Uncertainty-guided Metric Learning without Labels

Dhanunjaya Varma Devalraju and C Chandra Sekhar
Department of Computer Science and Engineering
Indian Institute of Technology Madras, India
cs21d006@smail.iitm.ac.in, chandra@cse.iitm.ac.in

Abstract

Unsupervised metric learning aims to learn the discriminative representations by grouping similar examples in the absence of labels. Many unsupervised metric learning algorithms combine clustering-based pseudo-label generation with embedding fine-tuning. However, pseudo-labels can be unreliable and noisy. This could affect metric learning and degrade the quality of the learned representations. In this work, we propose an approach to reduce the negative effect of label noise on learning discriminative embeddings by using context and prediction uncertainty. In particular, we refine the pseudo-labels by aggregating information from neighbors. We propose a function to weigh the pairs, leveraging their prediction confidence and uncertainty. We modify the metric learning loss function to incorporate this weight. Experimental results demonstrate the effectiveness of our proposed method on standard datasets for metric learning.

1. Introduction

Learning similarity between objects has been crucial in many machine learning and computer vision tasks. Metric learning encapsulates the concept of similarity. Metric learning techniques aim to learn an embedding space where similar objects are brought closer and dissimilar objects are moved apart. Deep neural networks have been commonly used to model the function that maps objects from an input space into an embedding space. The supervised metric learning uses the labels to generate pairs/triplets and learns metric from them. The unsupervised metric learning learns by exploring the structure present in the data. Supervised metric learning methods have shown remarkable progress in learning discriminative features or embeddings. Since labels are expensive and unlabeled data is abundant, recently, there has been a lot of focus on developing unsupervised techniques.

Existing approaches to unsupervised metric learning can be categorized into instance discrimination methods [7, 33, 37] and pseudo-labeling methods [3, 4, 9, 10, 13, 17, 34]. The instance discrimination method considers an instance and its transformations as one class and learns transformation invariant representations. However, these methods rely heavily on augmentation, making them not suitable for the domains where augmentation is not effective [25]. The pseudo-labeling method applies clustering algorithms like k-means [3, 4, 13, 17], hierarchical clustering [34], or graph clustering [9, 10] to cluster the samples in the feature space. These methods consider the cluster indices as pseudo-labels and learn the metric using a supervised metric learning loss function. The pseudo-labeling methods have shown impressive results in unsupervised metric learning. However, pseudo-labels generated from clustering are highly noisy. Such noisy labels may affect the metric learning performance and degrade the quality of the learned representations. To deal with label noise, existing methods use auxiliary tasks like rotation prediction [3] and attention consistency [17]. Some methods use relative orders [13] and weight function in terms of bilinear similarity [9, 10]. However, the problem persists, and improving the quality of pseudo-labels may improve the model performance.

The pseudo-labeling technique is also used in other settings like semi-supervised learning [8, 23, 25], and few shot learning [21]. In semi-supervised learning, the labeled data set is used to train/fine-tune an initial model. This trained model is used to predict labels for the unlabeled data. The predicted labels are considered as pseudo-labels. The unlabeled data with pseudo-labels and the labeled data are combined to learn the final model. These methods also suffer from label noise. To address this problem, the pseudo-label refinement/selection techniques [8, 23, 25] are used. In [11, 21, 25], it was shown that the pseudo-label noise is due to the poor network calibration that may produce an incorrect label with high confidence. They have also shown that network calibration is related to prediction uncertainty, and selecting samples with low uncertainty improves the model performance. Similarly, Zheng *et al.* [38] used prediction uncertainty to identify noisy labels and reduce the negative effect of label noise by incorporating the predic-

tion uncertainty as a weight term in the loss function.

Motivated by this, we propose an uncertainty-guided metric learning (UGML) framework for unsupervised metric learning. The proposed framework is a pseudo-labeling method that utilizes prediction uncertainty to alleviate the negative effect of label noise. However, the challenge is that, unlike semi-supervised learning, the unsupervised metric learning model cannot access any labels, not even a few labeled samples. Therefore, the noise could be high, causing unreliable uncertainty estimates that may induce little to no improvement in the model performance. To overcome this, we refine the model predictions by aggregating knowledge from neighbors (context) [15,19] before estimating the prediction uncertainty. The underlying assumption is that semantically similar samples should lie close on the data manifold [15].

In this work, we use a classifier network to estimate the prediction uncertainty. The proposed classifier is a convolutional neural network (CNN). Since the CNN is a non-probabilistic model, it can not be directly used to estimate uncertainty. In general, a Bayesian neural network (BNN) is required to estimate the uncertainty [14, 21]. The BNN places a distribution over the network weights and takes average over all possible weights for inference [14]. However, inference in a BNN is challenging [14]. Therefore, similar to [14, 21, 25], we also adopt the dropout method to approximate the BNN inference in a CNN. We add a few dropout layers to the classifier model and train the model using the pseudo-labels obtained from clustering. We enable the dropouts during the inference and perform several stochastic forward passes to get probability distributions. Then, we perform neighborhood aggregation to refine the model predictions. The refined distributions are used to compute the prediction mean and prediction variance. The prediction mean and prediction variance are used to obtain the refined pseudo-labels and uncertainty, respectively. Additionally, we propose incorporating a pairwise weight term into a metric learning loss to reduce the negative effect of label noise. The weight term is a function of prediction confidence (from prediction mean) and uncertainty (from prediction variance) to penalize the pairs with high uncertainty and low confidence. This formulation is inspired by the sample selection strategy employed by the uncertainty-aware pseudo-label selection [25].

The main contribution of this work can be summarized as follows:

- We introduce a novel uncertainty-guided metric learning framework (UGML), that reduces label noise and learns more discriminative embeddings.
- We develop a strategy to refine pseudo-labels and estimate uncertainty by using knowledge from neighborhood (context) and dropout.
- We propose a formulation for pairwise confidence and uncertainty that is incorporated into the metric learning loss function.

2. Related work

Unsupervised metric learning has mostly been studied in two directions, instance discrimination [7, 18, 33, 35, 37] and pseudo-labeling [3, 4, 9, 10, 13, 17, 34]. Instance discrimination methods are based on the idea of learning transformation invariant features. The Exemplar method [7] randomly samples a patch and applies stochastic data augmentation to form a surrogate class, and trains a classifier network to discriminate between these surrogate classes. In Exemplar, features are compared with classifier weights, limiting its effectiveness and discriminability. To overcome this, Wu *et al.* [33] extends the Exemplar to incorporate comparison over features by using a memory bank with noise contrastive estimation. This technique is also ineffective as the memory bank keeps the outdated features. To address this inefficiency issue, Ye *et al.* [37] proposed to compare the features from an instance and its augmentation and solve it as a binary classification problem. Li *et al.* [18] proposed the spatial assembly network that learns feature embedding invariant under generic spatial variations such as changes in the spatial layout of objects, object part configurations, and scene structures. However, all these instance discrimination methods fail to capture intra-class variations [15].

Clustering is the most natural method of identifying the semantic similarity among the examples in the absence of labels. The pseudo-labeling methods apply clustering algorithms like k-means [3, 4, 13, 17], hierarchical clustering [34], or graph clustering [9, 10] to cluster the samples in the feature space. The cluster indices are used to approximate the class-equivalence relations for the supervised metric learning loss functions. Kim *et al.* [15] used a combination of pairwise and contextual similarity to approximate the class-equivalence relation. Methods [3, 4, 13, 17, 34] that repetitively use the clustering algorithms incur substantial computational complexity. Further, all these pseudo-labeling methods suffer from high label noise. Existing techniques make use of auxiliary tasks to deal with label noise [3, 9, 10, 13, 17].

3. The Proposed Framework

Let $D = \{\mathbf{x}_i\}_{i=1}^{N}$ be the given unlabeled dataset with N examples. Let $f_{emb}(\cdot)$ be the model that transforms the examples from a given input space to a feature space. The $f_{emb}(\cdot)$ is usually a pre-trained CNN. Let $g_{emb}(\cdot)$ be an embedding layer that projects from the feature space to an embedding space. Then the metric learning problem involves learning the parameters of $f_{emb}(\cdot)$ and $g_{emb}(\cdot)$ given D such that the semantically similar examples (positive pairs)

are pulled together and the dissimilar examples (negative pairs) are pushed away in the embedding space. Supervised metric learning uses class labels to construct the positive and negative pairs, and then uses these pairs to learn the embedding. However, our goal is to learn the embedding space in the absence of labels. We propose a new framework that uses contextual information and prediction uncertainty to learn the embedding space in the absence of labels.

3.1. Framework overview

The proposed framework is shown in Fig. 1. The framework has a feature extractor, a classification network, and an embedding network. The feature extractor is a pre-trained CNN model used to generate initial features for the clustering algorithm to generate the pseudo-labels. The classification network has an encoder $f_{cls}(\cdot)$ followed by an embedding layer $g_{cls}(\cdot)$ and an output softmax layer $h_{cls}(\cdot)$. The output from the embedding layer $g_{cls}(\cdot)$ is used to estimate the contextual similarity. The contextual similarity and the probability distributions from the softmax layer $h_{cls}(\cdot)$ are used to estimate the confidence scores and uncertainty. The estimated confidence scores are used to refine the pseudo-labels obtained from clustering. In general, the pseudo-labels from clustering can be noisy and lead to an inferior performance and instability while training the embedding network [3]. The refined pseudo-labels help in reducing this noise. Further, the estimated uncertainty is used to weigh the refined pseudo-labels while training the embedding network. This further helps improve the model performance as the weight term specifies how reliable the pseudo-label associated with each example is. Our embedding network has an encoder $f_{emb}(\cdot)$ followed by an embedding layer $g_{emb}(\cdot)$.

As shown in Fig. 1, the proposed framework has four significant steps, namely: (1) Pseudo-label generation; (2) Classifier training for model uncertainty estimation; (3) Uncertainty estimation using Monte Carlo (MC) dropout and contextual similarity; (4) Learning the embedding.

3.2. Pseudo-label generation

Most of the existing unsupervised metric learning methods [3, 4, 9, 12, 13, 17] employ pseudo-labeling to obtain the supervision that facilitates the use of loss functions developed for supervised metric learning. The pseudo-label generation process generally involves clustering the examples in a feature space and using the cluster indices as the pseudo-labels. Further, it is a common practice to obtain the features for clustering from a pre-trained model in an unsupervised fashion [3]. We also use the same process as [3, 4, 13, 17] to generate pseudo-labels. As shown in Fig. 1 step 1, the pseudo-label generation step consists of a feature extractor followed by clustering. The feature extractor for image data can be any pre-trained CNN model trained with ImageNet [26] for classification. In this work, we use the k-means clustering algorithm to cluster the representations from the feature extractor. Upon clustering, the cluster indices are considered as pseudo-labels and are used to train the classifier and embedding networks in the proposed framework. Further, unlike methods in [3, 4, 13, 17], there is no cluster reassignment.

However, the major challenge is that pseudo-labels can be noisy and can lead to poor performance. We refine and reweigh the pseudo-labels to overcome this issue using contextual similarity and prediction uncertainty. We train a classifier network using the obtained pseudo-labels to estimate the contextual similarity and prediction uncertainty.

3.3. Classifier training for uncertainty estimation

Given the pre-trained CNN model ($f_{cls}(\cdot)$), we add a fully connected (embedding) layer ($g_{cls}(\cdot)$), a classification layer ($h_{cls}(\cdot)$) and dropouts to form a classification network. We then fine-tune this classification network using the pseudo-labels from clustering and the cross-entropy loss function. This classification network is used to refine the pseudo-labels by performing neighborhood aggregation and estimate the prediction uncertainty.

3.4. Batch creation for neighborhood aggregation

For neighborhood aggregation, it is ideal to use the entire dataset to fully utilize the context. However, for large datasets like SOP, the probability density matrix $\mathbf{P_i}$ for the i^{th} example has dimensions $15 \times 10,000$ (see Sec. 3.5), expanding to $15 \times 59,551 \times 10,000$ for the entire training set, making it inefficient to store and process the entire matrix in memory without efficient batch construction. Random sampling, on the other hand, could lead to the loss of contextual information. Furthermore, there is no guarantee that the harder classes that are close to one another will be chosen in a batch with random sampling. To overcome this, we propose a new approach to create a mini-batch by clustering classes that are close to each other in the feature space of the classification model, i.e., $g_{cls}(f_{cls}(\cdot))$. This approach first computes the centroids by gathering the features for each class based on the classifier network's predictions. Then, it employs hierarchical agglomerative clustering on centroids to identify the classes that are close to each other. Samples from these classes are then used to create the mini-batch.

3.5. Uncertainty estimation using MC dropout and contextual similarity

Similar to [21, 25], we use dropouts to compute the uncertainty estimates for the training data. Let $\psi(\cdot) = h_{cls}(g_{cls}(f_{cls}(\cdot)))$ be the classification network parameterized by the weight vector W. Once the classification network is trained, the dropouts are enabled, and several stochastic forward passes of training data are performed

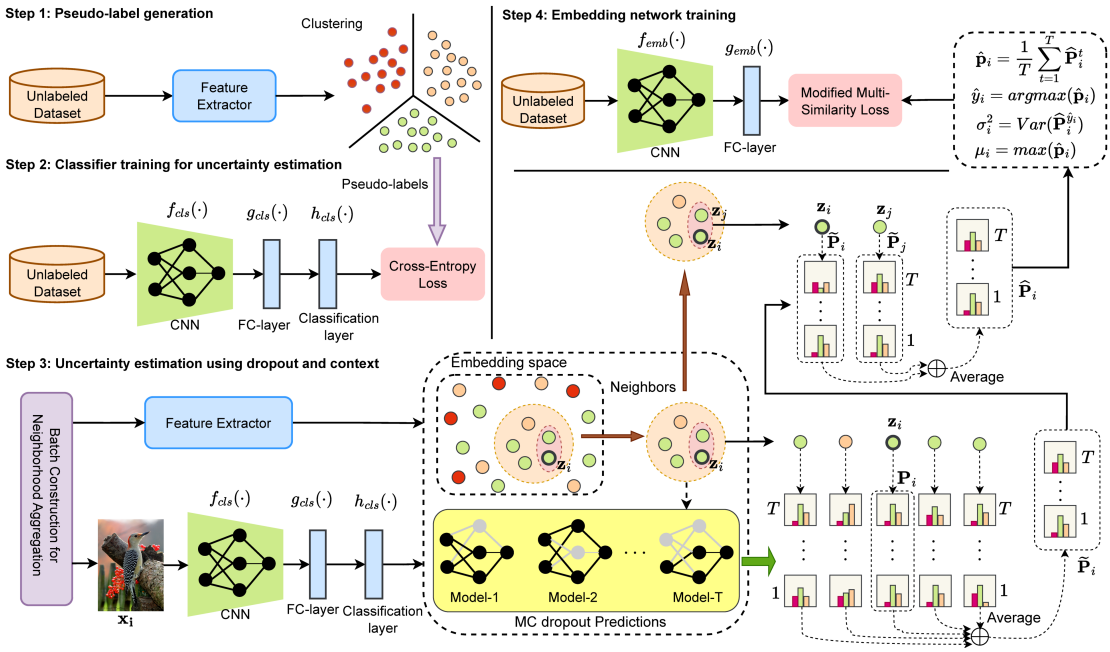

Figure 1. Overview of the proposed framework. The proposed framework has four main steps. Step 1: A pre-trained model to extract the image features, followed by k-means clustering to generate pseudo-labels; Step 2: The pseudo-labels are used to train a classifier; Step 3: The classifier along with MC dropout and context is used to estimate the classifier's uncertainty in the pseudo-labels; Step 4: The estimated uncertainty is used to weigh the positive and negative pairs in the multi-similarity loss while training the final embedding model.

through the classification network. For every example, such a stochastic forward pass produces a posterior probability distribution. The dropout-enabled stochastic forward pass is equivalent to sampling a masked classification network weights $\widetilde{W} \sim \mathbf{q}^*_\theta(W)$, where $\mathbf{q}^*_\theta(W)$ is the the dropout distribution [14, 21].

For each example $\mathbf{x}_i$, the T stochastic forward passes through the dropout enabled network produce a set of T probability distributions, denoted by $\mathbf{P}_i$, as shown below.

$$\mathbf{p}_i^t = softmax(\psi_{\widetilde{W}_t}(\mathbf{x}_i)); \;\; \widetilde{W}_t \sim \mathbf{q}^*_\theta(W) \tag{1}$$

$$\mathbf{P}_i = [\, \mathbf{p}_i^t \,]_{t=1}^{T} \tag{2}$$

Let C be the number of clusters. Then the vector $\mathbf{p}_i^t$ of length C specifies the probability distribution corresponding to i^{th} example at t^{th} forward pass. The matrix $\mathbf{P}_i$ of dimension $T \times C$ is a collection of $\mathbf{p}_i^t$ corresponding to T stochastic forward passes. The refined pseudo-labels and uncertainty estimates are obtained from $\mathbf{P}_i$, as done in [21, 25], which have access to a limited set of labeled samples. However, unlike these methods [21, 25], we do not have access to any labels. On the contrary, the pseudo-labels from clustering may suffer from heavy noise [13]. Hence, the uncertainty estimates from $\mathbf{P}_i$ alone may not be adequate for unsupervised metric learning. Therefore, we propose to refine $\mathbf{P}_i$ by aggregating information from neighbors, i.e., context [15, 19]. The refined $\mathbf{P}_i$ is, in turn, used to estimate the refined pseudo-labels and uncertainty. The underlying assumption is that semantically similar examples should lie close on the data manifold. Given two samples, the semantic similarity is measured by computing the pairwise similarity between the feature vectors from the feature extractor. The pairwise similarity s_{ij} between two examples $\mathbf{x}_i$ and $\mathbf{x}_j$ is given by

$$s_{ij} = \exp\left(-\frac{||\mathbf{z}_i - \mathbf{z}_j||_2^2}{\tau}\right) \tag{3}$$

where τ is the Gaussian kernel width, $\mathbf{z}_i$ and $\mathbf{z}_j$ are the feature vectors obtained from the feature extractor corresponding to $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively.

As shown in Fig. 1 step 3, given an example $\mathbf{x}_i$, the pairwise similarities between $\mathbf{x}_i$ and all the other examples in a given mini-batch (from sec 3.4) are used to find the k-nearest neighbors. These k-nearest neighbors are considered as context, and the information from neighbors is aggregated as follows:

$$\widetilde{\mathbf{P}}_i = \tfrac{1}{k} \textstyle\sum_{j \in \mathcal{N}_k(i)} \mathbf{P}_j \tag{4}$$

where $\mathcal{N}_k(i)$ is the set of indices of the k-nearest neighbors of i^{th} example ($\mathbf{x}_i$), including itself, i.e., $i \in \mathcal{N}_k(i)$. In order to further improve the reliability of the predictions from Eq. 4, we adopt the idea of query expansion [2, 5, 6]. Specifically, as stated in [15], we average the aggregated predictions of $\mathbf{x}_i$ and its neighbors as given below to get refined probabilities.

$$\widehat{\mathbf{P}}_i = \tfrac{1}{\lfloor \frac{k}{2} \rfloor} \textstyle\sum_{h \in \mathcal{N}_{\frac{k}{2}}(i)} \widetilde{\mathbf{P}}_h \tag{5}$$

Here, $\mathcal{N}_{\frac{k}{2}}(i)$ is a subset of $\mathcal{N}_k(i)$ and contains indices of top-$\frac{k}{2}$ nearest neighbors of i^{th} example ($\mathbf{x}_i$), including itself. From the refined probabilities $\widehat{\mathbf{P}}_i$, the confidence scores of label presence (or absence) are obtained by computing the prediction mean as the average over the T probability distributions, as shown below.

$$\hat{\mathbf{p}}_i = \frac{1}{T}\sum_{t=1}^{T}\widehat{\mathbf{P}}_i^t \tag{6}$$

Here, $\widehat{\mathbf{P}}_i^t$ is a vector of length C corresponding to i^{th} example at t^{th} forward pass. Once we have the prediction mean, the updated pseudo-label ($\hat{y}_i$) and its confidence (μ_i) corresponding to the example $\mathbf{x}_i$ are given by:

$$\hat{y}_i = \text{argmax}_c(\hat{\mathbf{p}}_i), \quad \mu_i = \max_c(\hat{\mathbf{p}}_i) \tag{7}$$

The updated pseudo-labels are used to identify the positive and negative pairs for every example $\mathbf{x}_i$ to train the embedding network (sec 3.6). Furthermore, similar to [14, 21], we compute the prediction variance ($Var(\widehat{\mathbf{P}}_i^{\hat{y}_i})$) associated with the pseudo-label $\hat{y}_i$ and consider it as the prediction uncertainty (σ_i^2).

$$\sigma_i^2 = Var(\widehat{\mathbf{P}}_i^{\hat{y}_i}) \tag{8}$$

Here, $\widehat{\mathbf{P}}_i^{\hat{y}_i}$ is a column vector of length T. The prediction uncertainty (σ_i^2) helps to identify more certain pseudo-labeled examples. If the uncertainty is higher, the model is less certain about the assigned pseudo-label. Therefore, we make the embedding network account for classifier uncertainty by incorporating the inverse of variance to weigh the loss incurred by an example. A smaller weight is used for the example with high uncertainty to reduce its effect on overall loss. Further, it may be possible that the model is less confident (i.e., small μ_i) even if the uncertainty is low. Therefore, the weight function that uses the confidence and uncertainty for a given example is devised as follows:

$$w_i = \frac{\mu_i}{\sigma_i} \tag{9}$$

However, the metric learning loss functions generally depend on the distance between pairs, triplets, or quadruplets. So, the per example weight function in Eq. 9 needs modification to account for the pair confidence and uncertainty. We use the average of the per example weights to get the per pair (or triplets or quadruplets) weights. Therefore, given two examples $\mathbf{x}_i$ and $\mathbf{x}_l$ along with their confidence and uncertainty values as μ_i, μ_l, $\frac{1}{\sigma_i}$ and $\frac{1}{\sigma_l}$, the per pair weight w_{il} is given by

$$w_{il} = \frac{1}{2}\left(\frac{\mu_i}{\sigma_i} + \frac{\mu_l}{\sigma_l}\right) \tag{10}$$

3.6. Training the embedding network

We use the multi-similarity loss [31] to train the embedding network. Two factors influence the choice of multi-similarity loss. Firstly, it exploits multiple similarities to extract informative pairs and shows an impressive performance in supervised metric learning. Secondly, it has been commonly used even in unsupervised metric learning [3, 13].

The multi-similarity loss [31] uses the hard example mining based on pairwise cosine similarity in the embedding space. This helps in extracting informative pairs and speeds up the training process. The vanilla multi-similarity loss is defined as follows:

$$\mathcal{L}_{MS} = \frac{1}{m}\sum_{i=1}^{m}\Big(\frac{1}{\alpha}\log\Big(1 + \sum_{l\in P_i^+} e^{-\alpha(S_{il}-\lambda)}\Big) + \frac{1}{\beta}\log\Big(1 + \sum_{l\in N_i^-} e^{\beta(S_{il}-\lambda)}\Big)\Big) \tag{11}$$

where m is the training batch size, and S is the cosine similarity between the embedding vectors. The P_i^+ and N_i^- are the sets of selected positive and negative pairs corresponding to i^{th} example, respectively. The positive and negative pairs are obtained using the refined pseudo-labels ($\hat{y}$) and hard example mining, i.e., given an anchor $\mathbf{x}_i$, the pair $\{\mathbf{x}_i, \mathbf{x}_j\}$ is chosen as negative pair if $S_{ij} > \min_{y_k=y_i} S_{ik} - \epsilon$. Similarly, the pair $\{\mathbf{x}_i, \mathbf{x}_j\}$ is chosen as positive pair if $S_{ij} < \max_{y_k\neq y_i} S_{ik} + \epsilon$, where $\epsilon > 0$ controls the margin.

We modified the vanilla multi-similarity loss in Eq. 11 to penalize the highly uncertain and less confident pairs by incorporating weight (w_{il}) from Eq. 10. The modified loss function is given below.

$$\widehat{\mathcal{L}}_{MS} = \frac{1}{m}\sum_{i=1}^{m}\Big(\frac{1}{\alpha}\log\Big(1 + \sum_{l\in P_i^+}(w_{il}\cdot e^{-\alpha(S_{il}-\lambda)})\Big) + \frac{1}{\beta}\log\Big(1 + \sum_{l\in N_i^-}(w_{il}\cdot e^{\beta(S_{il}-\lambda)})\Big)\Big) \tag{12}$$

We use this modified loss in Eq. 12, to train the embedding network. In the testing phase, the embeddings extracted from the $g_{emb}(\cdot)$ layer are used for the image retrieval task. Following [13], the values of the hyperparameters λ, α and β are set to 0.5, 2 and 40, respectively.

To analyse the modified loss function, we use the General Pair Weighting (GPW) framework [31]. A good pair-based loss function, when realized in GPW framework, is expected to assign higher weights ($\left|\frac{\partial\mathcal{L}(\mathbf{S},y)}{\partial S_{ij}}\right|$) to the informative pairs. When the vanilla multi-similarity loss (Eq. 11) is realized in the GPW framework, the positive and negative pair weights are obtained by computing gradient w.r.t S_{ij}.

The weight of a positive pair $\{\mathbf{x}_i, \mathbf{x}_j\} \in P_i^+$ is given by:

$$\left|\frac{\partial\mathcal{L}_{MS}}{\partial S_{ij}}\right|^+ = \frac{1}{e^{-\alpha(\lambda-S_{ij})} + \sum_{l\in P_i^+} e^{-\alpha(S_{il}-S_{ij})}} \tag{13}$$

and the weight of a negative pair $\{\mathbf{x}_i, \mathbf{x}_j\} \in N_i^-$ is given

by:

$$\left|\frac{\partial \mathcal{L}_{MS}}{\partial S_{ij}}\right|^{-} = \frac{1}{e^{\beta(\lambda - S_{ij})} + \sum_{l \in N_i^-} e^{\beta(S_{il} - S_{ij})}} \tag{14}$$

From Eq. 13, it can be observed that self-similarity (i.e., $e^{-\alpha(\lambda - S_{ij})}$) and relative similarity with other positive pairs (i.e., $e^{-\alpha(S_{il} - S_{ij})}$) are jointly used to compute the weight of a positive pair. Similar procedures apply for computing a negative pair weight, as in Eq. 14. These pair weights assist the loss function by assigning higher weights to the informative pairs that violate self-similarities and relative similarities.

Similarly, the positive and negative pairs weights for the modified multi-similarity loss defined in Eq. 12 obtained by computing gradient w.r.t S_{ij} are given in Eq. 15 and Eq. 16.

$$\left|\frac{\partial \widehat{\mathcal{L}}_{MS}}{\partial S_{ij}}\right|^{+} = \frac{w_{ij}}{e^{-\alpha(\lambda - S_{ij})} + \sum_{l \in P_i^+} w_{il} \cdot e^{-\alpha(S_{il} - S_{ij})}} \tag{15}$$

$$\left|\frac{\partial \widehat{\mathcal{L}}_{MS}}{\partial S_{ij}}\right|^{-} = \frac{w_{ij}}{e^{\beta(\lambda - S_{ij})} + \sum_{l \in N_i^-} w_{il} \cdot e^{\beta(S_{il} - S_{ij})}} \tag{16}$$

These pair weights employ pair-wise confidence and uncertainty to weigh the violation of its self-similarity and relative similarity. This aids the loss function in assigning higher weights to the more confident and less uncertain informative pairs.

4. Experimental Details

In this section, we describe the experiments used to evaluate the proposed framework for image retrieval task.

4.1. Datasets and Evaluation Protocol

The proposed framework is evaluated on three benchmark datasets for metric learning, namely CUB-200-2011 (CUB) [30], Cars-196 (Cars) [16], and Stanford Online Product (SOP) [22]. We follow the protocol provided in [22] to prepare the train and test sets. The CUB-200-2011 dataset have 200 bird species with a total of $11,788$ images. The first 100 species with $5,864$ images are used for training and the remaining 100 species with $5,924$ images are used for testing. The Cars-196 dataset comprises $16,185$ images of cars categorized into 196 different classes. The $8,054$ images of first 98 classes are used for training and the $8,131$ images of remaining 98 classes are used for testing. The Stanford Online Product dataset consists of $22,634$ products, with a total of $120,053$ images. The first $11,318$ products having $59,551$ images are used for training and the remaining $11,316$ products having $60,502$ images are used for testing.

For a fair comparison, we followed the protocol given in [36] and evaluated the proposed method on image retrieval task using the performance metric Recall@K. Recall@K is the fraction of search queries having at least one relevant sample among their k-nearest neighbors in a learned embedding space.

4.2. Feature Extraction

We use two different feature extractors, namely Regional Maximum Activation of Convolutions (R-MAC) [29] and Selective Convolutional Descriptor Aggregation (SCDA) [32]. Both are unsupervised methods that utilize CNNs pre-trained on the ImageNet dataset [26]. As reported in [32], SCDA features produce the best retrieval performance on the CUB dataset, while R-MAC performs best on the Cars dataset. Hence, we use SCDA features as the strong baseline for the CUB dataset and R-MAC features for the Cars datasets, demonstrating that UGML consistently outperforms these baseline methods.

For R-MAC features, similar to [9, 10, 12], we used the feature map from the final convolutions layer, right before the average pooling, of the GoogLeNet [28] pre-trained on ImageNet [26]. We considered three different input scales ($512, 512/\sqrt{2}$ and 256) forming the regions for the R-MAC. For each region ($h \times w \times d$) with d channels and $h \times w$ spatial resolution, max pooling is applied channel-wise to get a descriptor of size d. The descriptors from the regions are l_2 normalized, whitened, and sum pooled to get a feature vector of length $d = 512$. The sum pooled features are l_2 normalized at the end to get a final feature vector.

The SCDA features are obtained by performing descriptor selection and aggregation on the max-pooled feature maps from the final convolutional layer of VGG16 [27] pre-trained on the ImageNet [26] dataset. Descriptor selection involves finding the largest connected component to identify the descriptors that localize the primary object in the example. Descriptor aggregation involves applying max and average pooling on the identified descriptors and concatenating the max and average pooled features to get the SCDA features. A similar procedure is applied to the ReLU feature maps from the final convolutional block, scaled by 0.5, and concatenated with the SCDA features to form SCDA$^+$ features. We used the 4096 dimensional SCDA_flip$^+$ features obtained by concatenating the SCDA$^+$ features of an example and its horizontal flip. Further, the SCDA and SCDA$^+$ features are l_2 normalized before concatenation.

4.3. Implementation Details

We use Inception-V1 [28] pre-trained on ImageNet [26] as the CNN base for classification and embedding networks. Both networks are trained with the Adam optimizer and cosine annealing [20] for learning rate decay. Following [31], mini-batches are created by randomly selecting clusters and sampling M examples from each. We use a mini-batch of size 120, with M set to 4 for CUB and Cars, and 2 for SOP.

For the classification network, the CNN base is followed

Table 1. Performance of the unsupervised metric learning methods on the datasets *CUB*, *Cars* and *SOP* with network architecture GoogleNet [28] (Inception-V1). Top 3 results are marked by the colors Red (Top 1), Green (Top 2) and Blue (Top 3).

		CUB			Cars			SOP		
Method	Dim	R@1	R@2	R@4	R@1	R@2	R@4	R@1	R@10	R@100
MOM [12]	128	-	-	-	35.5	48.2	60.6	43.3	57.2	73.2
Exemplar [7]	128	38.2	50.3	62.8	36.5	48.1	59.2	45.0	60.3	75.2
NCE [33]	128	39.2	51.4	63.7	37.5	48.7	59.8	46.6	62.3	76.8
DeepCluster [4]	128	42.9	54.1	65.6	32.6	43.8	57.0	34.6	52.6	66.8
ISIF [37]	128	46.2	59.0	70.1	41.3	52.3	63.6	48.9	64.0	78.0
PSLR [35]	128	48.1	60.1	71.8	43.7	54.8	66.1	51.1	66.5	79.8
ROUL [13]	128	**56.7**	**68.4**	**78.3**	**45.0**	**56.9**	**68.4**	53.4	68.8	81.7
SAN [18]	128	**55.9**	**68.0**	**78.6**	44.2	55.5	66.8	58.7	73.1	84.6
STML [15]	128	**59.7**	**71.2**	**81.0**	**49.0**	**60.4**	**71.3**	**65.8**	**80.1**	**89.9**
Ours (SCDA)	128	55.7	67.8	77.9	41.5	53.0	64.3	**62.9**	**77.6**	**88.2**
Ours (R-MAC)	128	53.7	65.4	76.4	**45.1**	**56.8**	**67.6**	**62.3**	**77.0**	**87.9**
UDML-SS [3]	512	54.7	66.9	77.4	45.1	56.1	66.5	63.5	**78.0**	**88.6**
TAC-CCL [17]	512	57.5	68.8	78.8	46.1	56.9	67.5	**63.9**	77.6	87.8
UHML [34]	512	**58.9**	**70.6**	**80.4**	47.7	58.9	70.3	**65.1**	**78.2**	88.3
STML [15]	512	**60.6**	**71.7**	**81.5**	**50.5**	**61.8**	**71.7**	**65.3**	**79.8**	**89.8**
Ours (SCDA)	512	**58.8**	**70.7**	**81.0**	**48.5**	**59.5**	**70.4**	63.4	**78.0**	**88.4**
Ours (R-MAC)	512	56.7	68.4	79.0	**53.7**	**65.1**	**74.4**	62.9	77.3	88.1

by a 512-dimensional fully connected layer (g_{cls}) and a classification layer (h_{cls}) with the number of clusters (C) set to 100 for CUB and Cars, and 10,000 for SOP, following common practice in unsupervised metric learning [3,13,17]. Dropout applied to the third and fourth inception blocks of Inception-V1. The embedding network is the CNN base followed by a 128/512 dimensional embedding layer (g_{emb}). The values of τ, k, and T are set to 3, 5, and 15, respectively. More details are given in the supplementary material.

4.4. Comparison with State-of-the-Art Methods

We compare the proposed framework with the state-of-the-art methods available in the literature for embedding dimensions 128 and 512. The results on the datasets CUB, Cars, and SOP with the ImageNet pre-trained Inception-V1 base are summarized in Table 1. The top three results for each dataset and different embedding sizes are highlighted with Red, Green, and Blue colors. The best result is marked in Red, the second best in Green, and the third best in Blue.

As shown in Table 1, our results are comparable with the state-of-the-art results on all three datasets. With 512 dimensional embedding, the proposed method outperforms all other techniques with R-MAC features on the Cars dataset. It achieves the second best with SCDA features on the CUB dataset and the third best with SCDA features on the cars and SOP datasets. With 128 dimensional embedding, the proposed method achieves second best with SCDA features on the SOP dataset and third best with R-MAC features on the Cars and SOP dataset. Further, except for the model trained with 128 dimensional embedding on the cars dataset, the proposed method outperforms the instance discrimination methods Exemplar, NCE, ISIF, and PSLR, and our results are comparable to those of the SAN. However, compared to top-performing methods like STML, UHML, and ROUL, our results are sometimes comparable but occasionally fall short. The possible reason could be that the proposed framework relies heavily on initial features and does not update pseudo-labels after initial refinement. In contrast, STML, UHML, and ROUL periodically update pseudo-labels (UHML and ROUL) through costly clustering steps or class-equivalence (STML) using a complex student-teacher model.

Table 2. Ablation study on efficiency comparison.

	Training time (minutes)	
Methods	CUB	Cars
STML [15]	45	68
UGML(SCDA)	35	51
UGML(R-MAC)	**31**	**46**

UGML involves two training processes requiring 50 and 20 epochs, along with one-time operations such as feature extraction, clustering, and neighborhood aggregation, plus 15 forward passes (dropout) for uncertainty estimation. On the other hand, each epoch in STML requires one complete forward pass to extract information for NN batch construction, and additional forward passes through both the teacher and student (two heads) for computing contextualized semantic similarity and contrastive losses. Further, STML incurs additional processing for computing k-reciprocal nearest neighbors, updating teacher parameters, knowledge distillation, and handling two contrastive losses. Therefore, STML with three full forward passes for each epoch and the above additional overhead in the loop for 90 epochs makes it more complex than UGML, which requires fewer epochs and has one-time computation overheads. The training cost of STML and UGML, including all steps as well as cost for feature extraction, when run on a single RTX 3090 24GB GPU, is presented in Table 2. As indicated in Table 2, UGML is computationally more efficient than STML.

Table 3. Performance (R@1) on the CUB dataset using the Vision Transformers (ViT) from Unicom as the base.

Method	CUB		
	ViT-B/32	ViT-B/16	ViT-L/14
Unicom [1]	83.7	86.5	88.5
UGML(SCDA)	**85.0**	**87.7**	**89.0**

Table 4. Performance on the SOP dataset using a randomly initialized ResNet-18.

Method	SOP		
	R@1	R@10	R@100
PSLR [35]	42.3	57.7	72.5
ROUL [13]	45.4	60.5	74.8
SAN [18]	46.3	61.9	77.0
STML [15]	**60.7**	**74.8**	**85.2**
UGML(SCDA)	57.8	73.1	84.8

Furthermore, as seen in Table 1, our best results for the CUB and Cars datasets with 512 dimensional embeddings are comparable to STML. However, our performance lags behind STML with 128 dimensional embeddings, though it remains comparable with other top-performing methods like ROUL and SAN. This discrepancy can be attributed to the auxiliary higher-dimensional (1024) embedding space that STML learns. This higher-dimensional space facilitates the transfer of knowledge to the lower-dimensional spaces (512 or 128) used for evaluation, thereby narrowing the performance gap and resulting in minimal variation between the 512 and 128 dimensional embeddings.

4.5. Learning with Vision Transformer (ViT)

We evaluated UGML using various Vision Transformer models from Unicom [1] as the backbone, and the results on the CUB dataset are given in Table 3. Unicom [1] is a representation learning method trained on the large-scale image-text dataset LAION 400M, utilizing image and text features extracted from the pre-trained CLIP model [24]. As shown in Table 3, UGML outperforms Unicom baselines with ViT-B/32, ViT-B/16, and ViT-L/14 models.

4.6. Unsupervised Metric Learning from Scratch

We also validate the performance of the proposed framework on the SOP dataset using a randomly initialized ResNet-18 and 128 dimensional embedding layer. The model is trained with modified multi-similarity loss $\widehat{\mathcal{L}}_{MS}$ (Eq. 12) using refined pseudo-labels ($\hat{y}$) and uncertainty weights. The results are presented in Table 4. Our results are within a margin of 3% from the STML and outperform the second best method by a margin of 11.5%, 11.2% and 7.8% for Recall@1, Recall@10, and Recall@100.

4.7. Ablation study

To understand the effect of each component in the proposed UGML on the model performance, we studied them in isolation and summarized their performance in Table 5.

Table 5. Ablation study of UGML components on the CUB and Cars datasets.

Methods	CUB (SCDA)			Cars (R-MAC)		
	R@1	R@2	R@4	R@1	R@2	R@4
B1	57.4	69.4	79.8	51.6	62.7	72.4
B2	57.3	69.7	79.6	52.2	62.7	73.8
U1	57.9	69.6	79.8	53.0	64.4	73.7
M1	57.5	69.6	79.6	52.8	63.4	73.8
UGML	**58.8**	**70.7**	**81.0**	**53.7**	**65.1**	**74.4**

The UGML has two major components: (1) Pseudo-label refinement using neighborhood aggregation and (2) Pair-weights in terms of confidence and uncertainty. For comparison, we use two baselines, namely B1 and B2. B1 uses pseudo-labels from clustering (Step 1 in Fig. 1) and $\mathcal{L}_{MS}$ (Eq. 11) loss function to train the embedding network. Similarly, B2 uses predictions from the classification network (Step 2 in Fig. 1) and $\mathcal{L}_{MS}$ (Eq. 11) loss function to train the embedding network. U1 is an embedding network trained with the predictions from the classification network and $\widehat{\mathcal{L}}_{MS}$ (Eq. 12) loss function. M1 is an embedding network trained with the refined pseudo-labels ($\hat{y}$) and $\mathcal{L}_{MS}$ (Eq. 11) loss function, without uncertainty weight term. Table 5 presents the results for these four models and UGML using a 512 dimensional embedding. The results indicate that both pseudo-label refinement and uncertainty pair-weight components contribute to performance gains, and their combination produces better results. Further, the improvements are consistent across the datasets and various initial clustering feature choices.

Additional ablation studies and experimental results are included in the **supplemental material**.

5. Conclusion

We presented an approach that exploits the context and prediction uncertainty to improve the quality of learned representations in the absence of labels. The key idea is to refine the noisy pseudo-labels using context and re-weighting the informative pairs for metric learning loss using prediction confidence and uncertainty. Dropouts are used to estimate the prediction uncertainty of the underlying neural network. Multi-similarity loss, a pair-based loss, was modified to assign higher weights to the more confident and less uncertain informative pairs using the prediction confidence and uncertainty. The proposed approach achieves performance comparable to many existing methods on three benchmark datasets for metric learning while being computationally faster. We have empirically shown the effectiveness and consistency of the proposed framework with different unsupervised feature extraction techniques. The quality of the initial features from feature extraction limits the performance of our approach. In the future, we can extend this approach to reduce this dependency.

References

[1] Xiang An, Jiankang Deng, Kaicheng Yang, Jaiwei Li, Ziyong Feng, Jia Guo, Jing Yang, and Tongliang Liu. Unicom: Universal and compact representation learning for image retrieval. In *The Eleventh International Conference on Learning Representations*, 2023. 8

[2] Relja Arandjelović and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2911–2918. IEEE, 2012. 4

[3] Xuefei Cao, Bor-Chun Chen, and Ser-Nam Lim. Unsupervised deep metric learning via auxiliary rotation loss. *arXiv preprint arXiv:1911.07072*, 2019. 1, 2, 3, 5, 7

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 1, 2, 3, 7

[5] Ondřej Chum, Andrej Mikulík, Michal Perdoch, and Jiří Matas. Total recall ii: Query expansion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 889–896, 2011. 4

[6] Ondrej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8. IEEE, 2007. 4

[7] Alexey Dosovitskiy, Philipp Fischer, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(09):1734–1747, 2016. 1, 2, 7

[8] Jiali Duan, Yen-Liang Lin, Son Tran, Larry S Davis, and C-C Jay Kuo. Slade: A self-training framework for distance metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9644–9653, 2021. 1

[9] Ujjal Kr Dutta, Mehrtash Harandi, and Chellu Chandra Sekhar. Unsupervised deep metric learning via orthogonality based probabilistic loss. *IEEE Transactions on Artificial Intelligence*, 1(1):74–84, 2020. 1, 2, 3, 6

[10] Ujjal Kr Dutta and Chandra Sekhar C. A geometric approach for unsupervised similarity learning. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4202–4206, 2020. 1, 2, 6

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330, 2017. 1

[12] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Mining on manifolds: Metric learning without labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7642–7651, 2018. 3, 6, 7

[13] Shichao Kan, Yigang Cen, Yang Li, Vladimir Mladenovic, and Zhihai He. Relative order analysis and optimization for unsupervised deep metric learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13994–14003, 2021. 1, 2, 3, 4, 5, 7, 8

[14] Alex Kendall and Yarin Gal. What uncertainties do we need in Bayesian deep learning for computer vision? *Advances in Neural Information Processing Systems*, 30, 2017. 2, 4, 5

[15] Sungyeon Kim, Dongwon Kim, Minsu Cho, and Suha Kwak. Self-taught metric learning without labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7431–7441, 2022. 2, 4, 7, 8

[16] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 6

[17] Yang Li, Shichao Kan, and Zhihai He. Unsupervised deep metric learning with transformed attention consistency and contrastive clustering loss. In *European Conference on Computer Vision*, pages 141–157. Springer, 2020. 1, 2, 3, 7

[18] Yang Li, Shichao Kan, Jianhe Yuan, Wenming Cao, and Zhihai He. Spatial assembly networks for image representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13876–13885, 2021. 2, 7, 8

[19] Mattia Litrico, Alessio Del Bue, and Pietro Morerio. Guiding pseudo-labels with uncertainty estimation for source-free unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7640–7650, 2023. 2, 4

[20] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6

[21] Subhabrata Mukherjee and Ahmed Awadallah. Uncertainty-aware self-training for few-shot text classification. *Advances in Neural Information Processing Systems*, 33:21199–21212, 2020. 1, 2, 3, 4, 5

[22] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 6

[23] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021. 1

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 8

[25] Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *International Conference on Learning Representations*, 2021. 1, 2, 3, 4

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and

Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 3, 6

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 6

[28] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 6, 7

[29] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. *arXiv preprint arXiv:1511.05879*, 2015. 6

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

[31] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019. 5, 6

[32] Xiu-Shen Wei, Jian-Hao Luo, Jianxin Wu, and Zhi-Hua Zhou. Selective convolutional descriptor aggregation for fine-grained image retrieval. *IEEE Transactions on Image Processing*, 26(6):2868–2881, 2017. 6

[33] 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*, pages 3733–3742, 2018. 1, 2, 7

[34] Jiexi Yan, Lei Luo, Cheng Deng, and Heng Huang. Unsupervised hyperbolic metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12465–12474, 2021. 1, 2, 7

[35] Mang Ye and Jianbing Shen. Probabilistic structural latent representation for unsupervised embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5457–5466, 2020. 2, 7, 8

[36] Mang Ye, Jianbing Shen, Xu Zhang, Pong C. Yuen, and Shih-Fu Chang. Augmentation invariant and instance spreading feature for softmax embedding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):924–939, 2022. 6

[37] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019. 1, 2, 7

[38] Kecheng Zheng, Cuiling Lan, Wenjun Zeng, Zhizheng Zhang, and Zheng-Jun Zha. Exploiting sample uncertainty for domain adaptive person re-identification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3538–3546, 2021. 1