# Triplet is All You Need with Random Mappings for Unsupervised Visual Representation Learning

**Wenbin Li**[1], **Xuesong Yang**[1], **Meihao Kong** [1], **Lei Wang**[2], **Jing Huo**[1], **Yang Gao**[1], **Jiebo Luo**[3]
[1]Nanjing University, China, [2]University of Wollongong, Australia, [3]University of Rochester, USA
{liwenbin, huojing, gaoy}@nju.edu.cn
{yangxuesong, kongmeihao}@smail.nju.edu.cn
leiw@uow.edu.au, jluo@cs.rochester.edu

## Abstract

Contrastive self-supervised learning (SSL) has achieved great success in unsupervised visual representation learning by maximizing the similarity between two augmented views of the same image (positive pairs) and simultaneously contrasting other different images (negative pairs). However, this type of methods, such as SimCLR and MoCo, relies heavily on a large number of negative pairs and thus requires either large batches or memory banks. In contrast, some recent non-contrastive SSL methods, such as BYOL and SimSiam, attempt to discard negative pairs by introducing asymmetry and show remarkable performance. Unfortunately, to avoid collapsed solutions caused by not using negative pairs, these methods require sophisticated asymmetry designs. In this paper, we argue that negative pairs are still necessary but one is sufficient, *i.e., triplet is all you need*. A simple triplet-based loss can achieve surprisingly good performance without requiring large batches or asymmetry. Moreover, we observe that unsupervised visual representation learning can gain significantly from randomness. Based on this observation, we propose a simple plug-in *RandOm MApping (ROMA) strategy* by randomly mapping samples into other spaces and enforcing these randomly projected samples to satisfy the same correlation requirement. The proposed ROMA strategy not only achieves the state-of-the-art performance in conjunction with the triplet-based loss, but also can further effectively boost other SSL methods.

## 1 Introduction

Unsupervised visual representation learning aims to learn good image representations without human supervision. To achieve this, self-supervised learning (SSL) has received considerable attention and shown promise in recent years [1–8]. The recent advances can be classified into two categories: contrastive SSL and non-contrastive SSL. Contrastive SSL, such as SimCLR [3] and MoCo [2, 9], learns representations by closing the latent representations of two augmentations (views) of the same image together (positive pairs), while pushing the latent representations of different images farther away (negative pairs or inter-image). However, these methods normally rely on a large number of negative pairs, requiring either large batches or memory banks. To overcome this limitation, recently, non-contrastive SSL, such as BYOL [5] and SimSiam [4], has attempted to learn representations without using negative pairs (inter-image). However, to avoid collapsed representations caused by removing negative pairs, these methods usually have to employ some sophisticated asymmetry, such as asymmetric predictor network [5, 4], stop-gradients [5, 4], momentum encoder [5] and non-differentiable operators [7]. This raises an interesting question "*Are negative pairs really unnecessary for unsupervised visual representation learning?*".

In addition, to learn view-invariant representations, both contrastive and non-contrastive SSL use data augmentation to obtain pseudo labels, considering that data augmentation usually maintains the semantics of the original examples. Unfortunately, such an advantage could make models be prone to

Preprint. Under review.

the overfitting problem during training. This point has also been mentioned in SimCLR [3], where it shows that stronger data augmentation is needed to bring more benefits. Similarly, InfoMin [6] demonstrates that reducing the mutual information between augmented views of the same image (*e.g.,* stronger data augmentation) while keeping task-relevant information intact can approach the sweet spot in terms of downstream performance. In fact, the asymmetric designs in non-constrastive SSL can also be seen as a way to alleviate the overfitting problem. This naturally raises another question "*How to conveniently alleviate the overfitting problem in data augmentation based SSL methods?*".

In this paper, we propose a new method, *RandOm MAppings (ROMA)*, aiming to answer the above two questions. For the first question, we argue that *negative pairs are still necessary but one is sufficient, i.e., triplet is all you need.* To be specific, we employ a simple *triplet-based loss* built on triplets constructed within each minibatch for training. Compared with other methods, such a simple loss requires neither large batches [3] nor memory banks [2]. Also, it does not require any asymmetric designs like asymmetric predictor network [5, 4] or stop-gradients [5, 4]. In contrast, it enjoys advantages of both existing contrastive and non-contrastive SSL methods: (1) working well with small batches, (2) learning from both intra- and inter-image information, and (3) naturally avoiding collapsed representations.

As for the second question, we propose a "random mapping preemptive measure" into SSL to help unsupervised deep models efficiently deal with overfitting. In other words, we can learn more robust representations by forcing models maintain the required sample relationship even under random mappings. This is because sometimes the relationship of samples during optimization could be satisfied by chance or overfitting. Taking random mappings as a preemptive act can alleviate such a problem to some extent, and therefore bring a more robust way to evaluate the relationship between samples. In this work, we provide interpretation to the effect of the proposed random mapping strategy from the perspective of perturbation of samples or the coordinate system. Because all random mappings are confined to satisfy the same correlation requirement, we name our method ROMA, taking the meaning from "all roads lead to Rome".

By integrating the triplet-based loss and random mapping strategy, we obtain the proposed method ROMA, which achieves a new state of the art on multiple benchmark datasets. Under the linear evaluation protocol, ROMA achieves $79.24\%$ top-1 accuracy on ImageNet-100, which is a $3.4\%$ absolute improvement over SimCLR [3]. In addition, as a plug-in and free-lunch strategy, ROMA can also consistently improve the performance of existing SSL methods.

The main contributions of this work are summarized as follows:

- We demonstrate that negative pairs are still important but one is sufficient for contrastive SSL. A simple and effective triplet-based loss is also designed accordingly.
- We propose a novel plug-in random mapping strategy to enforce SSL models to learn under random mappings, and demonstrate that unsupervised representation learning can effectively benefit from incorporating randomness.
- We experimentally demonstrate that our method ROMA achieves a new state-of-the-art accuracy of $79.24\%$ on the ImageNet-100 linear readout benchmark with ResNet-50.

## 2 Related Work

**Handcrafted pretext task learning.** In the early stage of self-supervised learning (SSL), handcrafted pretext tasks are generally designed to make models learn from the pseudo-supervision constructed by some fantastic pretext tasks. The core of this kind of methods is how to design and mine effective information inside the image beyond human semantic annotations. Representative pretext tasks include relative patch prediction [10], image denoising [11], image inpainting [12], colorization [13], jigsaw puzzles [14] and rotation prediction [15]. Although these methods have been shown to be useful for some downstream tasks, there is usually a task gap between the ad-hoc pretext task and target downstream task, limiting the generality of learned representations.

**Contrastive learning.** Recently, contrastive learning based SSL [16–20, 1–3, 21, 9, 22, 6] has drawn increasing attention owing to its simple design and excellent performance. The core idea is to discriminate among different images, by maximizing the similarity between two augmented views of the same image (positive pairs) and repulsing other different images (negative pairs), *i.e.,* contrastive learning [23]. Along this way, Dosovitskiy *et al.* [24] and Wu *et al.* [1] propose to take each patch/image as an individual class via instance-level discrimination. MoCo [2] and MoCo

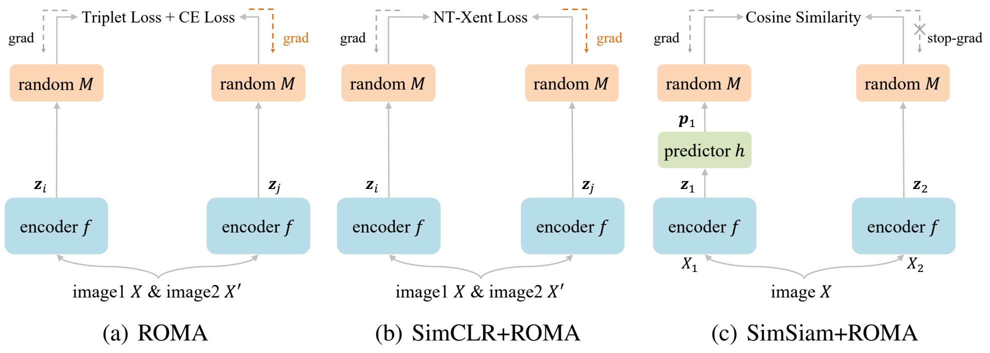

Figure 1: (a) ROMA architecture with a triplet+cross-entropy (CE) loss. (b) SimCLR architecture introducing the random mapping strategy, named SimCLR+ROMA. (c) SimSiam architecture using the random mapping strategy, named SimSiam+ROMA. All three models employ the same encoder $f$ (a backbone plus a projection MLP) in both branches except SimSiam uses an additional predictor $h$ in one branch. For each input positive image, SimSiam needs no negative examples, SimCLR requires a large number of negative examples, while ROMA only needs one negative example. A random PSD matrix $M \succeq 0$ is used to calculate the final loss, which will be discarded during test.

v2 [9] match an encoded query to a dictionary/queue of encoded keys (*i.e.,* a dictionary look-up task) with a slow-moving average network (momentum encoder) using an InfoNCE loss [21]. Differently, SimCLR [3] directly constructs negative samples within a much larger minibatch without requiring additional memory banks. Although this kind of methods performs excellently, they usually requires a large number of negative examples to work well, by using either memory banks or large batches.

**Non-contrastive learning.** Different from the above contrastive learning based SSL, recent literature has attempted to only use positive pairs and completely discard the negative examples, named non-contrastive learning based SSL [7, 5, 4, 8]. For example, BYOL [5] directly predicts the representation of one augmented view from another view of the same image with a Siamese network [25], where one branch of the network is a slow-moving average (momentum encoder) of another branch. After that, SimSiam [4] discards the momentum encoder, directly maximizes the similarity between two augmented views of one image, and demonstrates that the stop-gradient operation is critical for preventing collapsing. In addition, SwAV [7] performs online clustering to learn prototypes and indirectly compare two augmented views of the same image by using their cluster assignments built on the learned prototypes (clusters). Recently, Barlow Twins [8] maximizes the similarity between two augmented (distorted) views of one image while reducing redundancy between their components, by relying on very high-dimensional representations. Although these methods successfully discard the negative examples and thus somewhat alleviate the computation, they introduce a new collapse problem and require further effort to address this new problem.

As mentioned above, the existing contrastive SSL is confined to requiring massive negative examples, while the non-contrastive SSL easily suffers from the collapse problem. As a compromise, ROMA just employs one negative example to naturally avoid the collapse problem. More importantly, although only one negative example is used, ROMA can still achieve remarkable performance. In addition, a new plug-in random mapping strategy is proposed, which can further boost the unsupervised representation learning.

## 3 The Proposed Method

### 3.1 Triplet-based Loss Function

Following the existing contrastive and non-contrastive SSL methods [2–8], we use data augmentation to obtain the image-level pseudo labels, *i.e.,* each image is regarded as one individual class. As shown in Figure 1(a), given any two different images $\langle X, X' \rangle$ within a minibatch, we can construct a triplet $\langle X_i, \tilde{X}_i, X_j \rangle$ via data augmentation, where both $X_i$ and $\tilde{X}_i$ are augmented from $X$ (positive pairs), and $X_j$ is augmented from $X'$ (negative example). After that, this triplet will be processed by an encoder network $f$ that consists of a backbone and a projection multilayer perceptron (MLP), obtaining the corresponding feature vectors $\langle \boldsymbol{z}_i, \tilde{\boldsymbol{z}}_i, \boldsymbol{z}_j \rangle, \boldsymbol{z} \in \mathbb{R}^d$. A simple *triplet+cross-entropy loss* for a triplet can be formulated as

$$\mathcal{L}_{ij} = \left[\boldsymbol{z}_i^\top \tilde{\boldsymbol{z}}_i - \boldsymbol{z}_i^\top \boldsymbol{z}_j + \gamma\right]_+ - \lambda \cdot \log \frac{\exp(\boldsymbol{z}_i^\top \tilde{\boldsymbol{z}}_i/\tau)}{\exp(\boldsymbol{z}_i^\top \tilde{\boldsymbol{z}}_i/\tau) + \exp(\boldsymbol{z}_i^\top \boldsymbol{z}_j/\tau)} \,, \tag{1}$$

where $\boldsymbol{z}_i$, $\tilde{\boldsymbol{z}}_i$ and $\boldsymbol{z}_j$ have been $\ell_2$-normalized, $\gamma$ is a margin which is normally set as 1, and $[z]_+ = \max(0, z)$. Also, $\lambda$ is a hyper-parameter trading off the importance of the two terms of the loss, and $\tau$ denotes a temperature parameter.

As seen, the first term of Eq. (1) is a triplet loss [26], and the second term is a temperature-scaled cross-entropy loss for binary classification. Specifically, the temperature-scaled cross-entropy loss can also be seen as a *soft triplet loss* but with an adaptive margin, compensating for the limitation of the triplet loss with a fixed margin. Note that each part alone of Eq. (1) is not new and has been used in previous work [27, 28, 1–3, 6]. Nevertheless, our key contribution here is to demonstrate that such a simple triplet-based loss can achieve surprisingly good performance in unsupervised visual representation learning. This has not been clearly shown in the literature and it reveals that negative pairs are still necessary but one may be sufficient.

## 3.2 Free-lunch from Randomness

The existing contrastive and non-contrastive SSL methods normally use cosine similarity $Sim(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top \boldsymbol{v}/\|\boldsymbol{u}\|_2\|\boldsymbol{v}\|_2$ to calculate the similarity between points $\boldsymbol{u} \in \mathbb{R}^d$ and $\boldsymbol{v} \in \mathbb{R}^d$. During the optimization process of training, the relationship of points through such a measure could be satisfied by chance or due to overfitting. An interesting question is whether we can slightly modify this measure to deal with this situation so as to obtain a more robust evaluation on the relationship of points. Our answer is yes. To be specific, we assume both $\boldsymbol{u}$ and $\boldsymbol{v}$ have been $\ell_2$-normalized for simplicity. Then $Sim(\boldsymbol{u}, \boldsymbol{v})$ reduces to $\boldsymbol{u}^\top \boldsymbol{v}$ and it can be expressed as $Sim(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top I \boldsymbol{v}$, where $I$ is the identity matrix. To obtain a more robust evaluation on the relationship between $\boldsymbol{u}$ and $\boldsymbol{v}$, we replace $I$ with a random matrix $M \in \mathbb{R}^{d\times d}$ as follows,

$$Sim(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}^\top M \boldsymbol{v} \,, \tag{2}$$

In this way, we can conveniently introduce randomness into this similarity measure and then further into the objective function in Eq. (1), by using a series of random matrices $\{M_k\}|_{k=1}^n, M_k \in \mathbb{R}^{d\times d}$. More specifically, when using the proposed similarity measure in Eq. (2), the loss in Eq. (1) can be rewritten as follows,

$$\mathcal{L}_{ij}^{\text{random}} = \left[\boldsymbol{z}_i^\top M \tilde{\boldsymbol{z}}_i - \boldsymbol{z}_i^\top M \boldsymbol{z}_j + \gamma\right]_+ - \lambda \cdot \log \frac{\exp(\boldsymbol{z}_i^\top M \tilde{\boldsymbol{z}}_i/\tau)}{\exp(\boldsymbol{z}_i^\top M \tilde{\boldsymbol{z}}_i/\tau) + \exp(\boldsymbol{z}_i^\top M_i \boldsymbol{z}_j/\tau)} \,. \tag{3}$$

As will be demonstrated in the experimental study, the adoption of the random matrix $M$ consistently leads to better feature representation after the learning process. To gain better understanding on the use of $M$, we interpret its effect from the following two perspectives related to perturbation.

**Perturbation of the points.** To ensure that the similarity of two points is not caused by chance or overfitting, we can measure their similarity after perturbing the two points. Specifically, given a pair of points $\boldsymbol{u}$ and $\boldsymbol{v}$, we add a small perturbation $\Delta\boldsymbol{u} \in \mathbb{R}^d$ and $\Delta\boldsymbol{v} \in \mathbb{R}^d$ to $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively. In this case, their cosine similarity can be expressed as

$$\begin{aligned}
Sim(\boldsymbol{u}+\Delta\boldsymbol{u}, \boldsymbol{v}+\Delta\boldsymbol{v}) &= (\boldsymbol{u}+\Delta\boldsymbol{u})^\top(\boldsymbol{v}+\Delta\boldsymbol{v}) \\
&= \boldsymbol{u}^\top\boldsymbol{v} + \boldsymbol{u}^\top\Delta\boldsymbol{v} + \Delta\boldsymbol{u}^\top\boldsymbol{v} + \Delta\boldsymbol{u}^\top\Delta\boldsymbol{v} \\
&\quad \big(\textit{Assume } \Delta\boldsymbol{u} \textit{ and } \Delta\boldsymbol{v} \textit{ are related to } \boldsymbol{u} \textit{ and } \boldsymbol{v} \textit{ via a certain} \\
&\quad \textit{random linear transform } M_1 \textit{ and } M_2\textit{, respectively.}\big) \\
&\approx \boldsymbol{u}^\top\boldsymbol{v} + \boldsymbol{u}^\top M_2\boldsymbol{v} + \boldsymbol{u}^\top M_1\boldsymbol{v} \\
&\quad \big(\Delta\boldsymbol{u}^\top\Delta\boldsymbol{v} \textit{ can be omitted, since it is a higher-order term.)} \\
&= \boldsymbol{u}^\top(I + M_2 + M_1)\boldsymbol{v} \\
&\quad \big(\textit{Let } M = I + M_1 + M_2\big) \\
&= \boldsymbol{u}^\top M\boldsymbol{v} \,.
\end{aligned} \tag{4}$$

As seen, when we interpret our random matrix $M$ as $I + M_1 + M_2$, its effect can be thought of as adding a random perturbation to the points before evaluating their similarity as usual. This provides an explanation on why our insertion of random matrix $M$ could lead to better results.

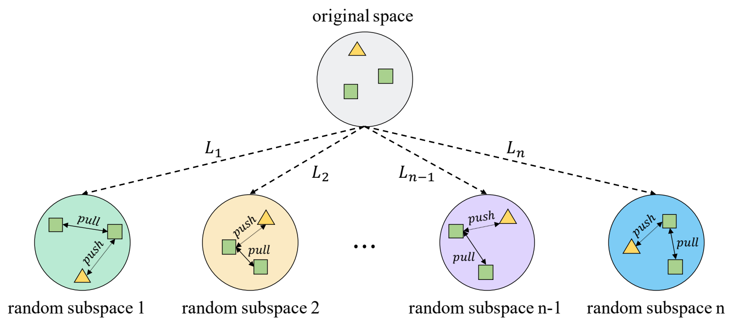

Figure 2: Illustration of random mappings for a triplet. $L_1, L_2, \ldots, L_n$ are different random transformation matrices. For each triplet, the original feature space can be randomly projected into another feature space with a random matrix $L_i$. In each of the obtained spaces, the relevant structure of the triplet, *i.e.,* that the similarity between the positive pair should be larger than the negative pair, will be enforced by pulling and pushing operations. Rectangles denote a positive pair and a triangle indicates a negative anchor.

**Perturbation of the coordinates.** In addition, we can also explain the effect of using the random matrix $M$ from a perspective of linear projection. In the above interpretation, we essentially assume the coordinate system is fixed and just perturb the points. On the contrary, we can also instead perturb the coordinate system with random rotation and scaling, *i.e.,* linear projection. In particular, if we assume $M$ is positive semi-definite (PSD) and symmetric in Eq. (2), *i.e.,* $M \succeq 0$, $M$ can be mathematically decomposed as $L^\top L$, where $L \in \mathbb{R}^{d\times d}$. After that, Eq. (2) can be converted as

$$Sim(\boldsymbol{u},\boldsymbol{v}) = \boldsymbol{u}^\top M \boldsymbol{v} = \boldsymbol{u}^\top L^\top L \boldsymbol{v} = (L\boldsymbol{u})^\top (L\boldsymbol{v})\,, \tag{5}$$

where $L$ is a random linear transformation matrix, randomly mapping the data points into a new (perturbed) coordinate system. In the new system, cosine similarity is then evaluated. By employing multiple different transformation matrices in Eq. (3), we hope to obtain a more reliable evaluation on the relationship of the two points. In this sense, we essentially advocate a "random mapping preemptive measure" for SSL, *i.e.,* to avoid the relationship among points from being satisfied by chance or due to overfitting, we shall take a more aggressive approach to evaluate their relationship. In other words, a stable similarity or dissimilarity relationship shall survive random perturbation. Considering that directly generating a random PSD matrix $M$ is difficult due to the requirement of the property of positive semi-definiteness, we instead randomly generate the linear projection matrix $\{L_k\}|_{k=1}^n$, making each $L_k^\top L_k$ naturally positive semi-definite. The illustration of random mappings for a triplet can be seen in Figure 2. In addition, the pseudo-code of ROMA is summarized in Algorithm 1.

## 3.3 Generality of ROMA

We highlight that the random mapping strategy is a general strategy. In other words, as a plug-in strategy, it can be applied to other state-of-the-art SSL methods, no matter they are contrastive based or non-contrastive based. Taking SimCLR [3] as an example, when the random mappings are incorporated, named *SimCLR+ROMA*, its random-based NT-Xent loss can be easily formulated as

$$\mathcal{L}_{ij}^{\text{SimCLR+ROMA}} = -\log \frac{\exp(\boldsymbol{z}_i^\top M \tilde{\boldsymbol{z}}_i/\tau)}{\sum_{j=1}^{2N} \mathbb{I}_{j\neq i} \exp(\boldsymbol{z}_i^\top M \boldsymbol{z}_j/\tau)}\,, \tag{6}$$

where $\mathbb{I}_{k\neq i}$ is an indicator function, $N$ is batch size, and $M$ is just a random PSD matrix. Its architecture can be seen in Figure 1(b).

Similarly, introducing the random mappings into SimSiam [4], we can obtain *SimSiam+ROMA* and its symmetrized cosine similarity loss can be easily converted as

$$\mathcal{L}^{\text{SimSiam+ROMA}} = -\frac{1}{2}\boldsymbol{p}_1^\top M \boldsymbol{z}_2 - \frac{1}{2}\boldsymbol{p}_2^\top M \boldsymbol{z}_1\,, \tag{7}$$

where $\langle \boldsymbol{z}_1, \boldsymbol{z}_2\rangle$ is a positive pair, outputs of the encoder $f$, and $\langle \boldsymbol{p}_1, \boldsymbol{p}_2\rangle$ is the subsequent representations of $\langle \boldsymbol{z}_1, \boldsymbol{z}_2\rangle$ by further processing via an additional predictor $h$. See Figure 1(c) for a more intuitive understanding. In addition, both SimCLR+ROMA and SimSiam+ROMA will be further investigated and discussed in the experimental part. Please see Section 5.1 for more details.

**Algorithm 1** ROMA Pseudocode, Pytorch-like

```
# f: backbone + projection mlp
for $X$, $X'$ in loader :                                # load two samples
    x1, x2, x3 = aug ($X$), aug ($X$), aug ($X'$)   # random augmentation, obtaining triplet
    z1, z2, z3 = f (x1), f (x2), f (x3)             # projections, N-by-D
    random_matrix = randn (D, D)                     # random matrix: D-by-D
    loss = L (z1, z2, z3, random_matrix)             # loss
    loss.backward ()                                 # back-propagate
    update (f)                                       # SGD update

def L (z1, z2, z3, random_matrix) :
    z1 = normalize (z1 * random_matrix, dim=1)  # random projection and l2-normalize
    z2 = normalize (z2 * random_matrix, dim=1)
    z3 = normalize (z3 * random_matrix, dim=1)

    pos_sim = (z1 * z2).sum(dim=1)              # positive pair similarity
    neg_sim = (z1 * z3).sum(dim=1)              # negative pair similarity

    Triplet_loss = clamp (margin + neg_sim - pos_sim, min=0.) # Triplet loss

    logits = cat ([pos_sim, neg_sim]) / temperature          # Scale by temperature
    labels = [1, 0]
    CE_loss = cross_entropy (logits, labels)                 # Cross entropy loss

    return Triplet_loss + weight * CE_loss
```

## 4 Implementation Details

**Datasets.** The main experiments are conducted on four benchmark datasets, *i.e.,* CIFAR-10 [29], CIFAR-100 [29], STL-10 [30] and ImageNet-100 [22]. **CIFAR-10** consists of 60000 $32 \times 32$ colour images in 10 classes, with 6000 images in each class. Following the original splits, we take 50000 and 10000 images for training (without labels) and test, respectively. **CIFAR-100** has 100 classes, containing 500 training images and 100 test images per class. We take 50000 images ignoring the labels for training, and take the remaining images for test. **STL-10** is an image recognition dataset with 10 classes, where there are $100,000$ $96 \times 96$ unlabeled training images, and 500 labeled training images and 800 test images in each class. In addition, because we are not able to have access to sufficient computing resources, we do not perform experiments on the commonly used ImageNet-1k [31]. Instead, we use **ImageNet-100**, a subset of ImageNet-1k with 100 well-balanced classes, as a substitute. This dataset is first introduced in [22], which contains around $0.13$ million images.

**Image augmentations.** For a fair comparison, we use the same data augmentations as SimCLR [3], consisting of random cropping and resizing, horizontal flipping, color distortion which involves a random sequence of brightness, contrast, saturation, hue adjustments, and an optional grayscale conversion. In addition, Gaussian blurring and solarization are also randomly applied. Please refer to SimCLR [3] for more details.

**Architectures.** We utilize *ResNet-50* [32] and CIFAR variant of *ResNet-18* network followed by a MLP-based projector network as our encoder $f$. The MLP-based projector network consists of three fully-connected (FC) layers, each with $2048$ output dimensions. Specifically, the first two FC layers are followed by a batch normalization (BN) layer [33] and a leaky rectified linear unit (LeakyReLU) layer with $0.2$ negative slope [34], while only the BN layer is used in the last FC layer. Furthermore, different from the mainstream methods [3, 5, 4], we add a random projection matrix with size of $2048 \times 1024$, *i.e.,* $L \in \mathbb{R}^{2048\times1024}$, after the encoder module to learn more robust representations.

**Optimization.** We use SGD optimizer with a momentum of $0.9$ and a cosine decay learning rate schedule to optimize our model. We train our model for 200 epochs with the base learning rate of $0.1$, batch size of 128 and weight decay of $1e-4$ on the ImageNet-100 dataset. For the CIFAR-10, STL-10 and CIFAR-100 datasets, the models are trained for $1000$ epochs with the base learning rate of $0.03$, batch size of 64 and weight decay of $5e-4$. Additionally, each learning rate is linearly scaled with the batch size (*i.e.,* LearningRate = BaseLearningRate $\times$ BatchSize/256). The hyper-parameters $\gamma$, $\lambda$ and $\tau$ in Eq. (3) are set to 1, 8 and 0.5, respectively. For the random mapping setting, a linear transformation matrix $L$ with size of $2048 \times 1024$ is randomly sampled from a standard normal distribution for ROMA.

**Linear evaluation.** Following the literature [13, 20, 3, 5, 4], we employ a linear evaluation protocol to evaluate the features learned by our model. To be specific, given the pre-trained ResNet backbone

| DataSet | Image Size | Method | Batch Size | BackBone | Neg. Pair | Stop-Grad | Pred. | Acc. (%) |
|---|---|---|---|---|---|---|---|---|
| *CIFAR-10* | $32 \times 32$ | SimCLR [3] | 512 | *ResNet-18* | ✓ | | | 91.29 |
| | | **SimCLR+ROMA** | 512 | *ResNet-18* | ✓ | | | 91.65 ($\uparrow$ 0.36) |
| | | MoCo v2 [9] | 512 | *ResNet-18* | ✓ | | | 89.79 |
| | | SimSiam [4] | 512 | *ResNet-18* | | ✓ | ✓ | 91.26 |
| | | **SimSiam+ROMA** | 512 | *ResNet-18* | | ✓ | ✓ | 91.72 ($\uparrow$ 0.46) |
| | | **ROMA w/o Random** | 64 | *ResNet-18* | ✓ | | | **91.85** |
| | | **ROMA** | 64 | *ResNet-18* | ✓ | | | **92.38** |
| *CIFAR-100* | $32 \times 32$ | SimCLR [3] | 512 | *ResNet-18* | ✓ | | | 63.11 |
| | | **SimCLR+ROMA** | 512 | *ResNet-18* | ✓ | | | 63.19 ($\uparrow$ 0.08) |
| | | MoCo v2 [9] | 512 | *ResNet-18* | ✓ | | | 60.46 |
| | | SimSiam [4] | 512 | *ResNet-18* | | ✓ | ✓ | 63.96 |
| | | **SimSiam+ROMA** | 512 | *ResNet-18* | | ✓ | ✓ | 65.47 ($\uparrow$ 1.51) |
| | | **ROMA w/o Random** | 64 | *ResNet-18* | ✓ | | | **65.89** |
| | | **ROMA** | 64 | *ResNet-18* | ✓ | | | **66.60** |
| *STL-10* | $64 \times 64$ | SimCLR [3] | 512 | *ResNet-50* | ✓ | | | 87.34 |
| | | **SimCLR+ROMA** | 512 | *ResNet-50* | ✓ | | | 87.55 ($\uparrow$ 0.21) |
| | | MoCo v2 [9] | 512 | *ResNet-50* | ✓ | | | 86.19 |
| | | SimSiam [4] | 256 | *ResNet-50* | | ✓ | ✓ | 85.12 |
| | | **SimSiam+ROMA** | 256 | *ResNet-50* | | ✓ | ✓ | 85.74 ($\uparrow$ 0.62) |
| | | **ROMA w/o Random** | 128 | *ResNet-50* | ✓ | | | **87.79** |
| | | **ROMA** | 128 | *ResNet-50* | ✓ | | | **88.08** |
| *ImageNet-100* | $224 \times 224$ | SimCLR [3] | 256 | *ResNet-50* | ✓ | | | 75.82 |
| | | **SimCLR+ROMA** | 256 | *ResNet-50* | ✓ | | | 76.58 ($\uparrow$ 0.76) |
| | | MoCo v2 [9] | 256 | *ResNet-50* | ✓ | | | 70.10 |
| | | SimSiam [4] | 256 | *ResNet-50* | | ✓ | ✓ | 73.33 |
| | | **SimSiam+ROMA** | 256 | *ResNet-50* | | ✓ | ✓ | 74.14 ($\uparrow$ 0.81) |
| | | **ROMA w/o Random** | 128 | *ResNet-50* | ✓ | | | **77.14** |
| | | **ROMA** | 128 | *ResNet-50* | ✓ | | | **79.24** |

Table 1: Linear evaluation (top-1) results on four benchmark datasets. Evaluation is on a single center crop. All unsupervised methods on CIFAR-10, CIFAR-100 and STL-10 are trained for 1000 epochs, and are trained for 200 epochs on ImageNet-100. *Neg. pair* indicates whether negative pairs are used. *Pred.* denotes whether an additional predictor network is used.

after unsupervised learning, a linear classifier is further trained on top of the frozen backbone and a global average pooling layer. In our experiments, the linear classifier is trained for 100 epochs using a SGD optimizer in a cosine decay schedule, where the base learning rate is 30, weight decay is 0, momentum is 0.9 and batch size is 128. After training, we perform the evaluation on the center cropped images in the evaluation/test set.

## 5 Experiments

In this section, we will first conduct experiments on four benchmark datasets to compare our proposed method with the state-of-the-art methods. Meanwhile, we will apply ROMA into SimCLR and SimSiam to verify the effectiveness of the proposed random mapping strategy. Finally, we conduct extensive ablation studies to reveal the impacts of different components in ROMA. Eight Nvidia GTX 2080Ti GPUs are used for all experiments.

### 5.1 Comparison with the State of the Art

We compare ROMA with the close related state-of-the-art methods, *i.e.,* SimCLR [3], MoCo v2 [9] and SimSiam [4], on four benchmark datasets, including *CIFAR-10*, *CIFAR-100*, *STL-10* and *ImageNet-100*. Two variants of ROMA are further constructed by introducing the random mapping strategy into SimCLR and SimSiam, named *SimCLR+ROMA* and *SimSiam+ROMA*, respectively. In addition, to verify the effectiveness of the proposed triplet-based loss in Eq. (1), another variant of ROMA is also constructed by removing the random mappings, named *ROMA w/o Random*. Following the literature [22, 3, 4], we adopt a *ResNet-50* backbone on STL-10 and ImageNet-100 datasets, and adopt a *ResNet-18* backbone on CIFAR-10 and CIFAR-100 datasets for all comparison methods. *For fairness, we reimplement all the comparison methods into the same framework being faithful to the original paper*. The linear evaluation results on top-1 accuracy are reported in Table 1.

**Effectiveness of triplet-based loss.** One of our concerns is whether the negative pairs are unnecessary in unsupervised visual representation learning. From the results in Table 1, we can see that ROMA w/o Random (which only uses the proposed triplet-based loss) can achieve remarkable performance on all benchmark datasets. For example, on the CIFAR-10 dataset, ROMA w/o Random can obtain $0.56\%$, $2.06\%$ and $0.59\%$ improvements over SimCLR, MoCo v2 and SimSiam, respectively. More importantly, ROMA w/o Random adopts a much smaller batch size (64), compared to other competitors (512). Similarly, on ImageNet-100, ROMA w/o Random can even gain $1.32\%$, $7.04\%$ and $3.81\%$ improvements over SimCLR, MoCo v2 and SimSiam, respectively. It verifies that negative pairs are still necessary and one is sufficient, *i.e.,* triplet is sufficient. Also, the above concern, *i.e.,* the first question in the introduction, has been answered.

**Effectiveness of random mappings.** As seen in Table 1, SimCLR+ROMA consistently performs better than SimCLR on all benchmark datasets. Specifically, SimCLR+ROMA obtains $0.36\%$, $0.08\%$, $0.21\%$ and $0.76\%$ improvements over SimCLR on CIFAR-10, CIFAR-100, STL-10 and ImageNet-100, respectively. Similarly, SimSiam+ROMA can also consistently performs better than SimSiam, and achieves $0.46\%$, $1.51\%$, $0.62\%$ and $0.81\%$ improvements, respectively. This successfully demonstrates the effectiveness and generality of the proposed random mapping strategy. Meanwhile, the second question in the introduction can also be answered.

**Effectiveness of ROMA.** As expected, ROMA, a combination of the random mapping strategy and triplet-based loss, achieves remarkable performance on all datasets, and performs much better than other state-of-the-art competitors. For example, ROMA achieves a new state-of-the-art result of $92.38\%$ on CIFAR-10 with a much smaller batch size of 64, which is $1.09\%$, $2.59\%$ and $1.12\%$ higher than SimCLR, MoCo v2 and SimSiam with the batch size of 512, respectively. Similarly, on the more challenging CIFAR-100, ROMA still performs significantly better than other competitors, gaining $3.49\%$, $6.14\%$ and $2.64\%$ improvements over SimCLR, MoCo v2 and SimSiam, respectively. Moreover, on STL-10, ROMA consistently performs better than SimCLR, MoCo v2 and SimSiam, with improvements of $0.74\%$, $1.89\%$ and $2.96\%$, respectively. Specifically, on the more challenging ImageNet-100 dataset, ROMA can still gain significantly improvements over SimCLR, MoCo v2 and SimSiam by $3.42\%$, $9.14\%$ and $5.91\%$, respectively, even with a smaller batch size of 128.

According to the above results, we can confidently verify the effectiveness of ROMA, no matter the triplet-based loss or the random mapping strategy. Compared with SimCLR and MoCo v2 that using a large number of negative pairs, ROMA only require one negative pair for each positive anchor, but can achieve competitive results. On the other hand, compared with SimSiam, which requires using stop-gradients and an additional predictor network to alleviate the collapse problem, ROMA can naturally avoid such a collapse problem but with a much better result, owing to using one negative pair and a suitable loss function. This gives us one insight that we may need to rethink the impact of negative pairs. Another insight is that we may need to pay more attention to the randomness to learn more robust representations.

## 5.2 Ablation Studies

**Loss function.** The well-designed objective loss function, *i.e.,* triplet+cross-entropy loss in Eq. (3), is an significant component in our work. Because there are two parts in this loss function, *i.e.,* a triplet loss and a cross-entropy (CE) loss, it will be interesting to analyze the influence of each individual part. Without loss of generality, we conduct experiments on both CIFAR-10 and ImageNet-100 datasets with the same settings in Section 5.1. In particular, three variants of ROMA are constructed, *i.e., Triplet*, *CE* and *Triplet+CE*, where *Triplet* means only the triplet loss is used, *CE* indicates only cross-entropy loss is used, and *Triplet+CE* means both parts are used. Note that all the three variants still employ the random mapping strategy. From Table 2, we can see that (1) each individual loss can achieve competitive results, compared to the state-of-the-art methods (see Table 1); (2) Triplet+CE gains nearly $0.7\%$ accuracy improvement over each individual loss. Similarly, Table 3 also consistently verifies the effectiveness of the combination loss, which obtains the highest $79.24\%$ top-1 accuracy on ImageNet-100.

| | Triplet | CE | Triplet+CE |
|---|---|---|---|
| Acc.(%) | 91.67 | 91.71 | **92.38** |

Table 2: Effect of loss functions on CIFAR-10 using ResNet-18.

| | Triplet | CE | Triplet+CE |
|---|---|---|---|
| Acc.(%) | 78.12 | 77.28 | **79.24** |

Table 3: Effect of loss functions on ImageNet-100 using ResNet-50.

As have been explained in Section 3.1, the triplet-based cross-entropy loss (for binary classification) can be regarded as a supplementary to the standard triplet loss, both of which benefits the final performance of ROMA. From another perspective, the whole training phase can also be considered

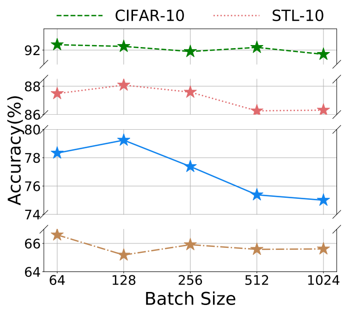

Figure 3: Effect of batch sizes.

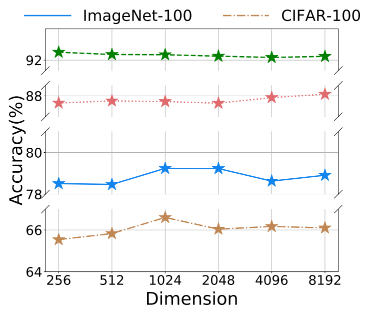

Figure 4: Effect of random output dimensions.

as a multi-task optimization process via triplet loss and cross-entropy loss, which improves the robustness of ROMA to some extent.

**Batch size.** For the contrastive SSL methods, such as SimCLR [3] and MoCo [2, 9], the excellent performance is largely dependent on large batch sizes, while the performance will suffer drops when batch size is reduced. In contrast, our ROMA is able to be more robust to smaller batch size, like some non-contrastive SSL methods, *e.g.,* SimSiam [4]. To empirically verify this point, we train our model of ROMA on four benchmark datasets using different batch sizes from 64 to 1024. As the batch size changes, the learning rate is adjusted by the equation of $LearningRate = BaseLearningRate \times BatchSize/256$. From Figure 3, we can observe that ROMA works very well with a small batch size on all four datasets, especially on CIFAR-10 and CIFAR-100. Moreover, when using a batch size of 128, ROMA performs the best on STL-10 and ImageNet-100.

**Random dimension.** The output dimension of the random projection matrix, *i.e.,* the dimension of the random subspace, is also an important factor in our method. Note that the feature dimension is usually set as 2048 in the literature. Clearly, we can not only randomly project samples into a lower-dimensional subspace (*e.g.,* 256), but also project them into a higher-dimensional subspace (*e.g.,* 8192) or just maintain the dimension (*i.e.,* 2048). To systematically study the impacts of different random dimensions, we conduct experiments on four benchmark datasets by varying the random output dimensions. The results are shown in Figure 4.

As seen, on different datasets, the impacts of output dimensions are relatively different. For example, on STL-10, a higher-dimensional projection performs better, but the difference is somewhat limited. In contrast, on CIFAR-100 and ImageNet-100, a 1024-dimensional projection performs the best. Intuitively, such an appropriate dimensionality reduction, *i.e.,* $2048 \times 1024$, could reduce some redundant features, benefitting the final performance. As shown in Figure 4, random projection matrices with the size of $2048 \times 1024$ generally obtains stable good results.

**Random frequency.** The frequency of random mapping determines how much knowledge can our method acquire from the latent subspace that generated by the random projection matrix. An inappropriate random frequency may cause our method to fail to acquire enough knowledge or waste too much time in a random subspace. Therefore, we compare the performance of using different frequencies, *i.e.* using no random (NoRandom), using random every batch (1Batch), using random every epoch (1Epoch) and using random every 10 epochs (10Epoch), on both CIFAR-10 and ImageNet-100. The results are reported in Table 4 and Table 5.

As seen, compared to NoRandom, three kinds of random frequencies can all boost the performance, which further verifies the effectiveness of the random mapping strategy. In addition, we can see that ROMA is not significantly sensitive to the random frequency. For example, on CIFAR-10, using random every 10 epochs (10Epoch), *i.e.,* a lower frequency, performs the best, while random every epoch (1Epoch), a higher frequency, achieves the highest accuracy on ImageNet-100. Therefore, in our other experiments, we use 1Epoch frequency for ImageNet-100, and 10Epoch frequency for other three datasets which are trained for 1000 epochs.

| NoRandom | 1Batch | 1Epoch | 10Epoch |
|---|---|---|---|
| 91.85 | 92.19 | 92.26 | **92.38** |

Table 4: Effect of random frequencies on CIFAR-10 using ResNet-18.

| NoRandom | 1Batch | 1Epoch | 10Epochs |
|---|---|---|---|
| 77.14 | 79.11 | **79.24** | 79.07 |

Table 5: Effect of random frequencies on ImageNet-100 using ResNet-50.

**Random strategy.** We further explore the impacts of different random strategies. Specifically, three distributions are employed, including *Bernoulli distribution (Bernoulli)*, *uniform distribution of* $U(-1, 1)$ *(Uniform)* , and *standard normal distribution (Normal)*, where the first is a discrete distribution, while the later two are continuous distributions. For example, if we employ the standard normal distribution to generate a random projection matrix $L$, it means that all entries of $L$ are identically and independently sampled from a standard normal distribution. Table 6 and Table 7 present the results on the CIFAR-10 and ImageNet-100 datasets, respectively.

From the results, we can observe that the normal distribution strategy outperforms Bernoulli and uniform strategies on both of datasets. Furthermore, we note that the continuous random strategy (*i.e.,* the Uniform and Normal distributions) performs better than the discrete random strategy (*i.e.,* the Bernoulli distribution) to some extent, which implies that the random continuous subspace is more conductive to learn more robust representations.

| | *Bernoulli* | *Uniform* | *Normal* |
|---|---|---|---|
| Acc.(%) | 91.28 | 91.57 | **92.38** |

Table 6: Effect of random strategies on CIFAR-10 using ResNet-18.

| | *Bernoulli* | *Uniform* | *Normal* |
|---|---|---|---|
| Acc.(%) | 77.66 | 78.67 | **79.24** |

Table 7: Effect of random strategies on ImageNet-100 using ResNet-50.

### 5.3 t-SNE Feature Visualization

To demonstrate the effectiveness of the proposed ROMA in a more intuitive way and show that the learned features are conducive to the classification, we visualize the feature spaces learnt by different methods in Figure 5. First, three models are trained on the CIFAR-10 dataset with a CIFAR variant of *ResNet-18* by using SimCLR, SimSiam and ROMA, respectively. After that, all test samples in CIFAR-10 are represented accordingly and then are reduced to a two-dimensional space by t-SNE. As seen, the samples are more separable in the feature space learned by ROMA than both SimCLR and SimSiam, showing that ROMA can learn better feature representations.

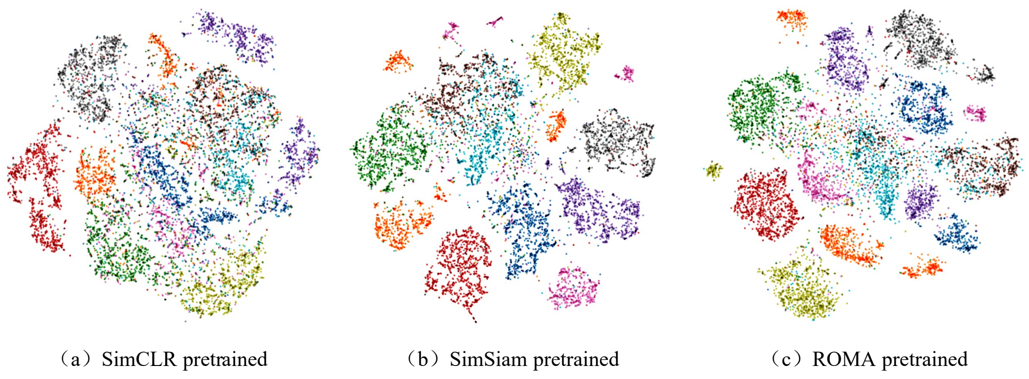

Figure 5: Feature visualizations on CIFAR-10 by t-SNE, showing the feature representation learned by SimCLR, SimSiam and ROMA, respectively. (a) SimCLR pretrained; (b) SimSiam pretrained ; (c) ROMA pretrained. Each color indicates one class.

## 6 Conclusion

In this paper, we present ROMA, a simple and effective SSL method for unsupervised visual representation learning. By using a simple triplet-based loss and a random mapping strategy, ROMA can achieve state-of-the-art results on four benchmark datasets. From the results, we have the following findings: (1) Contrary to the recent trends of discarding the negative pairs in SSL, we demonstrate that negative examples are still important but one is sufficient, only requiring a suitable loss function; (2) we show that unsupervised representation learning can gain significantly from randomness, such as using random mappings. We hope our work will draw the community to rethinking the impact of negative examples and paying attention to randomness.

## References

[1] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 3733–3742, 2018.

[2] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 9729–9738, 2020.

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the international conference on machine learning (ICML)*, pages 1597–1607, 2020.

[4] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2021.

[5] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. In *Proceedings of the conference on neural information processing systems (NeurIPS)*, 2020.

[6] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning. In *Proceedings of the conference on neural information processing systems (NeurIPS)*, 2020.

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Proceedings of the conference on neural information processing systems (NeurIPS)*, 2020.

[8] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv:2103.03230*, 2021.

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

[10] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, pages 1422–1430, 2015.

[11] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the international conference on machine learning (ICML)*, pages 1096–1103, 2008.

[12] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 2536–2544, 2016.

[13] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 649–666. Springer, 2016.

[14] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European conference on computer vision (ECCV)*, pages 69–84, 2016.

[15] Nikos Komodakis and Spyros Gidaris. Unsupervised representation learning by predicting image rotations. In *Proceedings of the international conference on learning representations (ICLR)*, 2018.

[16] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Proceedings of the conference on neural information processing systems (NeurIPS)*, 2019.

[17] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 6210–6219, 2019.

[18] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *Proceedings of the international conference on machine learning (ICLR)*, 2019.

[19] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 6707–6717, 2020.

[20] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv:1807.03748*, 2018.

[21] Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. In *Proceedings of the international conference on machine learning (ICML)*, pages 4182–4192, 2020.

[22] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv:1906.05849*, 2019.

[23] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1735–1742, 2006.

[24] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Proceedings of the conference on neural information processing systems (NeurIPS)*, 2014.

[25] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delaybromley1993signatureneural network. In *Proceedings of the conference on neural information processing systems (NeurIPS)*, volume 6, pages 737–744, 1993.

[26] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009.

[27] Omkar M Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. 2015.

[28] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *Proceedings of the international conference on machine learning (ICML) deep learning workshop*, volume 2. Lille, 2015.

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

[30] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the international conference on artificial intelligence and statistics (AISTATS)*, pages 215–223, 2011.

[31] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 248–255, 2009.

[32] 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 (CVPR)*, pages 770–778, 2016.

[33] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the international conference on machine learning (ICLR)*, pages 448–456, 2015.

[34] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the international conference on machine learning (ICML)*, page 3, 2013.