Implicit Contrastive Representation Learning with Guided Stop-gradient - Appendix

Byeongchan Lee*
Gauss Labs
Seoul, Korea
byeongchan.lee@gausslabs.ai

Sehyun Lee*
KAIST
Daejeon, Korea
sehyun.lee@kaist.ac.kr

A Algorithm

We present a PyTorch-style pseudocode for SimSiam with GSG. It can be written analogously for BYOL with GSG. The pseudocode can be vectorized more, but we offer it in its current form for a better understanding.

Algorithm 1 SimSiam with GSG

```
# pytorch style pseudocode

for batch in loader:
    loss = []
    for x1, x2 in zip(batch, shuffle(batch)):
        x11, x12 = aug(x1), aug(x1) # views
        x21, x22 = aug(x2), aug(x2) # aug: random augmentation

        z11, z12 = f(x11), f(x12) # projections
        z21, z22 = f(x21), f(x22) # f: encoder = backbone + projector (mlp)

        p11, p12 = h(z11), h(z12) # predictions
        p21, p22 = h(z21), h(z22) # h: predictor (mlp)

        d1121 = d(z11, z21) # Euclidean distances
        d1122 = d(z11, z22)
        d1221 = d(z12, z21)
        d1222 = d(z12, z22)
        m = min(d1121, d1122, d1221, d1222)

        # D: negative cosine similarity
        # sg: stop-gradient, sg(z) = z.detach()
        if d1121 == m:
            l = D(p11, sg(z12)) + D(p21, sg(z22))
        elif d1122 == m:
            l = D(p11, sg(z12)) + D(p22, sg(z21))
        elif d1221 == m:
            l = D(p12, sg(z11)) + D(p21, sg(z22))
        elif d1222 == m:
            l = D(p12, sg(z11)) + D(p22, sg(z21))
        loss.append(0.5 * l)

    loss = sum(loss) / len(loss) # mean
    loss.backward() # back-propagate
    update(f, h) # SGD update
```

*These authors contributed equally to this work.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

B Data augmentation details

B.1 ImageNet

The training set consists of 1,281,167 images, and the validation dataset consists of 50,000 images. Since the label of ImageNet's test set is not provided, we use the validation set for evaluation. The number of classes in ImageNet is 1,000. The images are variable-sized, so they are cropped to size 224×224 during the data augmentation process. We apply in turn the following data transformations in pre-training.

- `RandomResizedCrop`: Crop a random patch of the image with scale $(0.2, 1)$ and then resize it to a size of $(224, 224)$.
- `ColorJitter`: With a probability of 0.8, change the image's brightness, contrast, saturation, and hue with strength $(0.4, 0.4, 0.4, 0.1)$.
- `RandomGrayscale`: With a probability of 0.2, convert the image to grayscale.
- `GaussianBlur`: With a probability of 0.5, apply the Gaussian blur filter to the image with a radius uniformly sampled from $[0.1, 2]$.
- `RandomHorizontalFlip`: With a probability of 0.5, flop the image horizontally.
- `Normalize`: Normalize the image with mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$.

B.2 CIFAR-10

The training set consists of 60,000 images, and the test set consists of 10,000 images. The number of classes in CIFAR-10 is 10. The images have a fixed size of 32×32. We apply in turn the following data transformations in pre-training.

- `RandomResizedCrop`: Crop a random patch of the image with scale $(0.08, 1)$ and then resize it to a size of $(32, 32)$.
- `RandomHorizontalFlip`: With a probability of 0.5, flop the image horizontally.
- `ColorJitter`: With a probability of 0.8, change the image's brightness, contrast, saturation, and hue with strength $(0.4, 0.4, 0.4, 0.1)$.
- `RandomGrayscale`: With a probability of 0.2, convert the image to grayscale.
- `Normalize`: Normalize the image with mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$.

C Pre-training implementation details

C.1 ImageNet

For the ImageNet experiments, we follow the experimental setup in SimSiam's original paper [Chen and He, 2021] and `https://github.com/facebookresearch/simsiam`. The encoder f consists of a backbone and a projector. The backbone is ResNet-50 [He et al., 2016] with about 23M parameters, and the projector consists of three fully-connected (FC) layers. Batch normalization (BN) and ReLU are applied to each layer except for the output layer, where only BN is used. The input and output dimensions of the layers are 2,048. The predictor h consists of two FC layers. BN and ReLU are applied to the first layer, and nothing is applied to the second layer. The input and output dimensions of the first layer are 2,048 and 512, and the input and output dimensions of the second layer are 512 and 2,048 (bottleneck structure).

C.2 CIFAR-10

For the CIFAR-10 experiments, we follow the default setup in [Susmelj et al., 2020] and `https://github.com/lightly-ai/lightly`. The encoder f consists of a backbone and a projector. The backbone is a variant of ResNet-18 for CIFAR-10 with about 11M parameters, and the projector consists of two FC layers. BN and ReLU are applied to the first layer, and only BN is applied to the

second layer. The input and output dimensions of the first layer are 512 and 2,048, and the input and output dimensions of the second layer are 2,048. The predictor h is the same as in the case of ImageNet.

D Transfer learning implementation details

We present the implementation details for transfer learning. Other details can be found in [Ericsson et al., 2021] and `https://github.com/linusericsson/ssl-transfer`. We minimally change the code under the same computational parameters.

D.1 Image recognition

In the Caltech dataset, 30 images for each class are randomly selected to form the training set, and the remaining images form the test set. On the DTD and SUN397 datasets, the first dataset split provided by the authors is used. For validation sets, the pre-defined ones are used on the Aircraft, DTD, Flowers, and VOC2007 datasets, and on the rest of the datasets, 20% of the training set are randomly selected to form the validation set.

We resize the images so that the number of pixels of the shorter side of each image is 224 using bicubic interpolation. We crop the images at the center with size $(224, 224)$. The best weight decay rate is selected on the validation set over 45 evenly spaced values from 10^{-6} to 10^{5} on a logarithmic scale with base 10. We find the best hyperparameters on the validation set, retrain the networks on the training and validation sets, and evaluate on the test set. Since VOC2007 is a multi-label dataset, we fit a binary classifier for each class. L-BFGS [Liu and Nocedal, 1989] is used as a solver when doing a logistic regression.

D.2 Object detection

We initially normalize the images with mean $(123.675, 116.280, 103.530)$ and standard deviation $(58.395, 57.120, 57.375)$.

Pascal-VOC The images are resized so that the number of pixels of the shorter side of each image is one of $\{480, 512, 544, 576, 608, 640, 672, 704, 736, 768, 800\}$ for training and 800 for test. We use the batch size of 2. The base learning rate is chosen as 0.0025, and it decays by tenths at the $96,000$th and $128,000$th iterations, respectively. We warm up the networks for 100 iterations and train them for $144,000$ iterations.

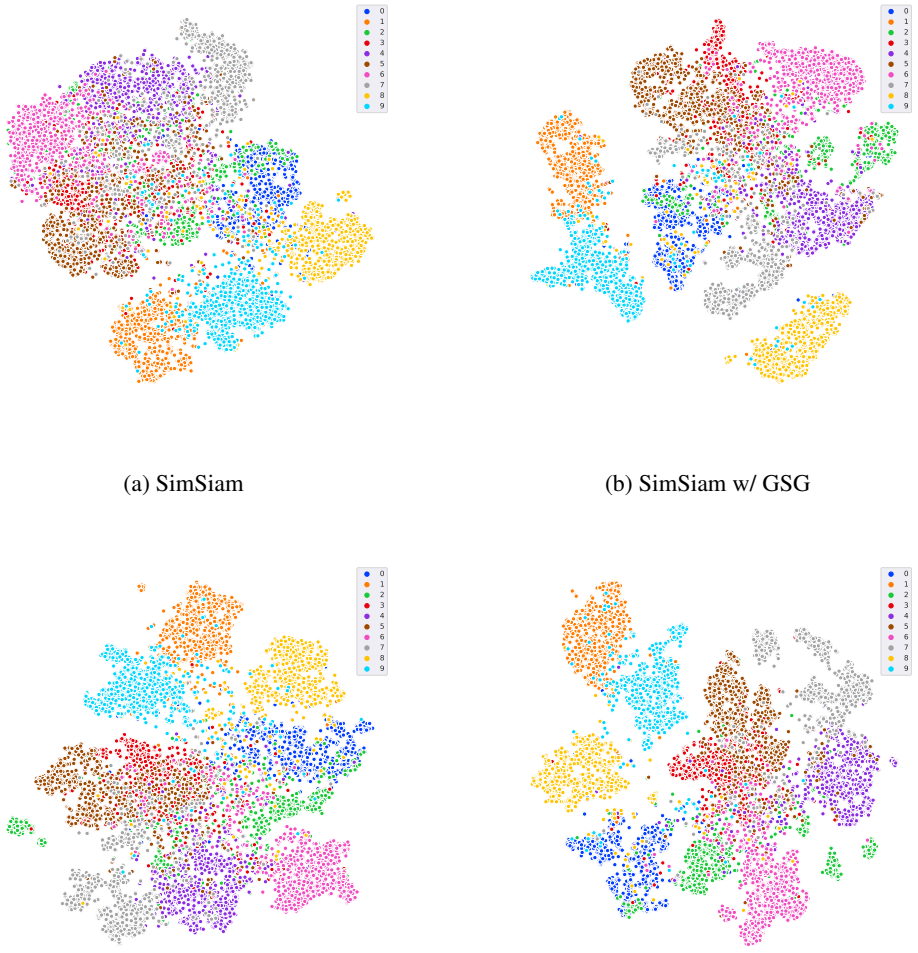
MS-COCO The images are resized so that the number of pixels of the shorter side of each image is one of $\{640, 672, 704, 736, 768, 800\}$ for training and 800 for test. We use the batch size of 16. We choose the base learning rate as 0.02, and it decays by tenths at the $210,000$th and $250,000$th iterations, respectively. We warm up the networks for 1000 iterations and train them for $270,000$ iterations.

D.3 Semantic segmentation

We use the SGD optimizer with momentum of 0.9, learning rate of 0.02, weight decay rate of 0.0001, and learning rate decay rate of 0.9. We use the batch size of 2. The networks are trained for $150,000$ iterations in total (30 epochs and $5,000$ iterations per epoch). The learning rate decays every 500 iterations.

E t-distributed stochastic neighbor embedding (t-SNE)

We present t-SNE visualizations [Van der Maaten and Hinton, 2008] of representations of images from the test set of CIFAR-10. Figure 1 shows that in both SimSiam and BYOL, applying our GSG made the clusters corresponding to the classes better separate.



(a) SimSiam (b) SimSiam w/ GSG

(c) BYOL (d) BYOL w/ GSG

Figure 1: t-SNE visualizations of CIFAR-10. Clusters of our algorithms are better separated than those of the original algorithms as expected.

F Semantic segmentation

We present semantic segmentation results (test image, ground truth, and predicted results of each algorithm) of the first ten images from the validation set of ADE20K. Figure 2 shows that the representations pre-trained by our algorithms transfer well to the semantic segmentation task. Also, the performance of our algorithms is similar to or better than that of the existing algorithms.

G Limitations and future directions

Since this is the first study in the direction of implicitly applying the idea of contrastive learning, we adopt algorithms with simple losses (such as SimSiam and BYOL) as the benchmark algorithms. To do so, we want to see vividly and not to obscure the effect of our method. The performance may be further improved when applied to algorithms that utilize more complicated strategies. For example, there are algorithms that use a multi-crop strategy [Caron et al., 2020] where both global and local views are considered [Caron et al., 2021, Zhang et al., 2022, Wanyan et al., 2023]. Combining

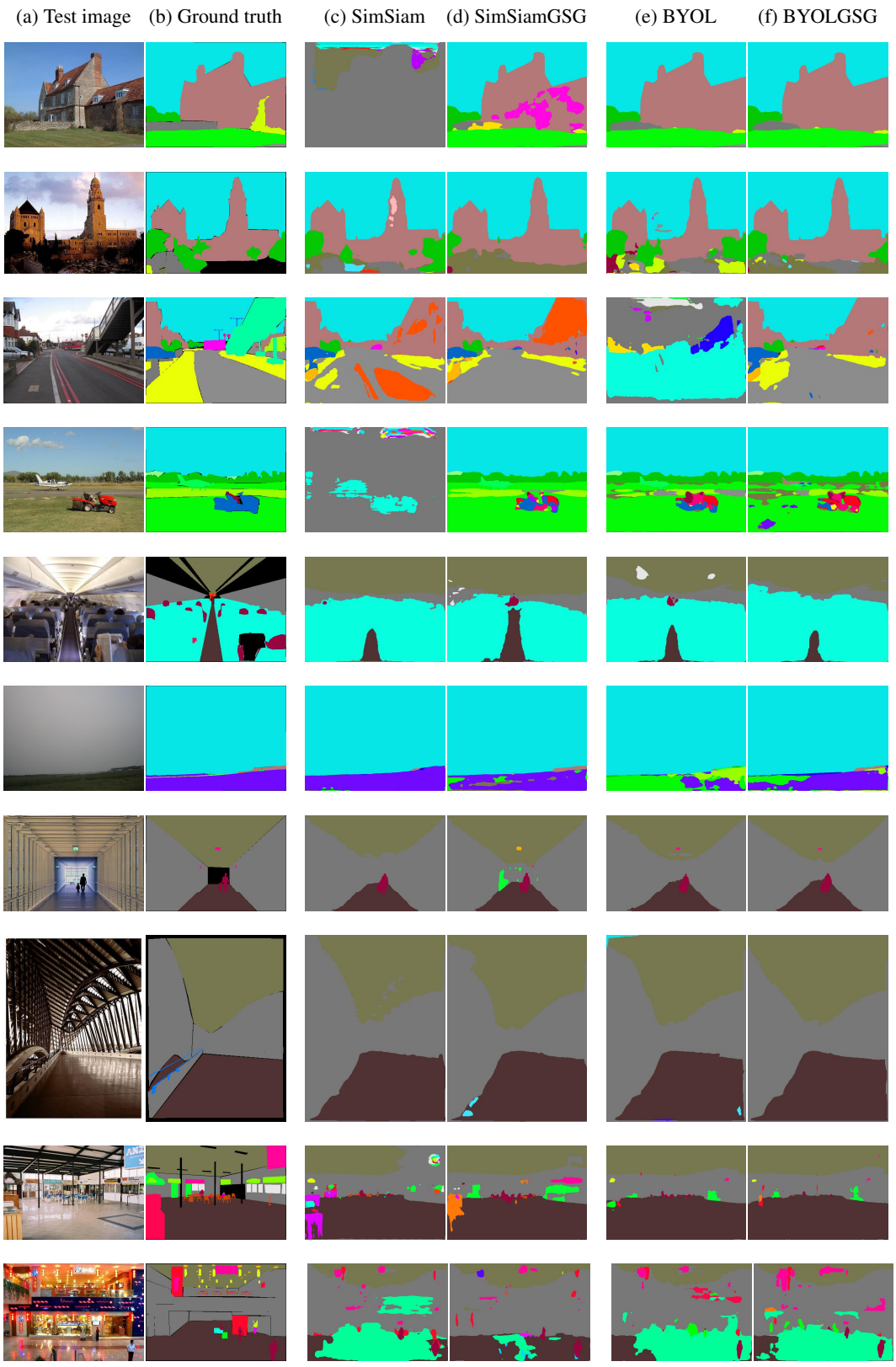


Figure 2: Semantic segmentation results of the first ten images from the validation set of ADE20K. The performance of our algorithms is similar or better than that of the original algorithms.

our ideas with these algorithms may further improve transfer learning (e.g., dense prediction tasks) performance on multi-object datasets. Since our methodology is an intermediate between contrastive and non-contrastive learning, it may help the community to better understand the differences and similarities between them [Garrido et al., 2022].

References

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*, 33:9912–9924, 2020.

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*, pages 9650–9660, 2021.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

Quentin Garrido, Yubei Chen, Adrien Bardes, Laurent Najman, and Yann Lecun. On the duality between contrastive and non-contrastive self-supervised learning. *arXiv preprint arXiv:2206.02574*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

Igor Susmelj, Matthias Heller, Philipp Wirth, Jeremy Prescott, and Malte Ebner et al. Lightly. *GitHub. Note: https://github.com/lightly-ai/lightly*, 2020.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Xinye Wanyan, Sachith Seneviratne, Shuchang Shen, and Michael Kirley. Dino-mc: Self-supervised contrastive learning for remote sensing imagery with multi-sized local crops. *arXiv preprint arXiv:2303.06670*, 2023.

Tong Zhang, Congpei Qiu, Wei Ke, Sabine Süsstrunk, and Mathieu Salzmann. Leverage your local and global representations: A new self-supervised learning strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16580–16589, 2022.