Appendix

In the appendix, we provide additional details, results and visualization. In Appendix A, we provide details about algorithm, high-parameters and implementation of patch selecting strategy of other ranking methods. In Appendix B, we provides more results complementing Fig. 5 . In Appendix C, we provide more visualization compared with other methods on different datasets.

Algorithm 1: Pseudocode in Pytorch Style

```
# me,md:  encoder and decoder
 networks of MAE
# g:  ranking model
# b,n,d:  batchsize,the number of
 visible patches and dimention of
 tokens
for x in loader:
   s = [] # pseudo labels
   xv, xm = RandomSplit(x)
   zv = me(xv) # zv:[b, n, d]
   xa = md(zv)
   for i in zv:
      # remove i-th visible token
      z_ = zv.remove(i)
      x_ = md(z_)
      s.append((xa-x_).abs().mean())
   y = g(xv) # y:[b, 196]
   # Select predictions of visible
    patchesy:[b, n]
   y = y.Select()
   loss = ListMLE(y, s)
   loss.backward() # back-propagate
   update(g) # AdamW

def ListMLE(y,s):
   i=s.argsort(descending=True) # π
   y_=y.gather(i,1)
   # Eq.(3)
   t=y_.exp().flip(1).cumsum(1).filp(1)
   loss=torch.log(t)-y_
   return loss.sum(1).mean()
```

A. Implementation Details

A.1. Pre-training

Hyper-parameters for both pre-training and fine-tuning of LTRP mostly follow MAE (all differences are listed in Tab. 3). The ranking model outputs a 196-d vector corresponding to the semantic density scores of all patches. Considering the sparse design, we only select the dimensions corresponding to the visible patches from this output, and compute the ranking loss. We also present pseudocode implementation of the pre-training in Algorithm 1.

config	value
masking ratio	0.9
epochs	400
weight decay	0.06
loss	ListMLE
MAE model	ViT-B
ranking model	ViT-S

Table 3. **Pre-training Setting**

A.2. Patch Selection

For fairly comparing the leading methods, we only utilize these methods to select patches and thus, their difference lies solely in the patch selection stage. We present the selected patches of different methods in Fig. 11. The detailed implementation of these methods is presented below (assume retaining k tokens):

GradCam. GradCam is a heatmap-based visualization technique. It highlights image regions based on a given category. We regard it as an image perception baseline of traditional supervised classification networks. The network in our GradCam is ViT-S. Notably, the target category is determined by the predicted category, rather than the label. After getting the heatmap with size of 14×14, we preserve patches with the highest values in the heatmap.

GFNet: The original GFNet with ResNet50 selects contiguous region with size of 128×128. In our implement, we select 8×8 patches around the predicted coordinates as informative patches. We utilize the final decision coordinate array from the original paper as our predicted coordinates.

IA-RED^2: We select patches with top k scores by using its official interpretation tool [1]. It predicts scores based on the multi-head interpreter in the 2^{th} group. It employs DeiT-S as its model.

EViT: We preserve the patches corresponding to attentive tokens outputted from the 4^{th} layer of a pre-trained EViT-DeiT-S, a variant of DeiT-S, according to the original paper.

MoCo: We also introduce contrastive learning methods into patch selection. Given the latest heatmap of MoCo with ResNet50, we resize it to 14×14, and also retrain k patches with the highest active values.

DGE: We apply ToMe to dig out the relationship among tokens of DGE. Specifically, we use a dot product similarity metric between the keys (K) of each token in the last layer. Then we choose top-k tokens from the result of class token. We pad images to 256x256 which is consistent with

[1] http://people.csail.mit.edu/bpan/ia-red/

the original input.

DynamicViT: The implementation of DynamicViT provides a list attribute *base_keep_rate* with length 3 in backbone. We first calculate the cube root of *(k /196)*. Then we append it, its square and its cube into this list. We can get a decision after forwarding and get the most informativeness patches from the function *get_keep_indics* with this decision.

AdaViT: We get the remained tokens according to the *token_select* attribute from outputs of backbone.

DINO: We utilize the original code [2] of attention's visualization in the last layer. But we employ the summation result among six heads (ViT-S). Because all head have their own interesting regions, thus summation can effectively capture the overall content of the entire image. We remain tokens with the largest activation value.

ToMe: We set the *r* value in the original code to our keep ratio. Then we can get the clustering group from the *source* attribute of backbone (ViT-S).

LTRP: We get predicted scores for all patches after directly feeding images into the trained ranking model. We select the top-K patches according to different keep ratios.

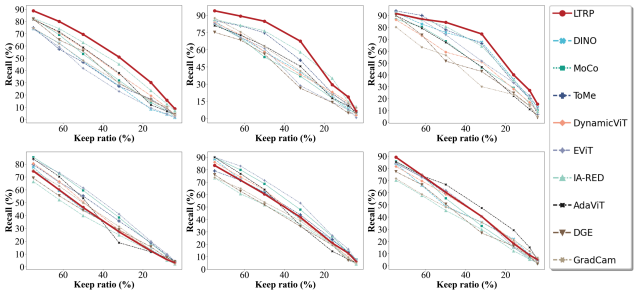


Figure 9. **Recall results** on six classes of ADE20K. LTRP outperforms others in capturing things (such as screen, hood and sculpture, see the upper row), but is slightly inferior in capturing stuff (such as pavement, path and swimming pool, see the bottom row).

B. Patch-level Evaluation

We provide more experiments regarding the patch-level evaluation in this part. First, we manually exclude categories overlapped with ImageNet-1K (learned categories). For MS-COCO, we filter out 13 unseen classes from the 80 classes including *person*, *traffic light*, *fire hydrant*, *stop sign*, *giraffe*, *frisbee*, *surfboard*, *fork*, *sandwich*, *carrot*, *toothbrush*, *sink*, *potted plant*. For PASCAL VOC, we only filter out *person* from the 20 classes. For ADE20K, we do not filter any class because it has 2,693 classes. Alternatively, we provide results of a few representative classes covering things and stuff.

[2] https://github.com/facebookresearch/dino/blob/main/visualize_attention.py

In Fig. 12, we provide results on learned classes of MS-COCO and PASCAL VOC. It has shown that LTRP has highly competitive results compared to other methods in evaluation based on object detection labels. However, it performs worse than the leading methods evaluations based on semantic segmentation masks. This can be interpreted by the fact that LTRP needs to focus on other non-label regions under the same *krs*. In Fig. 9, we notice that LTRP performs well in capturing things categories and slightly worse in capturing stuff. This can be explained as things categories contain more high-level semantics.

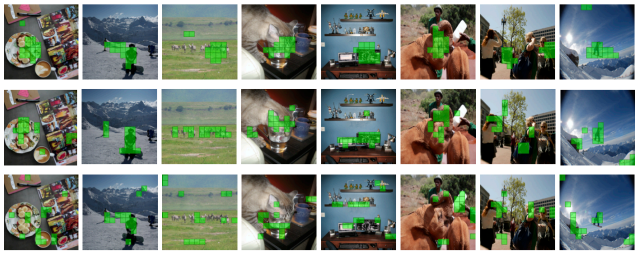


Figure 10. Visualization comparison of self-supervised methods (from top to bottom are MoCo, DINO and our LTRP).

C. More Visualization

We also visualize the results of choosing the most informative patches under an extremely low keep ratio by using different self-supervised methods. As shown in the top of Fig. 10, MoCo focuses on a few local parts and the focus of DINO is more discrete, mainly due to using different backbones (CNN and ViT). Nevertheless, compared to DINO, our LTRP can perceive even more discrete objects. We provide more visualization in Fig. 13 and **????**.

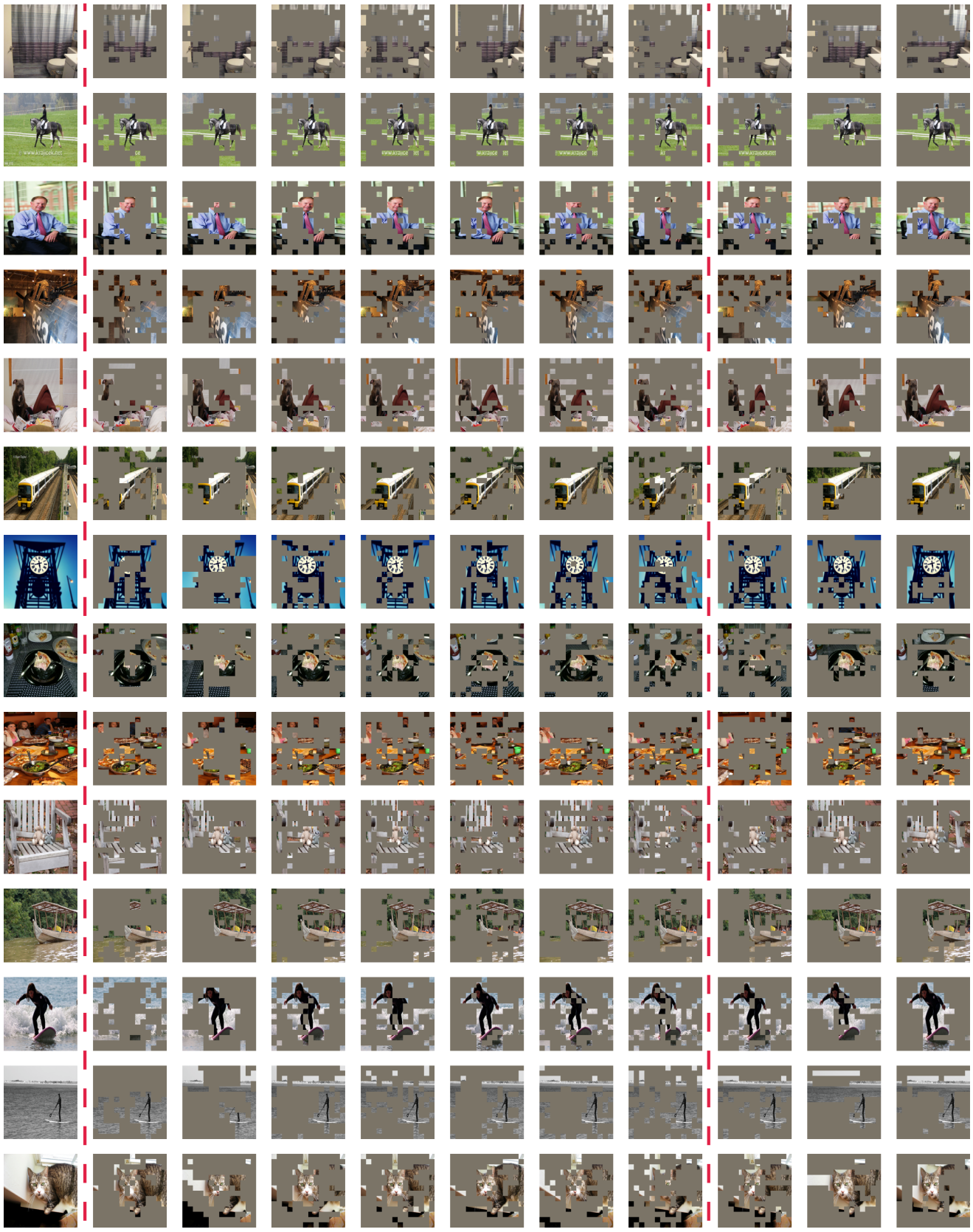


Figure 11. **More visualizations** of redundancy reduction of the compared methods on *validation* set of MS-COCO. They are all pre-trained (unsupervised) or trained (supervised) on ImageNet-1K. From left to right: raw image, GradCam, DGE, EViT, DynamicViT, IA-RED^2, AdaViT, ToMe, MoCo, DINO and LTRP. Supervised methods are located between the two red lines.

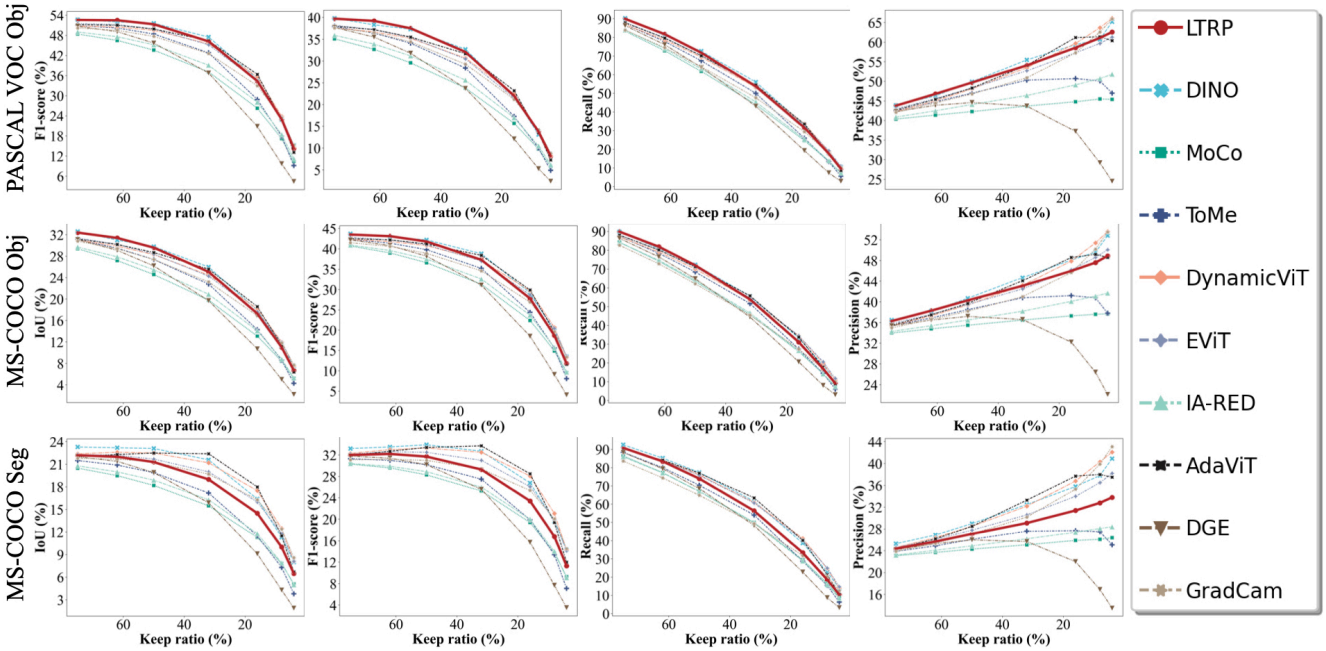


Figure 12. Experimental results on object detection and semantic segmentation datasets. For each dataset, only the learned categories are considered.

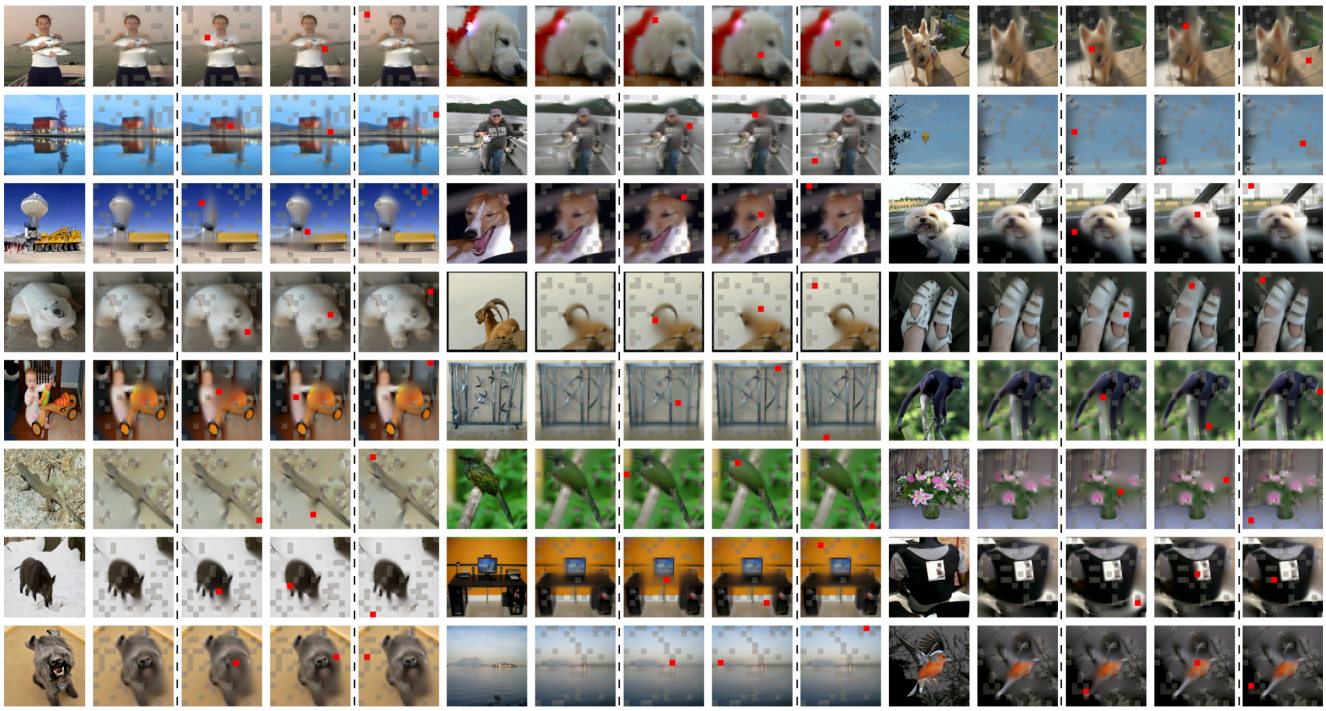
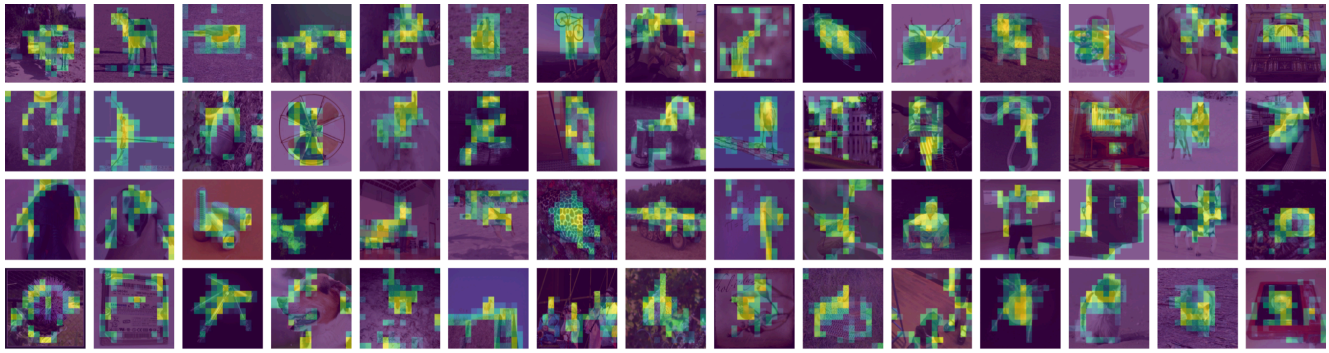
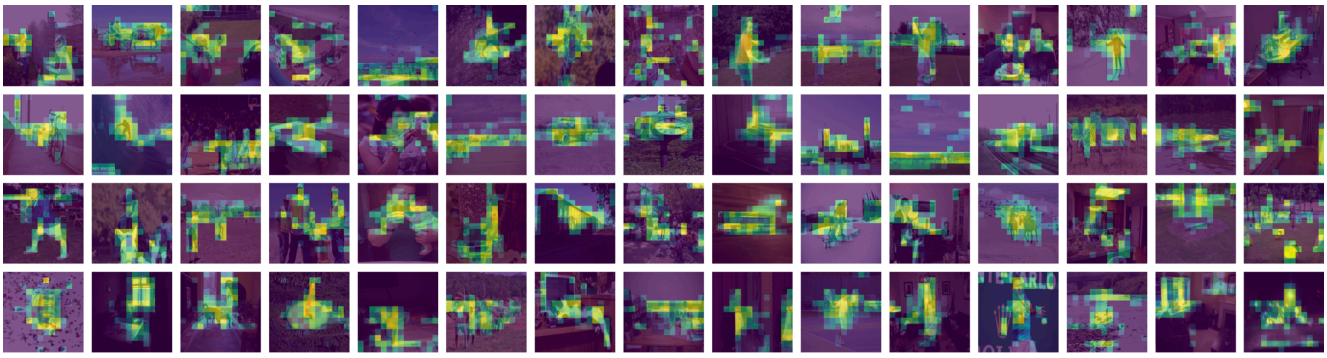


Figure 13. **More examples** of patch removing on *validation* set of ImageNet-1K, using an MAE-B trained on ImageNet-1K. For each quintuplet, we show the raw image, anchor image, two reconstructions with noticeable changes, and one with not detectable change. The red patch denotes the further removed visible patch.



(a) ImageNet-1K



(b) MS-COCO

Figure 14. **More samples** of normalized score maps on *validation* set of ImageNet-1K and MS-COCO generated by using LTRP trained on ImageNet-1K.