
Distribution Estimation to Automate Transformation Policies for Self-Supervision

Seunghan Yang*, Debasmit Das*, Simyung Chang, Sungrack Yun, Fatih Porikli
Qualcomm AI Research[†]
{seunghan, debadas, simychan, sungrack, fporikli}@qti.qualcomm.com

Abstract

In recent visual self-supervision works, an imitated classification objective, called pretext task, is established by assigning labels to transformed or augmented input images. The goal of pretext can be predicting what transformations are applied to the image. However, it is observed that image transformations already present in the dataset might be less effective in learning such self-supervised representations. Building on this observation, we propose a framework based on generative adversarial network to automatically find the transformations which are not present in the input dataset and thus effective for the self-supervised learning. This automated policy allows to estimate the transformation distribution of a dataset and also construct its complementary distribution from which training pairs are sampled for the pretext task. We evaluated our framework using several visual recognition datasets to show the efficacy of our automated transformation policy.

1 Introduction

Recently, self-supervised learning (SSL) has received great attention in the field of computer vision. In contrast to the supervised learning that requires ground-truth labels, SSL learns representations by defining a pretext task that helps construct labels from the input signals themselves. In literature, a number of pretext tasks have been proposed with various transformations and augmentations: *e.g.*, predicting the rotation degrees [1, 2], solving jigsaw puzzles [3], and minimizing the distance between representations of different augmented views of the same image (instance discrimination) [4, 5]. The constructed objective functions of the pretext task are applied to learn the representations, which are then re-used for downstream applications [6–12].

The performance of the self-supervised representation on downstream applications depends highly on the choice of the transformations used in the pretext task. However, there have been only few works that examined the choice of the transformations [13–15]. Especially, [13] expressed a Visual Transformation for Self-Supervision (VTSS) hypothesis that postulates the learned representations would be less useful if the pretext task involves the transformations already present in the dataset. It provided the empirical verification by manually inspecting the dataset, finding the present transformations (*e.g.*, rotation, translation, and scale), and evaluating the effectiveness of the found transformations with multiple downstream classification problems. However, even if the VTSS hypothesis is supported, the following two questions still need to be answered to utilize it in deep learning scenarios, "How to determine the transformation already present in the dataset?", "How to identify transformations that are effective for a self-supervision task?"

This paper proposes a novel method to automate the transformation policy for self-supervision tasks to address the above questions. First, we introduce a learning framework to estimate visual

*These two authors contributed equally.

[†]Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

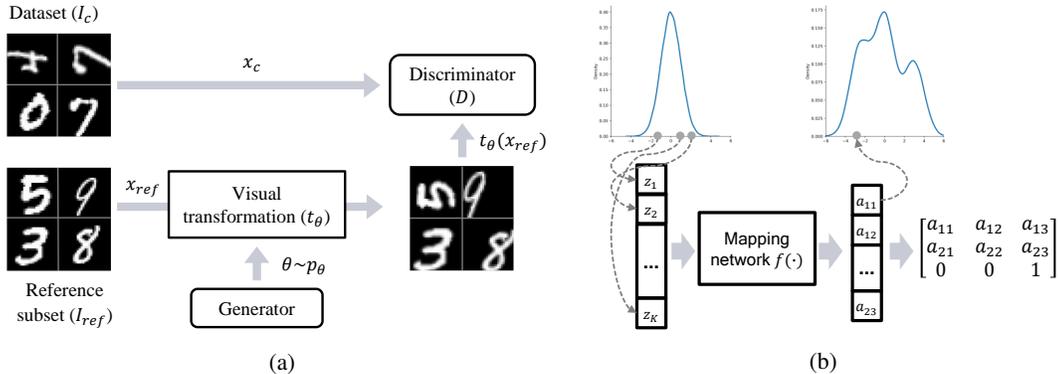


Figure 1: (a) Our framework to estimate the visual transformations present in a dataset I_c . (b) Generator produces visual transformation parameters through the mapping network that projects a *known* distribution to the desired distribution.

transformations. We learn a parameterized visual transformation function to estimate the distribution of the transformation present in the dataset. Then, we construct another distribution which is complementary to the estimated one to obtain the transformation instance not included in the dataset. For instance, if the estimated transformation is known to follow a uniform distribution between certain bounds, such as 0-90 degrees rotation, we can easily determine its complementary distribution ranging from 90-360. However, modeling the transformation distribution of a dataset without such a prior is not straightforward. To handle this challenge, we define a mapping network that estimates the transformation distribution of the dataset with a known distribution (e.g., Gaussian), and the mapping network is learned by adversarial learning [16]. Here, the histogram of the output values from the mapping network represents the transformation distribution, and then we can construct its complementary distribution to sample a transformation instance for the pretext task of self-supervised learning. Through extensive experiments on various datasets, including MNIST [17], Fashion MNIST [18], SVHN [19], CIFAR-10 [20], and CIFAR-100 [20], we provide empirical evidence to confirm the efficacy of our automated visual transformation policy.

In summary, the contributions of this paper can be summarized as follows: (i) A framework based on a generative adversarial network to obtain the distribution of transformations present in a dataset; (ii) Construction of the distribution complementary to the estimated distribution in (i); (iii) Generation of a useful pretext task for self-supervised learning with the transformation instances sampled from the complementary distribution; (iv) Comprehensive study, analysis and comparison of the representations learned from our transformation instances on multiple visual recognition benchmarks.

2 Related Work

Self-supervised learning. With the capability of learning a representation given unlabeled data, many research works on self-supervised learning have recently been presented and applied to various applications in computer vision, natural language processing [21, 22], and robotics [23–27]. Most previous works consider manually designing a pretext task involving specific visual transformations.

Analysis on the visual transformation and strategy. Only a few works address the impact of certain geometric transformations in the pretext task to learn representations for downstream tasks [13–15]. To our knowledge, VTSS [13] is the first to discuss that certain transformations are preferable for defining a pretext task. We automatically find good transformation parameters for defining pretext tasks and propose an automated policy choosing the transformation instances for the target dataset.

3 Proposed Method

3.1 Distribution estimation for visual transformations

Learning framework to estimate visual transformations. We propose a learning-based framework to estimate the distribution of visual transformations present in the dataset. In practice, most

collected datasets inherently assume that visual transformations they depict are from specific distributions. Generally, the priors on such distributions are not available, and thus we consider learning a parameterized model to represent such distributions.

For a dataset I_c , we choose a reference subset that includes the most representative and frequent data (e.g., upright images), denoted by $I_{ref} = \{x_{ref}^i\}_{i=1}^N$ where N is much smaller than the size of I_c (e.g., 1-3 images per class). Let \mathcal{T} be a set of all possible visual transformations such as affine, color, style-based visual transformation, etc. Assume it consists of K transformations, $\mathcal{T} = \{t_1, t_2, \dots, t_K\}$, where t_k denotes a transformation type, and $t_k(x)$ represents the corresponding transformed data. If we augment I_{ref} with the transformations present in I_c , it is intuitive that the transformation parameter distributions of $I_{trans} = \{t_1(x_{ref}^{1:N}), t_2(x_{ref}^{1:N}), \dots, t_K(x_{ref}^{1:N})\}$ and I_c are similar. We omit k for the simplicity in describing our algorithm for the subsequent paragraphs.

Based on this, we consider to model p_θ which is the distribution of transformation parameter for I_{trans} , from where a specific transformation t can be sampled. Here, we learn p_θ to represent p_c which is the distribution of transformation parameter for I_c , by minimizing the distance between p_θ and p_c with an adversarial learning (discriminator and generator). The idea is illustrated in Fig. 1a.

One of key components of our framework is modeling the learnable distribution p_θ with the parameterized transformation $t(x; \theta)$. A transformation parameter is sampled from p_θ to obtain a specific transformation $t(x; \theta)$. We consider training a mapping network $f(\cdot)$ to project a standard distribution into the desired distribution p_θ , and the reparameterization trick [28] is exploited to backpropagate through to the parameters θ of the distribution p_θ . In Appendix C, we describe how to parameterize visual transformations t including geometric and color transforms and how to flow back the loss gradients to sampling grid coordinates for geometric transforms.

As illustrated in Fig 1b, we learn the mapping network $f(\cdot)$ that maps a known distribution to the desired distribution. This can induce more complex distributions. The mapping network consists of fully connected layers with non-linear activations, and the inputs are random vectors sampled from known distributions, e.g., uniform or Gaussian distributions. The outputs of the mapping network are transformation parameters, and p_θ can be obtained from the histogram of the outputs. Thus, the network learns how to produce the desired distributions by combining known distributions. In summary, the network projects a simple distribution into any complex distribution that can model the distribution of transformations present in the dataset.

To train p_θ , we minimize the distance between two distributions p_θ and p_c . For this, we consider the following loss function, $L = E_{\theta \sim p_\theta} [l(t(x_{ref}; \theta), x_c)]$, where $t(\cdot; \theta)$ indicates a visual transformation parameterized by θ , and l is a distribution matching objective function such as MMD loss [29] and GAN-based loss functions [16, 30, 31]. In this work, we employ WGAN-GP [31], which expedites the optimization of the generator as well as enforce the Lipschitz constraint by using the gradient penalty not to fail to converge.

3.2 Automating transformation policies

A complementary set of current visual transformations. We train the network that maps a known distribution to the desired distribution, i.e., the outputs of this network follow p_θ . To approximate this distribution p_θ , we use a sampling procedure. Specifically, we feed random vectors sampled from the known distribution to the mapping network to produce a number of output samples, which are then aggregated to form a histogram of values. Based on this histogram, its complementary q_θ can be obtained by subtracting each histogram value from the peak histogram value and normalizing it.

Transformation policies for self-supervised tasks. We consider two policies for choosing a non-conflicting pretext task: manual and automated policy. In the manual policy, we sample the transformation instance from the parameter ranges where p_θ is 0 or low value. For example, suppose that I_c contains the rotated images from 0 to 120 degrees. We can sample the transformation instance for the pretext task from 120 to 360 degrees manually. In the automated policy, the transformation instance is sampled from the constructed complementary distribution q_θ . To sample a value following a specified distribution, we exploit the *inverse transform sampling* method, which is a well-known method for pseudo-random number sampling. First, the CDF of q_θ is obtained. Then, we obtain the transformation instances by mapping the samples from the uniform distribution $U(0, 1)$ to the inverse of the CDF. The obtained transformation instances can then be used for the pretext task.

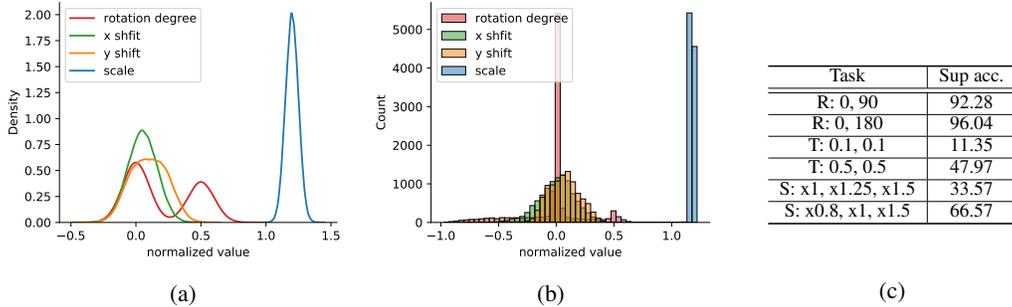


Figure 2: (a) The ground-truth distribution of Transformed MNIST. (b) Histogram of output values from the mapping network. (c) Experimental results of the pretext task parameters in the left column. Once the representation with the pretext task is learned, the classifier is trained with supervised learning using the fixed representations. We denote this classification accuracy by Sup acc.

4 Experiments

We demonstrate that our framework can estimate the distribution of visual transformations present in the dataset. Experimental details are in Appendix A. First, we estimated the distribution on Transformed MNIST, which is constructed by applying various affine transformations to the original MNIST [17]. Fig. 2a illustrates the ground-truth distribution of the transformations present in Transformed MNIST. Note that the rotation, translation, and scaling operations have their own units of measurements, but we normalize them for better visualization and more stable training: $[-1, 1]$.

For Transformed MNIST, as shown in Fig. 2b, the obtained histogram is similar to the ground truth distribution. Based on this, we can design the pretext tasks for given dataset. In particular, we define two types of self-supervision tasks. The first type is the transformation instance from p_θ , and the second type is that from q_θ , i.e., complementary to p_θ . In Fig. 2c, we compared these two types of pretext tasks where R, T, and S indicate rotation, translation, and scaling, respectively. Once the self-supervised network is trained with pretext tasks, the representations are used for downstream tasks (see Appendix B). For the rotation case, the representations learned with the task classifying 0 and 90 degrees encounter the transformation conflict (we can find 90 degree has high probability from Fig. 2a) and result in performance degradation compared to the task of 0 and 180 degree. Especially, in the case of translation and scaling, the performance gap becomes bigger. The results demonstrate that complementary pretext tasks lead to learn useful representation for downstream task.

Next, we applied our automated policy to the real dataset. We plot the histogram of transformation parameters for the SVHN dataset in Fig. 3. In VTSS [13], they described the results of pretext tasks by using prior knowledge or manual inspection on the dataset. In our case, we analyze the results based on the estimated distribution beyond the prior knowledge available for the dataset and define effective pretext tasks even significant degree of transformations already exist in the dataset. In VTSS, they observed that the pretext task based on scale-based transformations leads poor performance for SVHN. This is because the SVHN dataset has some degrees of scale variation (refer the histogram in Fig. 3). More experimental results using other datasets are described in Appendix D.

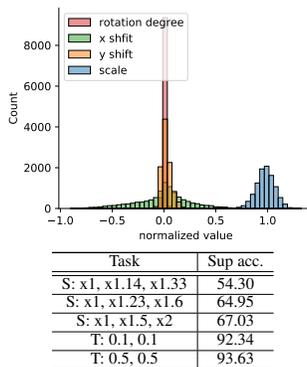


Figure 3: Experimental results on SVHN.

5 Conclusion

We propose a framework to automatically estimate the distribution of visual transformations present in the dataset. This enables efficiently creating pretext tasks that are not depicted in the dataset. We show that the representations learned from this task are more informative on downstream classification scenarios compared with those from the conflicting task.

References

- [1] Zeyu Feng, Chang Xu, and Dacheng Tao. Self-supervised representation learning by rotation feature decoupling. In *CVPR*, pages 10364–10374, 2019.
- [2] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *ICLR*, 2018.
- [3] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, pages 69–84. Springer, 2016.
- [4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, pages 1597–1607, 2020.
- [5] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020.
- [6] Suriya Singh, Anil Batra, Guan Pang, Lorenzo Torresani, Saikat Basu, Manohar Paluri, and CV Jawahar. Self-supervised feature learning for semantic segmentation of overhead imagery. In *BMVC*, volume 1, page 4, 2018.
- [7] Ahmed Taha, Moustafa Meshry, Xitong Yang, Yi-Ting Chen, and Larry Davis. Two stream self-supervised learning for action recognition. *CVPR-W*, 2018.
- [8] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018.
- [9] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *ICCV*, pages 8059–8068, 2019.
- [10] Sylvestre-Alvise Rebuffi, Sebastien Ehrhardt, Kai Han, Andrea Vedaldi, and Andrew Zisserman. Semi-supervised learning with scarce annotations. In *CVPR-W*, pages 762–763, 2020.
- [11] Xialei Liu, Joost Van De Weijer, and Andrew D Bagdanov. Exploiting unlabeled data in cnns by self-supervised learning to rank. *TPAMI*, 41(8):1862–1878, 2019.
- [12] Shubhankar Borse, Ying Wang, Yizhe Zhang, and Fatih Porikli. Inverseform: A loss function for structured boundary-aware segmentation. In *CVPR*, pages 5901–5911, 2021.
- [13] Dipan K Pal, Sreena Nallamothu, and Marios Savvides. Towards a hypothesis on visual transformation based self-supervision. *BMVC*, 2020.
- [14] Mandela Patrick, Yuki M Asano, Polina Kuznetsova, Ruth Fong, Joao F Henriques, Geoffrey Zweig, and Andrea Vedaldi. Multi-modal self-supervision from generalized data transformations. *arXiv preprint arXiv:2003.04298*, 2020.
- [15] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? *NeurIPS*, 2020.
- [16] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *NeurIPS*, 2014.
- [17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [18] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
- [19] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- [20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL-HLT*, 2018.

- [22] Jiawei Wu, Xin Wang, and William Yang Wang. Self-supervised dialogue learning. *ACL*, 2019.
- [23] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *ICML*, 2019.
- [24] Eric Jang, Coline Devin, Vincent Vanhoucke, and Sergey Levine. Grasp2vec: Learning object representations from self-supervised grasping. *CoRL*, 2018.
- [25] Frederik Ebert, Sudeep Dasari, Alex X Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. In *CoRL*, 2018.
- [26] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *ICRA*, pages 2146–2153. IEEE, 2017.
- [27] Adithyavairavan Murali, Lerrel Pinto, Dhiraj Gandhi, and Abhinav Gupta. Cassl: Curriculum accelerated self-supervised learning. In *ICRA*, pages 6453–6460. IEEE, 2018.
- [28] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [29] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. A kernel method for the two-sample-problem. *NeurIPS*, pages 513–520, 2006.
- [30] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.
- [31] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *NeurIPS*, 2017.
- [32] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *ICLR*, 2014.
- [33] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *NeurIPS*, 2015.
- [34] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, page 1419, 2016.
- [35] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pages 2217–2226, 2019.
- [36] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, pages 1–9, 2018.
- [37] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kaloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.
- [38] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017.
- [39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.
- [40] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009.
- [41] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschiot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.
- [42] Tete Xiao, Xiaolong Wang, Alexei A Efros, and Trevor Darrell. What should not be contrastive in contrastive learning. In *ICLR*, 2021.

A Distribution estimation for visual transformations

A.1 Implementation details

Our framework to estimate the distribution of visual transformations consists of two networks: a generator and a discriminator. The generator is constructed with three linear layers with 128 dimensional output except that the last layer is 6 dimensional output. We used LeakyReLU with 0.2 slope for the first two layers, and the last layer has the tangent hyperbolic function to produce values between -1 and 1 . The generator takes 10 dimensional inputs sampled from Gaussian distribution, and outputs 6 parameters for affine transformation (rotation, scale, and translation). We found that the affine transformation parameters can be accurately estimated with two generators: *i.e.*, the first one for the scale parameter and the other one for remaining five parameters. Similar with the affine case, the color transformation can be generated from 10 dimensional inputs (See the details in Appendix C). In this case, the generator outputs four dimensional parameters: brightness, saturation, contrast, and hue. And, the network architecture is the same with the affine transformation case, but the output dimension is four. We adopted a simple discriminator that has three linear layers with the LeakyReLU function. The input size of the first layer depends on the image size, and the output dimensions of the three layers are respectively 512, 256, and 1. The discriminator output represents a fake or a real, *i.e.*, an image from the given dataset (real) or a transformed image using the generator (fake).

We trained these networks with the aid of WGAN-GP [31]. We adopted the basic settings of WGAN-GP: $\lambda = 10$, $n_{critic} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, and $\beta_2 = 0.9$. With the batch size of 10 and learning rate of $5e - 5$, we trained the network for 500,000 iterations, which took 6 hours in our single NVIDIA TITAN Xp GPU. Fig. 4 illustrates three histograms obtained at different iteration points, and we observed that there is no big difference after 500,000 iterations.

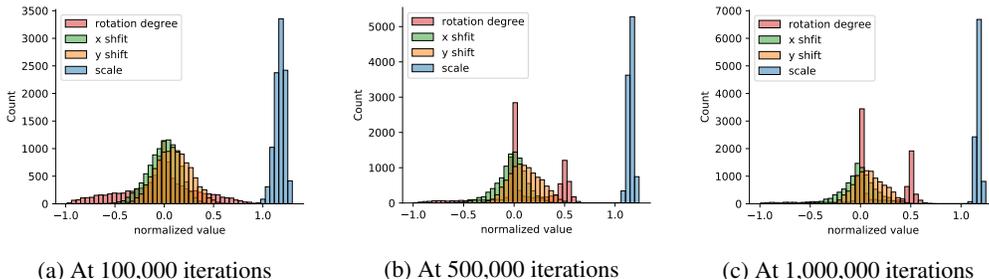


Figure 4: The histograms obtained from the outputs of the mapping network trained using Transformed MNIST at (a) 100,000, (b) 500,000, and (c) 1,000,000 iterations.

A.2 The method to reduce the artifact of a transformed image

Through adversarial learning, the discriminator is trained to distinguish the original images in the dataset from the images transformed by the generator given the reference subset. Our objective is to train the generator to deceive the discriminator such that the transformed images have a similar distribution with the original images. However, the discriminator may easily identify the generated images when they have artifacts like zero padding. These artifacts cause the network to learn an undesired property. For example, assume that the input image contains a dog of 45 degree rotated, and the objective is to find the amount of the rotation by generating a similar image using an upright dog image. However, when the image is rotated in the generator, the four corners are zero-padded, and hence it provides a crucial hint for the discriminator. Often, it leads to converge to a meaningless distribution which is not our objective. To mitigate this problem, we perform transformations first which is followed by a center-crop. For example, given a 32x32 image, a transformation is applied, and then the 24x24 image is obtained by the center-crop, which can cover 8 pixel shifts with no zero padding. For 256x256 image, the resulting image size is 196x196 where these two numbers were empirically selected through multiple trials. With this procedure, we can accurately estimate the distribution of visual transformations without an artifact effect.

B Self-supervised learning for downstream tasks

B.1 Implementation details

We used the NIN architecture [32] as the backbone feature extraction network to learn the representations for downstream tasks. The NIN consists of four convolutional blocks, and each block contains three convolutional layers. Convolution, batch normalization, and ReLU are sequentially placed in a single convolutional layer. In training a pretext task for rotation, translation, and scaling categories, the classifier of a single linear layer is added after global average pooling. We used SGD with the batch size of 128, momentum of 0.9, weight decay of $5e - 4$, and learning rate of 0.1. We dropped the learning rates by a factor of 5 after epochs 60, 120, and 160. We trained the network in total for 200 epochs. For a fair comparison among several pretext tasks, we used the same training procedure.

To evaluate the representation learned by self-supervision, a classifier is trained using the learned features. Specifically, in our experiments on Fashion MNIST [18], SVHN [19], CIFAR-10 [20], and CIFAR-100 [20], we followed the existing evaluation protocols [2, 13] by adding a classifier on top of the second convolutional block. The classifier consists of a single conv block, global average pooling, and one linear layer. Here, the conv block is the same with the conv3 in NIN architecture. We used SGD with batch size 128, momentum 0.9, weight decay $5e - 4$, and learning rate of 0.1, which is dropped by a factor of 5 after epochs 35, 70, and 85. We trained the network in total for 100 epochs.

B.2 Details of constructing pretext tasks

We introduce four pretext tasks, R, T, S, and RST, which indicate rotation, translation, scaling, and the combination of all, respectively. For the rotation case, the unit is the rotation degree. Translation and scaling units are the ratio of shift amount to the image size and the scaling ratio. Note that the translation task involves 5-way classification: left and right shift along with x and y axis, and 0 translation.

When constructing a pretext task of rotation, translation, and scaling, we also encounter the same artifact problem. The artifact induced by the transformation for a pretext task would be a big hint for the network, and hence the useful representation for downstream tasks may not be obtained. With this reason, we consider the pretext tasks of each transformation as follows: 1) For rotation, we use 90, 180, 270, and 360 degree which do not incur the zero-padding, 2) For translation and scaling, we perform the center crop depending on the task parameter: e.g. when we define the translation task of 3 pixels in 32x32 image, we crop the center image of 26x26.

C Visual transformation

C.1 Parameterize geometric transformation

We describe how to parameterize a distribution over the set of visual transformations, particularly affine transformations. This discussion can be extended to projective and spline transformations as well. For the affine transformation, we include translation, scaling, similarity, reflection, rotation, and shear. The affine matrix is parameterized as follows:

$$t = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}, \quad \text{where } a_{11}, a_{12}, \dots, a_{23} \in \theta. \quad (1)$$

These parameters are sampled from p_θ , and this matrix is applied to the coordinate system. We adapt Spatial Transformer Network [33] allowing loss gradients to flow back to the sampling grid coordinates as well as to the input image. For example, in case of affine transformation, the sampling grid is result of warping the regular grid with an affine matrix, as in (1). If we use a bilinear sampling kernel to map the output pixel value, the output image V is obtained as follows:

$$O_i^c = \sum_n^H \sum_m^W I_{nm}^c \max(0, 1 - |u_i^s - m|) \max(0, 1 - |v_i^s - n|), \quad (2)$$

where O_i^c indicates the output value for pixel i at location (u_i^t, v_i^t) of channel c , and (u_i^t, v_i^t) are the target coordinates of the regular grid. I_{nm}^c is the value at location (n, m) of the input image with

height H and width W , and (u_i^s, v_i^s) indicates the spatial location of the input corresponding to the output coordinate through the sampling grid. We can define the gradients with respect to I_{nm}^c and (u_i^s, v_i^s) such that the backpropagation of the loss can be flowed through this sampling mechanism.

C.2 Parameterize color transformation

In all previous experiments, we considered the transformations of rotation, scale, and translation. This section will describe how we can parameterize the color transformation. Specifically, we consider changing the brightness, saturation, contrast, and hue of an image. We introduce learnable parameters for these transformations and train the mapping network such that each parameter represents the transformation distribution of the dataset.

First, the brightness change can be simply expressed as follows:

$$x_{brt} = x\alpha_{brt}, \quad (3)$$

where α_{brt} is a learnable scale factor. Second, we can change the saturation of x by using a linear combination of the original image x and the gray-scaled version of the image x_{gray} as follows:

$$x_{sat} = x\alpha_{sat} + x_{gray}(1 - \alpha_{sat}), \quad \text{and} \quad x_{gray} = 0.299x_r + 0.587x_g + 0.114x_b, \quad (4)$$

where x_r , x_g , and x_b indicate three color channels of x . The scale factor α_{sat} is in the range of $[0, 1]$. And third, we can change the contrast of x by calculating a linear combination of x and the average of x_{gray} over all spatial dimensions as follows:

$$x_{con} = x\alpha_{con} + \text{mean}(x_{gray})(1 - \alpha_{con}), \quad (5)$$

where α_{con} is a scale parameter. Finally, we resort to a linear approximation for the hue change in the YIQ color space. We firstly convert RGB to YIQ, and then apply rotation to the IQ components. The transformation from RGB to YIQ is denoted by T_{YIQ} , and we can obtain the following expression:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = T_{YIQ} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \quad \text{where} \quad T_{YIQ} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix}. \quad (6)$$

In the YIQ color space, we can change the hue of an image by rotating the IQ components with a rotation matrix:

$$R_{\theta_{hue}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_{hue} & -\sin\theta_{hue} \\ 0 & \sin\theta_{hue} & \cos\theta_{hue} \end{bmatrix}, \quad (7)$$

where $\theta_{hue} = 2\pi\alpha_{hue}$, and α_{hue} is a learnable parameter. In summary, the hue change can be expressed by the following matrix multiplication:

$$x_{hue} = T_{RGB}R_{\theta_{hue}}T_{YIQ}x, \quad (8)$$

where T_{RGB} is the inverse matrix of T_{YIQ} . Using the procedure, we can employ learnable parameters for color transformation, and hence we can estimate the distribution of color transformations present in the dataset.

D Experimental results

D.1 Distribution estimation

We plot the histogram of transformation parameters for Fashion MNIST [18] in Fig. 5. As illustrated in Fig. 5, the Fashion MNIST dataset consists of well-aligned images, i.e., almost no transformation is present. This suggests that the network can learn powerful representations with pretext tasks constructed by any visual transformation.

Furthermore, we illustrated the histograms of output values obtained from the mapping network on four visual recognition datasets, CropDisease [34], EuroSAT [35], ISIC2018 [36, 37], and ChestX [38].

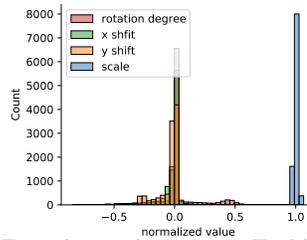


Figure 5: Experimental results on Fashion MNIST.

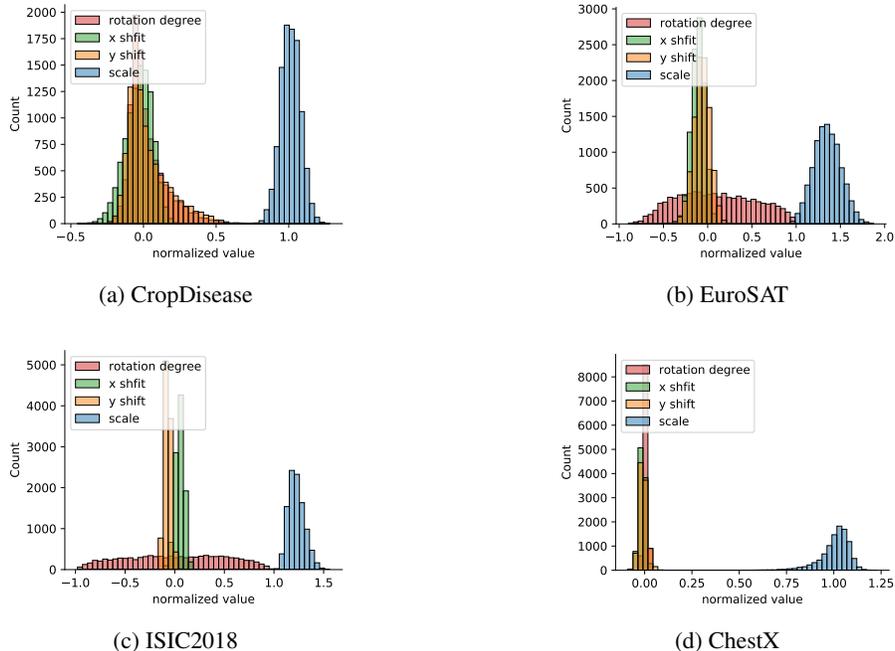


Figure 6: The estimated distributions of visual transformation on CD-FSL benchmarks.

Even when the images are big, we found that the mapping network can represent the transformation distribution well as shown in Fig 6. The EuroSAT and ISIC2018 dataset have the rotation transformation almost over the entire range, $[-\pi, \pi]$. Visually checking the image samples, we can find the two datasets already contain various rotated images.

D.2 Comparison with other methods on downstream tasks

To compare with RotNet [2] and VTSS [13], we set hyper-parameters with the same condition as others (using horizontal flip for data augmentation). We reported the accuracy in Table 1a.

We want to show that the pretext tasks selected by our estimated distribution can force the network to learn more meaningful representations than the manually selected one. In Table 1b, we define the pretext task, RST, which is the combined pretext task of three types of visual transformations. Baseline selects RST manually and ours defines RST based on the estimated distribution. We use rotation with 0, 90, 180, and 270, translation with 0.1, 0.1, and scaling $\times 1$, $\times 1.14$, $\times 1.33$ for the baseline RST while we define the pretext task carefully based on estimated distribution. Note that we do not use data augmentation technique to show the effectiveness of each pretext task separately.

E Discussion

E.1 Extension to contrastive loss

We consider a contrastive learning framework based on SimCLR [4]. We observed that the contrastive loss with the augmented versions appropriate to the target dataset helps the network to learn generalizable representations even when the domain gap between source and target is large, and

	FMNIST	SVHN	CIFAR-10	CIFAR-100
RotNet	91.94	91.29	89.15	63.62
VTSS	92.18	91.72	89.58	64.87
Ours	93.22	94.75	89.58	64.00

(a) Comparison with other methods.

	FMNIST	SVHN	CIFAR-10	CIFAR-100
Baseline	91.63	91.72	86.32	60.79
Ours	93.22	94.75	87.01	62.48

(b) Comparison with our baselines.

Table 1: Performance comparison on downstream tasks.

target data is scarce. Intuitively, this indicates proper pretext tasks for contrastive loss functions are effective mechanisms to derive meaningful representations for downstream tasks.

We apply the proposed automated transformation policy to generating augmented versions used for contrastive learning. As such, we retrain the base network, ResNet-50 [39], with the standard supervised loss on ImageNet [40], then fine-tune the base network, the added classifier, and the projector on the target dataset with both cross-entropy and contrastive losses. Specifically, we use supervised contrastive loss function [41] with the SimCLR [4] framework. It is to be noted that supervised contrastive loss function builds on the contrastive self-supervised literature by leveraging label information. We conduct experiments on cross-domain benchmark datasets (CropDisease [34], EuroSAT [35], ISIC2018 [36, 37], and ChestX [38]) where we split the total dataset such that 5% was used for training while the remaining was used for testing.

We compare four methods: fine-tuning the network using (a) cross-entropy loss (Baseline) and with additional contrastive loss of (b) random rotation from -180 to 180 (SimCLR with Rot.), (c) random affine transformations, which contain rotation from -180 to 180, translation from -0.5 to 0.5, and scaling from 0.5 to 1.5 (SimCLR with Aff.) and (d) automated transformation policies within the same range of (c) and obtained from the target dataset (SimCLR with ATP). Table 3 shows the results of fine-tuning with the contrastive loss function. We found that some pretext tasks, *i.e.*, data augmentations, are not helpful to the target tasks as pointed out by [42]. Our automated transformation policies for contrastive learning leads to better performance than the naive approach of using random augmentation. It shows that not only types of transformations, *e.g.*, color or spatial transformation, but also the degree of transformations, *e.g.*, 30 or 60 degrees rotation impact the performance significantly in contrastive learning.

Method	ChestX	ISIC	EuroSAT	CropDiseases
Baseline	41.16	67.00	96.35	96.43
+ SimCLR with Rot.	38.24	68.43	97.57	97.48
+ SimCLR with Aff.	40.65	68.97	97.66	97.41
+ SimCLR with ATP	42.33	72.97	97.72	97.60

Table 3: Effect of fine-tuning on cross-domain benchmark datasets. Rot., Aff., and ATP indicate random rotation, random affine transformation, and automated transformation policies, respectively.

E.2 Large-scale experiment

In the future, we will conduct experiments with pretraining on a large-scale dataset and transfer to other datasets for evaluation. Since the large-scale dataset consists of images following a complex distribution, the manual selection of pretext tasks cannot be the best one and it may produce transformation conflict. Still, we believe that our policies will work well on a large-scale dataset.