, 2018. 2

[26] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1, 2

[27] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2

[28] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[29] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. Image bert pre-training with online tokenizer. In *International Conference on Learning Representations (ICLR)*, 2022. 1, 2

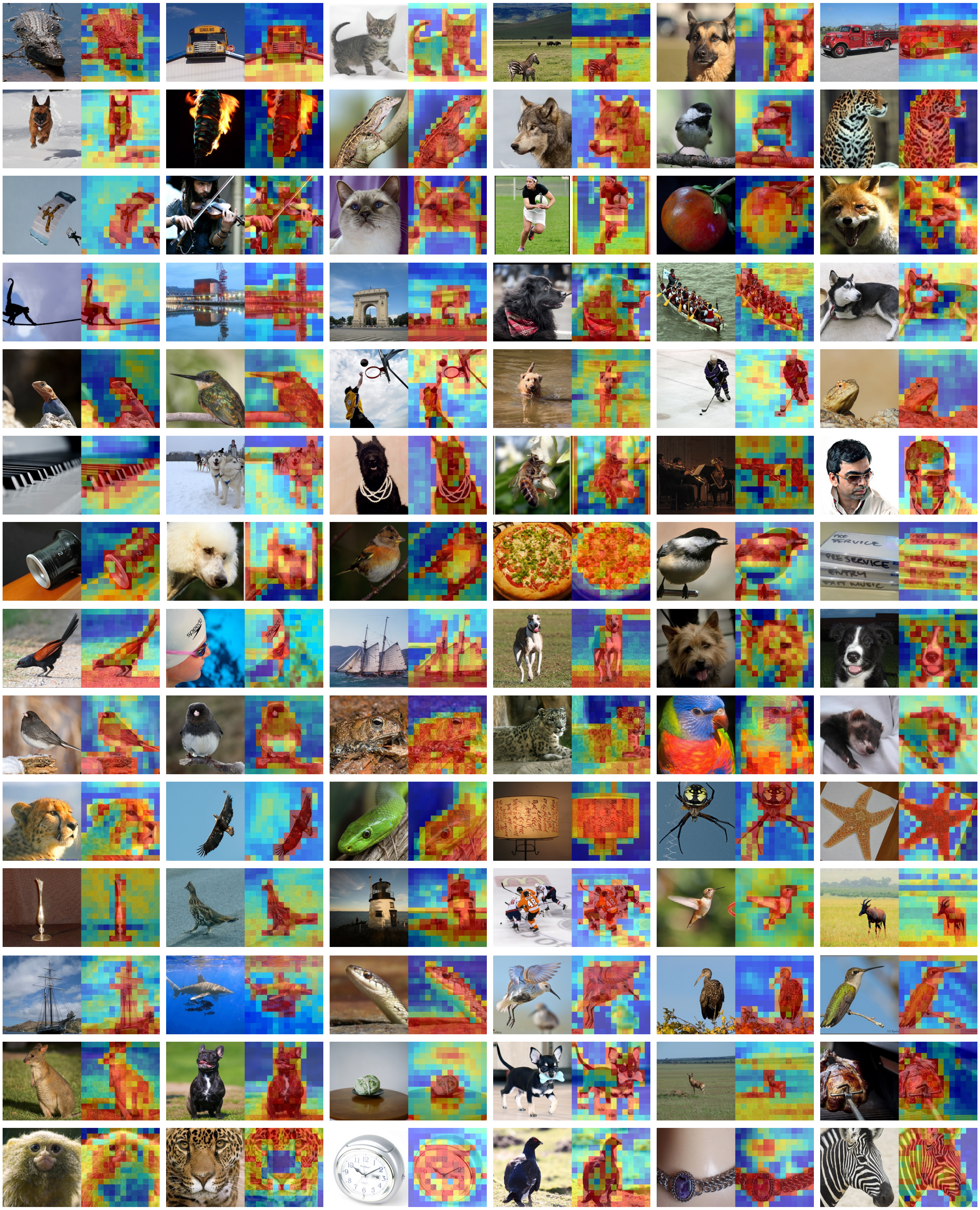

Figure S1. Qualitative results on **ImageNet-1K** *validation* set. For each tuple, we show the *input image* (left) and the patch-wise *predicted* reconstruction loss (right). Red means higher losses and blue indicates the opposite.

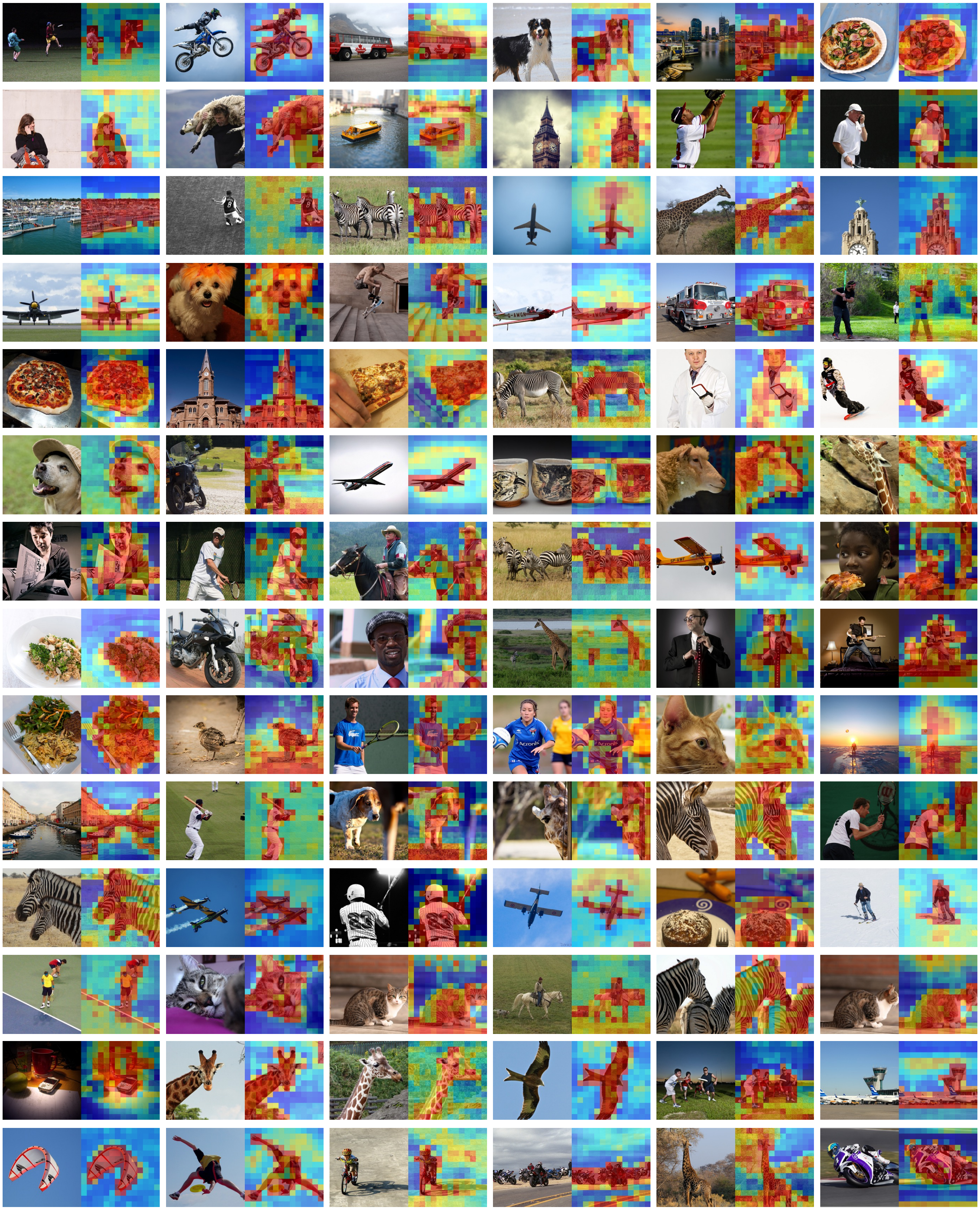

Figure S2. Qualitative results on **COCO** *validation* set. For each tuple, we show the *input image* (left) and the patch-wise *predicted* reconstruction loss (right). Red means higher losses and blue indicates the opposite.