Improving Transferability of Representations via Augmentation-Aware Self-Supervision

Hankook Lee [1] **Kibok Lee** [2] **Kimin Lee** [3] **Honglak Lee** [4 5] **Jinwoo Shin** [1]

Abstract

Recent unsupervised representation learning methods have shown to be effective in a range of vision tasks by learning representations invariant to data augmentations such as random cropping and color jittering. However, such invariance could be harmful to downstream tasks if they rely on the characteristics of the data augmentations, e.g., location- or color-sensitive. To avoid such failures and obtain more generalizable representations, we suggest to optimize an auxiliary self-supervised loss, coined *AugSelf*, that learns the difference of augmentation parameters (e.g., cropping positions, color adjustment intensities) between two randomly augmented samples. Our intuition is that AugSelf encourages to preserve augmentation-aware information in learned representations, which could be beneficial for their transferability. Furthermore, AugSelf can easily be incorporated into recent state-of-the-art representation learning methods with a negligible additional training cost. Extensive experiments demonstrate that our simple idea consistently improves the transferability of representations learned by supervised and unsupervised methods in various transfer learning scenarios.

1. Introduction

In the vision domain, the recent self-supervised learning methods (He et al., 2020; Chen et al., 2020a; Caron et al., 2020; Grill et al., 2020; Chen & He, 2020) learn representations to be invariant to a pre-defined set of augmentations. The choice of the augmentations plays a crucial role in representation learning (Chen et al., 2020a;b; Tian et al., 2020; Xiao et al., 2021). A common choice is a combination of random cropping, horizontal flipping, color jittering, grayscaling, and Gaussian blurring. With this choice, learned representations are invariant to color and positional information in images; in other words, the representations lose such information.

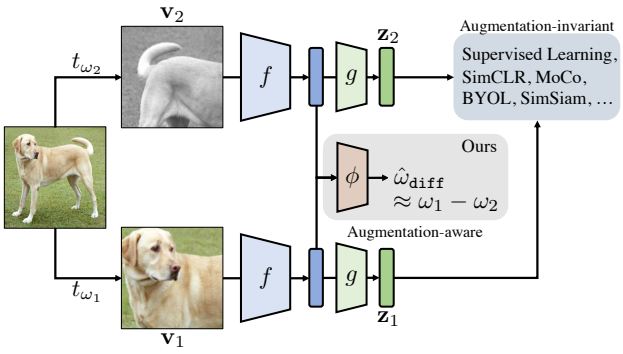


Figure 1. Illustration of the proposed method, AugSelf, that learns augmentation-aware information by predicting the difference between two augmentation parameters ω_1 and ω_2.

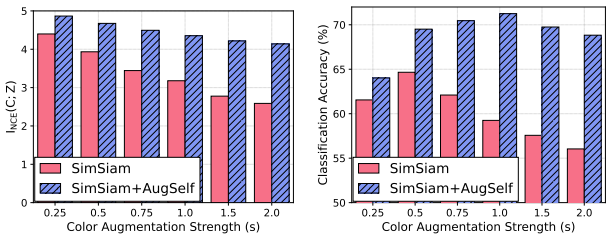
On the contrary, there have also been attempts to learn representations by designing pretext tasks that keep such information in augmentations, e.g., predicting positional relations between two patches of an image (Doersch et al., 2015), solving jigsaw puzzles (Noroozi & Favaro, 2016), or predicting color information from a gray image (Zhang et al., 2016). These results show the importance of augmentation-specific information for representation learning, and inspire us to explore the following research questions: *when is learning invariance to a given set of augmentations harmful to representation learning?* and, *how to prevent the loss in the recent unsupervised learning methods?*

Contribution. We first found that learning representations with an augmentation-invariant objective might hurt its performance in downstream tasks that rely on information related to the augmentations. For example, learning invariance against strong color augmentations forces the representations to contain less color information (see Figure 2a). Hence, it degrades the performance of the representations in color-sensitive downstream tasks such as the Flowers classification task (Nilsback & Zisserman, 2008) (see Figure 2b).

To prevent this information loss and obtain more generalizable representations, we propose an auxiliary self-

[1]KAIST [2]Amazon Web Services [3]University of California, Berkeley [4]University of Michigan [5]LG AI Research. Correspondence to: Jinwoo Shin <jinwoos@kaist.ac.kr>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.



(a) Mutual information (b) STL10→Flowers

Figure 2. (a) Changes of mutual information, i.e., $I_{\texttt{NCE}}(C;\mathbf{z})$, between color information $C(\mathbf{x})$ and the representation $\mathbf{z}=f(\mathbf{x})$ pretrained on STL10 (Coates et al., 2011) with varying the color jittering strength s. (b) The pretrained representations are evaluated in a color-sensitive benchmarks, Flowers (Nilsback & Zisserman, 2008), by linear evaluation (Kornblith et al., 2019).

supervised loss, coined AugSelf, that learns the difference of augmentation parameters between the two augmented samples (or views) as shown in Figure 1. For example, in the case of random cropping, AugSelf learns to predict the difference of cropping positions of two randomly cropped views. We found that AugSelf encourages the self-supervised representation learning methods, such as SimCLR (Chen et al., 2020a) and SimSiam (Chen & He, 2020), to preserve augmentation-aware information (see Figure 2a) that could be useful for downstream tasks. Furthermore, AugSelf can easily be incorporated into the recent unsupervised representation learning methods (He et al., 2020; Chen et al., 2020a; Grill et al., 2020; Chen & He, 2020) with a negligible additional training cost, which is for training an auxiliary prediction head ϕ in Figure 1.

We demonstrate the effectiveness of AugSelf under extensive transfer learning experiments. For example, AugSelf improves two self-supervised learning methods, MoCo (He et al., 2020) and SimSiam (Chen & He, 2020), in 28 of 30 tested scenarios (see Table 1 and 2). Somewhat interestingly, we also found that AugSelf is effective in the supervised pretraining scenario.

We remark that learning augmentation-invariant representations has been a common practice for both supervised and unsupervised representation learning frameworks, while the importance of augmentation-awareness is less emphasized. We hope that our work could inspire researchers to rethink the under-explored aspect and provide a new angle in representation learning.

2. Motivation

We first remark that the recent self-supervised learning methods (He et al., 2020; Chen et al., 2020a; Caron et al., 2020; Grill et al., 2020; Chen & He, 2020) learn augmentation-invariant representations. Formally, let $\mathbf{x}$ be an image, t_ω be an augmentation function parameterized by an augmentation parameter ω, $\mathbf{v}=t_\omega(\mathbf{x})$ be the augmented sample (or view) of $\mathbf{x}$ by t_ω, and f be a CNN feature extractor, such as ResNet (He et al., 2016). Then, the methods encourage the representations $f(\mathbf{v}_1)$ and $f(\mathbf{v}_2)$ to be invariant to the two randomly augmented views $\mathbf{v}_1=t_{\omega_1}(\mathbf{x})$ and $\mathbf{v}_2=t_{\omega_2}(\mathbf{x})$, i.e., $f(\mathbf{v}_1)\approx f(\mathbf{v}_2)$ for $\omega_1,\omega_2\sim\Omega$ where Ω is a pre-defined augmentation parameter distribution. For example, SimSiam (Chen & He, 2020) minimizes $\|h(f(\mathbf{v}_1))-\texttt{sg}(f(\mathbf{v}_2))\|_2^2$ where h is a 2-layer MLP and $\texttt{sg}$ is the stop-gradient operation. This objective can be reformulated as $\text{Var}_{\omega\sim\Omega}(f(\mathbf{v}))$ when h is optimal; hence, it learns augmentation-invariant representations. We describe more details of the recent methods in Appendix A.

Consequently, these methods encourage representations $f(\mathbf{x})$ to contain shared (i.e., augmentation-invariant) information between $t_{\omega_1}(\mathbf{x})$ and $t_{\omega_2}(\mathbf{x})$ and discard other information (Tian et al., 2020). For example, if t_ω changes color information, then to satisfy $f(t_{\omega_1}(\mathbf{x}))=f(t_{\omega_2}(\mathbf{x}))$ for any $\omega_1,\omega_2\sim\Omega$, $f(\mathbf{x})$ will be learned to contain no (or less) color information. To verify this, we pretrain ResNet-18 (He et al., 2016) on STL10 (Krizhevsky et al., 2009) using SimSiam (Chen & He, 2020) with varying the strength s of the color jittering augmentation. To measure the mutual information between representations and color information, we use the InfoNCE loss (Oord et al., 2018). We here simply encode color information as RGB color histograms of an image. As shown in Figure 2a, using stronger color augmentations leads to color-relevant information loss. In a color-sensitive classification task, Flowers (Nilsback & Zisserman, 2008), the learned representations containing less color information result in lower performance as shown in Figure 2b. This observation emphasizes the importance of learning augmentation-aware information in transfer learning scenarios.

3. Proposed Method

In this section, we introduce *auxiliary augmentation-aware self-supervision*, coined *AugSelf*, which encourages to preserve augmentation-aware information for generalizable representation learning. To be specific, we add an auxiliary self-supervision loss, which learns to predict the difference between augmentation parameters of two randomly augmented views, into existing augmentation-invariant representation learning methods (He et al., 2020; Chen et al., 2020a; Caron et al., 2020; Grill et al., 2020; Chen & He, 2020). For conciseness, let θ be the collection of all parameters in the model.

Since an augmentation function t_ω is typically a composition of different types of augmentations, the augmentation parameter ω can be written as $\omega=(\omega^{\texttt{aug}})_{\texttt{aug}\in\mathcal{A}}$ where $\mathcal{A}$ is the set of augmentations used in pretraining (e.g., $\mathcal{A}=\{\texttt{crop},\texttt{flip}\}$), and $\omega^{\texttt{aug}}$ is an augmentation-specific

Table 1. Linear evaluation accuracy (%) of ResNet-50 (He et al., 2016) pretrained on ImageNet100 (Russakovsky et al., 2015; Tian et al., 2019) in various downstream tasks.

Method	CIFAR10	CIFAR100	Food	MIT67	Pets	Flowers	Caltech101	Cars	Aircraft	DTD	SUN397
SimSiam	86.89	66.33	61.48	65.75	74.69	88.06	84.13	**48.20**	48.63	65.11	50.60
+ AugSelf (ours)	**88.80**	**70.27**	**65.63**	**67.76**	**76.34**	**90.70**	**85.30**	47.52	**49.76**	**67.29**	**52.28**
MoCo v2	84.60	61.60	59.37	61.64	70.08	82.43	77.25	33.86	**41.21**	64.47	46.50
+ AugSelf (ours)	**85.26**	**63.90**	**60.78**	**63.36**	**73.46**	**85.70**	**78.93**	**37.35**	39.47	**66.22**	**48.52**
Supervised	**86.16**	62.70	53.89	52.91	73.50	76.09	**77.53**	30.61	36.78	61.91	40.59
+ AugSelf (ours)	86.06	**63.77**	**55.84**	**54.63**	**74.81**	**78.22**	77.47	**31.26**	**38.02**	**62.07**	**41.49**

parameter (e.g., $\omega^{\texttt{crop}}$ decides how to crop an image). Then, given two randomly augmented views $\mathbf{v}_1 = t_{\omega_1}(\mathbf{x})$ and $\mathbf{v}_2 = t_{\omega_2}(\mathbf{x})$, the AugSelf objective is as follows:

$$\mathcal{L}_{\texttt{AugSelf}}(\mathbf{x}, \omega_1, \omega_2; \theta) = \sum\nolimits_{\texttt{aug} \in \mathcal{A}_{\texttt{AugSelf}}} \mathcal{L}_{\texttt{aug}}\big(\phi_\theta^{\texttt{aug}}(f_\theta(\mathbf{v}_1), f_\theta(\mathbf{v}_2)), \omega_{\texttt{diff}}^{\texttt{aug}}\big),$$

where $\mathcal{A}_{\texttt{AugSelf}} \subseteq \mathcal{A}$ is the set of augmentations for augmentation-aware learning, $\omega_{\texttt{diff}}^{\texttt{aug}}$ is the difference between two augmentation-specific parameters $\omega_1^{\texttt{aug}}$ and $\omega_2^{\texttt{aug}}$, $\mathcal{L}_{\texttt{aug}}$ is an augmentation-specific loss, and $\phi_\theta^{\texttt{aug}}$ is a 3-layer MLP for $\omega_{\texttt{diff}}^{\texttt{aug}}$ prediction. This design allows us to incorporate AugSelf into the recent state-of-the-art unsupervised learning methods (He et al., 2020; Chen et al., 2020a; Caron et al., 2020; Grill et al., 2020; Chen & He, 2020) with a negligible additional training cost. For example, the objective of SimSiam (Chen & He, 2020) with AugSelf can be written as follows:

$$\mathcal{L}_{\texttt{total}}(\mathbf{x}, \omega_1, \omega_2; \theta) = \mathcal{L}_{\texttt{SimSiam}}(\mathbf{x}, \omega_1, \omega_2; \theta) + \lambda \cdot \mathcal{L}_{\texttt{AugSelf}}(\mathbf{x}, \omega_1, \omega_2; \theta),$$

where λ is a hyperparameter for balancing losses.

In this paper, we mainly focus on the commonly-used augmentations in the recent unsupervised representation learning methods (He et al., 2020; Chen et al., 2020a; Caron et al., 2020; Grill et al., 2020; Chen & He, 2020): random cropping, random horizontal flipping, color jittering, and Gaussian blurring; however, we remark that different types of augmentations can be incorporated into AugSelf (see Section G). We describe the details of $\omega^{\texttt{aug}}$ and $\mathcal{L}_{\texttt{aug}}$ for each augmentation in Appendix B. Shortly, we use the ℓ_2 regression loss if $\omega^{\texttt{aug}}$ is continuous, otherwise the cross-entropy classification loss.

4. Experiments

Setup. We pretrain the standard ResNet-50 (He et al., 2016) on ImageNet100, a 100-category subset[1] of ImageNet (Russakovsky et al., 2015), for 500 epochs with recent self-supervised learning methods as baselines: a contrastive method, MoCo v2 (Chen et al., 2020b), and a non-contrastive method, SimSiam (Chen & He, 2020). For supervised pretraining, we pretrain ResNet-50 for 100 epochs. Other pretraining details are described in Appendix C.1. In this section, AugSelf predicts random cropping and color jittering parameters, i.e., $\mathcal{A}_{\texttt{AugSelf}} = \{\texttt{crop}, \texttt{color}\}$, with $\lambda = 0.5$. Other augmentations are tested in Appendix G.

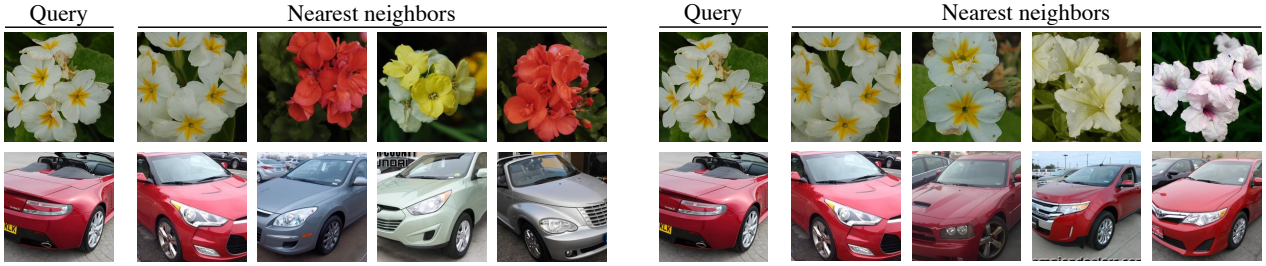
[1]We use the same split following Tian et al. (2019).

Table 2. Few-shot classification accuracy (%) with 95% confidence intervals averaged over 2000 episodes on CUB200 (Wah et al., 2011), and Plant Disease (Mohanty et al., 2016). (N, K) denotes N-way K-shot tasks.

	CUB200		Plant Disease	
Method	(5, 1)	(5, 5)	(5, 1)	(5, 5)
SimSiam	45.56±0.47	62.48±0.48	75.72±0.46	89.94±0.31
+ AugSelf (ours)	**48.08±0.47**	**66.27±0.46**	**77.93±0.46**	**91.52±0.29**
MoCo v2	41.67±0.47	56.92±0.47	65.73±0.49	84.98±0.36
+ AugSelf (ours)	**44.17±0.48**	**57.35±0.48**	**71.80±0.47**	**87.81±0.33**
Supervised	46.57±0.48	63.69±0.46	68.95±0.47	88.77±0.30
+ AugSelf (ours)	**47.58±0.48**	**65.31±0.45**	**70.82±0.46**	**89.77±0.29**

Linear evaluation in various downstream tasks. We evaluate the pretrained networks in downstream classification tasks on 11 datasets via the linear evaluation protocol (Kornblith et al., 2019). The detailed information of datasets and evaluation settings are Appendix D and F, respectively. As shown in Table 1, our AugSelf consistently improves (a) the recent self-supervised learning methods, SimSiam (Chen & He, 2020) and MoCo (Chen & He, 2020), and (b) supervised pretraining in almost all the downstream tasks. These consistent improvements imply that our method encourages to learn more generalizable representations.

Few-shot classification. We also evaluate the pretrained networks on various few-shot learning benchmarks: Caltech-UCSD Birds (Cubuk et al., 2020, CUB200) and Plant Disease (Mohanty et al., 2016). Note that the benchmarks require low-level features such as color information of birds and leaves, respectively, to detect their fine-grained labels. They are widely used in cross-domain few-shot settings (Chen et al., 2019; Guo et al., 2020). For few-shot learning, we perform logistic regression using the frozen representations $f(\mathbf{x})$ without fine-tuning. Table 2 shows the few-shot



(a) SimSiam (b) SimSiam + AugSelf

Figure 3. Top-4 nearest neighbors based on the cosine similarity using representations $f(\mathbf{x})$ learned by (a) SimSiam (Chen & He, 2020) or (b) SimSiam with AugSelf (ours).

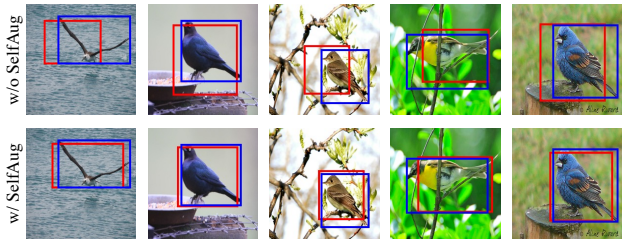


Figure 4. Examples of bounding box predictions on CUB200 (Cubuk et al., 2020). Blue and red boxes are ground-truth and model prediction, respectively.

Table 3. ℓ_2 errors of bounding box predictions on CUB200 (Wah et al., 2011) using representations pretrained by SimSiam (Chen & He, 2020), MoCo (He et al., 2020), and supervised learning without or with our AugSelf.

AugSelf	SimSiam	MoCo	Supervised
	0.00462	0.00487	0.00520
✓	**0.00335**	**0.00429**	**0.00473**

learning performance of 5-way 1-shot and 5-way 5-shot tasks. As shown in the table, our AugSelf improves the performance of SimSiam (Chen & He, 2020) and MoCo (He et al., 2020) in all cases with a large margin. For example, for the plant disease detection task (Mohanty et al., 2016), we obtain up to 6.07% accuracy gain in 5-way 1-shot tasks. These results show that our method is also effective in such transfer learning scenarios.

Retrieval. Figure 3 shows the retrieval results using pretrained models. For this experiment, we use the Flowers (Nilsback & Zisserman, 2008) and Cars (Krause et al., 2013) datasets and find top-4 nearest neighbors based on the cosine similarity between representations $f(\mathbf{x})$ where f is the pretrained ResNet-50 on ImageNet100. As shown in the figure, the representations learned by AugSelf are more color-sensitive.

Object localization. We also evaluate representations in an object localization task (i.e., bounding box prediction) that requires positional information. We experiment on CUB200 (Van Horn et al., 2015) and solve linear regression using representations pretrained by various pretraining methods without or with our method. Figure 4 shows the examples of the predictions and Table 3 reports the ℓ_2 errors of bounding box predictions. These results demonstrate that AugSelf is capable of learning positional information.

Comparison with LooC. Recently, Xiao et al. (2021) propose LooC that learns augmentation-aware representations via multiple augmentation-specific contrastive learning objectives. Our AugSelf has two advantages over LooC. First, AugSelf is more efficient since it requires the same number of augmented samples compared to the baseline self-supervised learning methods while LooC requires more. Second, AugSelf can be incorporated with non-contrastive methods, e.g., SimSiam (Chen & He, 2020); we found that SimSiam with AugSelf outperforms LooC in all cases. We provide detailed head-to-head comparisons in Appendix E.

Ablation study. We provide extensive ablation studies in Appendix G. In the section, we demonstrate the effect of each augmentation prediction task, compatibility with different augmentations, and compatibility with other self-supervised learning methods such as SwAV (Caron et al., 2020). We also validate AugSelf's capability to learn augmentation-aware information by solving augmentation-related pretext tasks (e.g., prediction of a color-channel permutation) using pretrained representations.

5. Conclusion

To improve the transferability of representations, we propose AugSelf, an auxiliary augmentation-aware self-supervision method that encourages representations to contain augmentation-aware information that could be useful in downstream tasks. Our idea is to learn to predict the difference between augmentation parameters of two randomly augmented views. Through extensive experiments, we demonstrate the effectiveness of AugSelf in various transfer learning scenarios. We believe that our work would guide many research directions for unsupervised representation learning and transfer learning.

References

Asano, Y. M., Rupprecht, C., and Vedaldi, A. Self-labelling via simultaneous clustering and representation learning. In *International Conference on Learning Representations*, 2020.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101 – mining discriminative components with random forests. In *Proceedings of the European Conference on Computer Vision*, 2014.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision*, pp. 132–149, 2018.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *Proceedings of International Conference on Machine Learning*, pp. 1597–1607. PMLR, 2020a.

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F., and Huang, J.-B. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019.

Chen, X. and He, K. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

Chen, X., Fan, H., Girshick, R., and He, K. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3606–3613, 2014.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. RandAugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703, 2020.

Doersch, C., Gupta, A., and Efros, A. A. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430, 2015.

Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *Conference on Computer Vision and Pattern Recognition Workshop*, pp. 178–178. IEEE, 2004.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., et al. Bootstrap Your Own Latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Guo, Y., Codella, N. C., Karlinsky, L., Codella, J. V., Smith, J. R., Saenko, K., Rosing, T., and Feris, R. A broader study of cross-domain few-shot learning. In *Proceedings of the European Conference on Computer Vision*, 2020.

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.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of International Conference on Machine Learning*, pp. 448–456. PMLR, 2015.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2661–2671, 2019.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

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

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Mohanty, S. P., Hughes, D. P., and Salathé, M. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

Nilsback, M.-E. and Zisserman, A. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, pp. 722–729. IEEE, 2008.

Noroozi, M. and Favaro, P. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European Conference on Computer Vision*, pp. 69–84. Springer, 2016.

Oord, A. v. d., Li, Y., and Vinyals, O. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Oreshkin, B., Rodríguez López, P., and Lacoste, A. Tadam: Task dependent adaptive metric for improved few-shot learning. In *Advances in Neural Information Processing Systems*, pp. 721–731, 2018.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and dogs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505. IEEE, 2012.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8026–8037, 2019.

Quattoni, A. and Torralba, A. Recognizing indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 413–420. IEEE, 2009.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 806–813, 2014.

Tian, Y., Krishnan, D., and Isola, P. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning? In *Advances in Neural Information Processing Systems*, pp. 6827–6839, 2020.

Van Horn, G., Branson, S., Farrell, R., Haber, S., Barry, J., Ipeirotis, P., Perona, P., and Belongie, S. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 595–604, 2015.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. SUN database: Large-scale scene recognition from abbey to zoo. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 3485–3492. IEEE, 2010.

Xiao, T., Wang, X., Efros, A. A., and Darrell, T. What should not be contrastive in contrastive learning. In *International Conference on Learning Representations*, 2021.

Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. In *Proceedings of the European Conference on Computer Vision*, pp. 649–666. Springer, 2016.

A. Augmentation-invariant representation learning

In this section, we describe how do recent self-supervised learning methods (He et al., 2020; Chen et al., 2020a; Caron et al., 2020; Grill et al., 2020; Chen & He, 2020) learn representations that are invariant to augmentations. Formally, let $\mathbf{x}$ be an image, t_ω be an augmentation function parameterized by an augmentation parameter ω, $\mathbf{v} = t_\omega(\mathbf{x})$ be the augmented sample (or view) of $\mathbf{x}$ by t_ω, and f be a CNN feature extractor, such as ResNet (He et al., 2016). Generally speaking, the methods encourage the representations $f(\mathbf{v}_1)$ and $f(\mathbf{v}_2)$ to be invariant to the two randomly augmented views $\mathbf{v}_1 = t_{\omega_1}(\mathbf{x})$ and $\mathbf{v}_2 = t_{\omega_2}(\mathbf{x})$, i.e., $f(\mathbf{v}_1) \approx f(\mathbf{v}_2)$ for $\omega_1, \omega_2 \sim \Omega$ where Ω is a pre-defined augmentation parameter distribution.

Instance contrastive learning approaches (He et al., 2020; Chen et al., 2020a; Wu et al., 2018) minimize the distance between an anchor $f(t_{\omega_1}(\mathbf{x}))$ and its positive sample $f(t_{\omega_2}(\mathbf{x}))$, while maximizing the distance between the anchor $f(t_{\omega_1}(\mathbf{x}))$ and its negative sample $f(t_{\omega_3}(\mathbf{x}'))$. Since contrastive learning performance depends on the number of negative samples, a memory bank (Wu et al., 2018), a large batch (Chen et al., 2020a), or a momentum network with a representation queue (He et al., 2020) has been utilized.

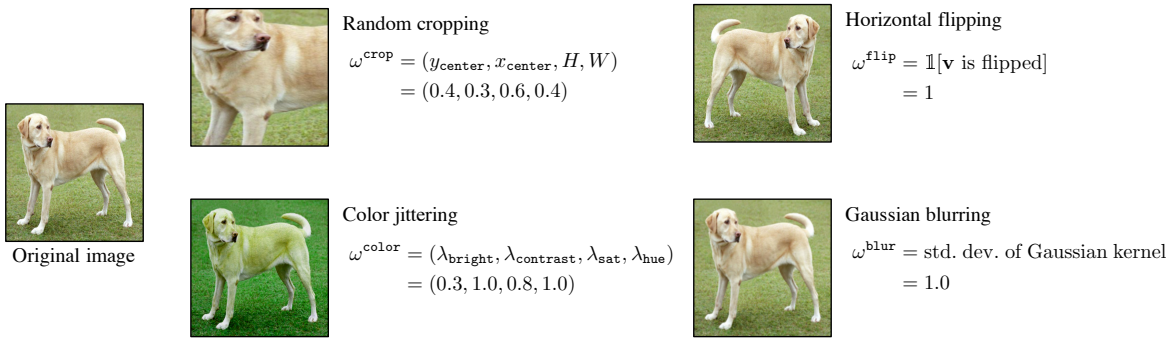


Figure 5. Examples of the commonly-used augmentations and their parameters $\omega^{\texttt{aug}}$.

Clustering approaches (Caron et al., 2020; 2018; Asano et al., 2020) encourage two representations $f(t_{\omega_1}(\mathbf{x}))$ and $f(t_{\omega_2}(\mathbf{x}))$ to be assigned into the same cluster, in other words, the distance between them will be minimized.

Negative-free methods (Grill et al., 2020; Chen & He, 2020) learn to predict the representation $f(\mathbf{v}_1)$ of a view $\mathbf{v}_1 = t_{\omega_1}(\mathbf{x})$ from another view $\mathbf{v}_2 = t_{\omega_2}(\mathbf{x})$. For example, SimSiam (Chen & He, 2020) minimizes $\|h(f(\mathbf{v}_2)) - \texttt{sg}(f(\mathbf{v}_1))\|_2^2$ where h is an MLP and $\texttt{sg}$ is the stop-gradient operation. In these methods, if h optimal, then $h(f(\mathbf{v}_2)) = \mathbb{E}_{\omega_1 \sim \Omega}[f(\mathbf{v_1})]$; thus, the expectation of the objective can be rewritten as $\text{Var}_{\omega \sim \Omega}(f(\mathbf{v}))$. Therefore, the methods can be considered as learning invariance with respect to the augmentations.

Supervised learning approaches (Sharif Razavian et al., 2014) with a huge labeled dataset are frequently used for transfer learning. They learn augmentation-invariant representations since they maximize $\frac{\exp(\mathbf{c}_y^\top f(t(\mathbf{x})))}{\sum_{y'} \exp(\mathbf{c}_{y'}^\top f(t(\mathbf{x})))}$ where $\mathbf{c}_y$ is the prototype vector of the label y; hence, $f(t(\mathbf{x}))$ is concentrated to $\mathbf{c}_y$, i.e., $\mathbf{c}_y \approx f(t_{\omega_1}(\mathbf{x})) \approx f(t_{\omega_2}(\mathbf{x}))$.

B. Specific forms of AugSelf

In this section, we elaborate on the details of $\omega^{\texttt{aug}}$ and $\mathcal{L}_{\texttt{aug}}$ for each augmentation. The examples of $\omega^{\texttt{aug}}$ are illustrated in Figure 5.

Random cropping. The random cropping is the most popular augmentation in vision tasks. A cropping parameter $\omega^{\texttt{crop}}$ contains the center position and cropping size. We normalize the values by the height and width of the original image $\mathbf{x}$, i.e., $\omega^{\texttt{crop}} \in [0, 1]^4$. Then, we use ℓ_2 loss for $\mathcal{L}_{\texttt{crop}}$ and set $\omega_{\texttt{diff}}^{\texttt{crop}} = \omega_1^{\texttt{crop}} - \omega_2^{\texttt{crop}}$.

Random horizontal flipping. A flipping parameter $\omega^{\texttt{flip}} \in \{0, 1\}$ indicates the image is horizontally flipped or not. Since it is discrete, we use the binary cross-entropy loss for $\mathcal{L}_{\texttt{flip}}$ and set $\omega_{\texttt{diff}}^{\texttt{flip}} = \mathbb{1}[\omega_1^{\texttt{flip}} = \omega_2^{\texttt{flip}}]$.

Color jittering. The color jittering augmentation adjusts brightness, contrast, saturation, and hue of an input image in a random order. For each adjustment, its intensity is uniformly sampled from a pre-defined interval. We normalize all intensities into $[0, 1]$, i.e., $\omega^{\texttt{color}} \in [0, 1]^4$. Similarly to cropping, we use ℓ_2 loss for $\mathcal{L}_{\texttt{color}}$ and set $\omega_{\texttt{diff}}^{\texttt{color}} = \omega_1^{\texttt{color}} - \omega_2^{\texttt{color}}$.

Gaussian blurring. This blurring operation is widely used in unsupervised representation learning. The Gaussian filter is constructed by a single parameter, standard deviation $\sigma = \omega^{\texttt{blur}}$. We also normalize the parameter into $[0, 1]$. Then, we use ℓ_2 loss for $\mathcal{L}_{\texttt{blur}}$ and set $\omega_{\texttt{diff}}^{\texttt{blur}} = \omega_1^{\texttt{blur}} - \omega_2^{\texttt{blur}}$.

C. Pretraining setup

C.1. ImageNet100 pretraining

We pretrain the standard ResNet-50 (He et al., 2016) architecture in the ImageNet100[2] (Russakovsky et al., 2015; Tian et al., 2019) dataset for 500 training epochs using SimSiam (Chen & He, 2020) and MoCo (He et al., 2020) methods. We use a batch size of 256 and a cosine learning rate schedule without restarts (Loshchilov & Hutter, 2017). Note that the pretraining setups are the same as they officially used for ImageNet pretraining described in (Chen & He, 2020; He et al., 2020; Chen et al., 2020b). In multi-GPU experiments, we use the synchronized batch normalization following Chen & He (2020). When incorporating our AugSelf into the methods, we use $\lambda = 0.5$ and $\mathcal{A}_{\texttt{AugSelf}} = \{\texttt{crop}, \texttt{color}\}$. Note that SimSiam (Chen & He, 2020) and MoCo (He et al., 2020) requires 32 and 29 hours on a single RTX3090 4-GPU machine, respectively.

SimSiam (Chen & He, 2020). We use a learning rate 0.05 and a weight decay of 0.0001. We use a 3-layer projection MLP head $g(\cdot)$ with a hidden dimension of 2048 and an output dimension of 2048. We use a batch normalization (Ioffe & Szegedy, 2015) at the last layer in the projection

[2]ImageNet100 is a 100-category subset of ImageNet (Russakovsky et al., 2015). We use the same split following Tian et al. (2019).

MLP. We use a 2-layer prediction MLP head $h(\cdot)$ with a hidden dimension of 512 and no batch normalization at the last layer in the prediction MLP. When optimizing the prediction MLP, we use a constant learning rate schedule following Chen & He (2020).

MoCo (He et al., 2020). We use a learning rate 0.03 and a weight decay of 0.0001. Following an advanced version of MoCo (Chen et al., 2020b, MoCo v2), we use a 2-layer projection MLP head $g(\cdot)$ with a hidden dimension of 2048 and an output dimension of 128. We use a batch normalization (Ioffe & Szegedy, 2015) at only the hidden layer. We also use a temperature scaling parameter of 0.2, an exponential moving average parameter of 0.999, and a queue size of 65536.

C.2. STL10 pretraining

We pretrain the standard ResNet-18 (He et al., 2016) architecture in the STL10 (Coates et al., 2011) dataset. We use the STL10-pretrained models for ablation studies in Appendix G. For all methods, we use the same optimization scheme: stochastic gradient descent (SGD) with a learning rate of 0.03, a batch size of 256, a weight decay of 0.0005, a momentum of 0.9. The learning rate follows a cosine decay schedule without restarts (Loshchilov & Hutter, 2017). When incorporating our AugSelf into the methods, we use $\lambda = 1.0$ and $\mathcal{A}_{\texttt{AugSelf}} = \{\texttt{crop}, \texttt{color}\}$, unless otherwise stated. We now describe method-specific hyperparameters one by one in the following.

SimCLR (Chen et al., 2020a). We use a 2-layer projection MLP head $g(\cdot)$ with a hidden dimension of 512 and an output dimension of 128. We do not use a batch normalization (Ioffe & Szegedy, 2015) at the last layer in the MLP. We use a temperature scaling parameter of 0.2 in contrastive learning.

MoCo (He et al., 2020). We use an advanced version of MoCo (Chen et al., 2020b, MoCo v2) with the same projection MLP architecture as SimCLR used. Other hyperparameters are the same as the ImageNet100 setup described in Section C.1.

BYOL (Grill et al., 2020). Following Grill et al. (2020), we use a 2-layer projection MLP head $g(\cdot)$ with a hidden dimension of 4096 and an output dimension of 256. We do not use a batch normalization (Ioffe & Szegedy, 2015) at the last layer in the MLP. We use the same architecture for the prediction MLP head $h(\cdot)$. The exponential moving average parameter is increased starting from 0.996 to 1.0 with a cosine schedule following Grill et al. (2020).

SimSiam (Chen & He, 2020). We use a 2-layer projection MLP with a hidden dimension of 2048 and an output dimension of 2048. Other hyperparameters are the same as the ImageNet100 setup described in Section C.1.

SWAV (Caron et al., 2020). We use a 2-layer projection MLP head $g(\cdot)$ with a hidden dimension of 2048 and an output dimension of 128 without batch normalization (Ioffe & Szegedy, 2015) at the last layer in the MLP. We use 100 prototypes, a smoothness factor of $\epsilon = 0.05$ and a temperature scaling parameter of $\tau = 0.1$. We do not use the multi-resolution cropping technique and the additional queue storing previous batches for simplicity.

C.3. Augmentations

In this section, we describe augmentations using PyTorch (Paszke et al., 2019) notations in the following. Note that we use random cropping, flipping, color jittering, grayscale, and Gaussian blurring in all the experiments except Table 7.

- **`RandomResizedCrop`**. The scale of cropping is randomly sampled from $[0.2, 1.0]$. The cropped images are resized to 96×96 and 224×224 for STL10 (Coates et al., 2011) and ImageNet (Russakovsky et al., 2015) pretraining, respectively.
- **`RandomHorizontalFlip`**. This operation is randomly applied with a probability of 0.5.
- **`ColorJitter`**. The maximum strengths of brightness, contrast, saturation and hue factors are 0.4, 0.4, 0.4 and 0.1, respectively. This operation is randomly applied with a probability of 0.8. In Section 2, we adjust the strengths by multiplying a strength factor s, i.e., $s > 1$ means stronger color jittering than the default configuration while $s < 1$ means weaker color jittering.
- **`RandomGrayscale`**. This operation is randomly applied with a probability of 0.2.
- **`GaussianBlur`**. The standard deviation is randomly sampled from $[0.1, 2.0]$. The kernel size is 9×9 and 23×23 for STL10 (Coates et al., 2011) and ImageNet (Russakovsky et al., 2015) pretraining, respectively. This operation is randomly applied with a probability of 0.5.
- **`Rotation`**. This rotates an image by $0°$, $90°$, $180°$, $270°$ randomly. This operation is randomly applied with a probability of 0.5 after the default geometric augmentations are applied.
- **`Solarization`**. This inverts each pixel value when the value is larger than a randomly sampled threshold, i.e., δ is sampled from $[0, 1]$ and then $x_{\texttt{new}}^{(i,j)} = \mathbb{1}[x_{\texttt{old}}^{(i,j)} < \delta] x_{\texttt{old}}^{(i,j)} + \mathbb{1}[x_{\texttt{old}}^{(i,j)} \geq \delta](1 - x_{\texttt{old}}^{(i,j)})$ for all pixels (i, j). This operation is randomly applied with a probability of 0.5 right after `ColorJitter` is applied.

Table 4. Dataset information.

Category	Dataset	# of classes	Training	Validation	Test	Metric
(a) Pretraining	STL10 (Coates et al., 2011)	10	105000	-	-	-
	ImageNet100 (Russakovsky et al., 2015; Tian et al., 2019)	1000	126689	-	-	-
(b) Linear evaluation	STL10 (Coates et al., 2011)	10	4500	500	8000	Top-1 accuracy
	CIFAR10 (Krizhevsky et al., 2009)	10	45000	5000	10000	Top-1 accuracy
	CIFAR100 (Krizhevsky et al., 2009)	100	45000	5000	10000	Top-1 accuracy
	Food (Bossard et al., 2014)	101	68175	7575	25250	Top-1 accuracy
	MIT67 (Quattoni & Torralba, 2009)	67	4690	670	1340	Top-1 accuracy
	Pets (Parkhi et al., 2012)	37	2940	740	3669	Mean per-class accuracy
	Flowers (Nilsback & Zisserman, 2008)	102	1020	1020	6149	Mean per-class accuracy
	Caltech101 (Fei-Fei et al., 2004)	101	2525	505	5647	Mean Per-class accuracy
	Cars (Krause et al., 2013)	196	6494	1650	8041	Top-1 accuracy
	Aircraft (Maji et al., 2013)	100	3334	3333	3333	Mean Per-class accuracy
	DTD (split 1) (Cimpoi et al., 2014)	47	1880	1880	1880	Top-1 accuracy
	SUN397 (split 1) (Xiao et al., 2010)	397	15880	3970	19850	Top-1 accuracy
(c) Few-shot	FC100 (Oreshkin et al., 2018)	20	-	-	12000	Average accuracy
	CUB200 (Wah et al., 2011)	200	-	-	11780	Average accuracy
	Plant Disease (Mohanty et al., 2016)	38	-	-	54305	Average accuracy

Table 5. Linear evaluation accuracy (%) under the same setup following Xiao et al. (2021). The augmentations in the brackets of LooC (Xiao et al., 2021) indicate which augmentation-aware information is learned. N is the number of required augmented samples for each instance, that reflects the effective training batch size. * indicates that the numbers are reported in (Xiao et al., 2021). The numbers in the brackets show the accuracy gains compared to each baseline.

Method	N	ImageNet100	CUB200	Flowers (5-shot)	Flowers (10-shot)
MoCo* (He et al., 2020)	2	81.0	36.7	67.9±0.5	77.3±0.1
LooC* (Xiao et al., 2021) (color)	3	81.1 (+0.1)	40.1 (+3.4)	68.2±0.6 (+0.3)	77.6±0.1 (+0.3)
LooC* (Xiao et al., 2021) (rotation)	3	80.2 (-0.8)	38.8 (+2.1)	70.1±0.4 (+2.2)	79.3±0.1 (+2.0)
LooC* (Xiao et al., 2021) (color, rotation)	4	79.2 (-1.8)	39.6 (+2.9)	70.9±0.3 (+3.0)	80.8±0.2 (+3.5)
MoCo (He et al., 2020)	2	81.0	32.2	78.5±0.3	81.2±0.3
MoCo (He et al., 2020) + AugSelf (ours)	2	82.4 (+1.4)	37.0 (+4.8)	81.7±0.2 (+3.2)	84.5±0.2 (+3.3)
SimSiam (Chen & He, 2020)	2	81.6	38.4	83.6±0.3	85.9±0.2
SimSiam (Chen & He, 2020) + AugSelf (ours)	2	**82.6** (+1.0)	**45.3** (+6.9)	**86.4±0.2** (+2.8)	**88.3±0.1** (+2.4)

D. Datasets

Table 4 summarizes detailed descriptions of (a) pretraining datasets, (c) linear evaluation benchmarks, and (c) few-shot learning benchmarks. For linear evaluation benchmarks, we randomly choose validation samples in the training split for each dataset when the validation split is not officially provided. For few-shot benchmarks, we use the meta-test split for FC100 (Oreshkin et al., 2018), and whole datasets for CUB200 (Wah et al., 2011) and Plant Disease (Mohanty et al., 2016). The evaluation details are described in Section F.

E. Additional experiments

Table 5 shows head-to-head comparisons between our AugSelf and LooC (Xiao et al., 2021) that learns augmentation-aware representations via multiple augmentation-specific contrastive learning objectives.

F. Evaluation protocol

Linear evaluation benchmarks. We follow the same linear transfer evaluation protocol (Chen et al., 2020a; Grill et al., 2020; Kornblith et al., 2019); we train linear classifiers upon the frozen features extracted from 224×224 (or 96×96 for STL10 pretraining) center-cropped images without data augmentation. To be specific, images are first resized to 224 pixels along the shorter side, and then cropped by 224×224 at the center of the images. Then, we minimize the ℓ_2-regularized cross-entropy objective using L-BFGS. The regularization parameter is selected from a range of 45 logarithmically spaced values from 10^{-6} to 10^5 using the validation split. After selecting the best hyperparameter, we train again the linear classifier using both training and validation splits and then report the test accuracy using the model. Note that we set the maximum number of iterations in L-BFGS as 5000 and use the previous solution as an initial point (i.e., warm start) for the next step.

Few-shot benchmarks. For evaluating representations in

Table 6. Linear evaluation accuracy (%) of ResNet-18 (He et al., 2016) pretrained by each augmentation prediction task without other methods such as SimSiam (Chen & He, 2020). We report SimSiam (Chen & He, 2020) results as reference. **Bold entries** are larger than the random initialization.

Pretraining objective	STL10	CIFAR10	CIFAR100	Food	MIT67	Pets	Flowers
Random Init	42.72	47.45	23.73	11.54	12.29	12.94	26.06
$\mathcal{L}_{\texttt{crop}}$	**68.28**	**70.78**	**43.44**	**22.26**	**26.17**	**27.68**	**38.21**
$\mathcal{L}_{\texttt{flip}}$	**46.45**	**53.80**	**24.89**	9.69	11.99	10.71	13.04
$\mathcal{L}_{\texttt{color}}$	**61.14**	**63.39**	**40.38**	**28.02**	**25.35**	**24.49**	**54.42**
$\mathcal{L}_{\texttt{blur}}$	**48.26**	46.60	20.44	8.73	11.87	**13.07**	17.20
SimSiam (Chen & He, 2020)	**85.19**	**82.35**	**54.90**	**33.99**	**39.15**	**44.90**	**59.19**

Table 7. Linear evaluation accuracy (%) of ResNet-18 (He et al., 2016) pretrained by SimSiam (Chen & He, 2020) with various combinations of our augmentation prediction tasks. **Bold entries** are the best of each task.

$\mathcal{A}_{\texttt{AugSelf}}$	STL10	CIFAR10	CIFAR100	Food	MIT67	Pets	Flowers
$\emptyset$	85.19	82.35	54.90	33.99	39.15	44.90	59.19
$\{\texttt{crop}\}$	**85.98**	82.82	55.78	35.68	43.21	47.10	62.05
$\{\texttt{color}\}$	85.55	**82.90**	58.11	40.32	43.56	47.85	71.08
$\{\texttt{crop}, \texttt{color}\}$	85.70	82.76	**58.65**	**41.58**	**45.67**	**48.42**	**72.18**

few-shot benchmarks, we simply perform logistic regression[3] using the frozen representations $f(\mathbf{x})$ and $N \times K$ support samples without fine-tuning and data augmentation in a N-way K-shot episode.

Object localization. For predicting bounding box information (i.e., top left coordinates and sizes of bounding boxes), we simply perform linear regression[3] using the frozen representations $f(\mathbf{x})$ and all training samples in CUB200 (Wah et al., 2011) without fine-tuning and data augmentation.

G. Ablation study

For ablation studies, we use STL10-pretrained ResNet-18 models. The pretraining details are described in Appendix C.2.

Effect of augmentation prediction tasks. We first evaluate the proposed augmentation prediction tasks one by one without incorporating invariance-learning methods. More specifically, we pretrain f_θ using only $\mathcal{L}_{\texttt{aug}}$ for each $\texttt{aug} \in \{\texttt{crop}, \texttt{flip}, \texttt{color}, \texttt{blur}\}$. Remark that training objectives are different but we use the same set of augmentations. Table 6 shows the transfer learning results in various downstream tasks. We observe that solving horizontal flipping and Gaussian blurring prediction tasks results in worse or similar performance to a random initialized network in various downstream tasks, i.e., the augmentations do not contain task-relevant information. However, solving random cropping and color jittering prediction tasks significantly outperforms the random initialization in all downstream tasks. Furthermore, surprisingly, the color jittering prediction task achieves competitive performance in the Flowers (Nilsback & Zisserman, 2008) dataset compared to a recent state-of-the-art method, SimSiam (Chen & He, 2020).These results show that augmentation-aware information are task-relevant and learning such information could be important in downstream tasks.

Based on the above observations, we incorporate random cropping and color jittering prediction tasks into SimSiam (Chen & He, 2020) when pretraining. More specifically, we optimize $\mathcal{L}_{\texttt{SimSiam}} + \lambda_{\texttt{crop}}\mathcal{L}_{\texttt{crop}} + \lambda_{\texttt{color}}\mathcal{L}_{\texttt{color}}$ where $\lambda_{\texttt{crop}}, \lambda_{\texttt{color}} \in \{0, 1\}$. The transfer learning results are reported in Table 7. As shown in the table, each self-supervision task improves SimSiam consistently (and often significantly) in various downstream tasks. For example, the color jittering prediction task improves SimSiam by 6.33% and 11.89% in Food (Bossard et al., 2014) and Flowers (Nilsback & Zisserman, 2008) benchmarks, respectively. When incorporating both tasks simultaneously, we achieve further improvements in almost all the downstream tasks. Furthermore, as shown in Figure 2, our AugSelf preserves augmentation-aware information as much as possible; hence our gain is consistent regardless of the strength of color jittering augmentation.

Different augmentations. We confirm that our method can allow to use other strong augmentations: rotation, which rotates an image by $0°$, $90°$, $180°$, $270°$ degrees randomly; and solarization, which inverts each pixel value when the value is larger than a randomly sampled threshold. Based on the default augmentation setting, i.e., $\mathcal{A}_{\texttt{AugSelf}} =$

[3]We use the scikit-learn `LogisticRegression` and `LinearRegression` modules for logistic regression and linear regression, respectively.

Table 8. Transfer learning accuracy (%) of ResNet-18 (He et al., 2016) pretrained by SimSiam (Chen & He, 2020) with or without our AugSelf using strong augmentations. `C`, `J`, `R` and `S` denote cropping, color jittering, rotation and solarization prediction tasks, respectively. **Bold entries** are the best of each augmentation.

Strong Aug.	$\mathcal{A}_{\texttt{AugSelf}}$	STL10	CIFAR10	CIFAR100	Food	MIT67	Pets	Flowers
None	$\emptyset$	85.19	82.35	54.90	33.99	39.15	44.90	59.19
	{C, J}	**85.70**	**82.76**	**58.65**	**41.58**	**45.67**	**48.42**	**72.18**
Rotation	$\emptyset$	80.11	77.78	50.24	36.40	36.39	41.43	61.77
	{C, J}	81.85	79.93	57.27	43.04	41.32	**47.30**	72.52
	{C, J, R}	**82.67**	**80.71**	**58.28**	**43.28**	**44.48**	46.65	**72.94**
Solarization	$\emptyset$	**86.32**	81.08	52.50	32.59	41.29	44.76	58.79
	{C, J}	86.03	**82.64**	57.94	**40.29**	**46.67**	48.81	**71.43**
	{C, J, S}	85.91	82.63	**58.18**	40.17	45.57	**49.02**	**71.43**

Table 9. Transfer learning accuracy (%) of various unsupervised learning frameworks with and without our AugSelf framework. **Bold entries** indicates the best for each baseline method.

Method	AugSelf (ours)	STL10	CIFAR10	CIFAR100	Food	MIT67	Pets	Flowers
SimCLR (Chen et al., 2020a)		84.87	78.93	48.94	31.97	36.82	43.18	56.20
	✓	**84.99**	**80.92**	**53.64**	**36.21**	**40.62**	**46.51**	**64.31**
BYOL (Grill et al., 2020)		86.73	82.66	55.94	37.30	42.78	50.21	66.89
	✓	**86.79**	**83.60**	**59.66**	**42.89**	**46.17**	**52.45**	**74.07**
SWAV (Caron et al., 2020)		82.21	81.60	52.00	29.78	36.69	37.68	53.01
	✓	**82.57**	**82.00**	**55.10**	**33.16**	**39.13**	**40.74**	**61.69**

$\{\texttt{crop}, \texttt{color}\}$, we additionally apply each augmentation with a probability of 0.5. We also evaluate the effectiveness of augmentation prediction tasks for rotation and solarization. Note that we formulate the rotation prediction as a 4-way classification task (i.e., $\omega^{\texttt{rot}}_{\texttt{diff}} \in \{0, 1, 2, 3\}$) and the solarization prediction as a regression task (i.e., $\omega^{\texttt{sol}}_{\texttt{diff}} \in [-1, 1]$). As shown in Table 8, we obtain consistent gains across various downstream tasks even if stronger augmentations are applied. Furthermore, in the case of rotation, we observe that our augmentation prediction task tries to prevent the performance degradation from learning invariance to rotations. For example, in CIFAR100 (Krizhevsky et al., 2009), the baseline loses 4.66% accuracy (54.90%→50.24%) when using rotations, but ours does only 0.37% (58.65%→58.28%). These results show that our AugSelf is less sensitive to the choice of augmentations. We believe that this robustness would be useful in future research on representation learning with strong augmentations.

Solving geometric and color-related pretext tasks. To validate that AugSelf is capable of learning augmentation-aware information, we try to solve two pretext tasks requiring the information: 4-way rotation ($0°$, $90°$, $180°$, $270°$) and 6-way color channel permutation (RGB, RBG, ..., BGR) classification tasks. We note that the baseline (SimSiam) and our method (SimSiam+AugSelf) do not observe rotated or color-permuted samples in the pretraining phase. We train a linear classifier on top of pretrained representation without finetuning for each task. As reported in Table 10, our AugSelf solves the pretext tasks well even without their prior knowledge in pretraining; these results validate that our method learns augmentation-aware information.

Table 10. Linear evaluation accuracy (%) in augmentation-aware pretext tasks.

Method	Rotation	Color perm
SimSiam	59.11	24.66
+ AugSelf	64.61	60.49

Compatibility with other methods. While we mainly focus on SimSiam (Chen & He, 2020) and MoCo (He et al., 2020) in the previous sections, our AugSelf can be incorporated into other unsupervised learning methods, SimCLR (Chen et al., 2020a), BYOL (Grill et al., 2020), and SwAV (Caron et al., 2020). Table 9 shows the consistent and significant gains by AugSelf across all methods and downstream tasks.