Supplementary Material

Self-supervised Label Augmentation via Input Transformations

A. Comparison with ten-crop

In this section, we compare the proposed aggregation method (SLA+AG) using rotation with a widely-used aggregation scheme, ten-crop (Krizhevsky et al., 2012), which aggregates the pre-softmax activations (i.e., logits) over a number of cropped images. As reported in Table 1, the aggregation using rotation performs significantly better than ten-crop.

Table 1. Classification accuracy (%) of the ten-crop and our aggregation using rotation (SLA+AG). The best accuracy is indicated as bold, and the relative gain over the baseline is shown in brackets.

Dataset	Baseline	ten-crop	SLA+AG
CIFAR10	92.39	93.33 (+1.02%)	**94.50** (+2.28%)
CIFAR100	68.27	70.54 (+3.33%)	**74.14** (+8.60%)
tiny-ImageNet	63.11	64.95 (+2.92%)	**66.95** (+6.08%)

B. Experiments with Composed Transformations

In this section, we present the more detailed experimental results of composed transformations described in the main text (Section 3.3 and Table 4). We additionally report the performance of the single inference (SLA+SI) with an additional dataset, Stanford Dogs. When using $M = 12$ composed transformations, we achieve the best performance, 20.8% and 15.4% relatively higher than baselines on CUB200 and Stanford Dogs, respectively.

Table 2. Classification accuracy (%) of SLA based on the set (each row) of composed transformations. We first choose subsets of rotation and color permutation (see first two columns) and compose them where M is the number of composed transformations. We use SLA with the composed transformations when training models. The best accuracy is indicated as bold.

Composed transformations $T = T_r \times T_c$			CUB200		Stanford Dogs	
Rotation T_r	Color permutation T_c	M	SLA+SI	SLA+AG	SLA+SI	SLA+AG
0°	RGB	1	54.24		60.62	
0°, 180°	RGB	2	56.62	58.92	63.57	65.65
0°, 90°, 180°, 270°	RGB	4	60.85	64.41	65.67	67.03
0°	RGB, GBR, BRG	3	52.91	56.47	63.26	65.87
0°	RGB, RBG, GRB, GBR, BRG, BGR	6	56.81	61.10	64.83	67.03
0°, 180°	RGB, GBR, BRG	6	56.14	60.87	65.45	68.75
0°, 90°, 180°, 270°	RGB, GBR, BRG	12	60.74	**65.53**	**66.40**	**69.95**
0°, 90°, 180°, 270°	RGB, RBG, GRB, GBR, BRG, BGR	24	**61.67**	65.43	64.71	67.80

C. Self-supervised Label Augmentation with Thousands of Labels

Since the proposed technique (SLA) increases the number of classes in a task, one could wonder that the technique is scalable with respect to the number of labels. To demonstrate the scalability of SLA, we train ResNet-50 (He et al., 2016) on ImageNet (Deng et al., 2009) and iNaturalist (Van Horn et al., 2018) datasets with the same experimental settings of tiny-ImageNet as described in the main text (Section 3.1) except the number of training iterations. In this experiment, we train models for 900K and 300K iterations (roughly 90 epochs) for ImageNet and iNaturalist, respectively. As reported in Table 3, our method also provides a benefit on the large-scale datasets.

Table 3. Classification accuracy (%) on ImageNet (Deng et al., 2009) and iNaturalist (Van Horn et al., 2018) with SLA using rotation. N indicates the number of labels in each dataset. The relative gain over the baseline is shown in brackets. Note that the reported accuracies are obtained from only one trial.

Dataset	N	Baseline	SLA+SI	SLA+AG	SLA+SD
ImageNet (Deng et al., 2009)	1000	75.16	75.81 (+0.86%)	77.16 (+2.66%)	76.17 (+1.34%)
iNaturalist (Van Horn et al., 2018)	8142	57.12	61.31 (+7.34%)	62.97 (+10.2%)	61.52 (+7.70%)

D. Combining with A Self-supervised Pre-training Technique

While self-supervised learning (SSL) techniques primarily target unsupervised learning or pre-training, we focus on joint-supervised-learning (from scratch) with self-supervision to improve upon the original supervised learning model. Thus our SLA framework is essentially a supervised learning method, and is not comparable with SSL methods that train with unlabeled data.

Yet, since our SLA is orthogonal from SSL, we could use a SSL technique as a pre-training strategy for our scheme as well. To validate this, we pre-train ResNet-18 (He et al., 2016) on CIFAR-10 (Krizhevsky et al., 2009) using a SOTA contrastive learning method, SimCLR (Chen et al., 2020), and then fine-tune it on the same dataset. As reported in Table 4, under the fully-supervised settings, pre-training the network only with SimCLR yields a marginal performance gain. On the other hand, when using SimCLR for pre-training and ours for fine-tuning, we achieve a significant performance gain, which shows that the benefit of our approach is orthogonal to pre-training strategies.

Table 4. Classification accuracy (%) on CIFAR-10 (Krizhevsky et al., 2009) with SimCLR (Chen et al., 2020) and our SLA framework.

Initialization	Fine-tuning	Accuracy (%)
Random initialization	Baseline	95.26
	SLA+SD (ours)	96.19
SimCLR pre-training	Baseline	95.44
	SLA+SD (ours)	96.55

E. Implementation with PyTorch

One of the strengths of the proposed self-supervised label augmentation is simple to implement. Note that the joint label $y = (i, j) \in [N] \times [M]$ can be rewritten as a single label $y = M \times i + j$ where N and M are the number of primary and self-supervised labels, respectively. Thus self-supervised label augmentation (SLA) can be implemented in PyTorch as follows. Note that `torch.rot90(X, k, ...)` is a built-in function which rotates the input tensor X by $90k$ degrees.

Listing 1. Training script of self-supervised label augmentation.

```
for inputs, targets in train_dataloader:
    inputs = torch.stack([torch.rot90(inputs, k, (2, 3)) for k in range(4)], 1)
    inputs = inputs.view(-1, 3, 32, 32)
    targets = torch.stack([targets*4+k for k in range(4)], 1).view(-1)

    outputs = model(inputs)
    loss = F.cross_entropy(outputs, targets)

    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Listing 2. Evaluation script of self-supervised label augmentation with single (SLA+SI) and aggregated (SLA+AG) inference.

```
for inputs, targets in test_dataloader:
    outputs = model(inputs)
    SI = outputs[:, ::4]

    inputs = torch.stack([torch.rot90(inputs, k, (2, 3)) for k in range(4)], 1)
    inputs = inputs.view(-1, 3, 32, 32)
    outputs = model(inputs)
    AG = 0.
    for k in range(4):
        AG = AG + outputs[k::4, k::4] / 4.

    SI_accuracy = compute_accuracy(SI, targets)
    AG_accuracy = compute_accuracy(AG, targets)
```

As described above, applying input transformations (e.g., line 2-3 in Listing 1) and label augmentation (e.g., line 4 in Listing 1) is enough to implement SLA. We here omit the script for SLA with self-distillation (SLA+SD), but remark that its implementation is also simple as SLA. We think that this simplicity could lead to the broad applicability for various applications.

References

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8769–8778, 2018.