– Supplementary Material –
Multi-Label Self-Supervised Learning with Scene Images

Ke Zhu Minghao Fu Jianxin Wu
State Key Laboratory for Novel Software Technology
Nanjing University, China
zhuk@lamda.nju.edu.cn, fumh@lamda.nju.edu.cn, wujx2001@nju.edu.cn

A. Experimental Settings

In the supplementary material, we will provide more detailed experimental settings of our Multi-Label Self-supervised learning (MLS) method, including the 7 small classification datasets, the architecture of MLS and the evaluation metrics & finetuning details.

A.1. The 7 small classification datasets

In our image classification experiment (*c.f*. Table 5 of our main paper), we adopt one multi-label image classification dataset VOC2007 [2] and 7 small single-label classification datasets, including CUB200 [10], Flowers [7], Cars [5], Aircrafts [6], Indoor67 [9], Pets [8] and DTD [1]. We conduct experiments on classification since it has been regarded as a difficult task for current scene-image SSL methods [11]. Specifically, CUB200 has 200 categories of birds, with 5,994 and 5,794 images for training and testing, respectively. Flowers contains 2,040 and 6,149 images for training and testing, with a total of 102 classes. Cars is a relatively large dataset of 196 classes, with 8,144 and 8,041 images for training and testing. Aircrafts dataset comprises 6,667 training and 3,333 testing images with 100 categories. Indoor67, with 67 classes, has 5,360 images for training and 1,340 images for testing. In contrast, Pets (3,680 and 3,669 for training and testing) and DTD (3,760 and 1,880 images for training and testing) are two datasets with smaller capacity, with only 37 and 47 categories, respectively.

A.2. Architecture settings

We have to emphasize that the structure of our MLS method largely resembles that of MoCo-v2. The only difference is that we add a backbone queue Q_g to store the feature g_1 of view1, with the rest architecture the same. The effect of using queue Q_g for label selection is illustrated in Fig. 3 and Table 7 in our main paper. Please note that the embeddings in both Q_g and Q_z *come from* the backbone feature and MLP feature of *view2* and are *detached* from the computation graph. Instead, the gradients are back-propagated from the embedding after MLP layer of view1 (z_1). Following MoCo-v2's implementation, the encoder of view2 *does not* receive gradient and is momentum updated from the weights of view1's encoder. The momentum update process is also applied to the view2's MLP layer (updated by view1's MLP parameters), too.

A.3. Metrics and finetuning

We will now describe our evaluation metrics in detail. For object detection on MS-COCO, we adopt AP, AP^{bbox}_{50}, AP^{bbox}_{75} since most previous detection methods [3, 4] adopt them for evaluation. The same is true for instance segmentation, where we use AP, AP^{seg}_{50}, AP^{seg}_{75} for evaluation. The number $_{50}$ and $_{75}$ both represent the threshold of IoU (Intersection-over-Union). For semantic segmentation in CityScapes dataset, we adopt mIoU, mAcc and aAcc, since previous SSL methods evaluating on this benchmarks all apply these metrics. For image classification, we adopt mAP (mean average precision) for multi-label classification on VOC2007 and top-1 accuracy for 7 small classification datasets as described before. Generally, we take one run (finetuning) on MS-COCO detection and segmentation since these results are more reliable. We take 5 runs on VOC2007 detection (following the advice of MoCo official repository) and report the mean results since the variance on this benchmarks is large. For CityScapes, we take 3 runs for a more stable result. About image classification benchmarks (whether single-label or multi-label), we take 2 runs since we haven't seen much variance during finetuning.

B. Properties of MLS

In this section, we will first describe the merit of our MLS method compared with other loss function (shown quantitatively in Table 6 of the main paper). We then analyze why the key procedure, global matching, works in the MLS. Finally, we will show more meaningful visualizations about the NN image patches discovered by our pipeline.

B.1. Compare with other loss functions

As already verified in Table 6 of our main paper, the proposed pure BCE loss is significantly better than the previous k-NN softmax loss (*c.f*. Eq. 2 in our main paper) and the BCE-pos loss. Now we will discuss intuitively why these loss functions are inferior to ours.

K-NN softmax loss. Firstly, we rewrite Eq. 2 in the main paper here for clarity, denoted as $\mathcal{L}_{knn}$:

$$\begin{aligned} &-\frac{1}{N_{pos}}\sum_{i=1}^{N_{pos}}\log\frac{\exp\left(q\cdot k_{+}^{i}/\tau\right)}{\exp\left(q\cdot k_{+}^{i}/\tau\right)+\sum_{k_{-}}\exp\left(q\cdot k_{-}/\tau\right)} \\ &=-\frac{1}{N_{pos}}\sum_{i=1}^{N_{pos}}\log\frac{\exp\left(q\cdot k_{+}^{i}/\tau\right)}{\sum_{k\in Q_z}\exp\left(q\cdot k/\tau\right)} \\ &=\mathcal{L}_{knn} \end{aligned} \tag{10}$$

where q is equal to z_1 (the embedding after MLP of view1). And that $\{k_{+}^{i}\}_{i=1}^{N_{pos}}$ refer to N_{pos} additional positives mined from the queue Q_z. Please refer to Fig. 2 in our main paper for a correspondence of mathematical symbol. Here we give a simplest example to illustrate the potential drawback of using this k-NN softmax loss.

Suppose $N_{pos}=2$, and the $\mathcal{L}_{knn}$ is formulated as:

$$\mathcal{L}_{knn}=-\frac{1}{2}\log\left[\frac{T(q,k_{+}^{1})}{\sum_{k\in Q_z}T(q,k)}+\frac{T(q,k_{+}^{2})}{\sum_{k\in Q_z}T(q,k)}\right] \tag{11}$$

where $T(x,y)=\exp\left(x\cdot y/\tau\right)$. Since k_{+}^{1} and k_{+}^{2} are *both* mined from the queue Q_z in the k-NN loss, we thus separate the original Q_z as:

$$Q_z=\{k_{+}^{1},k_{+}^{2}\}\ \cup\ Q_{neg}. \tag{12}$$

Now we unfold the negative term in the denominator of the above $\mathcal{L}_{knn}$ loss function as follows:

$$\begin{aligned} \sum_{k\in Q_z}T(q,k)&=\sum_{k\in\{k_{+}^{1},k_{+}^{2}\}}T(q,k)+\sum_{k\in Q_{neg}}T(q,k). \\ &=T(q,k_{+}^{1})+T(q,k_{+}^{2})+\sum_{k\in Q_{neg}}T(q,k). \end{aligned} \tag{13}$$

Finally, by integrating *this* Eq. 12-13 into the *above* Eq. 11, we rewrite the $\mathcal{L}_{knn}$ as (here $N_{pos}=2$):

$$\begin{aligned} &-\frac{1}{2}\log\frac{T(q,k_{+}^{1})}{T(q,k_{+}^{1})+T(q,k_{+}^{2})+\sum_{k\in Q_{neg}}T(q,k)} \\ &\quad-\frac{1}{2}\log\frac{T(q,k_{+}^{2})}{T(q,k_{+}^{2})+T(q,k_{+}^{1})+\sum_{k\in Q_{neg}}T(q,k)} \end{aligned} \tag{14}$$

We will intuitively show that Eq. 14 in the appendix lead to contradictory optimization (gradient) of q!

Actually, the goal of the first term in Eq. 14 is to *pull* q and k_{+}^{1} close while *push* q and k_{+}^{2} (plus those in Q_{neg}) apart. In contrast, the second term in Eq. 14 is to *push* q and k_{+}^{2} close while *push* q and k_{+}^{1} (plus those in Q_{neg}) apart! This fundamental drawback proposed in the k-NN softmax loss could hinder the model from learning quality representation, as its gradient is unstable and fluctuates during the training. Although we only consider a simplest case where $N_{pos}=2$, the same is true (and can be easily verified) for an arbitrary number N_{pos}.

This drawback can be partially derived from a property of softmax: *mutual exclusion*, such that there can only be one distinct positive in the loss term while the others are *all* suppressed. Experiment in Table 6 in the main paper has verified our hypothesis: the improvement by adding k-NN loss function is *marginal* compared with our pure BCE loss.

BCE-pos. The formulation of this loss function is relatively easy, since we directly omit the negative term in our BCE loss (*c.f*. Eq. 9 in the main paper) as follows:

$$\mathcal{L}_{BCE_{pos}}=-\frac{1}{D}\sum_{i=1}^{D}y_i\log\sigma(\frac{p_i}{\tau}) \tag{15}$$

Table 6 in the main paper shows that adding loss of Eq. 15 leads to noticeable improvement over the baseline. Besides, 'BCE-pos' loss function is similar with the form of other SSL methods (*e.g*., SimSiam and BYOL) where only positive terms are considered. This indicates that on top of Eq. 15, the loss that combining *multiple* positives terms with a cosine similarity form (as BYOL and SimSiam does) might also work well in our MLS pipeline.

BCE loss (*ours*). The formulation of the loss function we used is already shown in Eq. 9 in our main paper. It has the following advantages. On one hand, it is not mutually exclusive among all *pseudo* classes such that contradictory gradient will not be obtained as in Eq. 14. On the other hand, it provide multiple positive pairs mined from queue Q_z with abundant semantic meanings, boosting the accuracy of the baseline MoCo-v2 by a large margin (*c.f*. Table 6 in the main paper).

B.2. Why global matching helps in our MLS?

Some attentive readers might have noticed that our MLS used a global matching procedure to define positive and negative labels, and argued that this procedure might not be so accurate, or contain some false positives. This phenonmenon can be found in the Fig. 4 of our main paper, where the matched boxes could have contained partial objects which are *dissimilar* (*e.g*. an query car patch could possibly match an NN bounding box containing both a car and a motorbike). This mismatch could happen, since the embeddings in the dictionary are actually augmented with random-resized-crop (cf. Fig. 4), and thus it can be intrinsically difficult for these embeddings to contain *only* a pure

Figure 5. Visualization of the positives (top-10 NNs) selected by the proposed MLS method. The 6 query images are the *same* with those in the Fig. 4 of our main paper. For more clarity, top-10 NNs are provided here, which demonstrate the effectiveness of our method.

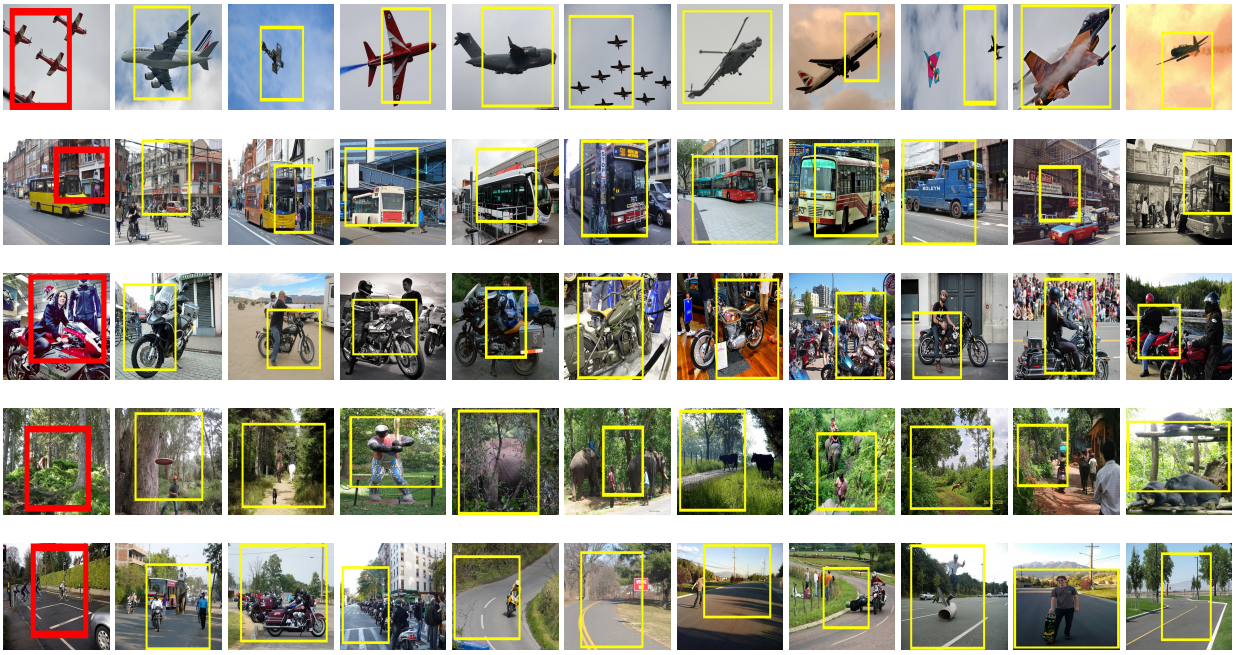

Figure 6. Visualization of the positives (top-10 NNs) selected by the proposed MLS method. For a more fair comparison, here we *randomly* sample another 5 query images from those 60 query images (*c.f*. Fig. 4 in our main paper), and find their top-10 NNs. The cropped query images are shown in those red rectangles, while those cropped NN patches are drawn with yellow rectangles.

object concept—the random crops may contain multiple objects with one dominating. But, global matching *still* works for our MLS.

We take the (*orange*, *banana*) query image in Fig. 4 as an example (*please* also refer to more NNs of this query image in Fig. 5-6 in the appendix). When we use it as the query image, the NNs are mostly in a form of (*orange*, X) or (*banana*, Y), where X and Y may contain concepts different from *orange* or *banana*. However, as the figures illustrate, the matched NNs are almost always *dominated* by orange or banana, that is, the matched NNs are mostly in a form of:

$$\{(banana, X), (orange, Y), (orange, X), ...\}. \quad (16)$$

As a result, the harmful gradient of X and Y will be dominated by *orange* or *banana* during optimization, and our global matching will be helpful in the MLS. The visualizations in Fig. 5-6 in the appendix also verify this hypothesis: the matched NN are mostly dominated by the concept same (or similar) as the query boxes. We have also conducted an quantitative experiment in Table 8 (in our main paper) to show the optimal working condition of this global matching: the number of globally matched NN should not be too low or too high, and a medium number of NN number (k=20) is enough.

B.3. More visualizations of MLS

As already shown in Fig. 4 in our main paper, our MLS method could effectively capture semantically similar correspondence across the datasets. To further and better analyze the quality of NNs and to reduce the randomness during selection, we visualize more NNs below.

We first consider enlarging the number of NNs by keeping the query images (patches) *same* as those in Fig. 4 of our main paper. As shown in Fig. 5, even the k becomes larger, our MLS can almost always find its NNs correctly, with visible semantic similarity across those images.

Besides, to reduce the randomness during query images (patches) sampling, we *randomly* sample another 5 query images, and then find their top-10 NNs from the queue Q_g. As shown in Fig. 6, our MLS pipeline can *truly* find similar objects with intra-class variance and multiple positive partial concepts. We conclude that involving more positives with our BCE loss is really helpful for scene images SSL!

References

[1] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014.

[2] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[3] Ross Girshick. Fast r-cnn. In *ICCV*, pages 1440–1448, 2015.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, pages 2961–2969, 2017.

[5] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3D object representations for fine-grained categorization. In *ICCV Workshop on 3D Representation and Recognition*, 2013.

[6] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[7] Maria-Elena Nilsback and Andrew Zisserman. A visual vocabulary for flower classification. In *CVPR*, pages 1447–1454, 2006.

[8] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, pages 3498–3505, 2012.

[9] Ariadna Quattoni and Antonio Torralba. Recognizing indoor scenes. In *CVPR*, pages 413–420, 2009.

[10] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[11] Enze Xie, Jian Ding, Wenhai Wang, Xiaohang Zhan, Hang Xu, Peize Sun, Zhenguo Li, and Ping Luo. DetCo: Unsupervised contrastive learning for object detection. In *ICCV*, pages 8392–8401, 2021.