Unsupervised Embedding Learning via Invariant and Spreading Instance Feature

Mang Ye† Xu Zhang‡ Pong C. Yuen† Shih-Fu Chang‡

† Hong Kong Baptist University, Hong Kong ‡ Columbia University, New York

{mangye,pcyuen}@comp.hkbu.edu.hk, {xu.zhang,sc250}@columbia.edu

Abstract

This paper studies the unsupervised embedding learning problem, which requires an effective similarity measurement between samples in low-dimensional embedding space. Motivated by the positive concentrated and negative separated properties observed from category-wise supervised learning, we propose to utilize the instance-wise supervision to approximate these properties, which aims at learning data augmentation invariant and instance spread-out features. To achieve this goal, we propose a novel instance based softmax embedding method, which directly optimizes the 'real' instance features on top of the softmax function. It achieves significantly faster learning speed and higher accuracy than all existing methods. The proposed method performs well for both seen and unseen testing categories with cosine similarity. It also achieves competitive performance even without pre-trained network over samples from fine-grained categories.

1. Introduction

Deep embedding learning is a fundamental task in computer vision [14], which aims at learning a feature embedding that has the following properties: 1) *positive concentrated*, the embedding features of samples belonging to the same category are close to each other [32]; 2) *negative separated*, the embedding features of samples belonging to different categories are separated as much as possible [52]. Supervised embedding learning methods have been studied to achieve such objectives and demonstrate impressive capabilities in various vision tasks [28, 30, 53]. However, annotated data needed for supervised methods might be difficult to obtain. Collecting enough annotated data for different tasks requires costly human efforts and special domain expertise. To address this issue, this paper tackles the unsupervised embedding learning problem (a.k.a. unsupervised metric learning in [21]), which aims at learning discriminative embedding features without human annotated labels.

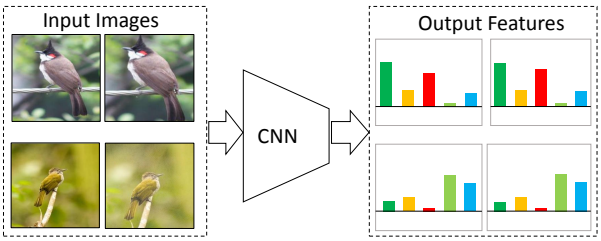

Figure 1: Illustration of our basic idea. The features of the same instance under different data augmentations should be invariant, while features of different image instances should be separated.

Unsupervised embedding learning usually requires that the similarity between learned embedding features is consistent with the visual similarity or category relations of input images. In comparison, general unsupervised feature learning usually aims at learning a good "intermediate" feature representation from unlabelled data [6, 26, 31, 34]. The learned feature is then generalized to different tasks by using a small set of labelled training data from the target task to fine-tune models (*e.g.*, linear classifier, object detector, etc.) for the target task [3]. However, the learned feature representation may not preserve visual similarity and its performance drops dramatically for similarity based tasks, *e.g.* nearest neighbor search [46, 48, 50].

The main challenge of unsupervised embedding learning is to discover visual similarity or weak category information from unlabelled samples. Iscen *et al*. [21] proposed to mine hard positive and negative samples on manifolds. However, its performance heavily relies on the quality of the initialized feature representation for label mining, which limits the applicability for general tasks. In this paper, we propose to utilize the instance-wise supervision to approximate the *positive concentrated* and *negative separated* properties mentioned earlier. The learning process only relies on instance-wise relationship and does not rely on relations between pre-defined categories, so it can be well generalized to samples of arbitrary categories that have not been seen before (*unseen testing categories*) [12].

For positive concentration: it is usually infeasible to mine reliable positive information with randomly initialized network. Therefore, we apply a random data augmentation (*e.g.*, transformation, scaling) to each image instance and use the augmented image as a positive sample. In other words, features of each image instance under different data augmentations should be invariant. *For negative separation*: since unlabelled data are usually highly imbalanced [27, 49], the number of negative samples for each image instance is much larger than that of positive samples. Therefore, a small batch of randomly selected instances can be approximately treated as negative samples for each instance. With such assumption, we try to separate each instance from all the other sampled instances within the batch, resulting in a spread-out property [52]. It is clear that such assumption may not always hold, and each batch may contain a few false negatives. However, through our extensive experiments, we observe that the spread-out property effectively improves the discriminability. In summary, our main idea is to learn a discriminative instance feature, which preserves *data augmentation invariant* and *spread-out* properties for unsupervised embedding learning, as shown in Fig. 1.

To achieve these goals, we introduce a novel instance feature-based softmax embedding method. Existing softmax embedding is usually built on classifier weights [8] or memorized features [46], which has limited efficiency and discriminability. We propose to explicitly optimize the feature embedding by directly using the inner products of instance features on top of softmax function, leading to significant performance and efficiency gains. The softmax function mines hard negative samples and takes full advantage of relationships among all sampled instances to improve the performance. The number of instance is significantly larger than the number of categories, so we introduce a Siamese network training strategy. We transform the multi-class classification problem to a binary classification problem and use maximum likelihood estimation for optimization.

The main contributions can be summarized as follows:

- We propose a novel instance feature-based softmax embedding method to learn data augmentation invariant and instance spread-out features. It achieves significantly faster learning speed and higher accuracy than all the competing methods.
- We show that both the data augmentation invariant and instance spread-out properties are important for instance-wise unsupervised embedding learning. They help capture apparent visual similarity between samples and generalizes well on unseen testing categories.
- The proposed method achieves the state-of-the-art performances over other unsupervised learning methods on comprehensive image classification and embedding learning experiments.

2. Related Work

General Unsupervised Feature Learning. Unsupervised feature learning has been widely studied in literature. Existing works can be roughly categorized into three categories [3]: 1) *generative models*, this approach aims at learning a parameterized mapping between images and predefined noise signals, which constrains the distribution between raw data and noises [46]. Bolztmann Machines (RBMs) [24, 40], Auto-encoders [20, 42] and generative adversarial network (GAN) [7, 10, 11] are widely studied. 2) *Estimating Between-image Labels,* it usually estimates between-image labels using the clustering technique [3, 9, 26] or kNN-based methods [41], which provide label information. Then label information and feature learning process are iteratively updated. 3) *Self-supervised Learning,* this approach designs pretext tasks/signals to generate "pseudo-labels" and then formulate it as a prediction task to learn the feature representations. The pretext task could be the context information of local patches [6], the position of randomly rearranged patches [31], the missing pixels of an image [34] or the color information from gray-scale images [51]. Some attempts also use video information to provide weak supervision to learn feature representations [1, 44].

As we discussed in Section 1, general unsupervised feature learning usually aims at learning a good "intermediate" feature representation that can be well generalized to other tasks. The intermediate feature representation may not preserve visual similar property. In comparison, unsupervised embedding learning requires additional visual similarity property of the learned features.

Deep Embedding Learning. Deep embedding learning usually learns an embedding function by minimizing the intra-class variation and maximizing the inter-class variation [32, 37, 45, 47]. Most of them are designed on top of pairwise [12, 30] or triplet relationships [13, 29]. In particular, several sampling strategies are widely investigated to improve the performance, such as hard mining [16], semi-hard mining [35], smart mining [13] and so on. In comparison, softmax embedding achieves competitive performance without sampling requirement [18]. Supervised learning has achieved superior performance on various tasks, but they still rely on enough annotated data.

Unsupervised Embedding Learning. According to the evaluation protocol, it can be categorized into two cases, 1) the testing categories are the same with the training categories (*seen testing categories*), and 2) the testing categories are not overlapped with the training categories (*unseen testing categories*). The latter setting is more challenging. Without category-wise labels, Iscen *et al.* [21] proposed to mine hard positive and negative samples on manifolds, and then train the feature embedding with triplet loss. However, it heavily relies on the initialized representation for label mining.

3. Proposed Method

Our goal is to learn a feature embedding network $f_\theta(\cdot)$ from a set of unlabelled images $X = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$. $f_\theta(\cdot)$ maps the input image $\mathbf{x}_i$ into a low-dimensional embedding feature $f_\theta(\mathbf{x}_i) \in \mathbb{R}^d$, where d is the feature dimension. For simplicity, the feature representation $f_\theta(\mathbf{x}_i)$ of an image instance is represented by $\mathbf{f}_i$, and we assume that all the features are ℓ_2 normalized, *i.e.* $\|\mathbf{f}_i\|_2 = 1$. A good feature embedding should satisfy: 1) the embedding features of visual similar images are close to each other; 2) the embedding features of dissimilar image instances are separated.

Without category-wise labels, we utilize the instance-wise supervision to approximate the *positive concentrated* and *negative seperated* properties. In particular, the embedding features of the same instance under different data augmentations should be invariant, while the features of different instances should be spread-out. In the rest of this section, we first review two existing instance-wise feature learning methods, and then propose a much more efficient and discriminative instance feature-based softmax embedding. Finally, we will give a detailed rationale analysis and introduce our training strategy with Siamese network.

3.1. Instance-wise Softmax Embedding

Softmax Embedding with Classifier Weights. Exemplar CNN [8] treats each image as a distinct class. Following the conventional classifier training, it defines a matrix $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_n]^T \in \mathbb{R}^{n \times d}$, where the j-th column $\mathbf{w}_j$ is called the corresponding classifier weight for the j-th instance. Exemplar CNN ensures that image instance under different image transformations can be correctly classified into its original instance with the learned weight. Based on Softmax function, the probability of sample $\mathbf{x}_j$ being recognized as the i-th instance can be represented as

$$P(i|\mathbf{x}_j) = \frac{\exp(\mathbf{w}_i^T \mathbf{f}_j)}{\sum_{k=1}^{n} \exp(\mathbf{w}_k^T \mathbf{f}_j)}. \tag{1}$$

At each step, the network pulls sample feature $\mathbf{f}_i$ towards its corresponding weight $\mathbf{w}_i$, and pushes it away from the classifier weights $\mathbf{w}_k$ of other instances. However, classifier weights prevent explicitly comparison over features, which results in limited efficiency and discriminability.

Softmax Embedding with Memory Bank. To improve the inferior efficiency, Wu *et al*. [46] propose to set up a memory bank to store the instance features $\mathbf{f}_i$ calculated in the previous step. The feature stored in the memory bank is denoted as $\mathbf{v}_i$, which serves as the classifier weight for the corresponding instance in the following step. Therefore, the probability of sample $\mathbf{x}_j$ being recognized as the i-th instance can be written as

$$P(i|\mathbf{x}_j) = \frac{\exp(\mathbf{v}_i^T \mathbf{f}_j / \tau)}{\sum_{k=1}^{n} \exp(\mathbf{v}_k^T \mathbf{f}_j / \tau)}, \tag{2}$$

where τ is the temperature parameter controlling the concentration level of the sample distribution [17]. $\mathbf{v}_i^T \mathbf{f}_j$ measures the cosine similarity between the feature $\mathbf{f}_j$ and the i-th memorized feature $\mathbf{v}_i$. For instance $\mathbf{x}_i$ at each step, the network pulls its feature $\mathbf{f}_i$ towards its corresponding memorized vector $\mathbf{v}_i$, and pushes it away from the memorized vectors of other instances. Due to efficiency issue, the memorized feature $\mathbf{v}_i$ corresponding to instance $\mathbf{x}_i$ is only updated in the iteration which takes $\mathbf{x}_i$ as input. In other words, the memorized feature $\mathbf{v}_i$ is only updated once per epoch. However, the network itself is updated in each iteration. Comparing the real-time instance feature $\mathbf{f}_i$ with the outdated memorized feature $\mathbf{v}_i$ would cumber the training process. Thus, the memory bank scheme is still inefficient.

A straightforward idea to improve the efficiency is directly optimizing over feature itself, *i.e*. replacing the weight $\{\mathbf{w}_i\}$ or memory $\{\mathbf{v}_i\}$ with $\mathbf{f}_i$. However, it is implausible due to two reasons: 1) Considering the probability $P(i|\mathbf{x}_i)$ of recognizing $\mathbf{x}_i$ to itself, since $\mathbf{f}_i^T \mathbf{f}_i$=1, *i.e*. the feature and "pseudo classifier weight" (the feature itself) are always perfectly aligned, optimizing the network will not provide any positive concentrated property; 2) It's impractical to calculate the feature of all the samples ($\mathbf{f}_k, k = 1, \ldots, n$) on-the-fly in order to calculate the denominator in Eq. (2), especially for large-scale instance number dataset.

3.2. Softmax Embedding on 'Real' Instance Feature

To address above issues, we propose a softmax embedding variant for unsupervised embedding learning, which directly optimizes the real instance feature rather than classifier weights [8] or memory bank [46]. To achieve the goal that features of the same instance under different data augmentations are invariant, while the features of different instances are spread-out, we propose to consider 1) both the original image and its augmented image, 2) a small batch of randomly selected samples instead of the full dataset.

For each iteration, we randomly sample m instances from the dataset. To simplify the notation, without loss of generality, the selected samples are denoted by $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m\}$. For each instance, a random data augmentation operation $T(\cdot)$ is applied to slightly modify the original image. The augmented sample $T(\mathbf{x}_i)$ is denoted by $\hat{\mathbf{x}}_i$, and its embedding feature $f_\theta(\hat{\mathbf{x}}_i)$ is denoted by $\hat{\mathbf{f}}_i$. Instead of considering the instance feature learning as a multi-class classification problem, we solve it as binary classification problem via maximum likelihood estimation (MLE). In particular, for instance $\mathbf{x}_i$, the augmented sample $\hat{\mathbf{x}}_i$ should be classified into instance i, and other instances $\mathbf{x}_j$, $j \neq i$ shouldn't be classified into instance i. The probability of $\hat{\mathbf{x}}_i$ being recognized as instance i is defined by

$$P(i|\hat{\mathbf{x}}_i) = \frac{\exp(\mathbf{f}_i^T \hat{\mathbf{f}}_i / \tau)}{\sum_{k=1}^{m} \exp(\mathbf{f}_k^T \hat{\mathbf{f}}_i / \tau)}. \tag{3}$$

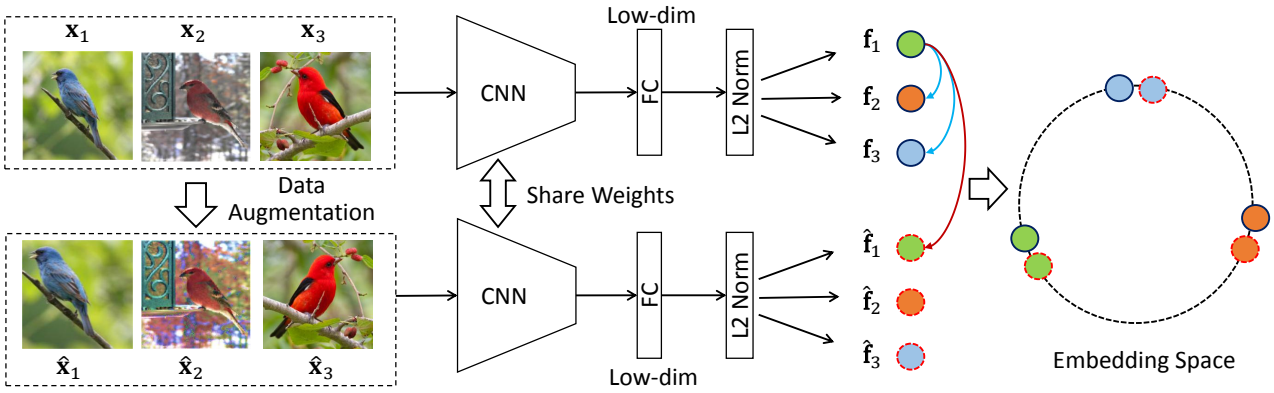

Figure 2: The framework of the proposed unsupervised learning method with Siamese network. The input images are projected into low-dimensional normalized embedding features with the CNN backbone. Image features of the same image instance with different data augmentations are invariant, while embedding features of different image instances are spread-out.

On the other hand, the probability of $\mathbf{x}_j$ being recognized as instance i is defined by

$$P(i|\mathbf{x}_j) = \frac{\exp(\mathbf{f}_i^T \mathbf{f}_j/\tau)}{\sum_{k=1}^{m} \exp(\mathbf{f}_k^T \mathbf{f}_j/\tau)}, \; j \neq i \tag{4}$$

Correspondingly, the probability of $\mathbf{x}_j$ *not* being recognized as instance i is $1 - P(i|\mathbf{x}_j)$.

Assuming different instances being recognized as instance i are independent, the joint probability of $\hat{\mathbf{x}}_i$ being recognized as instance i and $\mathbf{x}_j$, $j \neq i$ not being classified into instance i is

$$P_i = P(i|\hat{\mathbf{x}}_i) \prod_{j \neq i} (1 - P(i|\mathbf{x}_j)) \tag{5}$$

The negative log likelihood is given by

$$J_i = -\log P(i|\hat{\mathbf{x}}_i) - \sum_{j \neq i} \log(1 - P(i|\mathbf{x}_j)) \tag{6}$$

We solve this problem by minimizing the sum of the negative log likelihood over all the instances within the batch, which is denoted by

$$J = -\sum_i \log P(i|\hat{\mathbf{x}}_i) - \sum_i \sum_{j \neq i} \log(1 - P(i|\mathbf{x}_j)). \tag{7}$$

3.3. Rationale Analysis

This section gives a detailed rationale analysis about why minimizing Eq. (6) could achieve the augmentation invariant and instance spread-out feature. Minimizing Eq. (6) can be viewed as maximizing Eq. (3) and minimizing Eq. (4).

Considering Eq. (3), it can be rewritten as

$$P(i|\hat{\mathbf{x}}_i) = \frac{\exp(\mathbf{f}_i^T \hat{\mathbf{f}}_i/\tau)}{\exp(\mathbf{f}_i^T \hat{\mathbf{f}}_i/\tau) + \sum_{k \neq i} \exp(\mathbf{f}_k^T \hat{\mathbf{f}}_i/\tau)}, \tag{8}$$

Maximizing Eq. (3) requires maximizing $\exp(\mathbf{f}_i^T \hat{\mathbf{f}}_i/\tau)$ and minimizing $\exp(\mathbf{f}_k^T \hat{\mathbf{f}}_i/\tau), k \neq i$. Since all the features are ℓ_2 normalized, maximizing $\exp(\mathbf{f}_i^T \hat{\mathbf{f}}_i/\tau)$ requires increasing the inner product (cosine similarity) between $\mathbf{f}_i$ and $\hat{\mathbf{f}}_i$, resulting in a feature that is invariant to data augmentation. On the other hand, minimizing $\exp(\mathbf{f}_k^T \hat{\mathbf{f}}_i/\tau)$ ensures $\hat{\mathbf{f}}_i$ and other instances $\{\mathbf{f}_k\}$ are separated. Considering all the instances within the batch, the instances are forced to be separated from each other, resulting in the spread-out property.

Similarly, Eq. (4) can be rewritten as,

$$P(i|\mathbf{x}_j) = \frac{\exp(\mathbf{f}_i^T \mathbf{f}_j/\tau)}{\exp(\mathbf{f}_j^T \mathbf{f}_j/\tau) + \sum_{k \neq j} \exp(\mathbf{f}_k^T \mathbf{f}_j/\tau)}, \tag{9}$$

Note that the inner product $\mathbf{f}_j^T \mathbf{f}_j$ is 1 and the value of τ is generally small (say 0.1 in the experiment). Therefore, $\exp(\mathbf{f}_j^T \mathbf{f}_j/\tau)$ generally determines the value of the whole denominator. Minimizing Eq. (4) means that $\exp(\mathbf{f}_i^T \mathbf{f}_j/\tau)$ should be minimized, which aims at separating $\mathbf{f}_j$ from $\mathbf{f}_i$. Thus, it further enhances the spread-out property.

3.4. Training with Siamese Network

We proposed a Siamese network to implement the proposed algorithm as shown in Fig. 2. At each iteration, m randomly selected image instances are fed into in the first branch, and the corresponding augmented samples are fed into the second branch. Note that data augmentation is also be used in the first branch to enrich the training samples. For implementation, each sample has one randomly augmented positive sample and $2N - 2$ negative samples to compute Eq. (7), where N is the batch size. The proposed training strategy greatly reduces the computational cost. Meanwhile, this training strategy also takes full advantage of relationships among all instances sampled in a mini-batch [32]. Theoretically, we could also use a multi-branch network by considering multiple augmented images for each instance in the batch.

Methods	kNN
RandomCNN	32.1
DeepCluster (10) [3]	44.4
DeepCluster (1000) [3]	67.6
Exemplar [8]	74.5
NPSoftmax [46]	80.8
NCE [46]	80.4
Triplet	57.5
Triplet (Hard)	78.4
Ours	**83.6**

Table 1: kNN accuracy (%) on CIFAR-10 dataset.

4. Experimental Results

We have conducted the experiments with two different settings to evaluate the proposed method[1]. The first setting is that the training and testing sets share the same categories (*seen testing category*). This protocol is widely adopted for general unsupervised feature learning. The second setting is that the training and testing sets do not share any common categories (*unseen testing category*). This setting is usually used for supervised embedding learning [32]. Following [21], we don't use any semantic label in the training set. The latter setting is more challenging than the former setting and it could apparently demonstrate the quality of learned features on unseen categories.

4.1. Experiments on Seen Testing Categories

We follow the experimental settings in [46] to conduct the experiments on CIFAR-10 [23] and STL-10 [4] datasets, where training and testing set share the same categories. Specifically, ResNet18 network [15] is adopted as the backbone and the output embedding feature dimension is set to 128. The initial learning rate is set to 0.03, and it is decayed by 0.1 and 0.01 at 120 and 160 epoch. The network is trained for 200 epochs. The temperature parameter τ is set to 0.1. The algorithm is implemented on PyTorch with SGD optimizer with momentum. The weight decay parameter is 5×10^{-4} and momentum is 0.9. The training batch size is set to 128 for all competing methods on both datasets. Four kinds of data augmentation methods (*RandomResizedCrop*, *RandomGrayscale*, *ColorJitter*, *RandomHorizontalFlip*) in PyTorch with default parameters are adopted.

Following [46], we adopt weighted kNN classifier to evaluate the performance. Given a test sample, we retrieve its top-k ($k = 200$) nearest neighbors based on cosine similarity, then apply weighted voting to predict its label [46].

4.1.1 CIFAR-10 Dataset

CIFAR-10 datset [23] contains 50K training images and 10K testing images from the same ten classes. The image size are 32×32. Five methods are included for comparison: DeepCluster [3] with different cluster numbers, Exemplar CNN [8], NPSoftmax [46], NCE [46] and Triplet loss with and without hard mining. Triplet (hard) is the online hard negative sample within each batch for training [16], and the margin parameter is set to 0.5. DeepCluster [3] and NCE [46] represent the state-of-the-art unsupervised feature learning methods. The results are shown in Table 1.

[1]Code is available at https://github.com/mangye16/Unsupervised_Embedding_Learning

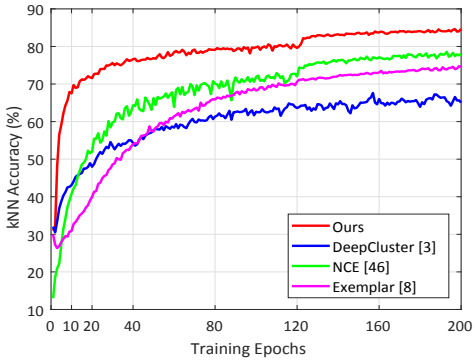

Figure 3: Evaluation of the training efficiency on CIFAR-10 dataset. kNN accuracy (%) at each epoch is reported, demonstrating the learning speed of different methods.

Classification Accuracy. Table 1 demonstrates that our proposed method achieves the best performance (83.6%) with kNN classifier. DeepCluster [3] performs well in learning good "intermediate" features with large-scale unlabelled data, but the performance with kNN classification drops dramatically. Meanwhile, it is also quite sensitive to cluster numbers, which is unsuitable for different tasks. Compared to Exemplar CNN [8] which uses the classifier weights for training, the proposed method outperforms it by 9.1%. Compared to NPSoftmax [46] and NCE [46], which use memorized feature for optimizing, the proposed method outperform by 2.8% and 3.2% respectively. The performance improvement is clear due to the idea of directly performing optimization over feature itself. Compared to triplet loss, the proposed method also outperforms it by a clear margin. The superiority is due to the hard mining nature in Softmax function.

Efficiency. We plot the learning curves of the competing methods at different epochs in Fig. 7. The proposed method takes only 2 epochs to get a kNN accuracy of 60% while [46] takes 25 epochs and [8] takes 45 epochs to reach the same accuracy. It is obvious that our learning speed is much faster than the competitors. The efficiency is guaranteed by directly optimization on instance features rather than classifier weights [8] or memory bank [46].

4.1.2 STL-10 Dataset

STL-10 dataset [4] is an image recognition dataset with colored images of size 96×96, which is widely used in unsupervised learning. Specifically, this dataset is originally designed with three splits: 1) *train*, 5K labelled images in ten

Methods	Training	Linear	kNN
RandomCNN	None	-	22.4
k-MeansNet* [5]	105K	60.1	-
HMP* [2]	105K	64.5	-
Satck* [54]	105K	74.3	-
Exemplar* [8]	105K	75.4	-
NPSoftmax [46]	5K	62.3	66.8
NCE [46]	5K	61.9	66.2
DeepCluster(100) [3]	5K	56.5	61.2
Ours	5K	69.5	74.1
Ours	105K	**77.9**	**81.6**

Table 2: Classification accuracy (%) with *linear classifier* and *kNN classifier* on STL-10 dataset. *Results are taken from [33], the baseline network is different.

classes for training, 2) *test*, 8K images from the same ten classes for testing, 3) *unlabelled*, 100K unlabelled images which share similar distribution with labelled data for unsupervised learning. We follow the same experimental setting as CIFAR-10 dataset and report classification accuracy (%) with both Linear Classifier (*Linear*) and kNN classier (*kNN*) in Table 2. Linear classifier means training a SVM classifier on the learned features and the labels of training samples. The classifier is used to predict the label of test samples. We implement NPSoftmax [46], NCE [46] and DeepCluster [3] (cluster number 100) under the same settings with their released code. By default, we only use 5K training images without using labels for training. The performances of some state-of-the-art unsupervised methods (k-MeansNet [5], HMP [2], Satck [54] and Exemplar [8]) are also reported. Those results are taken from [33].

As shown in Table 2 , when only using 5K training images for learning, the proposed method achieves the best accuracy with both classifiers (kNN: 74.1%, Linear: 69.5%), which are much better than NCE [46] and DeepCluster [3] under the same evaluation protocol. Note that kNN measures the similarity directly with the learned features and *Linear* requires additional classifier learning with the labelled training data. When 105K images are used for training, the proposed method also achieves the best performance for both kNN classifier and linear classifier. In particular, the kNN accuracy is 74.1% for 5K training images, and it increases to 81.6% for full 105K training images. The classification accuracy with linear classifier also increases from 69.5% to 77.9%. This experiment verifies that the proposed method can benefit from more training samples.

4.2. Experiments on Unseen Testing Categories

This section evaluates the discriminability of the learned feature embedding when the semantic categories of training samples and testing samples are not overlapped. We follow the experimental settings described in [32] to conduct experiments on CUB200-2011(*CUB200*) [43], Stanford Online Product (*Product*) [32] and Car196 [22] datasets. No semantic label is used for training. Caltech-UCSD Birds

Methods	R@1	R@2	R@4	R@8	NMI
Initial (FC)	39.2	52.1	66.1	78.2	51.4
	Supervised Learning				
Lifted [32]	43.6	56.6	68.6	79.6	56.5
Clustering[38]	48.2	61.4	71.8	81.9	59.2
Triplet+ [13]	45.9	57.7	69.6	79.8	58.1
Smart+ [13]	49.8	62.3	74.1	83.3	59.9
	Unsupervised Learning				
Cyclic [25]	40.8	52.8	65.1	76.0	52.6
Exemplar [8]	38.2	50.3	62.8	75.0	45.0
NCE [46]	39.2	51.4	63.7	75.8	45.1
DeepCluster[3]	42.9	54.1	65.6	76.2	53.0
MOM [21]	45.3	57.8	68.6	78.4	55.0
Ours	**46.2**	**59.0**	**70.1**	**80.2**	**55.4**

Table 3: Results (%) on CUB200 dataset.

Methods	R@1	R@10	R@100	NMI
Initial (FC)	40.8	56.7	72.1	84.0
Exemplar [8]	45.0	60.3	75.2	85.0
NCE [46]	46.6	62.3	76.8	85.8
DeepCluster[3]	34.6	52.6	66.8	82.8
MOM [21]	43.3	57.2	73.2	84.4
Ours	**48.9**	**64.0**	**78.0**	**86.0**

Table 4: Results (%) on *Product* dataset.

200 (*CUB200*) [43] is a fine-grained bird dataset. Following [32], the first 100 categories with 5,864 images are used for training, while the other 100 categories with 5,924 images are used for testing. Stanford Online Product (*Product*) [32] is a large-scale fine-grained product dataset. Similarly, 11,318 categories with totally 59,551 images are used for training, while the other 11,316 categories with 60,502 images are used for testing. Cars (*Car196*) dataset [22] is a fine-grained car category dataset. The first 98 categories with 8,054 images are used for training, while the other 98 categories with 8,131 images are used for testing.

Implementation Details. We implement the proposed method on PyTorch. The pre-trained Inception-V1 [39] on ImageNet is used as the backbone network following existing methods [30, 32, 37]. A 128-dim fully connected layer with ℓ_2 normalization is added after the pool5 layer as the feature embedding layer. All the input images are firstly resized to 256×256. For data augmentation, the images are randomly cropped at size 227×227 with random horizontal flipping following [21, 30]. Since the pre-trained network performs well on CUB200 dataset, we randomly select the augmented instance and its corresponding nearest instance as positive. In testing phase, a single center-cropped image is adopted for fine-grained recognition as in [30]. We adopt the SGD optimizer with 0.9 momentum. The initial learning rate is set to 0.001 without decay. The temperature parameter τ is set to 0.1. The training batch size is set to 64.

Evaluation Metrics. Following existing works on supervised deep embedding learning [13, 32], the retrieval performance and clustering quality of the testing set are evaluated. Cosine similarity is adopted for similarity mea-

Methods	R@1	R@2	R@4	R@8	NMI
Initial (FC)	35.1	47.4	60.0	72.0	38.3
Exemplar [8]	36.5	48.1	59.2	71.0	35.4
NCE [46]	37.5	48.7	59.8	71.5	35.6
DeepCluster[3]	32.6	43.8	57.0	69.5	38.5
MOM [21]	35.5	48.2	60.6	72.4	**38.6**
Ours	**41.3**	**52.3**	**63.6**	**74.9**	35.8

Table 5: Results (%) on Car196 dataset.

surement. Given a query image from the testing set, $R@K$ measures the probability of any correct matching (with same category label) occurs in the top-k retrieved ranking list [32]. The average score is reported for all testings samples. Normalized Mutual Information (NMI) [36] is utilized to measure the clustering performance of the testing set.

Comparison to State-of-the-arts. The results of all the competing methods on three datasets are listed in Table 3, 4 and 5, respectively. MOM [21] is the only method that claims for unsupervised metric learning. We implement the other three state-of-the-art unsupervised methods (Exemplar [8], NCE [46] and DeepCluster [3]) on three datasets with their released code under the same setting for fair comparison. Note that these methods are originally evaluated for general unsupervised feature learning, where the training and testing set share the same categories. We also list some results of supervised learning (originate from [21]) on CUB200 dataset as shown in Table 3.

Generally, the instance-wise feature learning methods (NCE [46], Examplar [8], Ours) outperform non-instance-wise feature learning methods (DeepCluster [3], MOM [21]), especially on *Car196* and *Product* datasets, which indicates instance-wise feature learning methods have good generalization ability on unseen testing categories. Among all the instance-wise feature learning methods, the proposed method is the clear winner, which also verifies the effectiveness of directly optimizing over feature itself. Moreover, the proposed unsupervised learning method is even competitive to some supervised learning methods on CUB200 dataset.

Qualitative Result. Some retrieved examples with cosine similarity on CUB200 dataset at different training epochs are shown in Fig. 4. The proposed algorithm can iteratively improve the quality of the learned feature and retrieve more correct images. Although there are some wrongly retrieved samples from other categories, most of the top retrieved samples are visually similar to the query.

Training from Scratch. We also evaluate the performance using a network (ResNet18) without pre-training. The results on the large-scale *Product* dataset are shown in Table 6. The proposed method is also a clear winner. Interestingly, MOM [21] fails in this experiment. The main reason is that the feature from randomly initialized network provides limited information for label mining. Therefore, MOM cannot estimate reliable labels for training.

Methods	R@1	R@10	R@100	NMI
Random	18.4	29.4	46.0	79.8
Exemplar [8]	31.5	46.7	64.2	82.9
NCE [46]	34.4	49.0	65.2	84.1
MOM [21]	16.3	27.6	44.5	80.6
Ours	**39.7**	**54.9**	**71.0**	**84.7**

Table 6: Results (%) on *Product* dataset using network without pre-trained parameters.

4.3. Ablation Study

The proposed method imposes two important properties for instance feature learning: data augmentation invariant and instance spread-out. We conduct ablation study to show the effectiveness of each property on CIFAR-10 dataset.

Strategy	Full	w/o R	w/o G	w/o C	w/o F
kNN Acc (%)	83.6	56.2	79.3	75.7	82.6

Table 7: Effects of each data augmentation operation on CIFAR-10 dataset. 'w/o': Without. 'R': *RandomResizedCrop*, 'G': *RandomGrayscale*, 'C': *ColorJitter*, 'F': *RandomHorizontalFlip*.

Strategy	Full	No DA	Hard	Easy
kNN Acc (%)	83.6	37.4	83.2	57.5

Table 8: Different sampling strategies on CIFAR-10 dataset.

To show the importance of data augmentation invariant property, we firstly evaluate the performance by removing each of the operation respectively from the data augmentation set. The results are shown in Table 7. We observe that all listed operations contribute to the remarkable performance gain achieved by the proposed algorithm. In particular, *RandomResizedCrop* contributes the most. We also evaluate the performance without data augmentation (*No DA*) in Table 8, and it shows that performance drops significantly from 83.6% to 37.4%. It is because when training without data augmentation, the network does not create any positive concentration property. The features of visually similar images are falsely separated.

To show the importance of spread-out property, we evaluated two different strategies to choose negative samples: 1) selecting the top 50% instance features that are similar to query instance as negative (hard negative); 2) selecting the bottom 50% instance features that are similar to query instance as negative (easy negative). The results are shown as "Hard" and "Easy" in Table 8. The performance drops dramatically when only using the *easy negative*. In comparison, the performance almost remains the same as the full model when only using *hard negative*. It shows that separating hard negative instances helps to improve the discriminability of the learned embedding.

4.4. Understanding of the Learned Embedding

We calculate the cosine similarity between the query feature and its 5NN features from the same category (*Positive*)

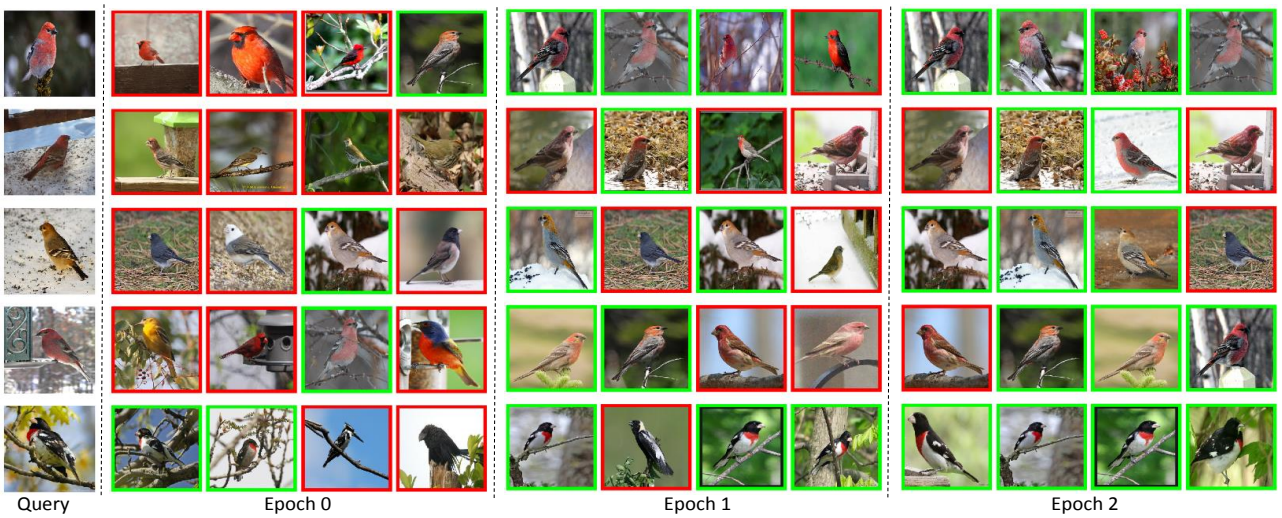

Figure 4: 4NN retrieval results of some example queries on CUB200-2011 dataset. The positive (negative) retrieved results are framed in green (red). The similarity is measured with cosine similarity.

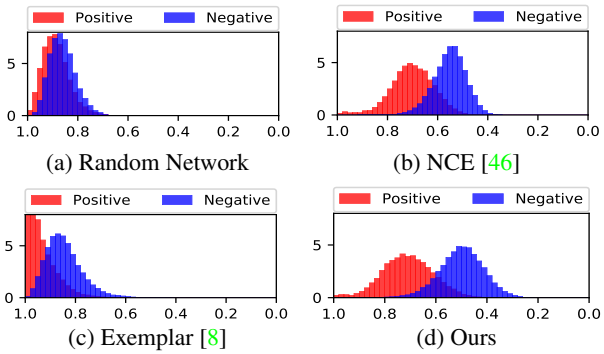

Figure 5: The cosine similarity distributions on CIFAR-10 [23]

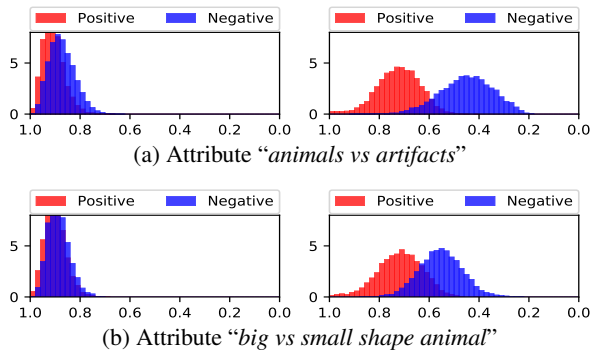

Figure 6: The cosine similarity distributions of randomly initialized network (left column) and our learned model (right column) with different attributes on CIFAR-10 [23].

as well as 5NN features from different categories (*Negative*). The distributions of the cosine similarity of different methods are shown in Fig. 5. A more separable distribution indicates a better feature embedding. It shows that the proposed method performs best to separate positive and negative samples. We could also observe that our learned feature preserves the best spread-out property.

It is interesting to show how the learned instance-wise feature helps the category label prediction. We report the cosine similarity distribution based on other category definitions (attributes in [19]) instead of semantic label in Fig. 6. The distribution clearly shows that the proposed method also performs well to separate other attributes, which demonstrates the generalization ability of the learned feature.

5. Conclusion

In this paper, we propose to address the unsupervised embedding learning problem by learning a data augmentation invariant and instance spread-out feature. In particular, we propose a novel instance feature based softmax embedding trained with Siamese network, which explicitly pulls the features of the same instance under different data augmentations close and pushes the features of different instances away. Comprehensive experiments show that directly optimizing over instance feature leads to significant performance and efficiency gains. We empirically show that the spread-out property is particularly important and it helps capture the visual similarity among samples.

Acknowledgement

This work is partially supported by Research Grants Council (RGC/HKBU12200518), Hong Kong. This work is partially supported by the United States Air Force Research Laboratory (AFRL) and the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-16-C-0166. Any opinions, findings and conclusions or recommendations expressed in this material are solely the responsibility of the authors and does not necessarily represent the official views of AFRL, DARPA, or the U.S. Government.

References

[1] Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *ICCV*, pages 37–45, 2015. 2

[2] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for rgb-d based object recognition. In *Experimental Robotics*, pages 387–402. Springer, 2013. 6

[3] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018. 1, 2, 5, 6, 7

[4] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, pages 215–223, 2011. 5

[5] Adam Coates and Andrew Y Ng. Selecting receptive fields in deep networks. In *NIPS*, pages 2528–2536, 2011. 6

[6] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, pages 1422–1430, 2015. 1, 2

[7] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016. 2

[8] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *PAMI*, 38(9):1734–1747, 2016. 2, 3, 5, 6, 7, 8

[9] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, pages 766–774, 2014. 2

[10] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016. 2

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 2

[12] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 1, 2

[13] Ben Harwood, BG Kumar, Gustavo Carneiro, Ian Reid, Tom Drummond, et al. Smart mining for deep metric learning. In *ICCV*, pages 2821–2829, 2017. 2, 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, pages 1026–1034, 2015. 1

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[16] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 2, 5

[17] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 3

[18] Shota Horiguchi, Daiki Ikami, and Kiyoharu Aizawa. Significance of softmax-based features in comparison to distance metric learning-based features. *arXiv preprint arXiv:1712.10151*, 2017. 2

[19] Chen Huang, Chen Change Loy, and Xiaoou Tang. Unsupervised learning of discriminative attributes and visual representations. In *CVPR*, pages 5175–5184, 2016. 8

[20] Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, pages 1–8, 2007. 2

[21] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Mining on manifolds: Metric learning without labels. 2018. 1, 2, 5, 6, 7

[22] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCVW*, pages 554–561, 2013. 6

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 5, 8

[24] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pages 609–616, 2009. 2

[25] Dong Li, Wei-Chih Hung, Jia-Bin Huang, Shengjin Wang, Narendra Ahuja, and Ming-Hsuan Yang. Unsupervised visual representation learning by graph-based consistent constraints. In *ECCV*, pages 678–694, 2016. 6

[26] Renjie Liao, Alex Schwing, Richard Zemel, and Raquel Urtasun. Learning deep parsimonious representations. In *NIPS*, pages 5076–5084, 2016. 1, 2

[27] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE TPAMI*, 38(3):447–461, 2016. 2

[28] Chaochao Lu and Xiaoou Tang. Surpassing human-level face verification performance on lfw with gaussianface. In *AAAI*, pages 3811–3819, 2015. 1

[29] R Manmatha, Chao-Yuan Wu, Alexander J Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *ICCV*, pages 2859–2867, 2017. 2

[30] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *ICCV*, pages 360–368, 2017. 1, 2, 6

[31] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, pages 69–84, 2016. 1, 2

[32] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, pages 4004–4012, 2016. 1, 2, 4, 5, 6, 7

[33] Edouard Oyallon, Eugene Belilovsky, and Sergey Zagoruyko. Scaling the scattering transform: Deep hybrid networks. In *ICCV*, pages 5619–5628, 2017. 6

[34] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, pages 2536–2544, 2016. 1, 2

[35] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015. 2

[36] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press, 2008. 7

[37] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, pages 1857–1865, 2016. 2, 6

[38] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *CVPR*, pages 2206–2214, 2017. 6

[39] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 6

[40] Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Robust boltzmann machines for recognition and denoising. In *CVPR*, pages 2264–2271, 2012. 2

[41] Daniel Tarlow, Kevin Swersky, Laurent Charlin, Ilya Sutskever, and Rich Zemel. Stochastic k-neighborhood selection for supervised and unsupervised learning. In *ICML*, pages 199–207, 2013. 2

[42] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008. 2

[43] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 6

[44] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, pages 2794–2802, 2015. 2

[45] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, pages 499–515, 2016. 2

[46] Zhirong Wu, Yuanjun Xiong, X Yu Stella, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pages 3733–3742, 2018. 1, 2, 3, 5, 6, 7, 8

[47] Tong Xiao, Hongsheng Li, Wanli Ouyang, and Xiaogang Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *CVPR*, pages 1249–1258, 2016. 2

[48] Mang Ye, Xiangyuan Lan, and Pong C. Yuen. Robust anchor embedding for unsupervised video person re-identification in the wild. In *ECCV*, pages 170–186, 2018. 1

[49] Mang Ye, Jiawei Li, Andy J Ma, Liang Zheng, and Pong C. Yuen. Dynamic graph co-matching for unsupervised video-based person re-identification. In *IEEE Transactions on Image Processing (TIP)*, 2019. 2

[50] Mang Ye, Andy J Ma, Liang Zheng, Jiawei Li, and Pong C. Yuen. Dynamic label graph matching for unsupervised video re-identification. In *ICCV*, pages 5142–5150, 2017. 1

[51] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, pages 649–666, 2016. 2

[52] Xu Zhang, X Yu Felix, Sanjiv Kumar, and Shih-Fu Chang. Learning spread-out local feature descriptors. In *ICCV*, pages 4605–4613, 2017. 1, 2

[53] Xuan Zhang, Hao Luo, Xing Fan, Weilai Xiang, Yixiao Sun, Qiqi Xiao, Wei Jiang, Chi Zhang, and Jian Sun. Alignedreid: Surpassing human-level performance in person re-identification. *arXiv preprint arXiv:1711.08184*, 2017. 1

[54] J Zhao, M Mathieu, R Goroshin, and Y Lecun. Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*, 2015. 6

Figure 7: Random selected retrieved examples on CUB200 dataset with the learnt instance features. Red denotes wrong labels and green represents correct labels.