A Simple Data Mixing Prior for Improving Self-Supervised Learning

Sucheng Ren[1] Huiyu Wang[2] Zhengqi Gao[3] Shengfeng He[1*] Alan Yuille[2]
Yuyin Zhou[4] Cihang Xie[4*]

[1] South China University of Technology [2]Johns Hopkins University
[3] Massachusetts Institute of Technology [4]UC Santa Cruz

Abstract

*Data mixing (e.g., Mixup, Cutmix, ResizeMix) is an essential component for advancing recognition models. In this paper, we focus on studying its effectiveness in the self-supervised setting. By noticing the mixed images that share the same source images are intrinsically related to each other, we hereby propose SDMP, short for **S**imple **D**ata **M**ixing **P**rior, to capture this straightforward yet essential prior, and position such mixed images as additional **positive pairs** to facilitate self-supervised representation learning.*

Our experiments verify that the proposed SDMP enables data mixing to help a set of self-supervised learning frameworks (e.g., MoCo) achieve better accuracy and out-of-distribution robustness. More notably, our SDMP is the first method that successfully leverages data mixing to improve (rather than hurt) the performance of Vision Transformers in the self-supervised setting. Code is publicly available at https://github.com/OliverRensu/SDMP.

1. Introduction

Data mixing is one of the key ingredients for improving recognition models. The concept of data mixing is firstly introduced in Mixup [47], which trains models on convex combinations of pairs of images and their labels. This idea subsequently inspires several follow-ups, including mixing images and cropped patches [45], mixing images and thumbnails [44], and mixing among cropped patches [4,38].

However, interestingly, data mixing plays little role in the recent surge of self-supervised learning. For instance, while naïvely replacing original images with their mixed counterparts substantially improves Vision Transformers (ViTs) in the supervised setting [39], it cannot improve ViTs under the self-supervised setting. Though many efforts [27, 31, 42] have been made recently by developing more sophisticated training strategies in data mixing, they are exclusively focusing on Convolutional Neural Networks (CNNs). As shown in Table 1 in Section 4, all these methods still fail to help (or even hurt) ViTs [16].

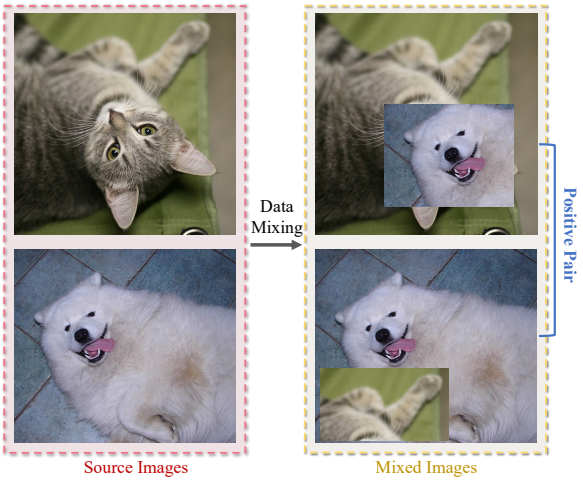


Figure 1. For the mixed images that share the same source (*e.g.*, a cat image and a dog image), they are semantically related and can be treated as additional positive pairs in self-supervised learning.

In this paper, *we aim to develop a generic training strategy in data mixing that can improve the self-supervised representation learning of both CNNs and ViTs*. By taking a closer look at the popular data mixing implementation where an image is mixed with another image *that sampled from the same batch but with the flipped order*[1], we observe such created mixed samples are inherently related in pairs (*e.g.*, an example is provided in Figure 1). This indicates that, now for one mixed image, there exist three related samples (*i.e.*, a pair of source images and a mixed image created with a different mixing parameter) in the same training batch. This intrinsic relationship qualifies the pair of mixed images to be treated as additional positive samples in self-supervised learning to facilitate representation learning.

*Corresponding authors: Shengfeng He (hesfe@scut.edu.cn), Cihang Xie (cixie@ucsc.edu)

[1]The traditional data mixing implementation randomly samples two batches and then mixes them with each other; while this instantiation of data mixing only samples one batch and then mixes pairs of images from the same batch. It generally will not hurt performance, and is very popular in many libraries (*e.g.*, timm [43]), due to its faster computational speed.

Motivated by the observation above, we hereby propose to leverage this **S**imple **D**ata **M**ixing **P**rior (dubbed *SDMP*) to holistically model the relationship among samples for enhancing self-supervised learning. Different from previous methods [27, 31, 32, 36, 42], SDMP not only considers the relationships between source images and the mixed counterparts, but also encodes the connections between mixed samples in representation learning. We further enhance SDMP's representation learning by semantically weighting the loss to accurately capture the relationships among samples.

Our empirical results verify that the proposed SDMP successfully helps a set of self-supervised learning frameworks gain better accuracy on different visual benchmarks and robustness on out-of-distribution samples, using both CNNs and ViTs. More essentially, we would like to highlight that our SDMP is the first strategy that enables data mixing to improve self-supervised ViTs. For example, by building upon the latest MoCo v3 [11], while existing training strategies [27, 31, 42] all hurt the top-1 ImageNet accuracy of ViT-S by 0.2% - 1.6%, SDMP successfully improves the top-1 ImageNet accuracy of ViT-S by 0.6%. We hope the technical insights and results provided in this work will be helpful for future works on studying data mixing in self-supervised learning.

2. Related Work

Self-supervised learning. Self-supervised learning aims to let models acquire semantically meaningful representations without human annotations. Traditional pretext tasks include reconstruction by autoencoder [3], colorization [48], rotation prediction [17] or combination of them [15, 34].

Contrastive learning, which aims to discriminate different views and samples, is one of the most successful pretext tasks. Its basic idea is to maximize the similarity of positive pairs and to minimize the similarity of negative pairs. However, discrimination based methods generally require a large amount of negative pairs, *e.g.*, SimCLR [8] takes a large training batch, MoCo [9, 21] requires a memory bank, and others [1, 5, 46] take a grouping or clustering. Later works [7, 10, 19] successfully remove the need for negative samples, enabling small batch training in self-supervised learning. In this work, we focus on improving self-supervised learning, using data mixing.

Data mixing. Mixup [47] is the first work on data mixing, which convexly combines data pairs and their corresponding labels to regularize network training, inspiring numerous followups, including Cutout [14], CutMix [45], SaliencyMix [40] and PuzzleMix [28].

Recent works begin to introduce data mixing into self-supervised learning. Verma *et al*. [42] utilize Mixup to create similar and dissimilar examples by mixing data samples differently, either at the input or hidden-state levels. [29, 31] explore the semi-contrastive encoding with mixup of negative and positive pairs. Unlike previous works which focus on CNNs, we are the first to explore data mixing for improving ViTs under the self-supervised setting. We additionally point out that properly modeling the relationships among intra-batch mixed data (which was largely overlooked) is essential for strengthening self-supervised learning.

Transformers. Transformer [13, 41] is the de-facto standard for natural language processing tasks. Recently, Dosovitskiy *et al*. [16] successfully introduce the pure Transformer architecture for computer vision, attaining competitive visual recognition performance compared to CNNs on a range of benchmarks. Nonetheless, the original ViT training framework strongly demands hundreds of millions of external (in-house) images [37] in training. Touvron *et al*. [39] relax this learning constraint by incorporating a set of strong regularization techniques into ViT training framework, where data mixing plays a vital role. In this work, we are particularly interested in exploring whether data mixing can improve ViT under the self-supervised setting.

3. Method

3.1. Which Images For Mixing?

Traditionally, data mixing generates mixed data by mixing two randomly sampled images (usually drawn from two different min-batches). While in this work, we follow the strategy of the widely used Python Deep Learning Package timm [43]—we mix the i-th image with another randomly selected j-th image that comes from the same batch. *We by default set j to n-i where n is the number of images in this batch, for facilitating implementation*. We call this mixing strategy as *intra-batch mixing*. Specifically, this intra-batch mixing strategy enables the mixed images now are related in pairs (as they share the same source images, see an example in Figure 1), which will facilitate the virtual label assignment introduced in Section 3.3.

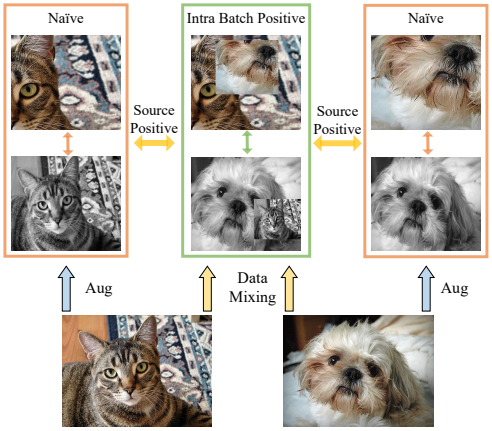
3.2. How To Mix?

To mix images, we mainly consider an element-wise data mixing method, Mixup [47], and two regional data mixing methods, *i.e.*, CutMix [45] and ResizeMix [35].

Mixup. Mixup element-wisely mix two sample while preserving the whole source data. Specifically, Mixup takes the weight λ_i following a Beta distribution Beta(α, α) and mixes the data as the following:

$$\begin{aligned} x_i^{\text{mix}} &= \lambda_i x_i + (1-\lambda_i) x_{n-i}, \\ x_i'^{\text{mix}} &= \lambda_i x_i' + (1-\lambda_i) x_{n-i}', \end{aligned} \tag{1}$$

where x_i and x_{n-i} indicate the i-th and the $(n-i)$-th image. x_i' and x_{n-i}' are another view of the source images x_i and x_{n-i}, created by data augmentation (*e.g.*, color jittering).



Algorithm 1 Loss computation of our SDMP

```
a, b = aug(x), aug(x) # two different views of input x
lam = Beta(alpha, alpha).sample() # mixing coefficient
a = mix(a, a.flip(), lam)
a = normalize(model(a))
x_one_hot = one_hot(arange(len(x)))
logits1 = matmul(a, normalize(model(b)).T) / t
label1 = lam * x_one_hot + (1-lam) * x_one_hot.flip()
loss1 = CrossEntropyLoss(logits1, label1)
clam = min(lam, 1-lam.flip()) + min(1-lam, lam.flip())
logits2 = matmul(a, normalize(model(mix(b, b.flip(), lam))).T) / t
label2 = 1/(1+clam) * x_one_hot + clam/(1+clam) * x_one_hot.flip()
loss2 = CrossEntropyLoss(logits2, label2)
loss = loss1 + loss2
```

Figure 2. Left panel: The positive pairs considered in self-supervised learning, *i.e.*, the pair of different views (denoted as *naïve*), the pair of the source image and the mixed image (denoted as *source positive*) and the pair of the mixed images (denoted as *intra batch positive*). Right panel: the pseudo code of the SDMP in PyTorch.

CutMix. Different from Mixup, CutMix [45] mixes data regionally by cropping and pasting a specific patch from one image to another, *i.e.*, the mixed data comes from one whole image and a local region of another image. Note that both Mixup and CutMix by default are included in Transformer's training recipe [39] under the supervised training setting.

ResizeMix. For CutMix, the cropped patch and the image itself could be label-irrelevant, *e.g.*, the background of the source image is selected. To accurately match the semantics of the patch and the source data, ResizeMix [35] proposes to take the resized source image as the patch, as the following:

$$\begin{aligned} P_i = R(x_i, H_i^p, W_i^p) \quad P_i' = R(x_i', H_i^p, W_i^p), \\ x_i^{\text{mix}} = \text{Paste}(P_{n-i}, x_i), \\ x_i'^{\text{mix}} = \text{Paste}(P_{n-i}', x_i'), \\ \lambda_i = 1 - \frac{W_{n-i}^p * H_{n-i}^p}{W_i * H_i} \end{aligned} \tag{2}$$

where $R(\cdot, h, w)$ indicate the resize function applied with the size of height h and width w. $\text{Paste}(P, x)$ indicates pasting the patch P onto a random region of the image x. H_i, W_i indicate the height and the width of the i-th image; H_i^p, W_i^p are randomly sampled patch height and patch width for the i-th image.

3.3. What is the label in self-supervised learning?

Given true labels are not available in self-supervised learning, we next show how to assign virtual labels accordingly. We provide two case studies of assigning virtual labels in popular self-supervised learning frameworks.

Case 1: contrastive learning. Contrastive learning positions self-supervised learning as an instance classification task, assigning only one positive label to the positive pair and setting all the rest samples as negative. However, such assumption will confuse models, especially when the training batch is large and contain multiple samples from the same or the similar category.

Our SDMP explicitly relaxes this assumption by introducing extra positive pairs. Specifically, we propose to assign the virtual positive labels to all the positive pairs, including 1) the source data and the mixed counterparts; and 2) the pair of mixed data that comes from the same source data. This label assignment enforces the model to learn to minimize the distance between more than just one pair.

More concretely, firstly, to model the relationship between the source data and the mixed counterpart, the mix data x_i^m and the other view of the source images x_i' and x_{n-i}' (obtained via augmentation) will be feed into the the encoder f and the momentum encoder f_k, respectively, as the "query" and the "key". Therefore, the first part of our data mixing contrastive loss, learning from source data, is:

$$\begin{aligned} y_i^m = & f(x_i^m) \quad y_i' = f_k(x_i') \quad y_{n-i}' = f_k(x_{n-i}') \\ \mathcal{L}_{\text{MoCo}}^s = & -\lambda_i \log \frac{\exp(< y_i^m, y_i' > /\tau)}{\sum_{j=0}^{n} \exp(< y_i^m, y_j' > /\tau)} \\ & -(1-\lambda_i) \log \frac{\exp(< y_i^m, y_{n-i}' > /\tau)}{\sum_{j=0}^{n} \exp(< y_i^m, y_j' > /\tau)}, \end{aligned} \tag{3}$$

where τ is the temperature to normalize the output y.

As mentioned in Section 3.1, we follow the intra-batch mixing strategy of timm [43] to mix one batch of data and its reversed-order version accordingly. Therefore such mixed images are related in pairs (*e.g.*, x_i^m and x_{n-i}^m) as they share the same source data. Moreover, by considering data aug-

mentation, we now have x_i^m, $x_i'^m$ and $x_{n-i}'^m$, which share the same source image x_i and x_{n-i}, as the positive samples. The second part of our data mixing MoCo loss, which aims to from intra-batch mixing data, should be written as:

$$\begin{aligned}\lambda_i^c &= \min(\lambda_i, 1-\lambda_{n-i}) + \min(1-\lambda_i, \lambda_{n-i}),\\ \mathcal{L}_{\text{MoCo}}^m &= -\frac{1}{1+\lambda_i^c}\log\frac{\exp(< y_i^m, y_i'^m > /\tau)}{\sum_{j=0}^{n}\exp(< y_i^m, y_j'^m > /\tau)}\\ &\quad -\frac{\lambda_i^c}{1+\lambda_i^c}\log\frac{\exp(< y_i^m, y_{n-i}'^m > /\tau)}{\sum_{j=0}^{n}\exp(< y_i^m, y_j'^m > /\tau)}\end{aligned} \tag{4}$$

where λ^c indicate the shared part between two mixed data.

To sum up, our mixing MoCo loss can be used as an extra view, or replace one of the symmetric view in the MoCo. We hereby take the replace version to describe the loss of our mix MoCo loss:

$$\mathcal{L} = \sum_{i=0}^{n}\left[\mathcal{L}_{\text{MoCo}}^s(x_i, x_i') + \mathcal{L}_{\text{MoCo}}^m(x_i'^m, x_i, x_{n-i}, x_i^m)\right] \tag{5}$$

Case 2: knowledge distillation. Recent research focus on removing the negative samples in self-supervised transformer. DINO [7] introduce knowledge distillation to self supervised training transformer. The student and the teacher network has parameters θ_s and θ_t and the output of student network should be close to the output of teacher model. There is a distillation loss:

$$H(P_t(x), P_s(x)) = -P_t(x)\log P_s(x) \tag{6}$$

where $P_s(x)$ and $P_t(x)$ are respectively student and teacher model output distribution (*i.e.*after a softmax function).

In our data mixing case, there are two teachers the student needs to distill from. The first one is distilling from the source teacher:

$$\mathcal{L}_{\text{DINO}}^s = H(\lambda_i P_t(x_i') + (1-\lambda_i)P_t(x_{n-i}'), P_s(x_i^m)), \tag{7}$$

Based on the loss, we can minimize the distance between mixed data and two source data. Another teacher is the mix teacher which take the mixed data as input:

$$\begin{aligned}\mathcal{L}_{\text{DINO}}^m &= \frac{1}{1+\lambda_i^c}H(P_t(x_i'^m), P_s(x_i^m)),\\ &\quad +\frac{\lambda_i^c}{1+\lambda_i^c}H(P_t(x_{n-i}'^m), P_s(x_i^m))\end{aligned} \tag{8}$$

The student can learn the probability of the mixed data from the mixed teacher output. The two teachers share the same network parameter but different input data. In practice, the teacher is not pretrained but the exponential moving average update on the student weights.

To sum up, the total loss of SDMP + DINO is:

$$\mathcal{L} = \mathcal{L}_{\text{DINO}} + \mathcal{L}_{\text{DINO}}^m + \mathcal{L}_{\text{DINO}}^s \tag{9}$$

where $\mathcal{L}_{\text{DINO}}$ is the original DINO [7] loss applied to the teacher-student pairs without using any data mixing.

4. Experiment

Given no prior works successfully apply data mixing to improve self-supervised ViTs, we take ViT [16] as the major backbone in experiments. We set contrastive learning and knowledge distillation as self-supervised learning pretext tasks, and measure the representation quality of the pretrained models by linear evaluation, end-to-end finetuning, and semi-supervised learning. The ImageNet top-1 accuracy [12] is reported for performance comparisons. In addition, we evaluate model robustness on out-of-distribution benchmarks [24–26].

4.1. Implementation Details

We take a small Transformer, ViT-S [39], as the default architecture in our experiments. The input patch size of ViT-S is 16×16. Therefore, the sequence length is 196 for the 224×224 input images. There are 12 transformer blocks, and the dimension of each block is 384. For the data augmentation, we follow the settings in BYOL [19], which includes random resize crop, color jittering, Gaussian Blurring, and solarization. We use Adam with weight decay as the optimizer [33]. We set the learning rate lr following the linear scaling rule [18]: $lr = 0.0005 * batchsize/256$; the default training batch size is 1024.

4.2. Classification on ImageNet-1K

4.2.1 Linear Evaluation

Linear Evaluation is the standard protocol [7, 22] in evaluating the performance of self-supervised learning by freezing the parameter in the backbone network and training a linear classifier. Following the previous method [22], we only take resize, crop and random flipping as data augmentation. Due to the variance of feature space in different self-supervised methods, we take a different learning rate for different pretraining methods. Note that most self-supervised method only take the last class token for linear evaluation, but DINO take the last four class tokens. We will follow the default settings in their method. When we apply the proposed SDMP on MoCo, we randomly replace one of the symmetric views with our mixing data. When we apply SDMP on DINO, we randomly replace some local crops with our mixing data. Our data mixing strategy will not bring extra computation costs in the student model.

The results are reported in Table 1, our method is compatible with MoCo and DINO and brings consistent improvements under the same training epochs and crops.

Method	Model	Param.	Epoch	Top-1 (%)
SimCLR [8]	Res50	23M	200	60.6
BYOL [19]	Res50	23M	200	61.9
SwAV [6]	Res50	23M	800	75.3
MoCo v1 [21]	Res50	23M	200	60.6
MoCo v2 [9]	Res50	23M	800	71.1
MoCo v3 [11]	Res50	23M	300	72.8
+ i-mix [31]	Res50	23M	300	72.8
+ SDMP (ours)	Res50	23M	300	73.5
Supervised [39]	ViT-S	21M	300	79.8
BYOL [19]	ViT-S	21M	300	71.4
MoCo v2 [9]	ViT-S	21M	300	72.7
SwAV [6]	ViT-S	21M	300	73.5
MoCo v3 [11]	ViT-S	21M	300	73.2
+ imix* [31]	ViT-S	21M	300	71.6
+ DACL* [42]	ViT-S	21M	300	72.3
+ MoChi* [27]	ViT-S	21M	300	73.0
+ SDMP (ours)	ViT-S	21M	300	73.8
DINO [7]	ViT-S	21M	300	76.0
+ SDMP (ours)	ViT-S	21M	300	76.4
MoCo v3	ViT-B	85M	300	76.7
+ SDMP (ours)	ViT-B	85M	300	77.2

Table 1. Comparison with different pretraining methods in Linear evaluation on ImageNet-1K. Our method consistently brings improvements without using extra crops or views. * indicate our reproduced results with Transformer.

Specifically, we only replace the view or crops. Therefore, without using extra crops or views, we achieve 0.6% accuracy improvement over the vanilla MoCo v3 baseline (*i.e.*, 73.2% *vs*. 73.8%) and 0.4% accuracy improvement over DINO (*i.e.*, 76.0% *vs*. 76.4%). In contrast, for all other training strategies in data mixing, including i-mix, DACL and MoChi, they cannot improve the performance of ViT-S over the vanilla MoCo v3 baseline. Lastly, we verify that SDMP scales well with large ViTs, *i.e.*, it successfully help ViT-B beat the vanilla MoCo v3 baseline by 0.5% accuracy.

4.2.2 End-to-End Fintuning

We follow the same training recipe in DeiT [39] for finetuning models, including data augmentation and regularization like mixup [47], cutmix [45], random flipping, random cropping, and random erasing. The whole network will be end-to-end finetuned for 100 epochs.

As shown in Table 2, we compare different self-supervised methods and the supervised baseline. "Supervised" indicates random initialization, while the rest use the pre-trained models as initialization. All self-supervised methods are pretrained for 300 epochs. Compared with the 100-epoch supervised training from random initialization, our method significantly outperforms it by a large margin.

Method	Model	Param.	Epoch	Top-1
Supervised	ViT-S	21M	100	75.8
Supervised	ViT-S	21M	300	79.8
MoCo v3	ViT-S	21M	100	78.7
+ SDMP	ViT-S	21M	100	79.1
DINO	ViT-S	21M	100	79.7
+ SDMP	ViT-S	21M	100	80.0

Table 2. End-to-end fintuning on ImageNet-1K. All methods are pretrained for 300 epochs. The "Epoch" in the table indicates the number of fintuning epochs.

These finetuning results even closely match or outperform the performance of the 300-epoch supervised training setting (*i.e.*, 79.1% or 80.0% *vs*. 79.8%). Compared with the vanilla baseline, our proposed SDMP brings substantial and consistent improvements over the self-supervised learning methods (*i.e.*, MoCo v3 and DINO) under the same pretraining and finetuning setups.

4.2.3 Semi-Supervised Learning

In semi-supervised learning, we followed the same procedures of self-supervised pretraining on the whole ImageNet and supervised finetuning with 10% data and 1% data. Compared with the end-to-end finetuning, semi-supervised learning utilizes fewer labels. Therefore, stronger regularization and data augmentation techniques are required for obtaining a more generalized pretrained model.

The results are reported in Table 3, our model consistently outperforms the method without applying data mixing, and the performance gap is much larger than that in the end-to-end finetuning. Additionally, we observe that the performance gap increases when using less data for fintuning. Therefore, the model trained with our proposed SDMP can be regarded as a more generalized model. Specifically, for MoCo with and without our proposed SDMP, the performance gap increases by 0.4% with 100% data (end-to-end fintuning settings), 0.7% (10% data semi-supervised learning) and 1.1% (1% data semi-supervised learning).

Method	Model	Param.	10%	1%
MoCo v3	ViT-S	21M	66.7	54.4
+ SDMP	ViT-S	21M	67.4	55.5
DINO	ViT-S	21M	67.2	55.6
+ SDMP	ViT-S	21M	68.0	56.3

Table 3. Semi-supervised learning on ImageNet-1K with 10% and 1% labeled data. All methods are pretrained for 300 epochs.

Method	ImageNet (%)	A (%)	R (%)	C (mCE)
MoCo	78.7	18.1	42.1	52.9
+ SDMP	79.1	18.9	42.8	53.4
DINO	79.7	20.0	44.9	54.7
+ SDMP	79.9	21.1	45.3	55.0

Table 4. Performance on ImageNet and out-of-distribution datasets. "A", "R", "C" refer to ImageNet-A [26], ImageNet-R [24], and ImageNet-C [25], respectively. Note that when measuring performance on ImageNet-C, we directly use top-1 accuracy rather than "mCE" [25] as the evaluation metric.

4.2.4 Robustness on Out-of-Distribution Datasets

To evaluate the robustness of our approach against out-of-distribution data, we test the performance on perturbed versions of ImageNet, *i.e.*, natural adversarial examples (ImageNet-A [26]), semantic shifts (ImageNet-R [24]), common image corruption (ImageNet-C [25]).

As shown in Table 4, our method not only improves the standard classification performance on ImageNet but also consistently improves the performance on different Out-of-Distribution datasets compared with the baseline methods, *i.e.*, MoCo and DINO. Notably, the robustness improvement is even more significant than the standard ImageNet classification performance boost. For instance, applying our proposed SDMP to standard ImageNet classification only yields a performance improvement of 0.2%. But for ImageNet-A classification, our method outperforms DINO by 1.1%. This suggests that SDMP can be an excellent addition to existing self-supervised learning frameworks to improve model robustness.

4.3. ResNet Results

In addition to ImageNet, we also test our method on CIFAR-10 and CIFAR-100 [30] to verify the generalization of our proposed data mixing strategy. In this section, we explore the effectiveness using CNN architectures, *i.e.*, ResNet [23], to show that our method can be also compatible with different architectures. CIFAR-10 and CIFAR-100 contain 32×32 small size images with 10 and 100 classes, respectively. There are 50000 training images and 10000 testing images. Specifically, we pretrain ResNet-50 and ResNet-101 on CIFAR-10 and CIFAR-100, and ResNet-50 on ImageNet-1K respectively. If we apply ResNet on CIFAR-10 and CIFAR-100, the first convolution and max-pooling layer will be replaced by a convolution layer with the kernel size of 3×3 and stride of 1. We re-implement i-mix [31] on MoCo v3 for fair comparison.

ResNet on CIFAR-10. As shown in Table 5, we compare ResNet-50 and ResNet-101 pre-trained models based on MoCo with and without applying data mixing. Though both improve the baseline method, we note that our method consistently yields more significant performance improvements than i-mix. Specifically, with ResNet-50 and 200 epoch pretraining, i-mix yields performance improvement of 1.9% while our proposed SDMP substantially produces 2.9% performance improvement.

Method	Model	Epoch	CIFAR10	CIFAR100
MoCo v3	Res50	200	86.5	63.2
+ i-mix	Res50	200	88.6	66.1
+ SDMP	Res50	200	89.5	68.2
MoCo v3	Res50	2000	93.7	69.0
+ i-mix	Res50	2000	95.4	77.3
+ SDMP	Res50	2000	95.8	78.7
MoCo v3	Res101	200	86.4	63.3
+ i-mix	Res101	200	89.4	67.7
+ SDMP	Res101	200	90.0	69.7
MoCo v3	Res101	2000	93.8	68.5
+ i-mix	Res101	2000	95.8	78.4
+ SDMP	Res101	2000	95.8	80.0

Table 5. Linear evaluation on CIFAR-10 and CIFAR-100.

ResNet on CIFAR-100. Similar to the results in CIFAR-10, our method also consistently yields more significant performance improvement compared to i-mix for CIFAR-100, as shown in Table 5. However, because CIFAR-100 is a relatively smaller dataset (5000 training sample per class in CIFAR-10 and 500 training sample per class in CIFAR100) and ResNet-101 is the larger model, the performance of ResNet-101 slightly drops compared with ResNet-50 (pretraining for 2000 epochs). Therefore, stronger data augmentation like SDMP is required to help larger models perform better, and with the proposed SDMP, ResNet-101 outperforms ResNet-50 when pretraining for 2000 epochs. Besides, when training is longer, the improvement is more significant. Specifically, compared with baseline MoCo, the proposed SDMP improves 6.4% in pretraining 200 epoch but improve 11.5% in pretraining 2000 epochs, which also shows the regularization power of our method.

ResNet on ImageNet. Compared with CIFAR-10 and CIFAR-100, ImageNet is a much larger dataset. Therefore data augmentation plays a less important role in pretraining. As shown in Table 1, i-mix hardly improves ResNet result on the MoCo v3 baseline. In contrast, our SDMP encodes more accurate relationship cross data and successfully boost accuracy by 0.7%.

Method	λ_i	λ_i^c	Linear	Finetuning
MoCo	None	None	73.2	78.7
+ SDMP	Static	Rand.	71.5	78.0
+ SDMP	Rand.	Static	72.5	78.4
+ SDMP	Rand.	Rand.	73.7	79.1
DINO	None	None	76.0	79.7
+ SDMP	Static	Rand.	75.5	79.4
+ SDMP	Rand.	Static	76.1	79.9
+ SDMP	Rand.	Rand.	76.4	79.9

Table 6. Ablation on the mixing weight in the loss. "None" indicate that data mixing is not applied. "Rand." indicate that λ is randomly sampled, where both the data mixing and the mixing loss take the same weight λ. "Static" indicate the λ in loss is static, predefined and irrelevant with the weight in data mixing.

4.4. Ablation Study

4.4.1 On the importance of λ

When mixing two images, we sample λ from Beta distributions as the weight. In pretraining, we regard such λ as the prior in our mixing loss. In this part, we ablate how this prior λ setup affects model performance.

Static weight. There are two parts in the loss related to the data mixing and the λ: source loss and mixing loss. To verify λ as a useful prior, we manually opt out the prior when computing the mixing loss (referred to as "static weight"). Specifically, we only keep the randomly sampled λ in data mixing for training. But for the loss computation, we set $\lambda_i, 1-\lambda_i$ equals to 0.5 in the source loss and λ_i^c equals to 0 in the mix loss separately. all $\lambda_i, 1-\lambda_i$ and λ_i^c equal to 1.

As shown in Table 6, for the weight of source loss (see in Eq. (7) and Eq. (3)), the performance of both DINO and MoCo significantly drops under the linear evaluation protocol when setting a fixed weight for computing the mix loss. But DINO is relatively more stable (with much less performance degradation). On the other hand, for end-to-end finetuning, the performance drop for both MoCo and DINO decreases. This suggests that for the mixing loss, the importance of the loss prior has declined compared to that in linear evaluation. As a comparison, we also randomly sample λ and apply it both for reweighting the mixing data and the mixing loss (referred to as "random weight"). Specifically, we find that applying either a static weight or a random weight for the mixing loss have no difference in DINO with end-to-end finetuning. We think this might be attributed to the intra batch similarity, because the intra batch sample pair (the second term in Eq. (3) or Eq. (7)) we build share the same source data but only with the different mixing weight. Comparing MoCo and DINO, our proposed data mixing strategy SDMP is more stable in DINO if we adjust the weight of the mixing and the source loss.

Method	λ_i	λ_i^c	Linear	Finetuning
MoCo	None	None	73.2	78.7
+ SDMP	Shared	Shared	72.3	78.2
+ SDMP	Ind.	Ind.	73.7	79.1
DINO	None	None	76.0	79.7
+ SDMP	Shared	Shared	74.5	79.0
+ SDMP	Ind.	Ind.	76.4	79.9

Table 7. "Ind." and "Shared" indicate using independent (per-sample) and shared (per-batch) λ in a training batch.

Per-sample weight vs. per-batch weight assignment. In our current method, we assign each sample a mixup weight λ which provides more diversity in a batch and enlarges the training difficulty. In contrast, if we assign a shared mixup weight for a batch, namely, $\lambda_1 = \lambda_2 = ... = \lambda_n$ in a batch, the training difficulty will be reduced since the same mixing pattern is applied to the entire training batch. The results are reported in Table 7, taking a shared weight of mixing in a batch reduce the training difficulty but lead to 0.9% performance drop in the linear evaluation and 0.5% performance drop in end-to-end finetuning. This indicates assigning the per-sample weight is more effective than assigning the per-batch weight for mixing the data in our method.

4.4.2 Data Mixing Strategies

our default setup is to select a data mixing strategy from the set {Mixup, Cutmix and ResizeMix} uniformly at random. To ablate the effects of applying different data mixing strategies, we then compare our default setup with two additional settings: 1) applying exclusively with the element-wise data mixing Mixup; and 2) applying exclusively with the regional data mixing Resizemix. We report the results in Table 8. We note that: 1) our SDMP consistently outperforms the MoCo or DINO baseline, even if only one data mixing strategy is applied; 2) our SDMP achieves the best results when {Mixup, Cutmix and ResizeMix} are all used.

Method	Model	Mixing	Epoch	Top-1 (%)
MoCo v3	ViT-S	None	100	64.7
+ SDMP	ViT-S	Mixup	100	65.1
+ SDMP	ViT-S	Resizemix	100	65.4
+ SDMP	ViT-S	Both	100	65.5
DINO	ViT-S	None	100	73.8
+ SDMP	ViT-S	Mixup	100	74.3
+ SDMP	ViT-S	Resizemix	100	74.4
+ SDMP	ViT-S	Both	100	74.4

Table 8. Ablations of different data mixing strategies.

4.4.3 Extra Version and Replace Version

The original MoCo by default see two different augmented views of the same input. Our proposed SDMP generates the mixed data, which can be then used to replace one of the existing views in MoCo or form an extra view. Therefore for a fair comparison, *the number of training epochs should be the same as MoCo for the replace version but reduced to 2/3 for the extra version*, given there remains 2 views for original MoCo and the replace version, but increases to 3 views for the extra version. Table 9 reports the performance comparison between the replace version and the extra version. We can see under this fair comparison, both the replace and the extra version outperform the MoCo baseline.

Method	Model	Epoch	Top-1 (%)
MoCo v3	ViT-S	150	66.7
+ SDMP (Replace)	ViT-S	150	67.4
+ SDMP (Extra)	ViT-S	100	67.5

Table 9. Comparison of extra version and replace version.

4.4.4 Local Crops in Training

The most important contribution of DINO is the local crop in training. We explore the function of local crops in SDMP. When we apply SDMP on DINO with local crops, we replace the local crops by our mixed data. In contrast, for the non local crops version, we replace one of the global crops with our mixed data. The results are shown in Table 10, without local crops, the performance of our proposed SDMP significantly drops and is even lower than the original version of DINO in linear evaluation. With the number of local crops growing, SDMP narrows down the gap and finally outperforms the original DINO when the number of local crops reaches 6, which demonstrates the superiority of our proposed SDMP. Therefore, the local crops help stabilize the training when introducing the mixing data.

Method	Global Clean	Global Mixed	Local Clean	Local Mixed	Top-1 (%)
DINO	2	✗	✗	✗	67.8
DINO	1	1	✗	✗	64.1
DINO	2	✗	2	✗	71.5
DINO	2	✗	1	1	70.9
DINO	2	✗	6	✗	73.8
DINO	2	✗	3	3	74.0
DINO	2	✗	8	✗	74.0
DINO	2	✗	4	4	74.4

Table 10. Top-1 accuracy of different variants of multi-crop. We pretrain all models with 100 epochs. Global clean and Local clean indicate the global crops and local crops without data mixing. Global mixed and local mixed indicate the global crops and local crops with data mixing.

Method	Model	Param.	Epoch	Linear
Supervised	ViT-S	21M	300	88.0
MoCo v3	ViT-S	21M	300	79.1
+ SDMP	ViT-S	21M	300	81.8
DINO	ViT-S	21M	300	82.0
+ SDMP	ViT-S	21M	300	83.2

Table 11. Linear evaluation on ImageNet-100. Different from previous transfer learning methods which pretrain on large-scale datasets and finetune on small-scale datasets, we pretrain our model and perform linear evaluation on ImageNet-100.

4.4.5 Generalization on Small-Scale Datasets

Compared with ResNet, it is known that ViTs require much more data to train. In this part, we explore whether our method can still help self-supervised ViT on the relatively small dataset, *i.e.*, ImageNet-100. As shown in Table 11, we can observe that the proposed SDMP can consistently improve MoCo (from 79.1% to 81.8%) and DINO (from 82.0% to 83.2%), demonstrating its effectiveness at different data scale regime.

5. Conclusion

In this paper, we develop a generic training strategy in data mixing for helping self-supervised training, especially for Vision Transformers. By following the intra-batch data mixing strategy in timm [43], we propose SDMP to capture the intrinsic relationships between mixed data in a precise manner. Experiments show that our method brings consistent improvements, and is compatible with kinds of self-supervised learning methods, architectures, and datasets.

Discussion & Limitation This work introduces an extra intra-batch relationship between mixed samples for different self-supervised learning frameworks, *i.e.*, MoCo and DINO. Future work should examine how to integrate our method to other recent self-supervised masked image modeling methods, especially for the masked image modeling based methods [2, 20, 49]. In addition, due to computational limitation, our experiments are mostly built upon the small-sized ViT (*i.e.*, ViT-S); it deserves to further validate our method at a larger scale.

Acknowledgements This work is supported by a gift from Open Philanthropy, ONR N00014-21-1-2812 and Google Cloud Research Credits program.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019. 2

[2] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 8

[3] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007. 2

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 1

[5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 2

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 5

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 2, 4, 5

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2, 5

[9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2, 5

[10] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 2

[11] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021. 2, 5

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 4

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 2

[14] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2

[15] Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2051–2060, 2017. 2

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. 1, 2, 4

[17] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018. 2

[18] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 4

[19] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 2, 4, 5

[20] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 8

[21] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 2, 5

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 4

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[24] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *ICCV*, 2021. 4, 6

[25] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*, 2019. 4, 6

[26] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021. 4, 6

[27] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *arXiv preprint arXiv:2010.01028*, 2020. 1, 2, 5

[28] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pages 5275–5285. PMLR, 2020. 2

[29] Sungnyun Kim, Gihun Lee, Sangmin Bae, and Se-Young Yun. Mixco: Mix-up contrastive learning for visual representation. *arXiv preprint arXiv:2010.06300*, 2020. 2

[30] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 6

[31] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. i-mix: A domain-agnostic strategy for contrastive representation learning. In *ICLR*, 2021. 1, 2, 5, 6

[32] Chunyuan Li, Xiujun Li, Lei Zhang, Baolin Peng, Mingyuan Zhou, and Jianfeng Gao. Self-supervised pre-training with hard examples improves visual representations. *arXiv preprint arXiv:2012.13493*, 2020. 2

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019. 4

[34] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9359–9367, 2018. 2

[35] Jie Qin, Jiemin Fang, Qian Zhang, Wenyu Liu, Xingang Wang, and Xinggang Wang. Resizemix: Mixing data with preserved object information and true labels. *arXiv preprint arXiv:2012.11101*, 2020. 2, 3

[36] Zhiqiang Shen, Zechun Liu, Zhuang Liu, Marios Savvides, Trevor Darrell, and Eric Xing. Un-mix: Rethinking image mixtures for unsupervised visual representation learning. *arXiv preprint arXiv:2003.05438*, 2020. 2

[37] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 2

[38] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *Asian Conference on Machine Learning*, 2018. 1

[39] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 1, 2, 3, 4, 5

[40] AFM Uddin, Mst Monira, Wheemyung Shin, TaeChoong Chung, Sung-Ho Bae, et al. Saliencymix: A saliency guided data augmentation strategy for better regularization. *arXiv preprint arXiv:2006.01791*, 2020. 2

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 2

[42] Vikas Verma, Thang Luong, Kenji Kawaguchi, Hieu Pham, and Quoc Le. Towards domain-agnostic contrastive learning. In *International Conference on Machine Learning*, pages 10530–10541. PMLR, 2021. 1, 2, 5

[43] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 1, 2, 3, 8

[44] Tianshu Xie, Xuan Cheng, Xiaomin Wang, Minghui Liu, Jiali Deng, Tao Zhou, and Ming Liu. Cut-thumbnail: A novel data augmentation for convolutional neural network. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 1627–1635, 2021. 1

[45] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *ICCV*, pages 6023–6032, 2019. 1, 2, 3, 5

[46] Xiaohang Zhan, Jiahao Xie, Ziwei Liu, Yew-Soon Ong, and Chen Change Loy. Online deep clustering for unsupervised representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6688–6697, 2020. 2

[47] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 1, 2, 5

[48] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016. 2

[49] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2021. 8