Un-Mix: Rethinking Image Mixtures for Unsupervised Visual Representation Learning

Zhiqiang Shen [1] **Zechun Liu** [1] **Zhuang Liu** [2] **Marios Savvides** [1] **Trevor Darrell** [2] **Eric Xing** [1]
[1]CMU [2]UC Berkeley

Abstract

In supervised learning, smoothing label or prediction distribution in neural network training has been proven useful in preventing the model from being over-confident, and is crucial for learning more robust visual representations. This observation motivates us to explore ways to make predictions flattened in unsupervised learning. Considering that human-annotated labels are not adopted in unsupervised learning, we introduce a straightforward approach to perturb input image space in order to soften the output prediction space indirectly, meanwhile, assigning new label values in the unsupervised frameworks accordingly. Despite its conceptual simplicity, we show empirically that with the simple solution – **Un**supervised image **mix**tures (Un-Mix), we can learn more robust visual representations from the transformed input. Extensive experiments are conducted on CIFAR-10, CIFAR-100, STL-10, Tiny ImageNet and standard ImageNet with popular unsupervised methods SimCLR, BYOL, MoCo V1&V2, *etc*. Our proposed image mixture and label assignment strategy can obtain consistent improvement by 1∼3% following exactly the same hyperparameters and training procedures of the base methods.

1. Introduction

In recent years, unsupervised visual representation learning has attracted increasing attention (Noroozi & Favaro, 2016; Zhang et al., 2016; Oord et al., 2018; Hjelm et al., 2018; Gidaris et al., 2018; Wu et al., 2018; He et al., 2019; Misra & van der Maaten, 2019; Tian et al., 2019; Chen et al., 2020a; Kim et al., 2020; Grill et al., 2020; Caron et al., 2020; Kalantidis et al., 2020) due to its enormous potential of being free from human-annotated supervision, *i.e.*, its extraordinary capability of leveraging the boundless unlabeled data. Previous studies in this field address this problem mainly in two directions: one is realized via a heuristic *pretext* task design that applies a transformation to the input image, such as colorization (Zhang et al., 2016), rotation (Gidaris et al., 2018), jigsaw (Noroozi & Favaro, 2016), *etc.*, and the corresponding labels are derived from the properties of the transformation on the unlabeled data. Another direction is contrastive learning based approaches (He et al., 2019; Chen et al., 2020a) in the latent feature space, such as maximizing mutual information between different views (Bachman et al., 2019; Tian et al., 2019), momentum contrast learning (He et al., 2019; Chen et al., 2020b) with instance discrimination task (Wu et al., 2018; Ye et al., 2019), larger batch sizes and nonlinear transformation (Chen et al., 2020a), symmetrized distance loss without negative pairs (Grill et al., 2020), contrasting cluster assignment (Caron et al., 2020). These methods have recently shown great promise for this task, achieving state-of-the-art accuracy.

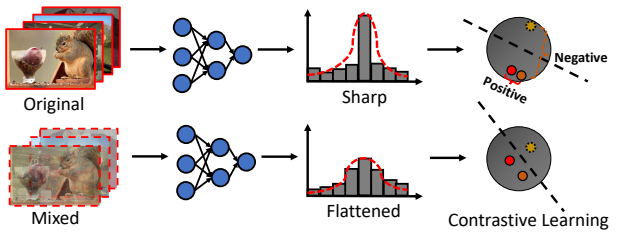

Figure 1. Illustration of our work's motivation. We take the contrastive-based unsupervised learning approaches as an example. Contrastive learning measures the similarity of sample pairs in the latent representation space. With *flattened prediction/label*, the model is encouraged to treat each incorrect instance as equally probable, which will smooth decision boundaries and prevent the learner from becoming over-confident.

The motivation of our work stems from some simple observations of *label smoothing* in supervised learning (Szegedy et al., 2016). Interestingly, we observed from visualizations of previous literature (Müller et al., 2019) that *label smoothing* tends to force the output prediction of networks being less confident (*i.e.*, lower maximum probability of predictions) but the overall accuracy can increase significantly. The explanation for this seemingly contradictory phenomenon is that with *label smoothing*, the learner is encouraged to treat each incorrect instance/class as equally probable. Thus more structure is enforced in latent representations, enabling less variation across predicted instances and/or across samples. This will further prevent the network

from overfitting to the training data. Otherwise, the network can often output incorrect and confident predictions when evaluated on slightly different test samples. Considering that contrastive learning essentially is classifying positive congruent and negative incongruent pairs with cross-entropy loss, such an observation reveals that a typical contrastive-based method can also encounter the over-confidence problem as in supervised learning.

The perspective of input and label spaces on un-/self-supervised learning. Contrastive learning methods adopt instance classification pretext, the features from different transformations (data augmentation) of the same images are compared directly to each other. The label of each pair images is binary (positive or negative) or continuous distance metrics. Augmentation is used as a transformation to make distance of the same image to be larger. Different from data augmentation that enlarges the dissimilar distance but the label for calculating loss is still unchanged, our proposed mixtures will manipulate the semantic distance between two images, while adjusting the label for unsupervised loss accordingly. In other words, **data augmentation only changes the distance of input space**, *i.e.*, heavier data augmentation makes two images look more different, **but remains unchanged in label space** during training. However, mixture will manipulate both input and label spaces simultaneously and the degree of change is controllable, which can further help capture the fine-grained representations from the unlabeled images and obtain models with more precise and smoother decision boundaries at latent features. As a result, neural networks trained with new spaces learn flatter class-agnostic representations, that is, with fewer directions of variance. The mechanism of image mixtures in unsupervised learning is generally different from the data augmentation. From this perspective, mixtures can be considered as a broader concept of data augmentation scheme in unsupervised learning.

We choose four recently proposed unsupervised methods: SimCLR (Chen et al., 2020a), MoCo V1&V2 (He et al., 2019; Chen et al., 2020b), BYOL (Grill et al., 2020) and Whitening (Ermolov et al., 2020) as our baseline approaches. We conduct extensive experiments on CIFAR-10, CIFAR-100, STL-10, Tiny/standard ImageNet classification, as well as downstream object detection task on PASCAL VOC and COCO to demonstrate the effectiveness of our proposed approach. We observe that our mixture learned representations are extraordinarily effective for the downstream detection task. For example, our 200-epoch trained model outperforms the baseline MoCo V2 by 0.6% ($\mathbf{AP}_{50}$), and is even better than the MoCo V2 800-epoch model.

Our contributions are summarized as follows:

- We provide empirical analysis to reveal the fact that *mixing input images and smoothing labels* could improve performance favorably for a variety of unsupervised learning methods. We applied two simple image mixture methods based on previous literature (Zhang et al., 2018; Yun et al., 2019) to encourage neural networks to predict less confidently.
- We show that input and label spaces matter. We provide evidence on why this flattening happens under ideal conditions of latent space, validate it empirically on practical situations of contrastive learning, and connect it to previous works on analyzing the discipline inside the unsupervised learning behavior. We explain the difficulties that arise with original image space when visualizing these trajectories of predictions. Thus we conclude that **good input and label spaces** are crucial for unsupervised optimization.
- Our proposed method is simple, flexible and universal. It can be utilized to nearly all mainstream unsupervised learning methods and only requires $\sim$**10 lines of PyTorch codes to incorporate in an existing framework**. We demonstrate our strategy with a variety of base approaches and datasets, including SimCLR, BYOL, MoCo V1&V2, *etc.*, on CIFAR-10, CIFAR-100, STL-10, Tiny ImageNet and standard ImageNet. Our method obtains consistent accuracy improvement by 1$\sim$3% across them. Code is available at: `https://github.com/szq0214/Un-Mix`.

2. Related Work

Un/Self-supervised Visual Feature Learning. Unsupervised learning aims to exploit the internal distributions of data and learns a representation without human-annotated labels. To achieve this purpose, early works mainly focused on reconstructing images from a latent representation, such as autoencoders (Vincent et al., 2008; 2010; Masci et al., 2011), sparse coding (Olshausen & Field, 1996), adversarial learning (Goodfellow et al., 2014; Donahue et al., 2016; Donahue & Simonyan, 2019). After that, more and more studies tried to design handcrafted pretext tasks such as image colorization (Zhang et al., 2016; 2017), solving jigsaw puzzles (Noroozi & Favaro, 2016), counting visual primitives (Noroozi et al., 2017), rotation prediction (Gidaris et al., 2018). Recently, contrastive based visual representation learning (Hadsell et al., 2006) has attracted much attention and achieved promising results. For example, Oord et al. (Oord et al., 2018) proposed to use autoregressive models to predict the future samples in latent space with probabilistic contrastive loss. Wu et al. (Wu et al., 2018) proposed a non-parametric memory bank to store the instance representation for tackling the computational issue. Hjelm et al. (Hjelm et al., 2018) proposed to maximize mutual information from the encoder between inputs and outputs of a deep network. Bachman et al. (Bachman et al., 2019) further extended this idea to multiple views of a shared context. Moreover, He et al. (He et al., 2019)

proposed to adopt momentum contrast to update the models and Misra&Maaten (Misra & van der Maaten, 2019) developed the pretext-invariant representation learning strategy that learns invariant representations from the pre-designed pretext tasks.

Smoothing Label/Prediction in Supervised Learning. Explicit label smoothing has been adopted successfully to improve the performance of deep neural models across a wide range of tasks, including image classification (Szegedy et al., 2016), object detection (Krothapalli & Abbott, 2020), machine translation (Vaswani et al., 2017), and speech recognition (Chorowski & Jaitly, 2016). Moreover, motivated by mixup, Verma et al. proposed to implicitly interpolate hidden states as a regularizer that encourages neural networks to predict less confidently (softer prediction) on interpolations of hidden representations. They found that neural networks trained with this kind of operation can learn flatter class representations which possess better generalization, as well as better robustness to novel deformations and even adversarial examples in testing data. Some recent work (Müller et al., 2019) further demonstrated that label smoothing implicitly calibrates the prediction of learned networks, so that the confidence of their outputs are more aligned with the true labels of the trained dataset. However, all of these studies lie in supervised learning. To the best of our knowledge, there is no existing work focusing on smoothing predictions for unsupervised learning and this is the first exploration in this direction.

3. Image Mixture Strategies

We start by introducing two atomic mixture operations before explaining ways to use them in the various unsupervised frameworks in Sec. 3.2. Many successful image mixture methods in supervised learning scheme have been proposed (Zhang et al., 2018; Yun et al., 2019) where they build upon the global/region-level images of explicit supervision.

3.1. Basic Mixture Operations

Mixup (Zhang et al., 2018) is a popular mixture method for pixel-wisely obtaining the weighted combination of two global images. All information of two input images will be reserved in the mixed sample, but the mixture will be visually misty, as shown in Fig. 2 (top).

$$I_{g\_m} \leftarrow \alpha I_1 + \beta I_2 \tag{1}$$

where $\{I_1, I_2\}$ denote the images that we want to mix and I_{g_m} is the output mixture. $\alpha, \beta \in [0, 1]$ are mixture coefficients and are restricted to $\alpha + \beta = 1$.

Cutmix (Yun et al., 2019) is the way to generate a new training sample by locally combining two samples of their region and whole image. Different from Mixup, this operation will replace pixels within particular locations of a region, as shown in Fig. 2 (bottom).

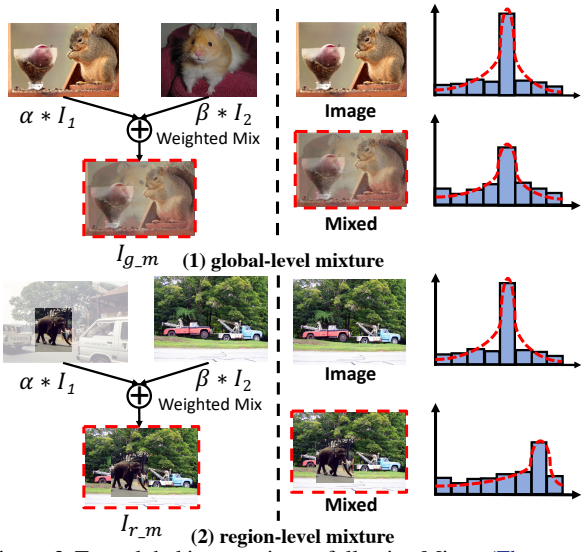

Figure 2. Top: global image mixture following Mixup (Zhang et al., 2018). Bottom: region-level mixture by adopting Cutmix (Yun et al., 2019). Since the input space has been changed by the mixture operations, the corresponding predictions will also be less confident in unsupervised frameworks, as illustrated on the right.

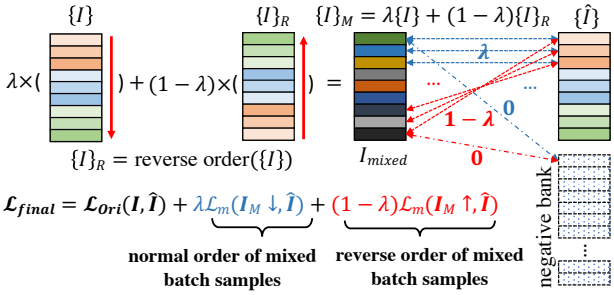

Figure 3. Illustration of self-mixture within a *mini*-batch. In each iteration, we randomly choose one mixture operation for all the current samples with a pre-defined probability P, thus the formulation of λ depends on the mixture type we choose.

$$I_{r\_m} \leftarrow \mathbf{M}_b \odot I_1 + (\mathbf{1} - \mathbf{M}_b) \odot I_2 \tag{2}$$

where $\mathbf{M}_b \in \{0, 1\}^I$ denotes a binary mask as defined in (Yun et al., 2019). $\mathbf{1}$ is a binary mask with all values equaling one. $\odot$ denotes element-wise multiplication.

Both Mixup and Cutmix can be regarded as the regularization techniques to prevent the models from being overfitting and make the predictions less confident.

3.2. Self-Mixtures within Per *Mini*-Batch of Training

In our method, the mixing ratio of two images is the core information that we would like to utilize in the unsupervised methods. Here we introduce a simple strategy to retain such information for loss calculation. We propose to mix the first image with the last one in each *mini*-batch of training, the second one is mixed with the penultimate, and so on. Our

strategy is visualized in Fig. 3, the advantages of such strategy are: (i) Different from employing individual ratio for each image in one *mini*-batch, the proposed scheme can be realized through calculating the batch loss with a weighted coefficient, which is well-regulated, controllable, more efficient during training and can facilitate the design of label assignment in unsupervised frameworks. (ii) As we will introduce below, the proposed method will make the distances between the mixtures and the origins to be consistent in each *mini*-batch. Hence, the calculation rule of loss function will be independent from the different frameworks that are used. For example, contrastive learning (consisting of positive and negative pairs) or positive only, using memory bank or without it, *etc.*, since scaling similarity distance is equivalent to weighting the loss value. We demonstrate that our method only requires $\sim$10 lines of PyTorch codes to incorporate into all of these frameworks, and further delivers significant improvement over these base approaches.

4. Label Assignment/Smoothing in the Unsupervised Frameworks

In this section, we first present different paradigms using mixtures in the unsupervised learning framework, including mixing both two branches and a single one. Then we discuss the circumstances that contain a memory bank or not, and the loss functions for different scenarios. Lastly, we provide a simple demonstration of how to apply multi-scale training for further boosting the performance.

4.1. Paradigms of Mixtures

Given an image I, we first augment it to two transformed views I_A and $\hat{I}_A$ by applying a pre-defined random transformation. The proposed mixtures are following the image transformations of the input sample. We define I_A^M and $\hat{I}_A^M$ as the mixed images which can be $\{I_{g\_m}, I_{r\_m}\}$ according to the type of mixture operation we choose in the current training iteration. The mixed images are forwarded to the target network f_θ, then a non-linear projection head p_θ is adopted to obtain the representations of the input sample for the unsupervised distance loss. In the following, we discuss two circumstances in such a framework, as shown in Fig. 4.

Both I_A and $\hat{I}_A$ are mixed (Fig. 4 (2)). This solution is to mix both the two views of an input image. Thus, the similarity between mixtures will remain unchanged. While such dynamic mixtures in both branches still admit undesirable equilibria, we found this strategy is effective on the relatively small datasets like CIFAR, STL-10, *etc.*, but does not help significantly on the large-scale ImageNet.

Only I_A is mixed (Fig. 4 (3)). This is the main strategy that we use in this work. Compared to the one above, this solution is more efficient since it only needs one additional branch of forward-propagation (two times for the normal and reverse orders of mixed batch samples). From our experimental results it is also more effective for obtaining performance gain over the baseline approaches.

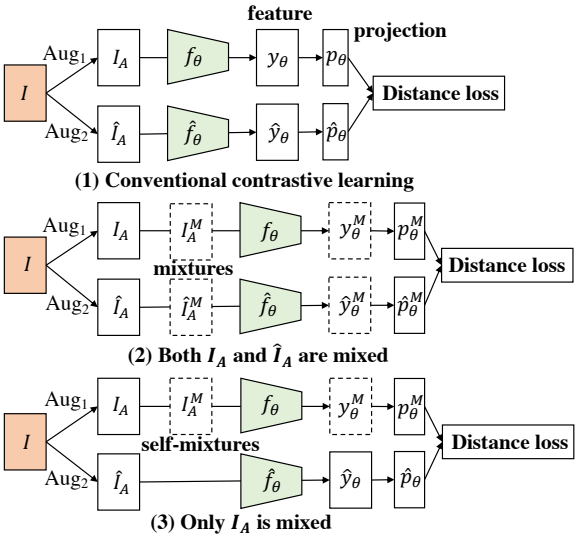

Figure 4. Comparison of different paradigms of utilizing mixtures in unsupervised learning. (1) is the conventional instance classification based framework. (2) and (3) are the strategies of applying image mixtures. "self-mixtures" denotes that the images of mixture operation only happens in current batch samples. The dashed bounding box represents the mixed image and its representation.

4.2. Dealing with Memory Banks

Our goal is to enhance visual features by leveraging additional mixture information and the different mixing ratios between two images in the unsupervised scheme. To this end, we propose a way to re-measure the distances of positive and negative pairs for unsupervised methods.

Without a memory bank. Under this circumstance, the unsupervised frameworks will be a positive pair only (*i.e.*, BYOL (Grill et al., 2020)) or contrastive-based (*i.e.*, SimCLR (Chen et al., 2020a)) pipelines. Within them, we still keep the distance of negative pairs as zero if there are, even they are one original and one mixed image. Therefore, we only need to design the new distance of the positive pairs, as shown in Fig. 6. In our self-mixture strategy, the new distance of a positive pair will be:

$$\begin{aligned} I_A^M &= \lambda I_1 + (1-\lambda) I_2, \\ \mathrm{dis}\left(I_A^M, \hat{I}_A\right) &= \begin{cases} \lambda & \text{if } \hat{I}_A = \hat{I}_1, \\ 1-\lambda & \text{if } \hat{I}_A = \hat{I}_2. \end{cases} \end{aligned} \tag{3}$$

where $\hat{I}_1$, $\hat{I}_2$ are another views of I_1, I_2 from the same images. In the conventional unsupervised scheme, they are a positive pair. λ is the mix ratio controlled by the type of mixture we use in the current iteration of training, if it is global, $\lambda = \alpha$ as in Eq. 1, otherwise, $\lambda = \frac{\mathbf{M}_b}{\mathbf{1}}$ as in Eq. 2.

With a memory bank. Using a memory bank solely affect the constitution of negative pairs as the distance/label be-

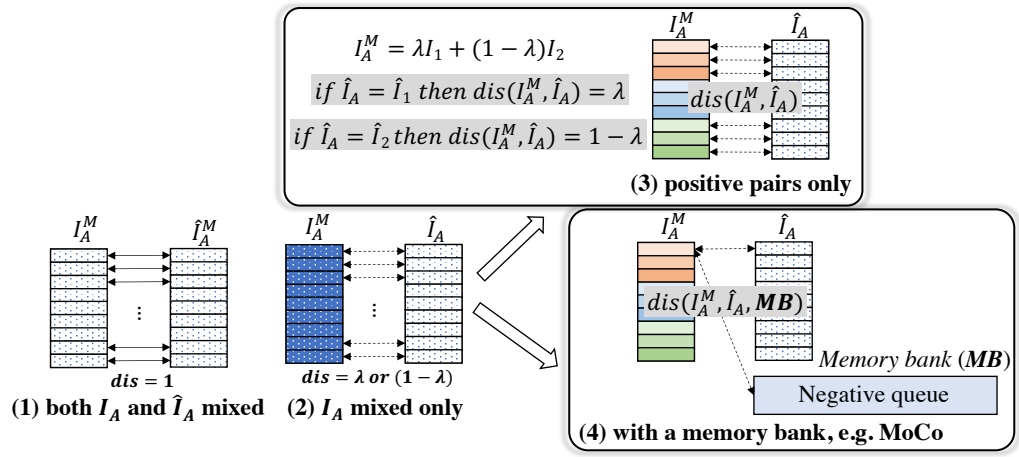

Figure 5. Illustration of our strategy when only I_A is mixed. We consider the unsupervised frameworks (i) contain a memory bank or not; (ii) positive pairs only or with negative pairs.

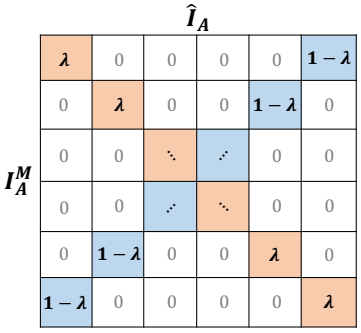

Figure 6. Distance matrix of proposed mixture strategy between the mixed I_A (*i.e.*, I_A^M) and $\hat{I}_A$ for calculating the distance loss. Here we take six images in the *mini*-batch as an example.

tween them is always zero in instance classification based contrastive learning. In particular, negative pairs can be $\{$*origin, origin*$\}$, $\{$*origin, mixture*$\}$, $\{$*mixture, mixture*$\}$. We found in experiments that maintaining one bank with the representations from original/unmixed images is enough to obtain good performance. However, this is inapplicable in the multi-scale training scheme, as we will introduce later.

4.3. Loss Functions

We now introduce the loss functions. We compute an extra loss from a mixed pair of the image. Given two mixed images I_A^M and $\hat{I}_A^M$ from two different augmentations of the same image (the case that both are mixed), we compute their loss together with the original one as the following:

$$\mathcal{L}_{both} = \mathcal{L}_{ori}(I_A, \hat{I}_A) + \underbrace{\mathcal{L}_m(I_A^M, \hat{I}_A^M)}_{\text{extra term of mixtures}} \tag{4}$$

where $\mathcal{L}_{ori}$ is the original loss function and $\mathcal{L}_m$ measures the fit between samples I_A^M and $\hat{I}_A^M$.

Finally we define the following sum of three loss terms from the original and mixed predictions as the ultimate objective:

$$\mathcal{L}_{ours} = \mathcal{L}_{ori} + \underbrace{\lambda \mathcal{L}_m(I_A^M(\downarrow), \hat{I}_A)}_{\text{normal order of mixtures}} + \underbrace{(1-\lambda)\mathcal{L}_m(I_A^M(\uparrow), \hat{I}_A)}_{\text{reverse order of mixtures}} \tag{5}$$

where the last two loss terms measure the fit between samples I_A^M and $\hat{I}_A$, as detailed above.

Enabling Multi-scale Training. Inspired by the prior study (Caron et al., 2020) that comparing random crops of an image is crucial for the networks to capture information of patterns from scenes or objects, we explore the possibility of employing multi-scale training in our mixture framework. Different from (Caron et al., 2020), we discuss the circumstance of multi-scale training where the framework contains a memory bank. In such a framework, the input sizes of multi-scale training are different for a network, but after flowing forward through an MLP projection head, the latent features will be of the same dimension. As memory bank is used to increase the number of negative pairs, so that we can share a single memory bank for all the scales of inputs to generate negative pairs. Unfortunately, we found that this sharing strategy leads to poor performance of representation ability which is incongruous with that we share the memory bank for original and mixed images above in the single-scale scenario. We conjecture that, in the multi-scale situation, such negative pairs are far away from the decision boundary so they are actually the "easy" negative samples. Adopting individual memory banks for each input size of samples is proven a better choice in our experiments. As shown in Fig. 10, the final loss is aggregated from all the scales of training samples.

5. Experiments

We demonstrate the effectiveness of our learned representation after unsupervised pretraining on a variety of datasets. We first evaluate it with a large number of baselines in linear evaluation. We then measure its transfer capability using

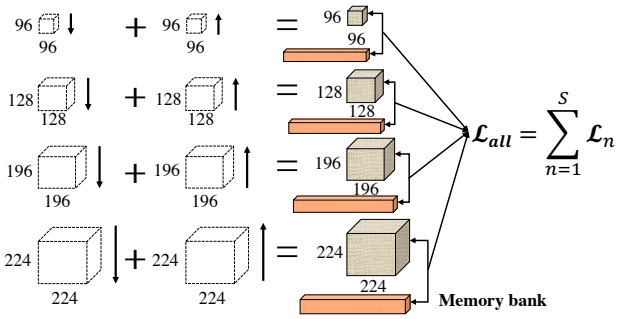

Figure 7. Illustration of the proposed multi-scale training strategy. We aggregate losses from all of the different scales of inputs.

object detection task on PASCAL VOC and COCO.

5.1. Datasets

CIFAR-10/100 (Krizhevsky et al., 2009). The two CIFAR datasets consist of tiny colored natural images with a size of 32×32. CIFAR-10 and 100 are drawn from 10 and 100 classes, respectively. In each dataset, the train and test sets contain 50,000 and 10,000 images.

STL-10 (Coates et al., 2011). STL-10 is inspired by CIFAR-10 with 10 classes, but each class has fewer labeled training examples (500 training images and 800 test images per class). The size of images is 96×96. It also contains 100,000 unlabeled images for unsupervised learning.

Tiny ImageNet. Tiny ImageNet is a lite and small version of ImageNet which contains 200 classes with images resized down to 64×64. The train and test sets contain 100,000 and 10,000 images, respectively.

ImageNet (Deng et al., 2009). The ImageNet, aka ILSVRC 2012 classification dataset (Deng et al., 2009) consists of 1000 classes, with a number of 1.28 million training images and 50,000 validation images.

5.2. Baseline Approaches

We perform our evaluation of image mixtures and label assignment strategy on the following four recently proposed unsupervised methods with state-of-the-art performance:

MoCo V1&V2 (He et al., 2019; Chen et al., 2020b) MoCo is a contrastive learning method using momentum updating for unsupervised visual representation learning. MoCo V2 further improves momentum contrastive learning by adopting an MLP projection head and more data augmentation from the following SimCLR (Chen et al., 2020a).

SimCLR (Chen et al., 2020a). SimCLR is a simple framework for contrastive learning without requiring specialized architectures or a memory bank. It introduces a learnable nonlinear transformation which substantially improves the quality of the learned representations.

BYOL (Grill et al., 2020). BYOL adopts online and target networks that learn from each other. It trains the online network to predict the target network representation of the same image under a different augmented view. At the same time, they update the target network with a slow-moving average of the online network without the negative pairs.

Whitening (Ermolov et al., 2020). Whitening is a new proposed loss function for unsupervised representation learning which is based on the whitening of the latent space features. The whitening operation has a scattering effect to avoid degenerate solutions of collapsing to a simple status.

Our baseline approach implementations follow the official codebases which are all publicly available[1][2][3].

5.3. Implementation Details in Pre-training

The goal of our experiments is to demonstrate the effectiveness of our proposed image mixture and label assignment upon various unsupervised learning frameworks, isolating the effects of other settings, such as the architectural choices, data augmentations, hyper-parameters. As this, we use the same encoder ResNet-18 for all non-ImageNet experiments and ResNet-50 for ImageNet. We use exactly the same training settings, hyper-parameters, *etc.*, as our comparisons. Therefore, all gains in this paper are "minima", and further tuning the hyper-parameters in the baseline approaches to fit our mixture strategies might achieve bigger improvement, while it is not the focus of this work.

Non-ImageNet Datasets. Following (Ermolov et al., 2020), on CIFAR-10 and CIFAR-100, we train for 1,000 epochs with learning rate 3×10^{-3}; on Tiny ImageNet, 1,000 epochs with learning rate 2×10^{-3}; on STL-10, 2,000 epochs with learning rate 2×10^{-3}. We also apply warm-up for the first 500 iterations, and a 0.2 learning rate drop at 50 and 25 epochs before the end.

Standard ImageNet. Unless otherwise stated, all the hyper-parameter configurations strictly follow the baseline MoCo V2 on ImageNet. For example, We use a *mini*-batch size of 256 with 8 GPUs on ImageNet, considering our primary objective is to verify the effectiveness of our proposed method rather than to suppress state-of-the-art results.

For image mixtures and label assignment, we use $\gamma = 1.0$ in beta sampling for all experiments, and $P = 0.5$ for non-ImageNet and 0 for ImageNet based on our ablation study.

5.4. Linear Classification

Our linear classification experiments consist of two parts: (i) ablation studies on small datasets including CIFAR-10, CIFAR-100, STL-10 and Tiny ImageNet with various base approaches to explore the optimal mixture hyperparameters and demonstrate the effectiveness of our strategy; (ii) the

[1] https://github.com/facebookresearch/moco.
[2] https://colab.research.google.com/github/facebookresearch/moco/blob/colab-notebook/colab/moco_cifar10_demo.ipynb.
[3] https://github.com/htdt/self-supervised.

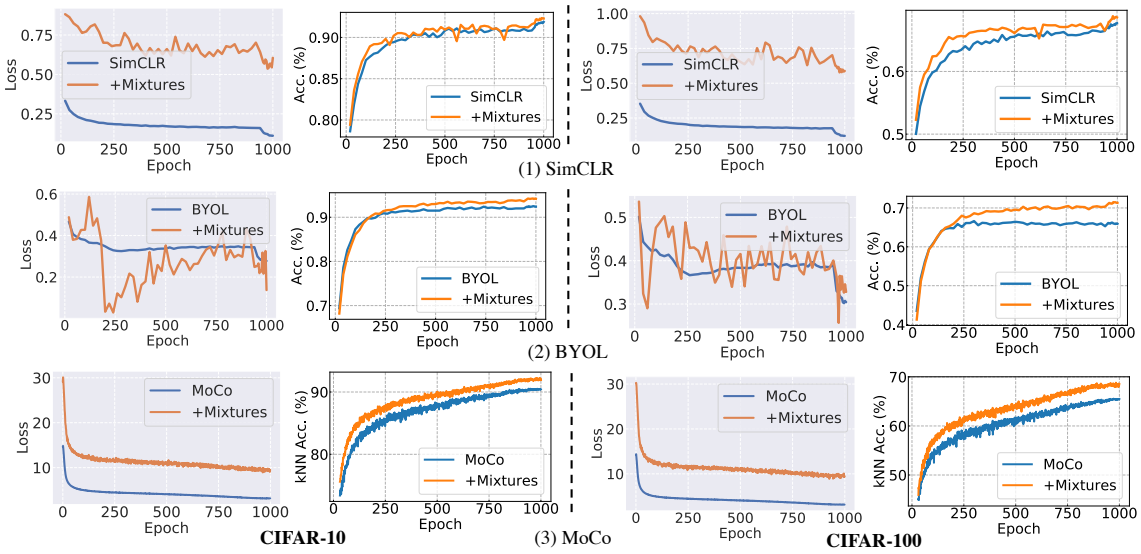

Figure 8. The curves of training loss and testing accuracy of SimCLR, BYOL and MoCo on CIFAR-10/100 datasets.

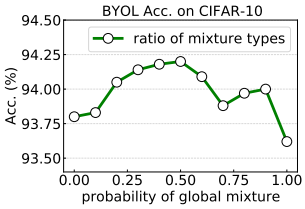

Figure 9. Acc. with various P.

Table 1. Sensitivity for γ.

CIFAR-10	γ in **Beta** sampling		
	1.0	0.8	0.5
Acc. (%)	**94.20**	94.12	93.93

Table 2. Sensitivity for P.

ImageNet	P of global mixture		
	1.0	0.5	0.0
Acc. (%)	67.6	68.3	**68.6**

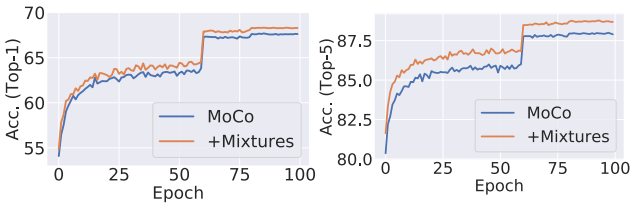

Figure 10. Linear classification accuracy of Top-1 (left) and Top-5 (right) with MoCo V2 and ours on ImageNet dataset.

final results on the standard ImageNet using MoCo V2.

Ablation Study. We investigate the following aspects in our methods: (i) the probability P between global and region mixtures; (ii) sensibility of γ in beta distribution sampling.

(1) Probability P for choosing global or region-level mixtures in each iteration. The result are shown in Fig. 9 (non-ImageNet) and Tab. 2 (ImageNet). It shows that $P = 0.5$ is optimal for small datasets and choosing region-level only (*i.e.*, $P = 0$) is best for large-scale ImageNet.

(2) Beta distribution hyperparameter γ. The combination ratio λ between two sample points is sampled from the beta distribution **Beta**(γ, γ). Our results on different γs are presented in Tab. 1, $\gamma = 1.0$ is the best and we use it for all our experiments, which means that λ is sampled from a uniform distribution $[0, 1]$.

Results on CIFAR-10/100, STL-10 and Tiny ImageNet. Our results are shown in Tab. 3 and Fig. 8. All experiments are conducted on a single scale since the input sizes of these datasets are small. Our method obtains consistency of $1\sim3\%$ gains. In particular, our loss values are usually larger than the baselines (except BYOL, which is unstable since it has no negative pairs), but the accuracy is still superior.

Results on IamgeNet with MoCo V2. As shown in Tab. 4, our method obtain 1.1% improvement than baseline MoCo V2. Employing multi-scale training can further boost accuracy by 2.3%. It is possible that tuning the hyperparameters in MoCo V2 like temperature to fit our mixed training samples has the potential to further improve the performance.

5.5. Downstream Tasks

In this section, we evaluate the transferability of our learned representation on the object detection task. We use PASCAL VOC (Everingham et al., 2010) and COCO (Lin et al., 2014) as our benchmarks and we exactly follow the same setups and hyperparameters of the prior works (He et al., 2019; Wu et al., 2018; Misra & van der Maaten, 2019) on the transfer learning stage. We use Faster R-CNN (Ren et al., 2015) and Mask R-CNN (He et al., 2017) implemented in Detectron2 (Wu et al., 2019) with a ResNet-50[4] backbone.

PASCAL VOC. We fine-tune our models on the split of `trainval07+12` and evaluate on the VOC `test2007` following (Wu et al., 2018; He et al., 2019; Misra & van der Maaten, 2019). It can be observed that significant improvements are consistently obtained by our proposed mixtures.

COCO. We fine-tune on the `train2017` and evaluate on the `val2017` split. The total training budget is 180K iterations. The whole schedule follows the Detectron2 (`coco_R_50_C4_2x`) default setting. Our results are shown in Tab. 5 (b), it can be observed that our results are consistently better than the baseline by a significant margin.

[4] https://github.com/pytorch/vision.

Table 3. Linear and 5-nearest neighbors classification results for different loss functions and datasets with a ResNet-18 backbone. Table is adapted from (Ermolov et al., 2020) for strictly fair comparisons. Note that MoCo is trained with symmetric loss, 1000 epochs and evaluated with 200 in kNN monitor* following `https://github.com/facebookresearch/moco` of CIFAR datasets.

Method	CIFAR-10				CIFAR-100				STL-10				Tiny ImageNet			
	linear	**ours**	5-nn	**ours**	linear	**ours**	5-nn	**ours**	linear	**ours**	5-nn	**ours**	linear	**ours**	5-nn	**ours**
SimCLR (Chen et al., 2020a)	91.80	**92.35**	88.42	**89.74**	66.83	**68.83**	56.56	**58.82**	90.51	**90.86**	85.68	**86.16**	48.84	**49.58**	32.86	**34.46**
BYOL (Grill et al., 2020)	91.73	**94.20**	89.45	**93.03**	66.60	**71.50**	56.82	**63.83**	91.99	**93.34**	88.64	**90.46**	51.00	**53.39**	36.24	**39.27**
Whitening (W = 2) (Ermolov et al., 2020)	91.55	**93.04**	89.69	**91.33**	66.10	**70.12**	56.69	**61.28**	90.36	**92.21**	87.10	**88.88**	48.20	**51.33**	34.16	**36.78**
Whitening (W = 4) (Ermolov et al., 2020)	91.99	**93.18**	89.87	**91.70**	67.64	**69.70**	56.45	**60.74**	91.75	**91.96**	88.59	**88.71**	49.22	**50.67**	35.44	**36.13**
MoCo (Symmetric Loss) (He et al., 2019)	–	–	90.49*	**92.25***	–	–	65.49*	**68.83***	–	–	–	–	–	–	–	–

Arch.	Method	#Params	Budget (#ep)	Top-1 Acc. (%)
R50	MoCo	24	200	60.6
R50	CMC	24	200	66.2
R50	SimCLR	24	200	66.6
R50	MoCo V2	24	200	67.5
R50	MoCo V2 + **Ours**	24	200	**68.6**$^{\uparrow 1.1}$
R50	MoCo V2 + **Ours**†	24	200	**69.8**$^{\uparrow 2.3}$
R50	PIRL	24	800	63.6
R50	SimCLR	24	1000	69.3
R50	MoCo V2	24	800	71.1
R50	MoCo V2 + **Ours**	24	800	71.8$^{\uparrow 0.7}$

Table 4. Comparison of linear classification on standard ImageNet. †denotes our result using multi-scale training. Note that all the hyperparameters follow the baseline MoCo V2 so they may not be optimal with our mixture, thus the gains are generally "minima".

.

5.6. Visualization and Analysis

Learned representations. To further explore what our model indeed learned, we visualize the embedded features in Fig. 11 from baseline MoCo (left) and our mixture model (right) using t-SNE with the last conv-layer features (128-dimension) from ResNet-18. Our model has more separate embedding clusters, especially on classes 9, 8 and 1.

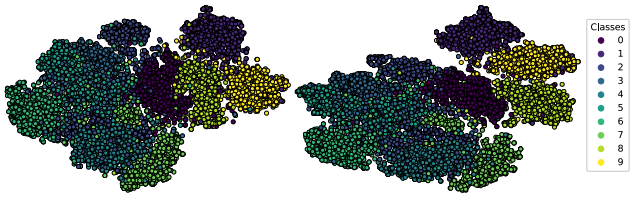

Figure 11. Visualization of feature embedding on CIFAR-10.

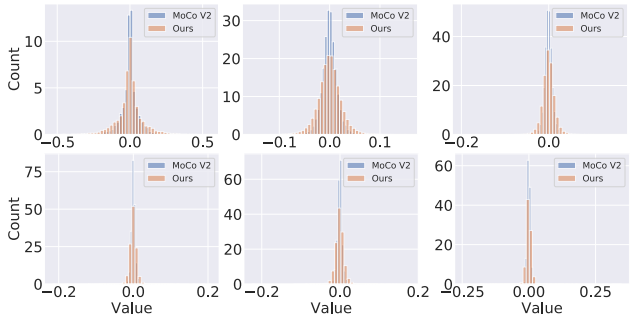

Figure 12. Illustration of weight distributions at *1, 10, 20, 30, 40 and 50-th* conv layers in a ResNet-50. Full layers are in Appendix.

How do image mixtures help representation ability? We visualize the histogram of weights in particular convolutional layers, as shown in Fig. 12. We can see our weight distributions always have a wider scope of values, while fewer elements are close to zero. This phenomenon reflects the larger capacity of our network since weights in our model have more status to be in.

Pre-train	AP_{50}	AP	AP_{75}
Random init.	60.2	33.8	33.1
Supervised IN-1M	81.3	53.5	58.8
MoCo V2 (200ep)	82.4	57.0	63.6
Ours (200ep)	**83.0**$^{\uparrow 0.6}$	**57.7**$^{\uparrow 0.7}$	**64.3**$^{\uparrow 0.7}$
MoCo V2 (800ep)	82.5	57.4	64.0
Ours (800ep)	**83.2**$^{\uparrow 0.7}$	**58.1**$^{\uparrow 0.7}$	**65.2**$^{\uparrow 1.2}$
Ours (200ep), MS	83.2	57.8	64.5

(a) Faster R-CNN, **R50-C4** on **PASCAL VOC**

Pre-train	AP	AP_{50}	AP_{75}
Random init.	35.6	54.6	38.2
Supervised IN-1M	40.0	59.9	43.1
MoCo V2 (200ep)	40.9	60.7	44.4
Ours (200ep)	**41.2**$^{\uparrow 0.3}$	**60.9**$^{\uparrow 0.2}$	**44.7**$^{\uparrow 0.3}$

(b) Mask R-CNN, **R50-C4 2**× on **COCO**

Table 5. Object detection results fine-tuned on PASCAL VOC (a) and COCO (b). On VOC, the training and evaluation sets are `trainval07+12` and `test2007`, and on COCO are the `train2017` and `val2017`. All models are fine-tuned for 24k iterations on VOC and 180k on COCO. On VOC dataset, we run three trials and report the means as the final results in this table.

6. Conclusion

We have investigated the feasibility of mixture operations in an unsupervised scheme, and proposed the strategy of image mixtures and corresponding label re-assignment for flattening inputs and predictions in various architectures of unsupervised frameworks. Through extensive experiments on SimCLR, BYOL, MoCo V1&V2, *etc.*, and downstream tasks like object detection, we have shown that neural networks trained with our newly constructed input space have better representation capability in terms of generalization and transferability, as well as better robustness for different pretext tasks or frameworks (contrastive or non-contrastive learning, with or without memory banks, multi-scale training). Easy to implement and only incurring a small additional computational cost, we hope the proposed method can be a useful technique for the unsupervised learning problem.

References

Bachman, P., Hjelm, R. D., and Buchwalter, W. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pp. 15509–15519, 2019.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

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

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

Chorowski, J. and Jaitly, N. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*, 2016.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

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

Donahue, J. and Simonyan, K. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pp. 10541–10551, 2019.

Donahue, J., Krähenbühl, P., and Darrell, T. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Ermolov, A., Siarohin, A., Sangineto, E., and Sebe, N. Whitening for self-supervised representation learning. *arXiv preprint arXiv:2007.06346*, 2020.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88 (2):303–338, 2010.

Gidaris, S., Singh, P., and Komodakis, N. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

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.

Hadsell, R., Chopra, S., and LeCun, Y. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pp. 1735–1742. IEEE, 2006.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Kalantidis, Y., Sariyildiz, M. B., Pion, N., Weinzaepfel, P., and Larlus, D. Hard negative mixing for contrastive learning. *arXiv preprint arXiv:2010.01028*, 2020.

Kim, S., Lee, G., Bae, S., and Yun, S.-Y. Mixco: Mix-up contrastive learning for visual representation. *arXiv preprint arXiv:2010.06300*, 2020.

Krizhevsky, A. et al. Learning multiple layers of features from tiny images. 2009.

Krothapalli, U. and Abbott, A. L. Adaptive label smoothing, 2020.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pp. 52–59. Springer, 2011.

Misra, I. and van der Maaten, L. Self-supervised learning of pretext-invariant representations. *arXiv preprint arXiv:1912.01991*, 2019.

Müller, R., Kornblith, S., and Hinton, G. E. When does label smoothing help? In *Advances in Neural Information Processing Systems*, pp. 4696–4705, 2019.

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

Noroozi, M., Pirsiavash, H., and Favaro, P. Representation learning by learning to count. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5898–5906, 2017.

Olshausen, B. A. and Field, D. J. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

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

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Ren, S., He, K., Girshick, R., and Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

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

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408, 2010.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

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.

Ye, M., Zhang, X., Yuen, P. C., and Chang, S.-F. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6210–6219, 2019.

Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6023–6032, 2019.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhang, R., Isola, P., and Efros, A. A. Colorful image colorization. In *European conference on computer vision*, pp. 649–666. Springer, 2016.

Zhang, R., Isola, P., and Efros, A. A. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1058–1067, 2017.

Appendix

In this appendix, we provide details omitted in the main text, including:

• Section A: Implementation details in unsupervised pre-training and linear evaluation on non-ImageNet and ImageNet. (Sec. 5.3 "Implementation Details in Pre-training" and Sec. 5.4. "Linear Classification" of the main paper.)

• Section B: Comparison of *both branch mixtures* and *single branch mixture*. (Sec. 4.1 "Paradigms of Mixtures" of the main paper.)

• Section C: A PyTorch-like code for our mixture approach. (Sec. 4.2 "Introduction" of the main paper.)

• Section D: Visualization of entire layers with a ResNet-50 on ImageNet dataset. (Sec. 5.6 "Visualization and Analysis" of the main paper.)

A. Implementation Details in Unsupervised Pre-training and Linear Evaluation

Following settings in (Ermolov et al., 2020), on CIFAR-10 and CIFAR-100, we train for 1,000 epochs with learning rate 3×10^{-3}; on Tiny ImageNet, 1,000 epochs with learning rate 2×10^{-3}; on STL-10, 2,000 epochs with learning rate 2×10^{-3}. We also apply warm-up for the first 500 iterations, and a 0.2 learning rate decay at 50 and 25 epochs before the end of training. Adam is used as the optimizer in all of the non-ImageNet experiments. The weight decay is used as 10^{-6}. The batch size is set to $K = 512$ samples. The dimension of the hidden layer of the projection head is 1024. Moreover, we use an embedding size of 64 for CIFAR-10 and CIFAR-100, and 128 for STL-10 and Tiny ImageNet.

Evaluation Protocol on Non-ImageNet. For linear evaluation on non-ImageNet datasets, we follow the baseline codebase configurations (Ermolov et al., 2020) with 500 epochs using the Adam optimizer without any data augmentation. The learning rate is exponentially decayed from 10^{-2} to 10^{-6}. The weight decay is 5×10^{-6}.

Evaluation Protocol on ImageNet. For linear evaluation on ImageNet dataset, we follow the baseline approach (He et al., 2019; Chen et al., 2020b) and use initial learning rate of 30 and weight decay of 0. We train with 100 epochs and the learning rate is multiplied by 0.1 at 60 and 80 epochs.

B. Comparison of *Both Branch Mixtures* and *Single Branch Mixture*

The comparisons between *both branch mixtures* and *single branch mixture* are provided in Table 6, "ours$_1$" and "ours$_2$" denote the results of these two paradigms of mixtures, respectively. From this table, we have several interesting observations: We found that the strategy of *both branch mixtures*, i.e., "ours$_1$" performs better than "ours$_2$" on SimCLR (Chen et al., 2020a) across all datasets, but is inferior with other three baseline approaches. We conjecture the reason is that SimCLR leverages negative pairs and our mixtures can increase the capacity/complexity of composition on the negative pair set, which is beneficial for the unsupervised representation learning. However, other three baseline approaches do not use negative pairs in their frameworks, so the advantage of enlarged capacity cannot be utilized by these methods. Also, "ours$_1$" has no additional mixture ratio information if comparing to "ours$_2$", so basically, "ours$_2$" is a superior design. Nevertheless, in most cases, both of these two paradigms are still better than the baseline results.

Algorithm 1 PyTorch-like Code for Our Mixture Strategy.

```
# P: probability of global or local level mixtures
# beta: hyperparameter for Beta distribution
# lam: mixture ratio in global-level mixture or
    bounding box location in region-level mixture

args.beta = 1.0
for x in loader: # load a minibatch x with N samples
   # Probability of choosing global or local level
       mixtures
   prob = np.random.rand(1)
   lam = np.random.beta(args.beta, args.beta)
   images_reverse = torch.flip(x[0], (0,))
   if prob < args.P:
      # global-level mixtures
      mixed_images = lam * x[0] + (1 - lam) *
          images_reverse
      mixed_images_flip = torch.flip(mixed_images,
          (0,))
   else:
      # region-level mixtures
      mixed_images = x[0].clone()
      bbx1, bby1, bbx2, bby2 = utils.rand_bbox(x[0].
          size(), lam)
      mixed_images[:, :, bbx1:bbx2, bby1:bby2] =
          images_reverse[:, :, bbx1:bbx2, bby1:bby2]
      mixed_images_flip = torch.flip(mixed_images,
          (0,))
      lam = 1 - ((bbx2 - bbx1) * (bby2 - bby1) / (x[0].
          size()[-1] * x[0].size()[-2]))

   # original loss term
   loss_ori = model(x)
   # In following two losses, we found using ``x[0]''
       may perform better on some particular datasets
   # loss for the normal order of mixtures
   loss_m1 = model([x[1], mixed_images])
   # loss for the reverse order of mixtures
   loss_m2 = model([x[1], mixed_images_flip])
   # final loss function
   loss = loss_ori + lam*loss_m1 + (1-lam)*loss_m2

   # update gradients
   optimizer.zero_grad()
   loss.backward()
   optimizer.step()
   ...
```

C. A PyTorch-like (Paszke et al., 2019) Code for Our Mixture Strategy

The pseudocode of our mixture method is shown in Algorithm 1. Note that this is a demonstration for the circumstance that the memory banks are not existing in the frameworks, such as *SimCLR* (Chen et al., 2020a), *BYOL* (Grill

Table 6. Classification accuracy of Top-1 on ResNet-18 using a linear classifier and a 5-nearest neighbors classifier with different loss functions and datasets. Ours$_1$ and ours$_2$ denote the results of *both branch mixtures* and *single branch mixture*, respectively. MoCo is trained with symmetric loss, 1000 epochs and evaluated with 200 in kNN monitor* following https://github.com/facebookresearch/moco of CIFAR datasets.

Method	CIFAR-10						CIFAR-100						STL-10					
	linear	**ours$_1$**	**ours$_2$**	5-nn	**ours$_1$**	**ours$_2$**	linear	**ours$_1$**	**ours$_2$**	5-nn	**ours$_1$**	**ours$_2$**	linear	**ours$_1$**	**ours$_2$**	5-nn	**ours$_1$**	**ours$_2$**
SimCLR (Chen et al., 2020a)	91.80	**93.11**	92.35	88.42	**90.06**	89.74	66.83	**69.47**	68.83	56.56	**59.34**	58.82	90.51	**91.16**	90.86	85.68	**87.29**	86.16
BYOL (Grill et al., 2020)	91.73	93.37	**94.20**	89.45	91.86	**93.03**	66.60	68.75	**71.50**	56.82	61.21	**63.83**	91.99	90.53	**93.34**	88.64	87.68	**90.46**
W-MSE 2 (Ermolov et al., 2020)	91.55	92.77	**93.04**	89.69	91.06	**91.33**	66.10	69.44	**70.12**	56.69	59.16	**61.28**	90.36	89.28	**92.21**	87.10	85.59	**88.88**
W-MSE 4 (Ermolov et al., 2020)	91.99	91.98	**93.18**	89.87	89.85	**91.70**	67.64	67.63	**69.70**	56.45	57.46	**60.74**	91.75	90.95	**91.96**	88.59	87.86	**88.71**
MoCo (Symmetric Loss) (He et al., 2019)	–	–	–	90.49*	90.35*	**92.25***	–	–	–	65.49*	66.01*	**68.83***	–	–	–	–	–	–

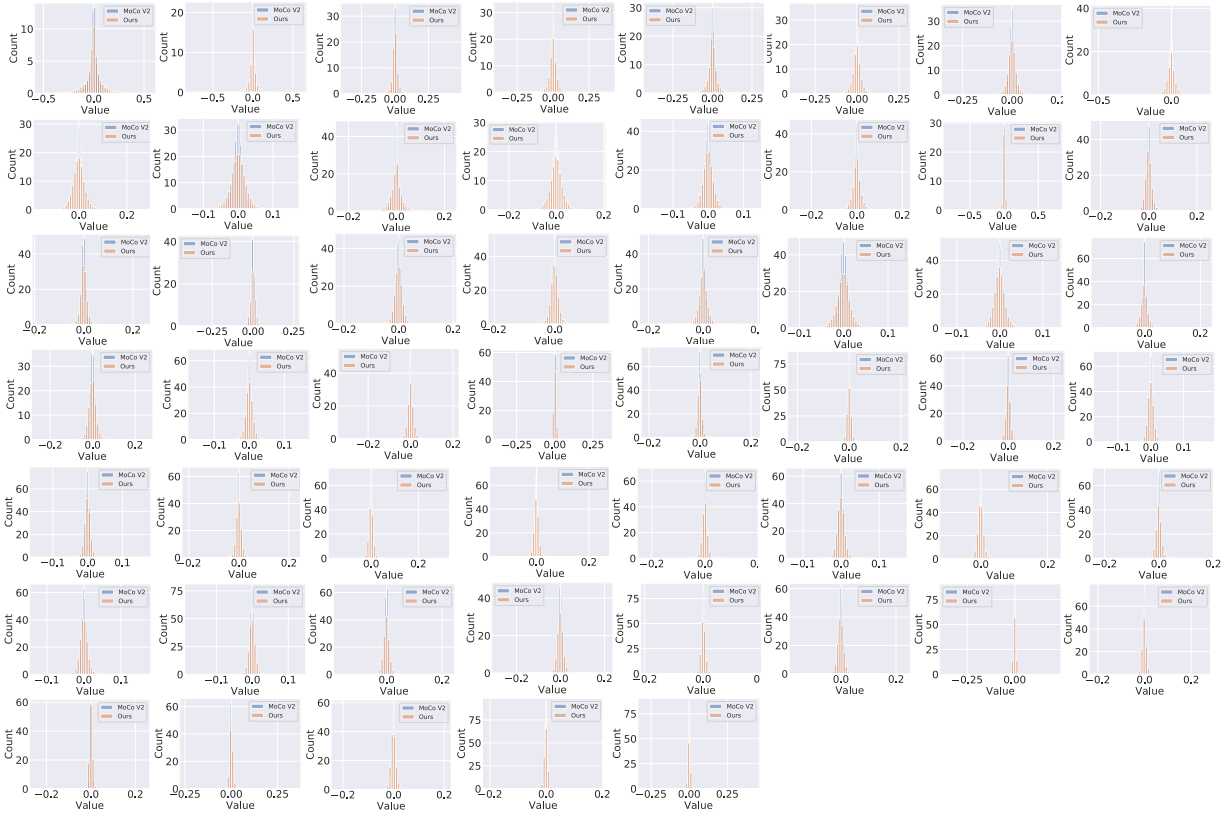

Figure 13. Weight distributions from all the layers of a ResNet-50 on ImageNet with baseline MoCo V2 and our mixture trained model.

et al., 2020) and *Whitening* (Ermolov et al., 2020) approaches. If employing a memory bank in the approach, it is required to deal with the memory bank mechanism so it needs a few additional non-core lines, but the central codes are entirely the same as we presented in Algorithm 1.

D. Visualizations of All Convolutional Layers

As shown in Figure 13, we visualize the weight distributions from all convolutional layers of a ResNet-50 on ImageNet with the baseline MoCo V2 model and our mixture trained model. It can be observed that our weight distributions always have a wider scope of values, meanwhile, fewer elements of weight values are close to zero.