MixMask: Revisiting Masked Siamese Self-supervised Learning in Asymmetric Distance

Kirill Vishniakov$^1$, Eric Xing$^{1,2}$, Zhiqiang Shen$^{1,2,3}$

$^1$Mohamed bin Zayed University of Artificial Intelligence, $^2$Carnegie Mellon University, $^3$Hong Kong University of Science and Technology

{kirill.vishniakov, eric.xing, zhiqiang.shen}@mbzuai.ac.ae

Abstract

Recent advances in self-supervised learning integrate Masked Modeling and Siamese Networks into a single framework to fully reap the advantages of both the two techniques. However, the previous erase-based masking scheme in masked image modeling is more aligned with the patchifying mechanism of ViT, it is not originally designed for siamese networks of ConvNet. Existing approaches simply inherit the default loss design from previous siamese networks and ignore the information loss after employing masking operation in the frameworks. In this paper, we propose a filling-based masking strategy called MixMask to prevent information loss due to the randomly erased areas of an image in the vanilla masking method. We further introduce a flexible loss function design that takes into account semantic distance change between two different mixed views for adapting the integrated architecture and avoiding mismatches between transformed input and objective in Masked Siamese ConvNets (MSCN). The flexible loss distance is calculated according to the proposed mix-masking scheme. Extensive experiments are conducted on various datasets of CIFAR-100, Tiny-ImageNet, and ImageNet-1K. The results demonstrate that the proposed framework can achieve better accuracy on linear probing, semi-supervised, and supervised finetuning, which outperforms the state-of-the-art MSCN by a significant margin. We also show the superiority on the downstream tasks of object detection and segmentation. Our source code is available at https://github.com/LightnessOfBeing/MixMask.

1. Introduction

Self-supervised learning is a widely used paradigm to learn representations from input data without human-annotated labels. In the computer vision domain, it has shown superior performance in many tasks, such as classification, detection, segmentation, etc. A popular self-supervised learning framework is a Siamese Network with two branches. A similarity loss [8, 17], contrastive loss [6, 19, 27] or distillation loss [5] is employed to calculate the distance of the two branches. Recently, masked image modeling (MIM) [1, 3, 18] has emerged and proven to be an effective approach to learn useful representation. To fully leverage the advantages from both the masked design and siamese networks, Masked Siamese Networks [1, 22] have been proposed.

Originally, erase-based masking greatly synergizes with the image patchify mechanism in ViT [11], which produces image patches that are independently processed by the encoder. Moreover, such independent patch-wise processing of an image allows to simply drop masked patches for decreasing the computational cost [18]. In contrast, ConvNets process data continuously. If the random image patches are dropped casually, they no longer contain global-level meaningful semantic information, and it even breaks the completeness of input, causing that the ConvNets cannot process anymore. In addition, multiple contrastive frameworks use embedding loss [6, 19], which is not designed to recover the erased regions but to discriminate between the produced embeddings. This is different from transformer-based MIM methods [3, 18] that will recover the masked regions using reconstruction loss. Therefore, we identify two main drawbacks of Masked Siamese ConvNets: (I) Information loss from the regular erase-based masking operation. It is straightforward that vanilla masking will drop semantic information from the input data and cannot be recovered by post-processing. For instance, if we mask 25% of image areas, 25% of information will be lost during training, and it will encumber the training efficiency. Thus, a better masking strategy is necessary to encourage models to learn better representations. (II) The default “symmetric” semantic distance loss in the siamese networks that does not take

$^1$Our symmetric and asymmetric refer to a semantic distance in a single loss term, while other literature use symmetric and asymmetric in the sense of adding an additional symmetric component to the loss where two views appear in the symmetric order.
into account the semantic distance change between different views of the same image after the masking operation. Clearly, when arbitrary regions of the image get erased, the semantic meaning of the image changes as well. An original image of a dog in the center has a substantially different meaning compared to the same image, where regions containing different body parts of a dog are masked and no longer visible. Such symmetric loss design brings relatively low influence when both of the branches are masked. However, it causes a negative effect inevitably when only one branch is masked in the architecture. Since asymmetric siamese networks have been observed to be more effective to learn, a more flexible loss design with soft distance is crucial to reflect the true semantic distance between the two asymmetric branches in masked siamese convnets.

To address these two drawbacks, in this work, we propose to use a filling-based masking strategy to avoid information loss by erasing operation. Specifically, instead of erasing random areas in an image solely, we will randomly select another image and fill its pixels to the masked areas. Compared to the erase-based strategy, this masking method contains more semantic information for the input space, and the siamese networks can also exploit more to enhance the representation during training. Our ablation experiments show that this is a better masking solution for the Masked Siamese ConvNets. We further introduce a soft flexible loss term with an asymmetric distance to fit the proposed MixMask and to evaluate the similarity of two branches in masked siamese networks adaptively. The soft distance is calculated by comparing the masking ratios of two branches, and the resulting distance essentially reflects the semantic similarity between them. Since the masking ratio changes across different iterations, the corresponding soft similarity loss also changes accordingly for different iterations in a dynamic way.

Comprehensive experiments are conducted on CIFAR-100, Tiny-ImageNet, and ImageNet-1K datasets. We integrate our method in multiple Siamese ConvNets such as MoCo, BYOL, SimCLR, and SimSiam. Our method improves the various baselines by remarkable margins on all datasets. We also examine our learned models on semi-supervised and supervised fine-tuning, object detection and segmentation downstream tasks.

Our contributions in this work are as follows:

- We propose a simple yet effective filling-based masking strategy to prevent information loss from input space for self-supervised Siamese ConvNets.
- We introduce a flexible loss function design with soft distance to adapt the integrated masking and siamese architecture, as well as to avoid mismatches between transformed input and objective in Masked Siamese ConvNets.

Extensive experiments are conducted on various datasets and siamese frameworks to demonstrate the effectiveness of the proposed method. We also verify our approach on the semi-supervised and supervised fine-tuning, object detection and segmentation downstream tasks.

2. Related Work

Self-supervised Learning. Self-supervised learning is a popular technique for representation learning whose key ingredient is the usage of huge amounts of unlabeled data. Early self-supervised approaches in vision were based on pre-text tasks [15, 26, 30, 38]. A milestone in SSL was a simple contrastive learning framework, introduced by [6, 9], which utilized a siamese architecture together with an InfoNCE [27] contrastive loss. MoCo [19] employed a memory bank to store negative samples. Several methods indicated that the presence of negative pairs is not necessary. BYOL [29] used asymmetric architecture with EMA, where the online network is trying to predict the representations of the target network. SimSiam [8] provided a simple siamese framework that applied a stop-gradient operation on one branch and an additional predictor step on the other. Recently, self-supervised learning adopted the use of vision transformers [11]. DINO [5] utilized distillation loss together with vision transformers in a siamese framework. MAE [18] explored how Masked Autoencoders can be used for the problem of self-supervised representation learning in vision domain.

Masked Siamese Networks. Advances in masked image modeling and siamese self-supervised learning posed a natural question of finding a way to combine these two techniques. Masked Siamese Networks were proposed in [1]. MSN generates two views, an anchor and a target, and applies masking operation on the anchor branch. It further uses prototype cluster assignment to assign masked anchor representations to the same cluster as the unmasked target. Vision Transformer has been chosen as an encoder, which greatly combines with masking operation due to its underlying design, which splits the original image into patches of a smaller size. In contrast, CNNs are less suitable for working with masked images as they operate on the pixel level instead of the patch level, and thus, masking operation destroys the image continuity that CNNs heavily rely on. Masked Siamese ConvNets were proposed in [22], where authors explore and provide guidelines on how to make a CNN-based siamese self-supervised learning framework compatible with masking.

Mixture-based Data Augmentations. Different mixture-based methods were proven to be useful data augmentations in the computer vision domain. Such methods usu-
ally mix two images to obtain a new image, their mixture, and further modify the loss function to reflect the semantic distance change in the produced mixture image. Initially, Mixup [37] and Manifold Mixup [32] were proposed for supervised learning to mix pairs of images in the same batch on a global pixel level with a soft coefficient $\lambda$. Further, a local region-based method CutMix [36] and attentive scheme [33] were proposed following the same loss design as Mixup. Finally, in Un-Mix [31], authors have shown how to combine Mixup and CutMix for self-supervised learning.

**Masked Image Modeling.** Masked Image Modeling (MIM) is a task whose goal is to learn useful representations by trying to reconstruct a masked image to its original view. Masked Autoencoder is a common model which can be used in a MIM framework. MAE [18] proposed a simple transformer-based masked autoencoder architecture that tries to reconstruct the original image using MSE loss. BEIT [3] proposed a self-supervised pretraining via tokenizing the input image to visual tokens with masking. MAE for spatiotemporal representation learning was proposed by [13]. Several works [2,14,16] have studied how to make masked autoencoders in the case of multimodal data.

### 3. Approach

In this section, we first introduce each component of our framework elaborately, including: (i) a filling-based masking strategy; (ii) an asymmetric loss formulation with soft distance to match the proposed mix-masking scheme; (iii) permutation strategies if incorporating with other mixing approaches. Then, we provide an overview of the proposed architecture comparing to the basic model. Finally, we discuss in detail the design principles, insights, and differences from other counterparts.

#### 3.1. Perspective on Masking Strategy

![Image](image.png)

Figure 1. Illustration of the proposed filling-based masking strategy. The dashed box shows Erase/Gaussian noise [22] masking strategy. A formal definition of a switch image in the case of reverse permutation is given in Equation 2.

**Masking Scheme: Erasing or Filling?** In Fig. 1, the Erase/Gaussian noise masking strategy is illustrated inside the dashed box. In such strategy erased regions can be filled with a Gaussian noise [22]. Different from Masked Image Modeling (MIM), which is to reconstruct the masked contents for learning good representation, the **Masked Siamese Networks** will not predict the information in removed areas, so erasing will only lose information and is not desired in the learning procedure. In contrast to erase-based masking, our filling-based strategy will repatch the removed areas using an auxiliary image, as shown in the right part of Fig. 1. After that, we will **switch** the content between the main image and auxiliary image to generate a new image for information completeness of two input images. For a given original image $I_i$, we define its mixture as mix of the pair $(I_i, I_{n-i})$ and **switch** image of $I_i$ as mixture of the pair $(I_{n-i}, I_i)$. Our soft objective calculation is designed to fit the format of such masked images in siamese networks, as described in the next section.

#### 3.2. Perspective on Distance in Siamese Networks

**Objective Calculation: Inflexible or Soft?** It has been observed [31] that different pretext and data processes (e.g., masking, mixture) will change the semantic distance of two branches in the siamese networks, hence the default symmetric loss will no longer be aligned to reflect the true similarity of the latent representations. It thus far has not attracted enough attention for such a problem in this area. In this work, we introduce a soft objective calculation method that can fit the filling-based masking strategy in a better way. To calculate the soft distance, firstly, we start by generating a binary mask with a fixed grid size that will later be used to mix a batch of images, denoted as $I_i$ from a single branch. In case when we use a reverse permutation to obtain the mixture, each image in the batch with index $i$ is mixed according to the mask with the image in the same batch but with index $n-i$ as described in Equation 1:

$$
\text{mix}_i = \text{mix}(I_i, I_{n-i}) = m \odot I_i + (1-m) \odot I_{n-i} \\
\text{switch}_i = \text{mix}(I_{n-i}, I_i) = m \odot I_{n-i} + (1-m) \odot I_i, \tag{1}$$

where $n$ is batch size and $m$ is mask. The mixed image contains parts from both $I_i$ and $I_{n-i}$ whose spatial locations in the mixture are defined by the contents of the binary mask. For the reverse permutation switch mixture will interact with two images, but for random permutation, it will involve an additional image. Furthermore, we calculate a mixture coefficient $\lambda$, which is equal to the ratio of the masked area to the total area of the image using the Equation 3:

$$
\lambda = \frac{\sum_{x,y} \mathbb{1}[\text{mask}(x,y) = 1]}{\text{width} \cdot \text{height}}, \tag{3}$$

where $\mathbb{1}$ is the indicator function that measures the masked area of an image.

**Loss Function.** The final loss is defined as a summation of the original loss and mixture loss:

$$
\mathbb{L} = \mathbb{L}_{\text{Orig}} + \mathbb{L}_{\text{MixMask}}. \tag{4}
$$
Our MixMask Siamese ConvNets

MSCN

1. symmetric loss
2. asymmetric loss
3. filling-based masking
4. shared

Figure 2. Illustration of the Masked Siamese ConvNets (left) and our proposed framework (right). MixMask branch incorporates asymmetry into the loss function design by generating images with different rates of similarity to the images in the original branch. In MixMask branch image of the truck is presented twice with different levels of similarity to the image in the original branch due to the regions masked with contents of another image.

Here we take a contrastive loss from MoCo [19] as an example, the $\mathbb{L}_{\text{Orig}}$ will be:

$$
\mathbb{L}_{\text{Orig}} = -\log \frac{\exp (q \cdot k / \tau)}{\sum_{i=0}^{K} \exp (q \cdot k_{i} / \tau)}.
$$

The mixture term $\mathbb{L}_{\text{MixMask}}$ will contain two terms which are scaled with $\lambda$ coefficient:

$$
\mathbb{L}_{\text{MixMask}} = \lambda \cdot \mathbb{L}_{\uparrow} + (1 - \lambda) \cdot \mathbb{L}_{\downarrow} =
\begin{align*}
&= -(\lambda \cdot \log \frac{\exp (q_{\uparrow} \cdot k / \tau)}{\sum_{i=0}^{K} \exp (q_{\uparrow} \cdot k_{i} / \tau)}) + \\
&(1 - \lambda) \cdot \log \frac{\exp (q_{\downarrow} \cdot k / \tau)}{\sum_{i=0}^{K} \exp (q_{\downarrow} \cdot k_{i} / \tau)},
\end{align*}
$$

where $q_{\uparrow}$ and $q_{\downarrow}$ are normal and reverse orders of mixed queries in a mini-batch respectively, $k$ is the unmixed single key, $\lambda$ is calculated according to the Equation 3 and $\tau$ is the temperature.

Permutation Strategies: Same or Shuffled? We also provide insights on how our method can be combined with Un-Mix [31]. Namely, we explore how different permutation strategies used to produce an image mixture affect the final result of the model trained when Un-Mix and MixMask are applied together. Recall, that by default, in Un-Mix a reverse permutation is used to generate the image mixture, i.e. the image with index $i$ is mixed with the image with index $n - i$. We empirically show that to get the best performance for the model, which is trained both with Un-Mix and MixMask, we need to use a different permutation on the MixMask branch. In this case, we can generate a more diverse set of mixed images as different pairs of images are mixed in Un-Mix and MixMask branches respectively. This permutation strategy ultimately leads to a better generalization ability of the model. A detailed illustration of the strategy is shown in Fig. 4.

3.3. Framework Overview

Our framework overview is shown in Fig. 2. In this figure, the left is the conventional Masked Siamese ConvNets (MSCN), right is our proposed MixMask with asymmetric distance loss. The motivation behind this design is that directly erasing regions will lose a significant proportion of information in the Siamese ConvNets, which cannot be recovered by post-training. This is quite different from the mechanism of Masked Autoencoders (MAE) that predict masked areas to learn good representations. According to this, we propose a filling-based scheme to overcome the drawback. The soft distance loss is designed to fit the true similarity of the two branches. Despite its conceptual simplicity, we empirically show that with the integrality of mix-masking and objective, we can learn more robust and generalized representations from the masked input.

3.4. Discussions

Mask Pattern: Blocked or Discrete? Masking strategies determine the difficulty for the network to generate representations in siamese networks, even if the masking ratio is the same, the representation will still be different for different masking patterns. It will directly influence the information of input to further affect the latent representations. Our observation on Masked Siamese ConvNets is opposite to that in MIM methods, which found discrete/random masking is better [18, 35]. From our empirical experiments, on CIFAR-100, blocked and discrete masking patterns achieve similar accuracy and discrete is slightly better, however, on Tiny-ImageNet and ImageNet-1K, blocked
Un-Mix branch MixMask branch

\( I_4 \)
\( I_3 \)
\( I_2 \)
\( I_1 \)
\( I_4 \)
\( I_3 \)
\( I_2 \)
\( I_1 \)
\( I_1 \)
\( I_2 \)
\( I_3 \)
\( I_4 \)
\( M_1 \)
\( M_2 \)
\( M_3 \)
\( M_4 \)
\( U_n_1 \)
\( U_n_2 \)
\( U_n_3 \)
\( U_n_4 \)

Mini-batch

(a) Same Permutation

(b) Different Permutation

Figure 3. Illustration of the different mask patterns with a mask grid size of 8. (a) and (b) are input images. (c) is the discrete/random mask pattern, and (d) and (e) are mixed images using this mask. (f) is the blocked mask pattern, and (g) and (h) are mixed images with a blocked mask.

Figure 4. Illustration of different permutation strategies when Un-Mix and MixMask are used together. \( I_i \) denotes an image in mini-batch, \( U_n_i \) denotes a mixture image obtained using Un-Mix, and \( M_i \) denotes a mixture image generated using MixMask. In (a), we use the same reverse permutation \( P_1 \) for both the Un-Mix branch and the MixMask branch, and thus we mix the same pair of images twice. In (b), we use two different permutations \( P_1 \) (reverse) and \( P_2 \) (different random permutation) therefore, different pairs of images are mixed in different branches yielding a more diverse set of training data.

mask clearly shows superiority over discrete/random. We explain this as that if the input size is small, the mask pattern is not so necessary since the semantic information of the object is still preserved. On larger datasets like Tiny-ImageNet and ImageNet-1K, discrete/random masking will entirely destroy the completeness of the object in an image, as shown in Fig. 3 (d, e), while, this is crucial for ConvNet to extract a meaningful representation of the object. Therefore, blocked masking shows superior ability on the MSCN and is a better choice than random masking.

**Symmetric or Asymmetric losses in Siamese ConvNets?**

There are several paradigms in Siamese networks on input and objective spaces: (i) *Input symmetric + objective symmetric* (regular Siamese models). (ii) *Input asymmetric + objective asymmetric* (Un-Mix [31]) (iii) *Input asymmetric (slightly) + objective symmetric* (MSCN [22]). Here, *input symmetric* indicates that we use the same probability of data augmentations to generate two different views of samples from the same image. *Input asymmetric* means one view/branch contains extra data augmentations like Mixup or CutMix. *Objective symmetric* presents that regular contrastive loss or similarity loss is used, and *objective asymmetric* indicates that the similarity will be calibrated by a soft coefficient. Generally, both input and objective are asymmetric and can force the model to learn subtler and fine-grained information, thus the representation is more robust for downstream tasks.

**Relationship to Counterparts in Self-supervised Learning.**

Un-Mix introduced Mixup and CutMix into the siamese networks for self-supervised learning. Our MixMask is basically a generalized CutMix whereas designed for siamese networks with arbitrary shape of mask areas. Thus, if considering the self-supervised learning scenario, Un-Mix’s region-level mixture can be regarded as a special case of our MixMask scheme. MixMask itself has shown strong representation learning ability in our empirical study, while it is interesting to see that MixMask also has compatibility with Un-Mix and can be employed together to further improve the performance and achieve SOTA accuracy. We highlight that our key contribution in this work is a new masking strategy to make masking compatible with Siamese ConvNets, while Un-Mix focuses on the strategy of incorporating mixture data augmentations into SSL.

4. Experiments

**Base Models.** In our experimental section, we use base models including: MoCo V1&V2 [7, 19], Un-Mix [31], SimCLR [6], BYOL [29] and SimSiam [8].

**Datasets and Training Settings.** We conduct experiments on CIFAR-100 [23], Tiny-ImageNet [24] and ImageNet-1K [10] datasets. For CIFAR-100 and Tiny-ImageNet, we train each framework for 1000 epochs with ResNet-18 [21] backbone. For ImageNet-1K, we pretrain ResNet-50 for
200 epochs and then finetune a linear classifier on top of the frozen features for 100 epochs and report Top-1 accuracy. We use MoCo (and MoCo V2 for ImageNet-1K) as a base framework unless stated otherwise. We report Top-1 on linear evaluation, except the case of MoCo on CIFAR-100 and Tiny-ImageNet for which we provide k-NN accuracy. * indicates that we build our method upon Un-Mix. We provide full training configurations and additional k-NN evaluation results in the supplementary material.

4.1. Ablation Study

In the experiments, we use a blocked masking strategy, masking ratio of 0.5, and grid size of 2, 4, 8 for CIFAR-100, Tiny-ImageNet, and ImageNet-1K if not stated otherwise.

**Masking Strategy.** We first explore how different masking strategies affect the final result. We consider three different hyperparameters: grid size, grid strategy, and masking ratio. To make our experiments more reliable, we consider three datasets with different image sizes: CIFAR-100: 32×32, Tiny-ImageNet: 64×64, and ImageNet-1K: 224×224.

The first parameter grid size specifies the granularity of the n×n square grid of the image. We consider the following values of n = 2, 4, 8, 16, 32, 48. However, we have different upper bounds for different datasets depending on the spatial size of the images. From our experiments, we can conclude that a very large grid size completely disrupts the semantic features of the image and leads to poor performance. Our results indicate that optimal grid size increases proportionally to the input size of the image. We obtain the optimal grid size for CIFAR-100 is 2, for Tiny-ImageNet is 4, and for ImageNet-1K is 8. The masks with very small and very large grid sizes show bad performance. We analyze this happens because a small grid size does not provide enough variance in the mask structure whilst a large grid size destroys the important semantic features of the image.

We consider two different strategies for the random mask generation, namely discrete/random mask and blocked mask. We generate a blocked mask according to the algorithm described in [3]. A discrete mask does not have any underlying structure, whilst a blocked mask is generated in a way to preserve global spatial continuity and, thus, is more suitable for capturing global semantic features. For CIFAR-100, we observe a negligible difference between blocked and discrete masks. On the other hand, for Tiny-ImageNet and ImageNet-1K, which have larger spatial sizes of the image, blocked mask performs better than discrete.

**Table 1. Ablation study on masking and switch strategies using MoCo V1/V2 on various datasets with original and MixMask branches. k-NN accuracy averaged over 3 runs is reported.**

| Input size   | CIFAR-100 | Tiny-ImageNet | ImageNet-1K |
|--------------|-----------|---------------|-------------|
| Grid Size    |           |               |             |
| 2×2          | 68.11     | 46.40         | 68.44       |
| 4×4          | 67.55     | 47.44         | 68.95       |
| 8×8          | 67.51     | 46.48         | 69.18       |
| 16×16        | –         | 46.10         | 68.54       |
| 32×32        | –         | –             | 68.65       |
| 48×48        | –         | –             | 68.79       |
| Masking Ratio|           |               |             |
| 0.25/0.75    | 66.78     | 45.60         | 68.53       |
| 0.5          | 68.11     | 47.44         | 69.18       |
| uniform(0,1) | 67.56     | 45.46         | 68.71       |
| uniform(0.25, 0.75) | 67.78 | 47.04 | 68.85 |
| Switch Mixture|         |               |             |
| Yes          | 68.11     | 47.44         | 69.18       |
| No           | 67.25     | 45.60         | 67.82       |

**Switch Mixture.** We also examine the efficacy of the second term in the mixture loss in Equation 6, which is computed using switch images and multiplied with soft coefficient (1 − λ). Certainly, having two terms in the mixture loss part is beneficial and gives better results on all datasets as it provides more training signal.

**Training Budgets.** We test our method with different training budgets, including 200 epochs; 400 epochs; 600 epochs; 800 epochs, and 1000 epochs on CIFAR-100 using MoCo, SimCLR, and SimSiam. The results are shown in Fig. 5. We can observe our method achieves consistent improvement over various frameworks. We also provide training loss and accuracy curves for 1000 epoch training configuration in the supplementary material.

**Compatibility with Un-Mix.** We examine whether our method can be applied together with Un-Mix in the same framework. We consider two cases which are illustrated in Fig. 4: in the first case, both Un-Mix and MixMask branches use the same permutation to generate the mixed siamese-based self-supervised learning approaches.

For the masking ratio, we consider two cases with constant values of 0.5 and 0.25/0.75 (as there is no difference between these two due to the loss function design) and two cases when the value is sampled from the uniform distribution with different bounds. We obtain the best results for masking ratio 0.5 under different settings, conjecturing that this value causes the blocked mask to generate consistent global views for both of the images being mixed. Sampling masking ratio from a uniform distribution with bounds 0.25 and 0.75 yields better results than using 0 and 1 as bounds.

We believe this shows that extreme values of masking ratio close to either 0 or 1 generate a mixture where one image heavily dominates over the other when in the optimal mixture, areas of each image should be roughly proportional.
images. In the second case, mixtures for Un-Mix and MixMask are generated using different permutations. Our results highlight the importance of using different permutations when mixing images on Un-Mix and MixMask branches respectively. Clearly, when the same permutations are used, even though Un-Mix and MixMask perform different operations on images, there is still some redundancy and duplicated information in the produced mixtures, as each pair of images is being mixed twice (Fig. 4 (a)). On the contrary, when different permutations are used, different sets of pairs are generated in different branches (Fig. 4 (b)). Thus, we produce a more diverse and richer set of mixed training samples yielding better performance. The loss function for the case of using MixMask together with Un-Mix is given as:

\[
L^* = L_{\text{Orig}} + L_{\text{MixMask}} + L_{\text{Un-Mix}}
\]

\[
L_{\text{MixMask}} = \lambda_{\text{MixMask}} \cdot L_{\uparrow} + (1 - \lambda_{\text{MixMask}}) \cdot L_{\downarrow}
\]

\[
L_{\text{Un-Mix}} = \lambda_{\text{Un-Mix}} \cdot L_{\text{rp}} + (1 - \lambda_{\text{Un-Mix}}) \cdot L_{\text{rp}^*},
\]

where \(L_{\text{rp}}/L_{\text{rp}^*}\) is the embedding loss which is computed using mixed queries obtained from Un-Mix, where a random permutation \(rp\) was used to compose pairs of images for mixing; in that case \(rp^*\) stands for inverse permutation of \(rp\), it can be obtained using unshuffle operation on \(rp\). Note that in general, we can use any random permutation to mix images, here, for the sake of simplicity, we are using reverse order permutation on the MixMask branch, which mixes images with indices \(i\) and \(n - i\) respectively. However, it is important to ensure that these permutations are different in MixMask and Un-Mix branches when using both branches together, as shown in Fig. 4. That is why we use a different permutation on the Un-Mix branch. We provide a pseudocode for the case when using both image mixtures in the supplementary material.

Furthermore, we consider the effect of using two branches together on the parameter of the probability of global mixture in Un-Mix. In [31], it has been shown that the optimal parameter for the global mixture (Mixup) on ImageNet-1K is \(P = 0\) (local mixture only). We challenge this assumption in the setting with two mixture branches as we observe adding the MixMask branch can change the optimal value of \(P\) in Un-Mix. Empirically, we yield the best performance for \(P = 0.5\) as shown in Table 3, theorizing that CutMix is essentially the special case of MixMask applied with a blocked mask, which enables the opportunity for the model to benefit from Mixup part of Un-Mix.

### Results for Different Base Frameworks

We consider the generalizability of our method by applying it on top of four different self-supervised learning frameworks. When applying our method upon Un-Mix, we yield superior performance in all cases. We can also obtain a good performance...
Table 6. Results of semi-supervised and supervised finetuning on ImageNet-1K with 1%, 10%, 100% labels. We use \( lr = 0.001 \) on stem and \( lr = 1 / 0.1 / 0.01 \) on the classification layer for 1% / 10% / 100% labels, respectively. All the models are pretrained with 200 epochs.

|                | Top-1 Acc | Top-5 Acc | Top-1 Acc | Top-5 Acc | Top-1 Acc | Top-5 Acc |
|----------------|-----------|-----------|-----------|-----------|-----------|-----------|
| Vanilla MoCo V2| 44.81     | 73.40     | 63.30     | 86.15     | 76.17     | 92.93     |
| Un-Mix         | 47.41     | 75.62     | 65.02     | 87.40     | 76.16     | 93.22     |
| MixMask        | 49.63     | 77.66     | 66.30     | 87.92     | 76.70     | 93.47     |
| MixMask*       | 49.75     | 77.61     | 66.73     | 88.30     | 76.80     | 93.48     |

Table 7. Object detection and segmentation results on the COCO dataset and detection results on the Pascal VOC dataset. MixMask* performs better than Un-Mix across different metrics. All the models are pretrained with 200 epochs.

|                | Object detection | Segmentation | Object detection |
|----------------|------------------|-------------|-----------------|
|                | AP   | AP\(_{50}\) | AP\(_{75}\) | AP\(_{m}\) | AP\(_l\) | AP   | AP\(_{50}\) | AP\(_{75}\) | AP\(_{m}\) | AP\(_l\) | AP   | AP\(_{50}\) | AP\(_{75}\) |
| Vanilla MoCo V2| 40.65 | 60.41 | 44.34 | 24.09 | 45.48 | 54.08 | 35.36 | 57.01 | 37.83 | 16.37 | 39.38 | 52.38 | 57.19 | 82.22 | 63.83 |
| Un-Mix         | 41.24 | 61.14 | 44.90 | 23.88 | 46.10 | 55.43 | 35.96 | 57.82 | 38.17 | 16.47 | 39.49 | 53.71 | 57.69 | 82.98 | 64.53 |
| MixMask        | 40.98 | 60.93 | 44.46 | 23.87 | 45.88 | 55.00 | 35.83 | 57.58 | 38.39 | 16.82 | 39.38 | 53.32 | 57.56 | 82.83 | 64.00 |
| MixMask*       | 41.65 | 61.30 | 45.05 | 23.52 | 46.57 | 55.99 | 36.08 | 57.97 | 38.49 | 16.54 | 39.98 | 53.17 | 57.73 | 83.22 | 64.61 |

4.4. Results on Semi and Supervised Finetuning

For the semi-supervised and supervised finetuning on ImageNet-1K, we follow the protocol described in [4]. We explore three different data regimes of 1%, 10%, and 100% of labels. The results are given in Table 6. MixMask and MixMask* both clearly show superior performance over Un-Mix, especially under low data settings.

4.5. Results on Object Detection and Segmentation

We test our method on the downstream task of object detection and segmentation. For that, we finetune a Faster-RCNN [28], and Mask-RCNN [20] models implemented in Detectron2 [34] on Pascal VOC 2007 [12], and MS COCO 2017 [25] datasets. For VOC 2007, we follow the standard evaluation protocol in [19] with 24k training iterations. For COCO, we use mask_rcnn_R_50_C4_2x configuration from [34]. The results in Table 7 verify the superiority of the proposed MixMask.

5. Conclusion

In this work, we have proposed a new approach of MixMask which combines a dynamic and asymmetric loss design together with a mask-filling strategy for the Masked Siamese ConvNets. We have shown that our asymmetric loss function performs better than the standard symmetric loss used in MSCN. Furthermore, we have solved the problem of information loss in vanilla masking operation, which hinders the training efficiency, by employing a filling-based masking strategy, where another image is used to fill the erased regions. In addition, we have provided a recipe for how to combine MixMask with other mixture methods and explained the effect of the different masking parameters on the quality of the learned representations. Extensive experiments are conducted on CIFAR-100, Tiny-ImageNet, and ImageNet-1K across different frameworks and downstream tasks. Our method outperforms the state-of-the-art baseline by a significant margin.
References

[1] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florant Bordes, Pascal Vincent, Armand Joulin, Michael Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. arXiv preprint arXiv:2204.07141, 2022. 1, 2

[2] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. Multimae: Multi-modal multi-task masked autoencoders. arXiv preprint arXiv:2204.01678, 2022. 3

[3] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. arXiv preprint arXiv:2106.08254, 2021. 1, 3, 6

[4] Zhaowei Cai, Avinash Ravichandran, Subhransu Maji, Charles Fowlkes, Zhuowen Tu, and Stefano Soatto. Exponential moving average normalization for self-supervised and semi-supervised learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 194–203, 2021. 8

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 9650–9660, 2021. 1, 2

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In International conference on machine learning, pages 1597–1607. PMLR, 2020. 1, 2, 5, 11

[7] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297, 2020. 5, 11

[8] 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. 1, 2, 5, 11

[9] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), volume 1, pages 539–546. IEEE, 2005. 2

[10] 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. 5, 11

[11] 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. arXiv preprint arXiv:2010.11929, 2020. 1, 2

[12] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2):303–338, 2010. 8

[13] Christoph Feichtenhofer, Haoqi Fan, Yanghao Li, and Kaiming He. Masked autoencoders as spatiotemporal learners. arXiv preprint arXiv:2205.09113, 2022. 3

[14] Xinyang Geng, Hao Liu, Lisa Lee, Dale Schuurams, Sergey Levine, and Pieter Abbeel. Multimodal masked autoencoders learn transferable representations. arXiv preprint arXiv:2205.14204, 2022. 3

[15] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations, 2018. 2

[16] Rohit Girdhar, Alaaeldin El-Nouby, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Omnima: Single model masked pretraining on images and videos. arXiv preprint arXiv:2206.08356, 2022. 3

[17] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohuan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. Advances in neural information processing systems, 33:21271–21284, 2020. 1

[18] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16000–16009, 2022. 1, 2, 3, 4, 6

[19] 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. 1, 2, 4, 5, 8, 11

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 2961–2969, 2017. 8

[21] 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. 5

[22] Li Jing, Jiachen Zhu, and Yann LeCun. Masked siamese convnets. arXiv preprint arXiv:2206.07700, 2022. 1, 2, 3, 5, 8

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5, 11

[24] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. CS 231N, 7(7):3, 2015. 5, 11

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In European conference on computer vision, pages 740–755. Springer, 2014. 8

[26] Mehdi Norouzi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In European conference on computer vision, pages 69–84. Springer, 2016. 2

[27] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748, 2018. 1, 2

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28, 2015. 8
[29] Pierre H Richemond, Jean-Bastien Grill, Florent Altché, Corentin Tallec, Florian Strub, Andrew Brock, Samuel Smith, Soham De, Razvan Pascanu, Bilal Piot, et al. Byol works even without batch statistics. *arXiv preprint arXiv:2010.10241*, 2020. 2, 5, 11

[30] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video, 2018. 2

[31] Zhiqiang Shen, Zechun Liu, Zhuang Liu, Marios Savvides, Trevor Darrell, and Eric Xing. Un-mix: Rethinking image mixtures for unsupervised visual representation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022. 2, 3, 4, 5, 7, 11

[32] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliakas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pages 6438–6447. PMLR, 2019. 3

[33] Devesh Walawalkar, Zhiqiang Shen, Zechun Liu, and Marios Savvides. Attentive cutmix: An enhanced data augmentation approach for deep learning based image classification. *arXiv preprint arXiv:2003.13048*, 2020. 3

[34] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. [https://github.com/facebookresearch/detectron2](https://github.com/facebookresearch/detectron2), 2019. 8, 11

[35] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9653–9663, 2022. 4, 6

[36] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 3

[37] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 3

[38] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization, 2016. 2
Supplementary material

A. Base Models & Datasets

In this section, we provide a description of self-supervised learning frameworks and datasets that we used in the experiments. To test our method, we tried to select a diverse set of frameworks that incorporate different mechanisms to avoid model collapse and follow different design paradigms, i.e., vanilla contrastive learning with negative pairs vs knowledge distillation.

A.1. Base Models

**MoCo V1&V2** [7, 19] is a self-supervised contrastive learning framework that employs a memory bank to store negative samples. MoCo V2 is an extension of the original MoCo, which introduces a projection head and stronger data augmentations.

**Un-Mix** [31] is an image mixture technique with state-of-the-art performance for unsupervised learning, which uses CutMix and Mixup at its core. It smooths decision boundary and reduces overconfidence in model predictions by introducing an additional mixture term to the original loss value, which is proportional to the degree of the mixture.

**SimCLR** [6] is a siamese framework with two branches that uses contrastive loss to attract positive and repel negative instances using various data augmentations.

**BYOL** [29] is a self-supervised learning technique that does not use negative pairs. It is composed of two networks, an online and a target. The task of an online network is to predict the representations produced by the target network. EMA from the online network is used to update the weights of the target.

**SimSiam** [8] The authors examined the effect of the different techniques which are commonly used to design siamese frameworks for representation learning. As a result, they proposed a simple framework with two branches that relies on the stop gradient operation on one branch and an extra prediction module on the other.

A.2. Datasets

**CIFAR-100** [23] consists of 32×32 images with 100 classes. There are 50,000 train images and 10,000 test images, 500 and 100 per class, respectively.

**Tiny-ImageNet** [24] is a dataset containing 64 × 64 colored natural images with 200 classes. The test set is composed of 10,000 test images, whilst the train contains 500 images per category, totaling 100,000 images.

**ImageNet-1K** [10] has images with a spatial size of 224×224. 1,281,167 images span the training set, which includes 1K different classes, whilst the validation set includes 50K images.

B. Training Configurations

In this section we provide hyperparameter settings for:

- Training on CIFAR-100 and Tiny-ImageNet in Table 8.
- Pretraining and linear probing on ImageNet-1K configurations are shown in Table 9.
- Configurations for semi-supervised and supervised fine-tuning on ImageNet-1K are given in Table 10.
- For object detection and segmentation we use the Detectron2 [34] library and follow the protocol described in the config file coco_R_50_C4_2x.yaml.

C. Additional $k$-NN Evaluation Results

In Table 11, we provide supplementary $k$-NN evaluation results for testing our method on different permutation strategies. The obtained results are consistent with the result from linear evaluation.

D. Training Loss and Accuracy Curves

In Fig. 7, we present the training loss and $k$-NN accuracy curves for different base frameworks trained for 1,000 epochs on CIFAR-100 dataset. MixMask consistently outperforms baseline on all methods. MixMask has a higher (in case of SimSiam lower because it can attain the value of -1) training loss than baseline due to the presence of the additional asymmetric loss term.
Table 8. Training settings on CIFAR-100 and Tiny-ImageNet. Slash separated values correspond to CIFAR-100 and Tiny-ImageNet, respectively.

| MoCo | SimCLR & BYOL | SimSiam |
|------|---------------|---------|
| hparam | value | hparam | value | hparam | value |
| backbone | resnet18 | backbone | resnet18 | backbone | resnet18 |
| optimizer | SGD | optimizer | Adam | optimizer | SGD |
| lr | 0.06 | lr | 0.003/0.002 | lr | 0.03 |
| batch size | 512 | batch size | 512 | batch size | 512 |
| opt momentum | 0.90 | proj layers | 2 | opt momentum | 0.90 |
| epochs | 1,000 | epochs | 1,000 | epochs | 1,000 |
| weight decay | 5e-4 | weight decay | 5e-4 | weight decay | 5e-4 |
| embed-dim | 128 | embed-dim | 64/128 | embed-dim | 128 |
| moco-m | 0.99 | Adam l2 | 1e-6 | warmup epochs | 10 |
| moco-k | 4,096 | proj dim | 1,024 | proj layers | 2 |
| unmix prob | 0.50 | unmix prob | 0.50 | unmix prob | 0.50 |
| moco-t | 0.10 | byol tau | 0.99 | | |

Table 9. Hyperparameter values for pretraining and linear probing on ImageNet-1K. This configuration achieves the highest score of 69.51. All experiments are conducted on 4 × NVIDIA A100 SXM4 40GB GPU.

| hparam | value | Linear probing | value |
|--------|-------|----------------|-------|
| backbone | resnet50 | backbone | resnet50 |
| optimizer | SGD | optimizer | SGD |
| lr | 0.03 | lr | 30 |
| batch size per gpu | 256 | batch size per gpu | 256 |
| num gpus | 4 | num gpus | 4 |
| total batch size | 1,024 | total batch size | 1,024 |
| opt momentum | 0.90 | opt momentum | 0.90 |
| lr schedule | cosine | lr schedule | [60, 80] |
| epochs | 200 | epochs | 100 |
| weight decay | 0 | weight decay | 0 |
| moco-m | 0.999 | | |
| moco-k | 65,536 | | |
| moco-t | 0.2 | | |
| unmix probability | 0.5 | | |
| mask type | block | | |
| grid size | 8 | | |

Table 10. Hyperparameter values for semi-supervised and supervised finetuning on ImageNet-1K. Slash separated values correspond to 1%, 10% and 100% percent data regimes, respectively.

Table 11. Results for k-NN evaluation on CIFAR-100 and Tiny-ImageNet using different permutation strategies when Un-Mix and MixMask are applied together. Applying different permutations produces the best performance in almost all cases.

| CIFAR-100 | Tiny-ImageNet |
|-----------|---------------|
| Permutations | BYOL | SimCLR | BYOL | SimCLR |
| Same | 64.42 | **61.82** | 36.48 | 34.90 |
| Different | **65.01** | 61.48 | **40.02** | **34.96** |

E. More Illustrations of Different Mask Patterns and Image Mixtures

We provide additional illustrations for the different mask patterns and images generated by them. In Fig. 8 illustrations we use mask with grid size 8. All original images are sampled from ImageNet-1K.

F. Pseudocode for the Case When MixMask Used with Un-Mix

In Algorithm 1, we provide a pseudocode for the case when MixMask is used together with Un-Mix.
Figure 7. Training losses (top row) and k-NN evaluation accuracies (bottom row) on CIFAR-100 for experiments with 1,000 epochs for different self-supervised frameworks. MixMask (red) outperforms vanilla baseline (blue) on all frameworks by a significant margin.

Figure 8. Illustration of different mask patterns with grid size of 8. (a) and (b) are input images. (c) is the discrete/random masking pattern, and (d) and (e) are mixed images using this mask. (f) is the blocked mask pattern, and (g) and (h) are mixed images with a blocked mask.
Algorithm 1 Pseudocode for MixMask and Un-Mix used together

```python
for x in train_loader:
    x_rev = torch.flip(x[0], (0,))
    lam_unmix = np.random.beta(1.0, 1.0)
    # generate un-mix mixture using reverse permutation
    if np.random.rand(1) < unmix_prob:  # mixup
        x_unmix = lam_unmix * x[0] + (1 - lam_unmix) * x_rev
    else:  # cutmix
        x_unmix = x[0].clone()
        bbx1, bby1, bbx2, bby2 = rand_bbox(x[0].size(), lam_unmix)
        x_unmix[:, :, bbx1:bbx2, bby1:bby2] = x_rev[:, :, bbx1:bbx2, bby1:bby2]
        lam_unmix = 1 - ((bbx2 - bbx1) * (bby2 - bby1) / (im_width * im_height))

    index_mask = torch.randperm(len(x[0]))  # generate a different random permutation
    mask = get_mask(mask_args)  # generate mask
    x_mixmask = mask * x[0] + (1 - mask) * x[0][index_mask]  # mix according a new random permutation
    lam_mask = mask.sum() / (mask_width * mask_height)  # calculate lambda for mixmask branch

    output, target, output_um, output_um_flip, output_mask, output_mask_inv = model(x[0], x[1], x_unmix, x_mixmask, index_mask=index_mask)

    l_o = criterion(output, target)
    l_um = criterion(output_um, target)
    l_um_flip = criterion(output_um_flip, target)
    l_mask = criterion(output_mask, target)
    l_mask_inv = criterion(output_mask_inv, target)

    l = l_o + lam_unmix * l_um + (1 - lam_unmix) * l_um_flip + lam_mask * l_mask + (1 - lam_mask) * l_mask_inv  # compute total loss
```
