Applied Intelligence
https://doi.org/10.1007/s10489-022-03959-6

A framework to enhance generalization of deep metric learning methods using general discriminative feature learning and class adversarial neural networks

Karrar Al-Kaabi [1,2] · Reza Monsefi [1] · Davood Zabihzadeh [3]

Accepted: 2 July 2022

Abstract
Deep Metric Learning (DML) methods automatically extract features from data and learn a non-linear transformation from the input to a semantically embedding space. Many DML methods focused to enhance the discrimination power of the learned metric by proposing novel sampling strategies or loss functions. This approach is very helpful when both the training and test examples are selected from the same set of categories. However, it is less effective in many applications of DML such as image retrieval and person-reidentification. Here, the DML should learn general semantic concepts from observed classes and employ them to rank or identify objects from unseen categories. Neglecting the generalization ability of the learned representation and just emphasizing to learn a more discriminative embedding on the observed classes may lead to the overfitting problem. To address this limitation, we propose a framework to enhance the generalization power of existing DML methods in a Zero-Shot Learning (ZSL) setting by general yet discriminative representation learning and employing a class adversarial neural network. To learn a *general representation*, we employ feature maps of intermediate layers in a deep neural network and enhance their discrimination power through an attention mechanism. Besides, a *class adversarial network* is utilized to force the deep model to seek class invariant features. We evaluate our work on widely used machine vision datasets in a ZSL setting. Extensive experimental results confirm that our framework can improve the generalization of existing DML methods, and it consistently outperforms baseline DML algorithms on unseen classes.

Keywords Deep metric learning · Similarity embedding · Zero-shot learning · General discriminative feature learning · Adversarial neural network

1 Introduction

Distance measures have a major role in the success of many machine learning or pattern recognition algorithms. Standard distance or similarity measures such as Euclidean or cosine similarity often fail to capture the semantic concepts needed for a specific task. Thus, we need data-dependent metrics that measure semantically similar pairs close together while evaluating dissimilar ones far apart.

Most research in metric learning is dedicated to Mahalanobis distance defined as:

$$d_{\boldsymbol{M}}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)^2=\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right)^{\top}\boldsymbol{M}\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right), \tag{1}$$

where $\boldsymbol{x}_i,\boldsymbol{x}_j \in \mathbb{R}^d$ and $\boldsymbol{M} \in \mathbb{R}^{d \times d}$ is a positive semi-definite (p.s.d) matrix. Since $\boldsymbol{M} \succcurlyeq \boldsymbol{0}$, it can be factorized as $\boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L} \in \mathbb{R}^{d \times r}$ and $r = rank\ (\boldsymbol{M}) \leq d$. Thus,

$$\begin{aligned} d_{\boldsymbol{M}}\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)^2 &= \left(\boldsymbol{x}_i-\boldsymbol{x}_j\right)^{\top}\boldsymbol{M}\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right) \\ &= \left(\boldsymbol{x}_i-\boldsymbol{x}_j\right)^{\top}\boldsymbol{L}\boldsymbol{L}^{\top}\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right) = \left\|\boldsymbol{L}^{\top}\left(\boldsymbol{x}_i-\boldsymbol{x}_j\right)\right\|_2^2. \end{aligned} \tag{2}$$

The above equation indicates that Mahalanobis distance learning is equivalent to finding a *linear* transformation matrix $\boldsymbol{L}$. Despite its success, in many nonlinear datasets with complex class boundaries, a linear transformation is unable to

✉ Reza Monsefi
monsefi@um.ac.ir

Karrar Al-Kaabi
karrar.alkaabi@mail.um.ac.ir; karrara.hussein@uokufa.edu.iq

Davood Zabihzadeh
d.zabihzadeh@hsu.ac.ir

[1] Computer Engineering Department, Engineering Faculty, Ferdowsi University of Mashhad (FUM), Mashhad, Iran

[2] Faculty of Veterinary Medicine, University of Kufa Al-Najaf, Kufa, Iraq

[3] Computer Engineering Department, Hakim Sabzevari University, Sabzevar, Iran

Published online: 02 August 2022

extract discriminative features. Besides, Mahalanobis metric learning methods cannot process complex data such as images, text, and videos directly. Therefore, we need to first extract features from data and then apply Mahalanobis matrix learning algorithms.

To overcome these issues, *deep metric learning* (DML) methods are proposed that jointly perform feature extraction and learning non-linear transformation. Hence, they obtain an optimized representation of the input data for the distance learning task. Specifically, DML finds a non-linear function $\boldsymbol{f_w(x)}$ parameterized by $\mathbf{w}$ that maps $\boldsymbol{x}$ from the input space to a semantic embedding space. In this space, similar data items should be close together while dissimilar pairs should be kept at a distance.

A typical DML model has three components: 1) deep neural network model, 2) sampling algorithm, and 3) loss function.

The function $\boldsymbol{f_w}$ is implemented using a deep neural network such as CNN where the Softmax output layer is replaced by a Fully Connected (FC) named embedding layer. The embedding consists of a weight matrix $\boldsymbol{L} \in \mathbb{R}^{h \times d}$ that projects the output of the last hidden layer into the semantic embedding space.

To learn the parameters of the deep model (i.e., $\boldsymbol{w}$), the network is often trained by pairwise or triplet constraints defined as follows:

- Pairwise: $S = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \,|\, \boldsymbol{x}_i \text{ and } \boldsymbol{x}_j \text{ are similar}\}$ (Similar set), and
  $D = \{(\boldsymbol{x}_i, \boldsymbol{x}_j) \,|\boldsymbol{x}_i \textit{ and } \boldsymbol{x}_j \text{ are dissimilar}\}$ (Dissimilar set).
- Triplets: $T = \left\{ \left(\boldsymbol{x}_i, \boldsymbol{x}_i^+, \boldsymbol{x}_i^-\right) \;\middle|\; \boldsymbol{x}_i \text{ should be more similar to } \boldsymbol{x}_i^+ \text{ than to } \boldsymbol{x}_i^- \right\}$. Here, $\boldsymbol{x}_i$, $\boldsymbol{x}_i^+$, and $\boldsymbol{x}_i^-$ are named anchor, positive, and negative data, respectively.

Many DML algorithms adopt the Siamese [1] or the Triplet [2, 3] architecture shown in Fig. 1. Also, other forms of constraints such as quadruplets [4], and n-pairs [5] are introduced in recent research.

Many sampling strategies are developed to obtain informative constraints such as pairs, and triplets from the input minibatch of data passed to the network. Some seminal mechanisms include easy [6], hard [7], and semi-hard [8] negative mining. The sampled constraints are forwarded to the loss layer that encourages the separation of positive and negative pairs. The loss gradient is then backpropagated through the network to adjust the deep network parameters.

A loss function has a key role in the training process of a deep model. Two popular DML loss functions are *contrastive* and *triplet* utilized in the Siamese and Triplets networks respectively. Besides, many other loss functions are developed to promote the performance of the given DML task.

Many DML methods emphasize learning a more discriminative embedding function by providing novel sampling strategies or loss functions. This approach yields a good result where training and testing data are collected from the same set of categories. However, this assumption is not correct in major applications of DML such as CBIR,[1] person re-identification, ZSL,[2] and FSL.[3] Here, the model should learn general informative concepts from seen classes and utilize them to rank or identify objects from unseen categories. Neglecting the generalization power of the semantic embedding and just seeking to learn more discriminative representations of seen classes is subject to the overfitting problem.

To address this challenge, we propose a novel framework to enhance the generalization of DML methods in the ZSL setting using general yet discriminative feature learning and a class adversarial neural network.

In a typical deep neural network, the middle layers of the network extract general and small patterns of given data. The output of each layer is passed to the next layer to extract more discriminative and large patterns from the input. The last hidden layer of the network produces the most discriminative features, but it is dependent on the observed classes. It also concentrates only on specific regions from the input image needed to discriminate training classes. That increases the risk of neglecting other important regions required to classify unseen categories. This problem is illustrated in Fig. 2. The figure shows the generated Grad-CAMs of some images using fine-tuned BN-Inception neural network on the CUB200–2011 dataset [9]. As seen, the deep model has mainly focused on the head of birds to classify the images and omit some other discriminative regions that might be helpful to identify unseen objects.

Let $\boldsymbol{u}$ be the output of the last hidden layer in the network. Most existing DML approaches use only the feature vector $\boldsymbol{u}$ to learn a discriminative embedding. Here, we propose to employ outputs of intermediate layers since they have a better generalization on unseen categories. Also, they almost cover the entire input image. Now, the problem is the low discrimination power of these feature maps. We handle this issue by attending these features to the discriminative $\boldsymbol{u}$ feature vector. We assign weights to the feature maps of the selected layers according to discrimination scores obtained by an attention mechanism. Then, we form the final feature vector by combining the weighted feature maps.

Besides, an adversarial network is employed that enforces the deep model seeking class invariant features for the DML task. The adversarial network increases the classification loss on seen classes during the semantic embedding learning

[1] Content-Based Information Retrieval
[2] Zero-Shot Learning
[3] Few-Shot Learning

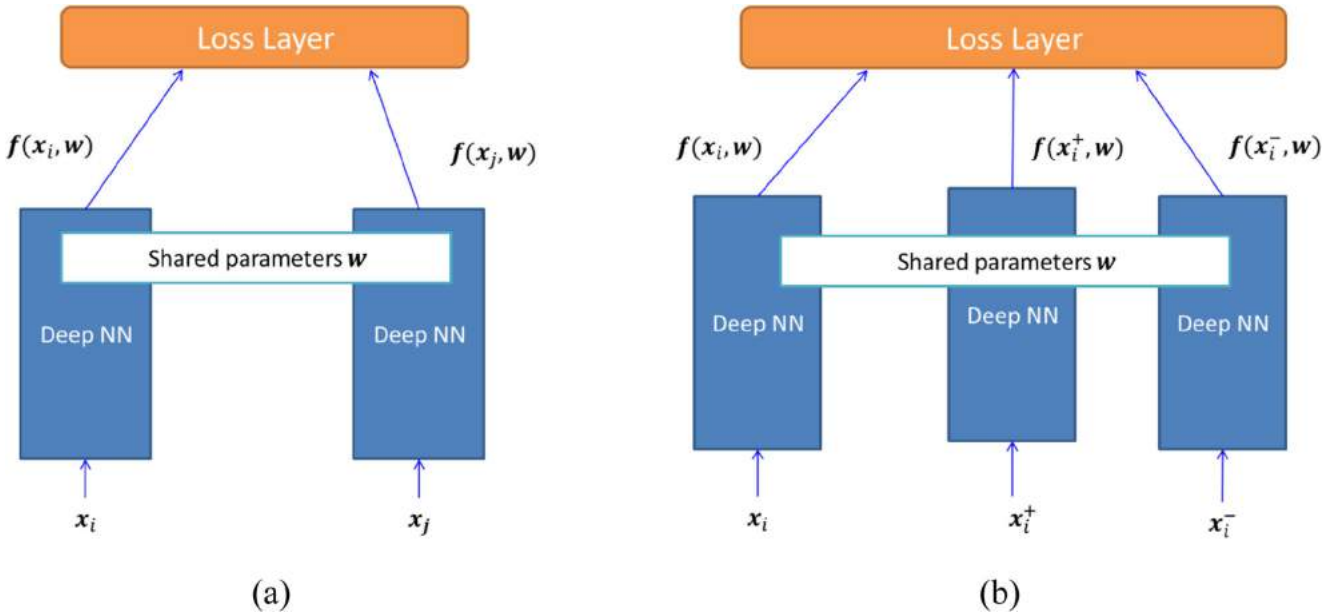


Fig. 1 **a** The Siamese architecture. **b** The Triplet architecture

process. This approach causes the deep network to explore more general discriminative information that is less dependent only on the available classes for the metric learning task. Therefore, it improves the generalization capability of the learned semantic embedding and prevents the overfitting on observed categories.

In summary, the major contributions of this paper are as follows:

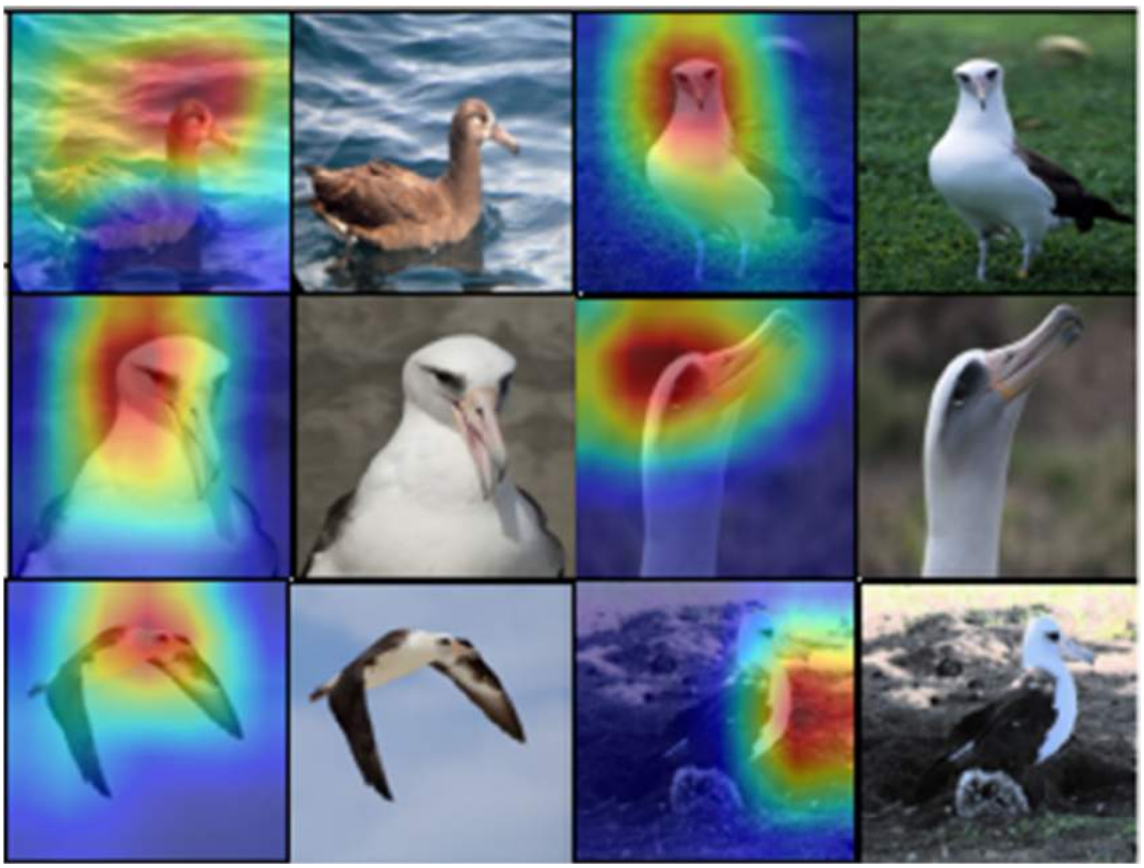


Fig. 2 Generated Grad-CAMs of some Birds using fine-tuned BN-Inception neural network on the CUB200–2011 [9] dataset

I. Learning general and discriminative representation by utilizing intermediate feature maps of the deep model and enhancing their discrimination power by attending them to the feature vector obtained from the last hidden layer.
II. A general framework to enhance the generalization power of DML is proposed using a class adversarial neural network. The framework can be applied to many existing DML methods and improve their performance in a ZSL setting.
III. We investigate several strategies to implement a class adversarial module and observed the best performance is obtained by confusing the classifier via adaptively setting the adversarial coefficient.
IV. Extensive experiments are performed on popular image information retrieval datasets in a ZSL setting, and our approach improves the results of state-of-the-art DML methods on these datasets.

The rest of the paper is organized as follows. Section 2 reviews related deep semantic embedding learning methods. Section 3 presents our framework to enhance the generalization power of DML in a ZSL setting. Strategies to implement the framework are presented in Section 4. Experimental results and comparison with state-of-the-art methods are reported in Section 5. Finally, the conclusion of the work and recommendations for future studies is given in Section 6.

2 Related work

[1] developed based on a Siamese network is the first pioneer work in the deep metric learning domain. Here, the energy between two pairs is defined as:

$$E_{\boldsymbol{W}}(\boldsymbol{x}_1,\boldsymbol{x}_2) = \|\boldsymbol{f}_{\boldsymbol{W}}(\boldsymbol{x}_1)-\boldsymbol{f}_{\boldsymbol{W}}(\boldsymbol{x}_2)\|. \tag{3}$$

The aim is to learn the network weights to minimize the energy function for similar images and maximize it for dissimilar ones. To this end, the following loss function is proposed:

$$\begin{aligned}\mathcal{L}(\boldsymbol{W},Y,X_1,X_2) &= (1-Y)\mathcal{L}_G(E_{\boldsymbol{W}}) + Y\mathcal{L}_I(E_{\boldsymbol{W}}) \\ &= (1-Y)\frac{2}{Q}(E_{\boldsymbol{W}})^2 + Y2Q\exp\left(-\frac{2.77}{Q}E_{\boldsymbol{W}}\right),\end{aligned} \tag{4}$$

where binary variable Y indicates that the corresponding (X_1, X_2) is similar or dissimilar pair:

$$Y = \begin{cases} 0, & \textit{for similar pair } (X_1,X_2) \ \ (\textit{genuine pair}) \\ 1, & \textit{otherwise}. \end{cases}$$

Loss functions defined on pairwise constraints such as (4) are named *Contrastive*. These functions are based on absolute distance values. However, in many applications, the *relative* distances between positive and negative pairs are more important. The triplet loss focuses on relative distances while contrastive loss concentrates on absolute distances. Also, a triplet considers both negative and positive constraints at the same time. Therefore, triplet-based methods often have superior performance compared to pairwise counterparts.

The margin-based Hinge loss function utilized in many triplet methods is defined as:

$$\begin{aligned} l\left(\left(\boldsymbol{x}_i,\boldsymbol{x}_i^+,\boldsymbol{x}_i^-\right)\right) &= [\alpha-(d^- - d^+)]_+ \\ &= \begin{cases} 0, & \textit{if } (d^- - d^+) \geq \alpha \\ \alpha-(d^- - d^+), & \textit{otherwise}. \end{cases} \end{aligned} \tag{5}$$

Here, $d^+ = \|\boldsymbol{f}_{\boldsymbol{W}}(\boldsymbol{x}_i)-\boldsymbol{f}_{\boldsymbol{W}}(\boldsymbol{x}_i^+)\|$ indicates the Euclidean distance in the embedding space between the anchor and positive, and $d^- = \left\|\boldsymbol{f}_{\boldsymbol{W}}(\boldsymbol{x}_i)-\boldsymbol{f}_{\boldsymbol{W}}\left(\boldsymbol{x}_i^-\right)\right\|$ shows the distance between the anchor and negative.

Most research in DML is focused on developing a new sampling strategy and loss functions. In the following, we review seminal work in each field.

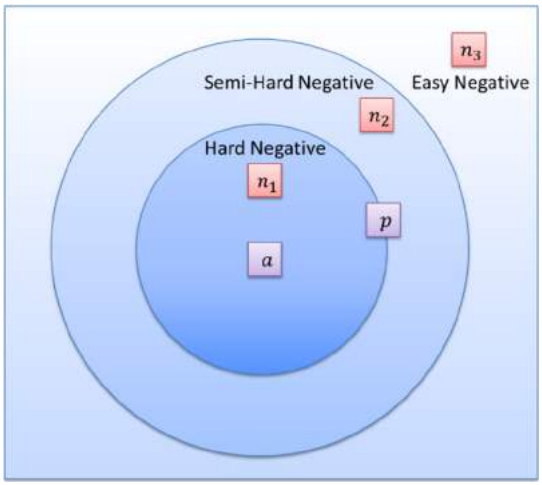
2.1 Sampling strategies

The simplest approach to generate positive and negative pairs is to randomly sample pairs based on class labels. For example, we can generate a similar pair by randomly selecting two datapoints from the same class and create a dissimilar pair by choosing two data items from different classes. This approach named easy sampling [6] often results in poor performance. [7] proposed hard negative mining that selects $\boldsymbol{x}_i^-$ from opposite classes subject to $d^- < d^+$ condition. Facenet [8] choose a *semi-hard* negative instance for each positive pair $(\boldsymbol{x}_i,\boldsymbol{x}_i^+)$. Here, the negative has more distance from the anchor compared to the positive example but still violates the margin constraint (*i. e.*, $d^+ < d^-$, $(d^- - d^+) < \alpha$). Figure 3 illustrates the differences between hard, semi-hard, and easy triplets.

Since hard negative samples are too close to the anchor, the gradient of loss has high variance and a low signal-to-noise ratio [10]. To alleviate this issue, [10] proposed a distance weighting sampling approach that allows exploiting different types of triplets in a noisy environment.

N-pair [5] samples from all negative classes as shown in Fig. 4. Hence, the loss function benefits from $N-1$ negative samples (one from each negative class) simultaneously leading to a better convergence rate in comparison with triplet-based methods.

[11] deals with the problem that many potential triplets in a dataset are easy and so do not affect the learned embedding.



a:anchor, p:positive, n_1,n_2, and n_3:negative

hard negative: $d^- < d^+$

semi-hard negative: $(d^- - d^+) < \alpha$

easy negative: $(d^- - d^+) \geq \alpha$

Fig. 3 Illustration of different triplet sampling strategies

Hence, DML lacks enough informative samples for training. To overcome the challenge, it generates a harder synthetic triplet from an easy sample (z, $\mathbf{z}^+$, $\mathbf{z}^-$) by approaching $\mathbf{z}^-$ to z while preserving its label information.

Generally, sampling algorithms lack different aspects such as capturing the global structure of data, low convergence rate, and extra time to mine informative samples. To address these challenges, proxy-based DML methods [12–15] proposed that can directly learn a semantic embedding from training data without utilizing a sampling strategy.

2.2 Loss functions

A Loss function has a crucial role in the success of a DML algorithm. The *contrastive* and *triplet* are two widely used losses denoted in (4), and (5) equations, respectively. Many other losses are proposed to enhance the convergence rate and the discrimination power of the learned metric. [16] adapts the margin in the triplet loss by constructing a hierarchical class-level tree. While this approach enhances the performance of DML, it suffers from the high computation time required to construct and update the tree.

The angular loss [17] constrains the *angle* at the negative instance of the triangle determined by a triplet. Compared to the contrastive and triplet losses, the loss benefits from robustness against feature variance and having a better convergence rate.

[4] aims to learn a better semantic embedding by sampling quadruplets instead of triplets. As shown in Fig. 5, a quadruplet is formed by adding an extra positive example to a triplet, and the loss function uses two different margin values.

The histogram loss [18] computes a distribution-based similarity between positive and negative pairs using the histogram. It then minimizes the probability that a positive pair has a lower similarity score than a negative.

[19] proposes the General Pair Weighting (GPW) mechanism that casts the sampling in DML into a pair weighting problem. Also, it introduces the multi-similarity loss under the GPW framework that explores pairs and assigns weights to them iteratively.

[20] presents Part loss for person re-identification task. It divides the identified body in a person image into five parts and then enforces the network to learn a representation for each different body part. By combining discriminative

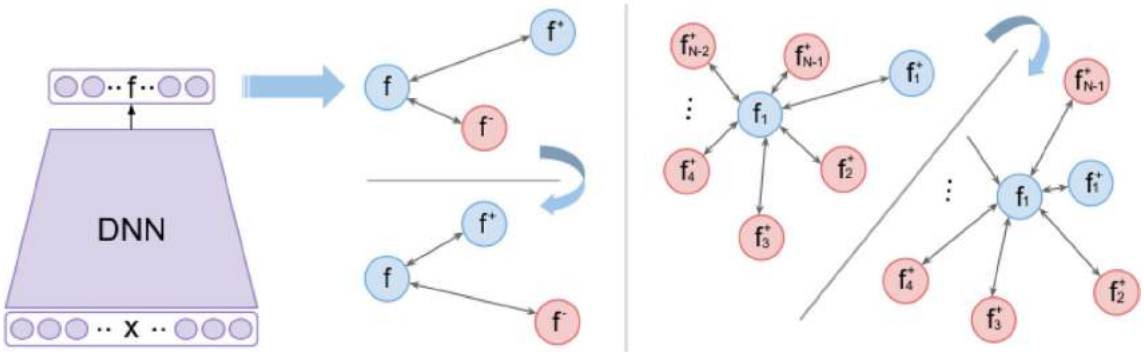


Fig. 4 Difference between triplet (left) and N-pair (right) distance learning. N-pair samples n-tuples that each contains an anchor (f_1), a positive example (f_1^+), and N-1 negative samples (f_2^+,…, f_{N-1}^+) [5]

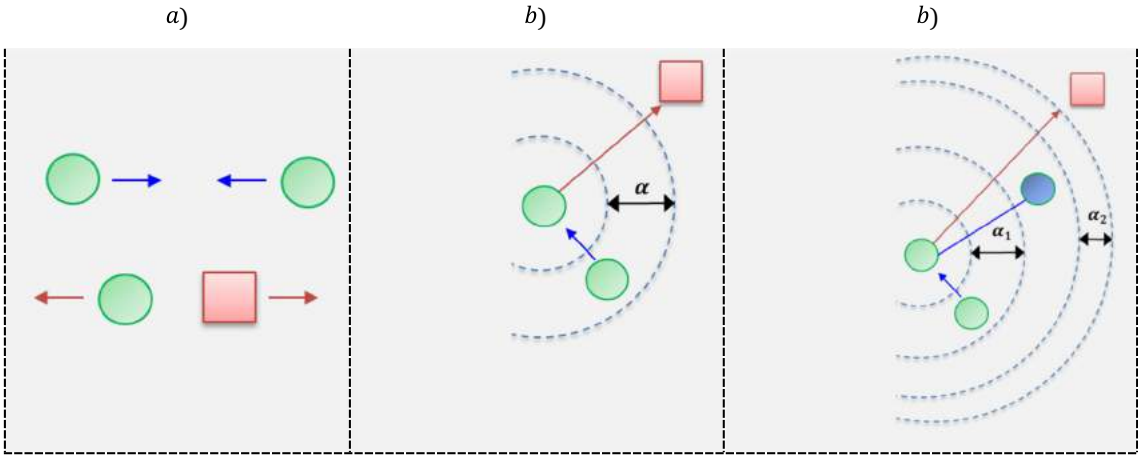


Fig. 5 The quadruplet loss vs contrastive and triplet loss. **a** Contrastive Loss **b** Triplet Loss **c** Quadruplet Loss

features obtained from different regions, the model achieves a better generalization to identify unseen persons.

For each similar pair, [21] considers all negative samples within the input mini-batch. Here, the loss function attempts to keep away all negative samples from the positive. N-pair loss [5] is defined on the input n-tuple as follows:

$$l_{n-pair}\left(\left(\boldsymbol{x},\boldsymbol{x}^{+},\boldsymbol{x}_{1}^{-},\boldsymbol{x}_{2}^{-},\ldots,\boldsymbol{x}_{N-1}^{-}\right)\right)=-\ln\left(1+\sum_{j=1}^{N-1}\exp\left(S_{j}^{-}-S^{+}\right)\right)$$
$$\text{where } S^{+}=\boldsymbol{f}_{\boldsymbol{w}}(\boldsymbol{x})^{\top}\boldsymbol{f}_{\boldsymbol{w}}(\boldsymbol{x}^{+}),\ \ S_{j}^{-}=\boldsymbol{f}_{\boldsymbol{w}}(\boldsymbol{x})^{\top}\boldsymbol{f}_{\boldsymbol{w}}\left(\boldsymbol{x}_{j}^{-}\right). \tag{6}$$

Hence, the loss increases the similarity gap (i.e., $S^{+}-S_{j}^{-}$) between a positive pair and all $N-1$ negatives simultaneously. Compared to the triplet loss, n-pair loss achieves a higher convergence rate and better captures the global structure of semantic embedding by considering more training examples at a time.

Some recent work learns semantic embedding without the need to sample tuples. For example, the clustering loss [14] directly optimizes the NMI.[4] As illustrated in Fig. 6, it considers one proxy per class, and the loss brings examples to their proxies closer while penalizing different proxies approaching each other.

Proxy NCA [13] optimizes triplet loss where the triplets are formed by an anchor datapoint and proxies of positive and negative classes. Here, the proxies (one per class) are learned jointly with the semantic embedding. [15] shows that the classification loss is equivalent to a smoothed triplet loss where each class is represented by a single center. Then, it presents SoftTriple loss that extends classification loss by considering multiple proxies per class. It also provides an effective regularization term to determine the appropriate number of centers.

2.3 Generalizability of DML

Major applications of DML such as ZSL, FSL, person-reidentification, CBIR, and clustering are challenging due to the following characteristics:

1. a large number of classes,
2. few examples in some categories,
3. unseen classes during the test phase.

Much research in DML focus on increasing the discrimination power of the learned embedding using available training classes and neglect the generalizability of the embedding on unseen categories.

MSML [22] deals with the problem by utilizing both high-level semantic and low-level but abundant visual features. It learns multi-scale feature maps and a multi-scale relation generation network (MRGN).

To enhance the generalizability of DML, [23] extends the triplet to a K-tuplet network where an anchor point deals with K negative examples at a time. That is similar to the test phase in FSL, where a query image needs a comparison with multiple different classes. The paper also provides a sampling mechanism to mine semi-hard informative K-tuples. However, these ideas come from previous research such as [5] and are mainly effective to capture the global structure of the embedding.

Adversarial learning is applied in DML to generate hard synthetic samples from the original training information. Unlike many approaches that ignore easy negatives, DAML[5] [24] utilizes them to generate hard synthetic triplets. (Wang, [25]) presents a similar idea. Here, as illustrated in Fig. 7, a hard negative generator is adversarially trained jointly with the DML to enhance the robustness of the learned embedding.

[4] Normalized Mutual Information

[5] Deep Adversarial Metric Learning

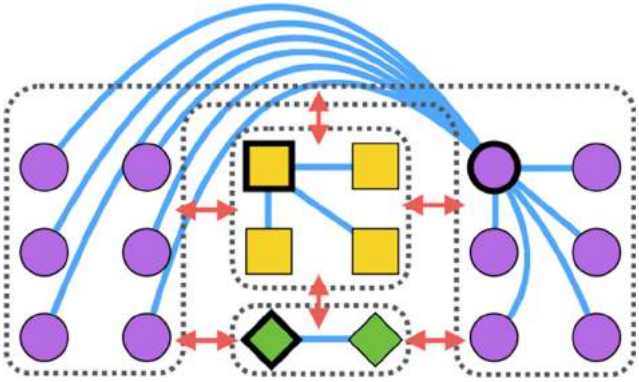


Fig. 6 [14] optimizes the NMI clustering metric by considering one proxy per class

In [26], an adversarial learning approach is exploited to enhance the performance of semantic embedding in a multi-modal environment. To this end, it maps the text and corresponding image of multi-modal data into a shared embedding, where an adversarial classifier is employed to minimize the gap between these modalities (i.e., text and image).

Proxy-Synthesis [27] interpolates proxies using a MixUp [28] technique to generate synthetic proxies. They mimic unseen classes in training DML. Proxy-Synthesis improves the generalization of the learned embedding on unobserved categories; however, it forms a larger batch from the input by generating synthetic proxies and samples. Also, this method is only applicable to proxy-based losses.

Chen and Deng [29] proposes to use adversarial learning to enhance the performance of DML on unseen classes in the ZSL setting. It introduces a regularization term named energy confusion that reduces overfitting on the seen classes during the metric learning process. The same idea is also adopted in [30] to boost the discrimination power of an unconstrained palmprint recognition model. The energy confusion term is defined as:

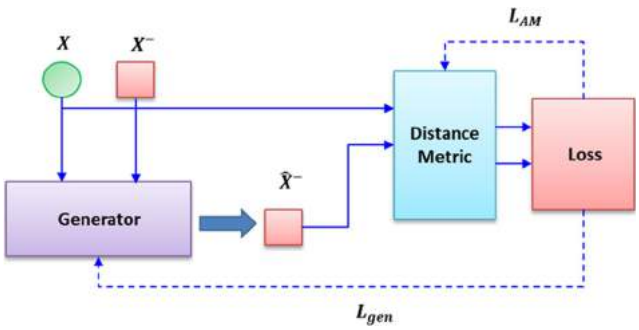


Fig. 7 Training distance metric and adversarial generator simultaneously to generate hard synthetic triplets (Wang, [25])

$$\mathcal{L}_{ec}(\boldsymbol{\theta}_f; X_I, X_J) = \mathbb{E}_{\widetilde{X}_I, \widetilde{X}_J}\left[\left\|\widetilde{X}_I - \widetilde{X}_J\right\|_2^2\right] = \sum_{i,j} p_{ij}\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2, \tag{7}$$

where $\boldsymbol{\theta}_f$ denotes DML parameters. $\widetilde{X}_I$ and $\widetilde{X}_J$ are random variables from probability distributions of i^{th} and j^{th} classes, respectively. $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the corresponding observations from these classes. Also, p_{ij} indicates joint probability distribution. Assuming independence between the classes and $\widetilde{X}_I \sim \text{uniform}(X_I), \widetilde{X}_J \sim \text{uniform}(X_J)$, p_{ij} is

$$p_{ij} = p_i p_j = \frac{1}{N_I}\frac{1}{N_J},$$

where N_I and N_J are the number of instances in the i^{th} and j^{th} classes. According to (7), minimizing $\mathcal{L}_{ec}$ encourages overlap between two different classes that is in contradiction with traditional metric learning.

Our aim is also to promote the generalizability of DML. However, instead of using regularization terms, we utilize an adversarial classifier to increase the classification loss on seen classes during semantic embedding learning. Besides, we learn global and discriminative features using feature maps of the intermediate layers through attention mechanisms.

3 Proposed model

As illustrated in Fig. 8, the proposed model consists of four components: 1) feature extractor, 2) tuplet Sampler, 3) metric embedding layer, and 4) classifier.

First, the input image is passed through the deep network, and feature maps of the layers are generated. The feature maps of the selected intermediate layers are weighted using an attention mechanism and then combined to form the final feature vector. The extracted features of the input minibatch are fed to the tuplet sampler to generate training side information. The sampled tuples are forwarded to the metric embedding layer and the metric loss is evaluated. Besides, the feature vectors are given to the classifier to obtain the classification loss. The proposed hybrid loss function is evaluated, and the gradient is backpropagated to adjust the model parameters. The loss enforces the model to learn a discriminative semantic embedding that is not limited to observed categories and generalizes well on unseen classes. In the following, we discuss each component in more detail.

3.1 Feature extractor

A deep neural network such as CNN, Encoder, or LSTM can be utilized to extract features from data. Since DML applications are focused on images, CNN models are very popular in

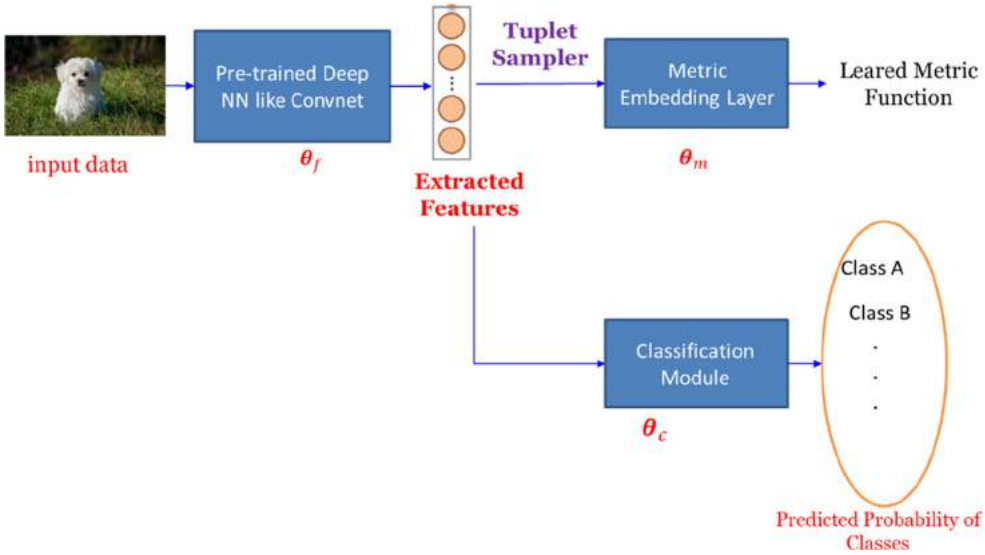


Fig. 8 The proposed model to enhance the generalization of a DML Task

this domain. A CNN learns hierarchical features from data, where the initial layers extract more general and small patterns, and the last layers learn abstract discriminative concepts.

As mentioned, most DML algorithms only utilize the feature vector $\boldsymbol{u} \in \mathbb{R}^I$ of the last hidden layer from a pre-trained convnet such as InceptionV2, VGG19, or ResNet50, which increases the dependency of the learned metric on the seen classes. To resolve this issue, we divide the deep model into several sequential modules that increasingly learn discriminative and abstract patterns from the input. Figure 9 illustrates the proposed architecture.

In Fig. 9, $\boldsymbol{M}^{(i)} \in \mathbb{R}^{c_i \times H_i \times V_i}$ shows the output feature map from the i-th module. The task of the Att_i is to attend features in $\boldsymbol{M}^{(i)}$ to the discriminative feature vector $\boldsymbol{u}$. The attention weight measures the discrimination power of the features. The final feature vector ($\boldsymbol{p}^{(i)} \in \mathbb{R}^I$) is formed by combining the weighted feature maps. Finally, the Inter-Module Fusion Component merges the attended features vectors. We implement this module by *concatenating* the attended vectors ($\boldsymbol{p}^{(i)}$ $i = 1, 2, .., m$). In the following, In the following, we discuss some attention mechanisms that can be employed in our model.

I. Multiplicative Attention:

Let $\boldsymbol{q}_j \in \mathbb{R}^{c_i}$, $j = 1, 2, \ldots, (H_i \times V_i)$ be a *pixel* in the feature map $\boldsymbol{M}^{(i)}$. In this approach, we measure the similarity of $\boldsymbol{q}_j$ to feature vector $\boldsymbol{u}$ as [31]:

$$S\left(\boldsymbol{q}_j, \boldsymbol{u}\right) = \left\langle \boldsymbol{W}_i^{(1)} \boldsymbol{q}_j, \boldsymbol{W}_i^{(2)} \boldsymbol{u} \right\rangle. \tag{8}$$

In the above equation, the weight matrices $\boldsymbol{W}_i^{(1)}$ and $\boldsymbol{W}_i^{(2)}$ are learned in an end-to-end training paradigm to increase the similarity between $\boldsymbol{q}_j$ and $\boldsymbol{u}$. Since we aim to promote the

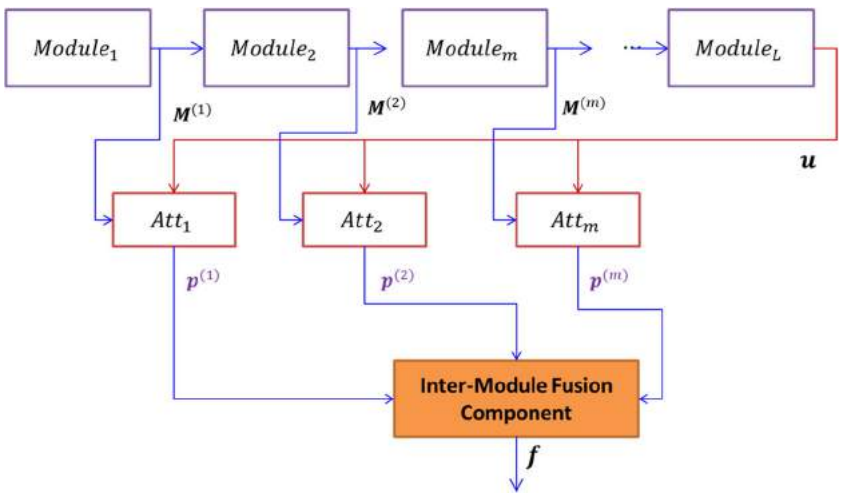


Fig. 9 The Architecture of feature extractor to learn global yet discriminative features from the input image

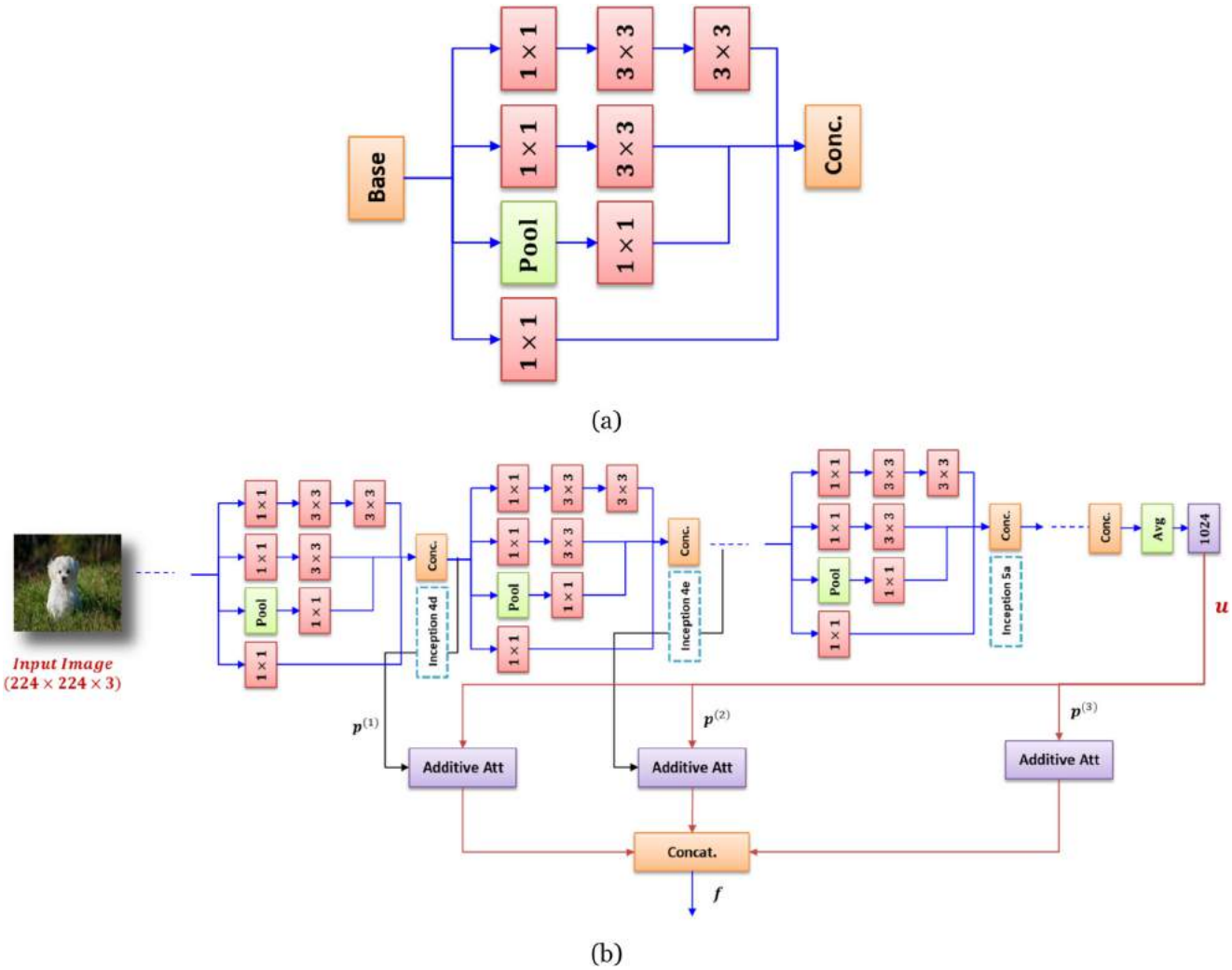


Fig. 10 **a** Inception module architecture in the *Inception V2*, **b** The Simplified architecture of the implemented feature extractor

similarity of $\boldsymbol{q}_j$ to **u**, the weight matrix $\boldsymbol{W}_i^{(2)}$ can be dropped to achieve a simpler mechanism (with fewer parameters) as follows:

$$S\left(\boldsymbol{q}_j, \boldsymbol{u}\right) = \left\langle \boldsymbol{W}_i^{(1)} \boldsymbol{q}_j, \boldsymbol{u} \right\rangle, \tag{9}$$

where $\boldsymbol{W}_i^{(1)} \in \mathbb{R}^{l \times c_i}$.

II. Additive Attention

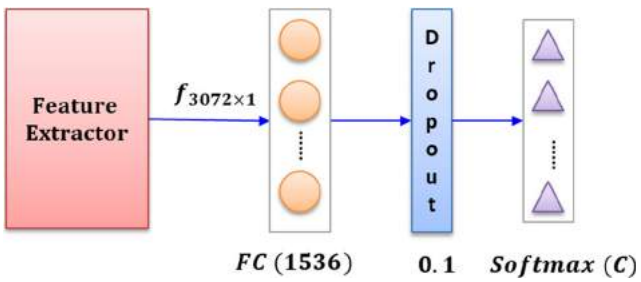


Fig. 11 Architecture of the classification module

This mechanism evaluates the similarity of $\boldsymbol{q}_j$ to $\boldsymbol{u}$ [31] as:

$$S\left(\boldsymbol{q}_j, \boldsymbol{u}\right) = \boldsymbol{w}_i^{\intercal} \sigma\left(\boldsymbol{W}_i^{(1)} \boldsymbol{q}_j + \boldsymbol{W}_i^{(2)} \boldsymbol{u}\right), \tag{10}$$

where σ is an activation function and $\boldsymbol{w}_i$ is a weight vector. In practice, the additive form outperforms the multiplicative mechanism. However, it demands more computations and memory. Similarly, we can simplify the Eq. (10) as:

$$S\left(\boldsymbol{q}_j, \boldsymbol{u}\right) = \boldsymbol{w}_i^{\intercal} \sigma\left(\boldsymbol{W}_i^{(1)} \boldsymbol{q}_j + \boldsymbol{u}\right). \tag{11}$$

III. Multi-dimensional Attention [31]:

This mechanism replaces the weight vector $\boldsymbol{w}_i$ with a matrix $\boldsymbol{W}_i \in \mathbb{R}^{l \times c_i}$:

Table 1 Statistics of the image datasets used in our experiments

Data Set	#classes	#samples	Evaluation Protocol
CUB-200-2011 [9]	200	11,788	The initial 100 classes (including 5864 images) for training and the rest (100) for testing (5924 images).
CARS-196 [33]	196	16,185	The first 98 classes (including 8054 images) for training and the remaining (98) for testing (8131 images).
Oxford Flowers-102 [34]	102	6552	The first 51 classes (including 2807 images) for training and the remaining (51) for evaluation (3745 images).

$$\boldsymbol{S}\left(\boldsymbol{q}_j, \boldsymbol{u}\right) = \boldsymbol{W}_i^{\mathsf{T}} \sigma\left(\boldsymbol{W}_i^{(1)} \boldsymbol{q}_j + \boldsymbol{u}\right). \tag{12}$$

Thus, the output is a similarity vector $\boldsymbol{a}_j \in \mathbb{R}^{c_i}$ where each component in $\boldsymbol{a}_j$ measures the importance or weight of the corresponding feature in $\boldsymbol{q}_j$.

After computing the similarity function, the Softmax is applied to normalize attention weights. More precisely, let $\boldsymbol{a}^{(i)} = \left[S\left(\boldsymbol{q}_j, \boldsymbol{u}\right)\right]_{j=1}^{H_i \times V_i}$ be the attention vector obtained from module i using multiplicative or additive mechanisms. We normalize the $\boldsymbol{a}^{(i)}$ using the Softmax function as follows:

$$\widetilde{a}_j^{(i)} = p\left(z_j \,|\boldsymbol{q}_j, \boldsymbol{u}\right) = \left[\text{Softmax}\left(\boldsymbol{a}^{(i)}\right)\right]_j = \frac{\exp\left(S\left(\boldsymbol{q}_j, \boldsymbol{u}\right)\right)}{\sum_{k=1}^{H_i \times V_i} \exp(S(\boldsymbol{q}_k, \boldsymbol{u}))}. \tag{13}$$

Here, z_j is an indicator that shows which pixel in the i-th module is attended more to the reference vector $\boldsymbol{u}$. In other words, the $\widetilde{a}_j^{(i)} = p\left(z_j \,|\boldsymbol{q}_j, \boldsymbol{u}\right)$ denotes the discrimination score or attention weight of $\boldsymbol{q}_j$. The output of the attention is the weighted mean of pixels:

$$\boldsymbol{p}^{(i)} = \sum_{j=1}^{H_i \times V_i} \widetilde{a}_j^{(i)} \boldsymbol{q}_j, \; i = 1, 2, .., m. \tag{14}$$

In the case of multi-dimensional attention, we perform the same procedure to each feature independently to obtain the importance weights for that feature. In particular, let A_{jk} be the k-th component of the $\boldsymbol{S}(\boldsymbol{q}_j, \boldsymbol{u})$ vector. A_{jk} $k = 1, 2, \ldots, c_i$ shows the unnormalized weight of k-th feature in $\boldsymbol{q}_j$. We normalize the weights using the Softmax as:

$$\widetilde{A}_{jk}^{(i)} = p\left(z_{jk} \,|\boldsymbol{q}_j, \boldsymbol{u}\right) = \frac{\exp\left(A_{jk}\right)}{\sum_{l=1}^{H_i \times V_i} \exp(A_{lk})}. \tag{15}$$

Then, the output of the k-th component of $\boldsymbol{p}^{(i)}$ is obtained as follows:

$$p_k^{(i)} = \sum_{j=1}^{H_i \times V_i} \widetilde{A}_{jk}^{(i)} \boldsymbol{q}_{jk}, \; i = 1, 2, .., m, \;\; k = 1, 2, \ldots, c_i. \tag{16}$$

In the following, we show the Feature Extractor as a function $\boldsymbol{f}(\boldsymbol{\theta}_f; .)$ where $\boldsymbol{\theta}_f$ denotes the network parameters.

Table 2 Specifications of Hyperparameters of DML methods and their adjustments

Hyper-parameter	Description	Value
model−lr	Learning rate of pre-trained Inception-V2 network.	10^{-4}
embedding−lr	Learning rate of the embedding layer.	10^{-3}
margin	The margin of triplet sampling and hinge loss	0.01
α	The degree in Angular loss.	45°
proxy−lr	The learning rate of the Proxy-NCA.	Adjusted from {0.01, 0.15,0.02}
μ and α	The generation ratio and mix up interpolation factor in Proxy-synthesis.	$\mu \in \{0.5,1,2\}$ and $\alpha = \{0.2,0.4,0.6\}$
dropout rate	The dropout rate in the classification module.	0.1
λ_0	Balances the trade-off between the metric and classification losses in Adapt-Adv-DML.	Adjusted from {0.1, 0.5,1}
l_{thresh}	The classification loss threshold.	1.5

Table 3 Information Retrieval Results in a ZSL Setting on the CUB-200-2011 [9] dataset

Method	Extension	Recall@1	Recall@2	Recall@4	Recall@8	kNN-Acc	NMI
Triplet Loss:	Base	45.39	58.86	70.41	80.44	49.14	58.17
	Energy	39.94	52.65	66.12	78.53	42.35	55.87
	Adapt-Adv	**53.66**	**65.83**	**76.65**	**84.99**	**56.14**	**62.07**
Angular Loss:	Base	49.59	62.15	73.36	83.05	51.98	59.26
	Energy	49.95	62.19	72.75	82.63	51.92	59.91
	Adapt-Adv	**52.68**	**65.11**	**75.86**	**84.5**	**55.60**	**61.78**
Proxy-NCA:	Base	54.02	66.83	78.07	85.7	56.74	63.62
	Energy	53.83	66.49	77.67	86.38	56.48	63.62
	Proxy-Syn	55.18	66.33	76.83	85.43	57.21	64.12
	Adapt-Adv	**57.38**	**68.91**	**78.58**	**87.07**	**60.20**	**65.86**

3.2 Tuplet sampler

As mentioned, many DML algorithms use pairs or triplet constraints. Some work extends the triplet sampling to quadruplets [4] and N-pairs [5]. Recently, proxy-based methods are introduced that aim to learn the semantic embedding without a sampling procedure [13, 15].

As seen in Fig. 8, the proposed approach can be easily applied to any DML method. In our work, the sampling procedure is selected based on the DML algorithm. The input of this algorithm is a mini-batch B of extracted features along with their labels.

$B = \{((\boldsymbol{f}_1, y_1), (\boldsymbol{f}_2, y_2), \ldots, (\boldsymbol{f}_n, y_n))\}$, where $\boldsymbol{f}_i = \boldsymbol{f}(\boldsymbol{\theta}_f; \boldsymbol{x}_i)$

The output is a set of tuples denoted by:

$$T = \{T_1, T_2, \ldots, T_N\}.$$

Note that proxy-based DML methods do not need a sampling procedure. In this case, we directly pass the mini-batch B to the metric embedding layer.

3.3 Metric embedding layer

This layer is built on top of the feature extractor. Its input of the layer is a mini-batch of tuples generated by the sampler. As our approach can be applied to almost any DML method, we can generally denote this layer by the function $\boldsymbol{g}(\boldsymbol{\theta}_m; .)$ that transforms the extracted features to the semantic embedding space.

Let $\mathcal{L}_m$ be the metric loss function, we can optimize the parameters $\{\boldsymbol{\theta}_f, \boldsymbol{\theta}_m\}$ by solving the following optimization problem:

$$\widehat{\boldsymbol{\theta}}_f, \widehat{\boldsymbol{\theta}}_m = \arg\min_{\boldsymbol{\theta}_f, \boldsymbol{\theta}_m} \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_m\left(T_i; \boldsymbol{\theta}_f, \boldsymbol{\theta}_m\right). \quad (17)$$

Minimizing the above loss function helps learning an appropriate semantic embedding space for the classes included in the training dataset. However, it may not be suitable for the new concepts (or even observed categories with few samples). Thus, we should force the feature extractor to mine a more

Table 4 Information Retrieval Results in a ZSL Setting on the CARS-196 [33] dataset

Method	Extension	Recall@1	Recall@2	Recall@4	Recall@8	kNN-Acc	NMI
Triplet Loss:	Base	44.13	57.51	70.1	79.79	46.28	52.50
	Energy	34.93	48.49	61.8	74.16	37.57	48.93
	Adapt-Adv	**62.51**	**74.23**	**83.22**	**89.45**	**65.18**	**57.71**
Angular Loss:	Base	59.76	71.16	79.79	86.87	61.33	55.49
	Energy	60.34	71.37	80.09	87.27	62.29	55.52
	Adapt-Adv	**66.99**	**77.58**	**85.39**	**90.63**	**68.33**	**59.49**
Proxy-NCA:	Base	65.67	76.67	84.68	90.65	67.78	61.20
	Energy	64.11	75.18	83.91	89.69	66.40	60.16
	Proxy-Syn	66.35	76.43	84.32	90.33	68.48	6025
	Adapt-Adv	**69.94**	**79.66**	**86.37**	**91.42**	**71.76**	**63.53**

Table 5 Information Retrieval Results in a ZSL Setting on the on the Oxford 102 Flowers [34] dataset

Method	Extension	Recall@1	Recall@2	Recall@4	Recall@8	kNN-Acc	NMI
Triplet Loss:	Base	81.55	88.89	93.59	96.58	82.22	70.16
	Energy	78.24	86.28	92.02	96.05	80.24	68.42
	Adapt-Adv	**89.32**	**94.26**	**96.48**	**97.84**	**90.39**	**76.59**
Angular Loss:	Base	84.89	90.89	94.63	97.12	86.17	72.39
	Energy	85.21	90.81	94.31	96.8	85.93	71.39
	Adapt-Adv	**89.64**	**93.94**	**96.56**	**98.16**	**90.31**	**76.05**
Proxy-NCA:	Base	86.3	91.24	95.33	97.38	88.38	73.79
	Energy	87.42	92.26	95.59	97.68	88.44	75.42
	Proxy-Syn	87.76	93.38	95.91	97.7	88.54	74.27
	Adapt-Adv	**91.67**	**95.27**	**97.62**	**98.69**	**92.42**	**80.46**

general representation and discard the features specific to the observed classes. To this end, we propose to combine the DML process with the classification module in an adversarial manner.

3.4 Classification module

This module can be implemented by a multi-layer neural network with the Softmax activation function at the last layer. Generally, we can denote this module by $\boldsymbol{h}(\boldsymbol{\theta}_c; \boldsymbol{f}_i)$ where $\boldsymbol{\theta}_c$ is the parameters and $\boldsymbol{f}_i$ is the extracted feature vector from the input image (i.e., $\boldsymbol{f}_i = \boldsymbol{f}(\boldsymbol{\theta}_f, \boldsymbol{x}_i)$). We train this module using the standard Cross-Entropy loss:

$$\mathcal{L}_{cross-entropy}\left(\boldsymbol{x}_i, y_i; \boldsymbol{\theta}_f, \boldsymbol{\theta}_c\right) = -\log p_{y_i} = -\log \sum_{k=1}^{C} \mathbf{1}_{(y_i=k)} [\boldsymbol{h}(\boldsymbol{\theta}_c; \boldsymbol{f}_i)]_k, \tag{18}$$

where p_{y_i} is the probability of the correct label obtained by applying the Softmax on the last layer of the module and $[\boldsymbol{h}(\boldsymbol{\theta}_c; \boldsymbol{f}_i)]_k$ denotes the k-th component of the classification output. Generally, we indicate the classification loss by $\mathcal{L}_c(\boldsymbol{x}_i; \boldsymbol{\theta}_f, \boldsymbol{\theta}_c)$.

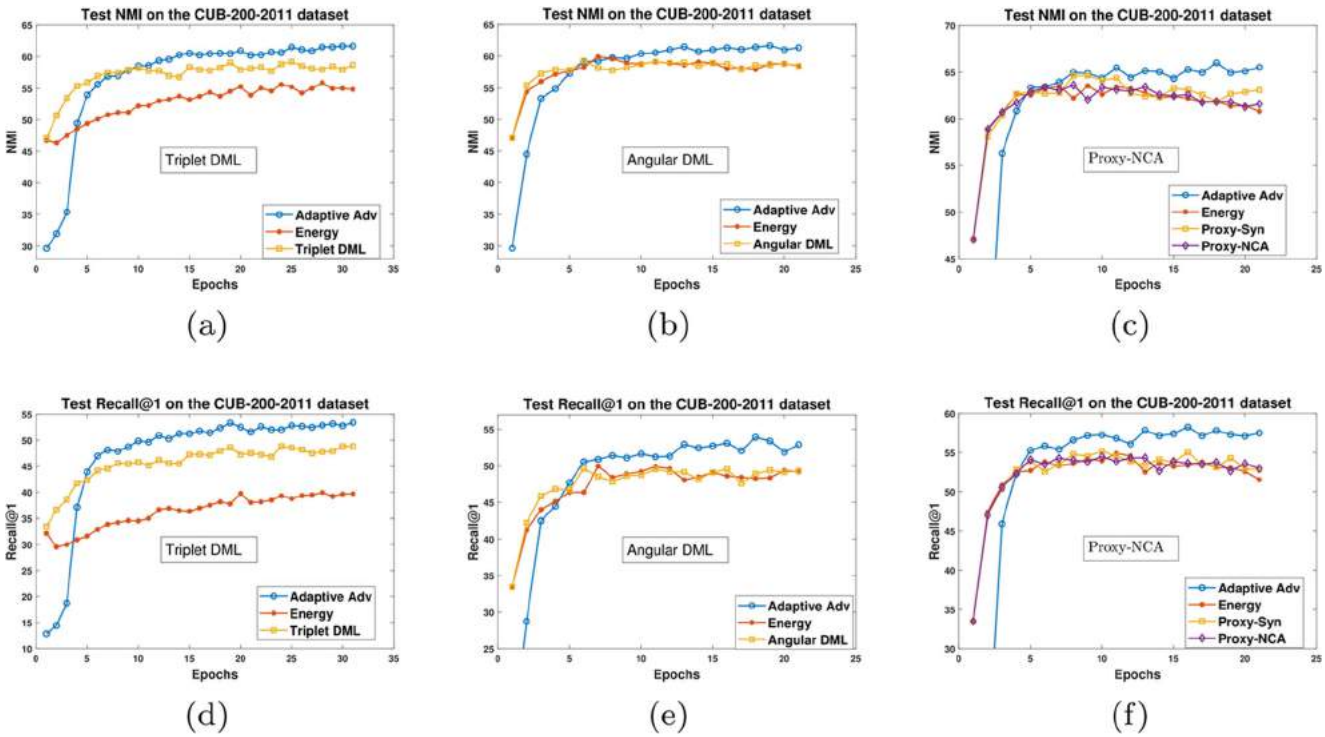


Fig. 12 NMI and Recall@1 of the evaluated methods on the CUB-200-2011 [9] dataset

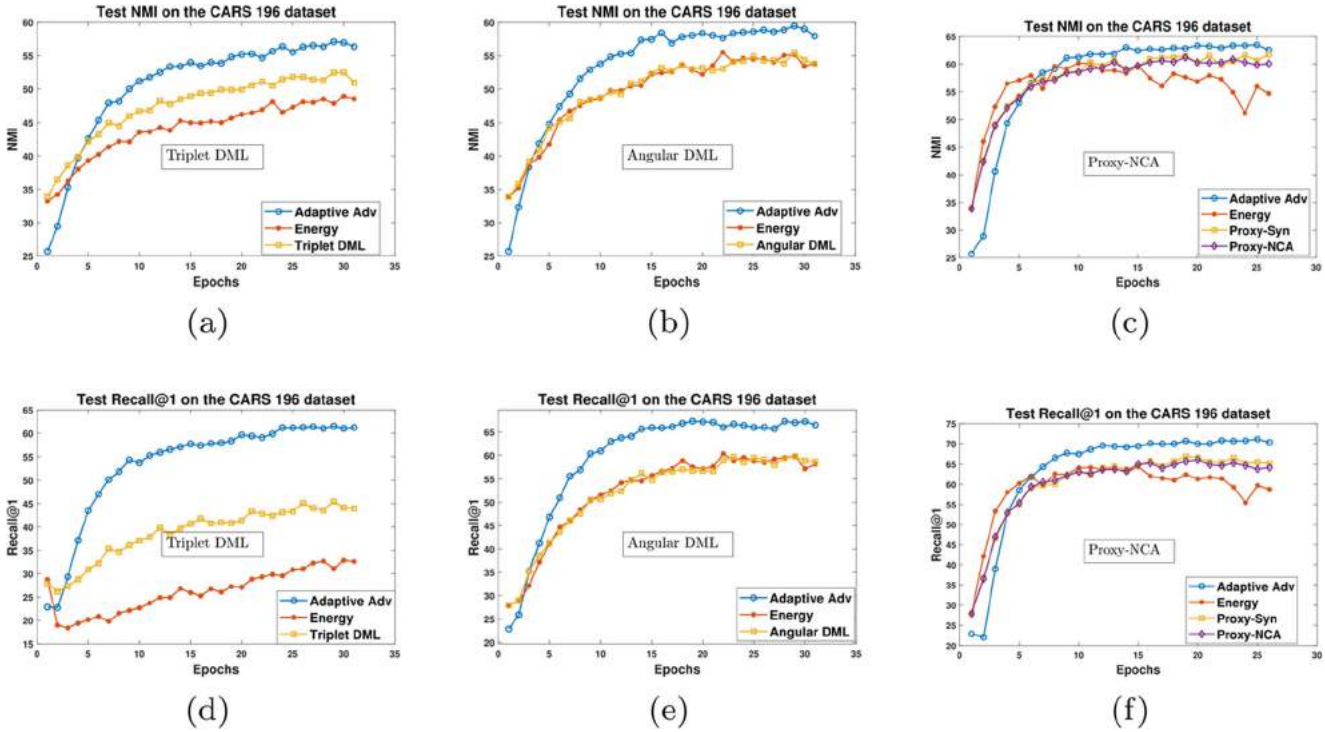


Fig. 13 NMI and Recall@1 of the evaluated methods on the CARS-196 [33] dataset

3.5 Model integration

In the training process, we aim to learn a semantic embedding space by minimizing the metric loss (17) on the $\{\boldsymbol{\theta}_f, \boldsymbol{\theta}_m\}$ parameters. Meanwhile, we should enforce the feature extractor to discard class-specific features by utilizing the classification module. To this end, we propose the following adversarial approach.

3.5.1 Adaptive adversarial approach

In this approach, we utilize the classification module in an adaptive adversarial setting. We observed that in the initial stages of training, minimizing the classification loss on seen classes helps to learn a better discriminative representation. However, in the later stages, it is necessary to *increase* the classification loss (*jointly with minimizing the metric loss*) to enforce the feature extractor to learn more general discriminative features that are not limited to available classes in the training set. Therefore, we propose the final loss function as:

$$\mathcal{L}_{adapt-adv}\left(B, T; \boldsymbol{\theta}_f, \boldsymbol{\theta}_m, \boldsymbol{\theta}_c\right) = \mathcal{L}_m\left(T, \boldsymbol{\theta}_f, \boldsymbol{\theta}_m\right) - \lambda\, \mathcal{L}_c\left(B, \boldsymbol{\theta}_f, \boldsymbol{\theta}_c\right). \tag{19}$$

When $\lambda < 0$, the proposed loss is optimized by minimizing both the metric and classification loss on seen classes. After a certain classification loss is met, we change the sign of λ and force the optimizer to increase the classification loss.

To optimize $\mathcal{L}_{adapt-adv}$, we first freeze the classifier parameters (i.e., $\boldsymbol{\theta}_c$) and minimize the loss on the feature extractor and metric embedding layer parameters as follows:

Table 6 Ablation studies of general discriminative feature learning and the class adversarial module on the Oxford 102 Flowers [34] dataset

Method	Extension	NMI	Recall@1	Recall@2	Recall@4	Recall@8	kNN-Acc
Triplet Loss:	Base	70.16	81.55	88.89	93.59	96.58	82.22
	Tri+GDFL	73.66	86.41	91.86	95.09	97.6	88.12
	Tri+Adv	72.73	84.54	90.68	94.71	97.36	86.03
	Adapt-Adv	**76.59**	**89.32**	**94.26**	**96.48**	**97.84**	**90.39**

$$\widehat{\boldsymbol{\theta}}_f, \widehat{\boldsymbol{\theta}}_m = \arg\min_{\boldsymbol{\theta}_f, \boldsymbol{\theta}_m} R_{adv}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_m\left(T_i; \boldsymbol{\theta}_f, \boldsymbol{\theta}_m\right) - \frac{\lambda}{n}\sum_{i=1}^{n}\mathcal{L}_c\left(\boldsymbol{x}_i; \boldsymbol{\theta}_f, \boldsymbol{\theta}_c\right). \quad (20)$$

Subsequently, the classifier tries to minimize the classification loss by maximizing the loss as follows:

$$\widehat{\boldsymbol{\theta}}_c = \arg\max_{\boldsymbol{\theta}_c} R_{adv} \equiv \arg\min_{\boldsymbol{\theta}_c} \frac{\lambda}{n}\sum_{i=1}^{n}\mathcal{L}_c\left(\boldsymbol{x}_i; \boldsymbol{\theta}_f, \boldsymbol{\theta}_c\right). \quad (21)$$

It motivates a minimax game between the classifier and the feature extractor. On one hand, the feature extractor confuses the classifier by maximizing $\mathcal{L}_c$, and on the other hand, the classifier tries to find class-discriminative information in the feature representations to identify the correct label. Algorithm 1 summarizes the steps of the proposed adversarial approach.

Algorithm1. The proposed adversarial approach

Inputs: $\{(\boldsymbol{x}_i, y_i), i = 1,2, \ldots, L\}$, λ : Controls trade-off between metric learning and classifier objectives

Output: Learned Embedding $\boldsymbol{h}$, Feature Extractor $\boldsymbol{f}$

1.Initialize the feature extractor with a pre-trained Convnet

2. for iter = 1,2, ... MAX_Iter

2.1. B← Get a mini-batch of data and forward it through the network to generate features vectors.

2.2. Obtain the tuples T from the input batch B using the sampling module[1].

2.3. Freeze the classification module.

2.4. Optimize deep feature extractor and metric embedding parameters:

$$\widehat{\boldsymbol{\theta}}_f, \widehat{\boldsymbol{\theta}}_m = \arg\min_{\boldsymbol{\theta}_f, \boldsymbol{\theta}_m} R_{adv} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_m(T_i;\ \boldsymbol{\theta}_f, \boldsymbol{\theta}_m) - \frac{\lambda}{n}\sum_{i=1}^{n}\mathcal{L}_c(\boldsymbol{x}_i;\ \boldsymbol{\theta}_f, \boldsymbol{\theta}_c),$$

using sophisticated neural network optimization algorithms like Adam.

2.5. Freeze the feature extractor module.

2.6. Optimize the classifier module:

$$\widehat{\boldsymbol{\theta}}_c = \arg\max_{\boldsymbol{\theta}_c} R_{adv} \equiv \arg\min_{\boldsymbol{\theta}_c} \frac{\lambda}{n}\sum_{i=1}^{n}\mathcal{L}_c(\boldsymbol{x}_i;\ \boldsymbol{\theta}_f, \boldsymbol{\theta}_c),$$

using sophisticated neural network optimization algorithms like Adam

end;

4 Implementation details

To implement the proposed approaches, we utilize the pre-trained *Inception v2* neural network [32] as the feature extractor.[6]

The network consists of 5 inception modules. Figure 10a illustrates the architecture of each module. Here, we utilize the feature maps of intermediate layers *inception_4d_output, inception_4e_output, and inception_5a_output.* The size of feature maps in these layers are 608 × 14 × 14, 1056 × 7 × 7, and 1024 × 7 × 7, respectively. The output of the last hidden layer is $\boldsymbol{u} \in \mathbb{R}^{1024}$. We use the *additive* attention mechanism denoted in Eq. (11) to promote the discrimination power of the intermediate features. The output feature vectors of attention modules are then *concatenated* in the *Inter-Module Fusion Component*. Figure 10b shows the Simplified architecture of the implemented feature extractor.

To implement the adaptive adversarial approach, we used a *Gradient Reversal Layer* (GRL) layer between the classification module and the feature extractor. The GRL acts as an identity operation in the forward phase. However, during the backward stage, it multiplies the gradient with −λ and then passes the results to the previous layer. We adjust the λ based on the classification loss ($\mathcal{L}_c$) in each epoch as follows:

$$\lambda = -\tanh(\mathcal{L}_c - l_{thresh})\lambda_0, \quad (22)$$

where l_{thresh} indicates a classification loss threshold. Hence, in the initial training epochs where $\mathcal{L}_c$ is high, the λ is adjusted with a negative value. As the $\mathcal{L}_c$ is decreased substantially (i.e., $\mathcal{L}_c < l_{thresh}$), the adversarial mechanism is activated and tries to prevent overfitting on the seen classes. The value of λ_0 is chosen from the range [.1, 1] as described in the experimental section.

The classification module is implemented by a standard feed-forward neural network with one hidden layer followed by a dropout operation. Utilizing dropout layers in the

[6] downloaded from: https://github.com/dichotomies/proxy-nca

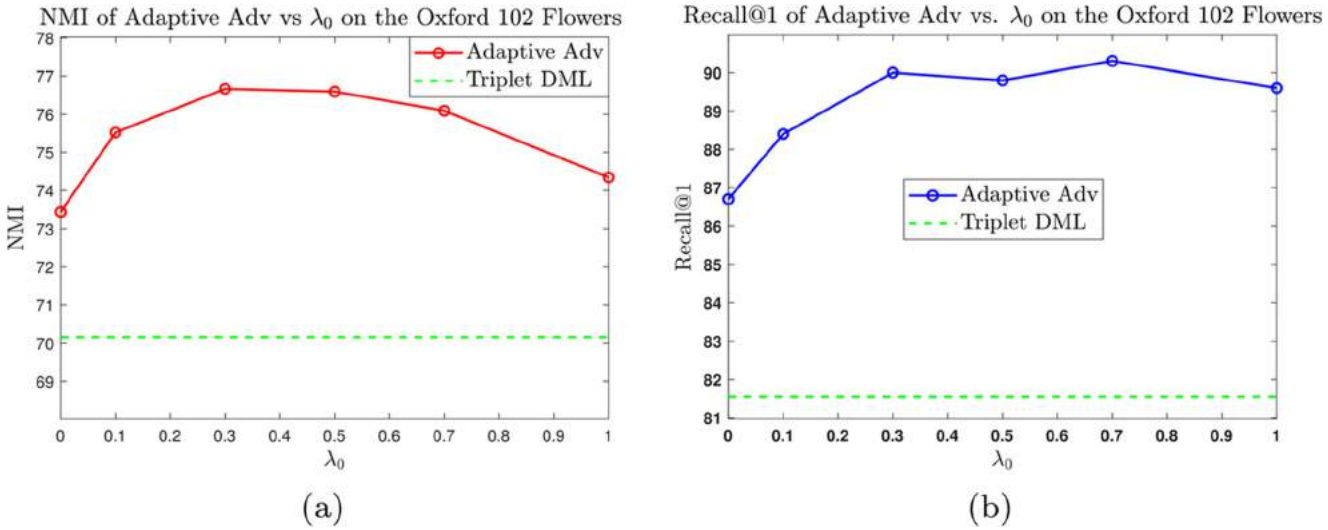


Fig. 14 NMI and Recall@1 of Adapt-Adv DML vs λ_0 values on the Oxford 102 Flowers [34] dataset

classification module simulates a large number of different network architectures by randomly dropping some nodes at the training phase. Thus, increasing the classification loss on the simulated classifiers leads to obtain a more generalizable representation. Besides, dropout is an effective yet cheap regularization technique that avoids overfitting and decreases the generalization error of deep neural networks. The architecture of the module is shown in Fig. 11.

The work is implemented using *Pytorch* deep learning library and the source code is publicly available at: https://github.com/d-zabihzadeh/Adaptive-Adv-DML.

5 Experimental results

In this section, we evaluate the performance of our method named Adapt-Adv-DML[7] on some challenging datasets in the machine vision domain. To this end, we select Triplet hinge loss,[8] Angular Loss[9] [17], and Proxy-NCA[10] [13] as baseline DML algorithms and examine how much applying our framework to these methods boosts their overall performance in a ZSL setting. Also, we compare our work with the recent Energy confusion [29, 30] and Proxy-Synthesis [27] methods on the evaluated datasets.

Subsequently, Hyperparameter analysis and the ablation study of components in our methods are provided. We also visualize the attention weights obtained by the proposed feature extraction method.

[7] Adaptive Adversarial Deep Metric Learning
[8] Source code: https://github.com/KevinMusgrave/pytorch-metric-learning
[9] Source code: https://github.com/tomp11/metric_learning
[10] Source code: https://github.com/dichotomies/proxy-nca
[11] Normalized Mutual Information

5.1 Data description

The CUB-200-2011, Cars, and Flower102 are widely used datasets selected in our work. The statistics of these datasets are summarized in Table 1.

5.2 Evaluation metrics

To evaluate the proposed methods, we adopt standard information retrieval metrics: Recall@k and *NMI*.[11] Recall@k measures the proportion of relevant images in the top-k retrieved results.

NMI indicates the quality of clustering. Let $C = \{c_1, c_2, \ldots, c_n\}$ be the clustering assignment set provided by a clustering method. Given the true labels $Y = \{y_1, y_2, \ldots, y_n\}$, NMI is computed as:

$$NMI = 2 \times \frac{I(Y;C)}{H(Y) + H(C)}, \tag{23}$$

where *I(;)* is the mutual information, and $H(.)$ indicates the entropy.

Also, we evaluate the accuracy of kNN ($k = 5$) in the ZSL setting. Here, a test image is classified correctly provided *3 or more* relevant (with the same label) images be among the *5-top* results.

5.3 Experimental setup

For a fair comparison, we adopt the pre-trained *Inception-V2* for all evaluated methods. We found that the learning rate=10^{-4} in the network is appropriate for fine-tuning. Thus, we set $lr = 10^{-4}$ for all evaluated methods. Also, similar

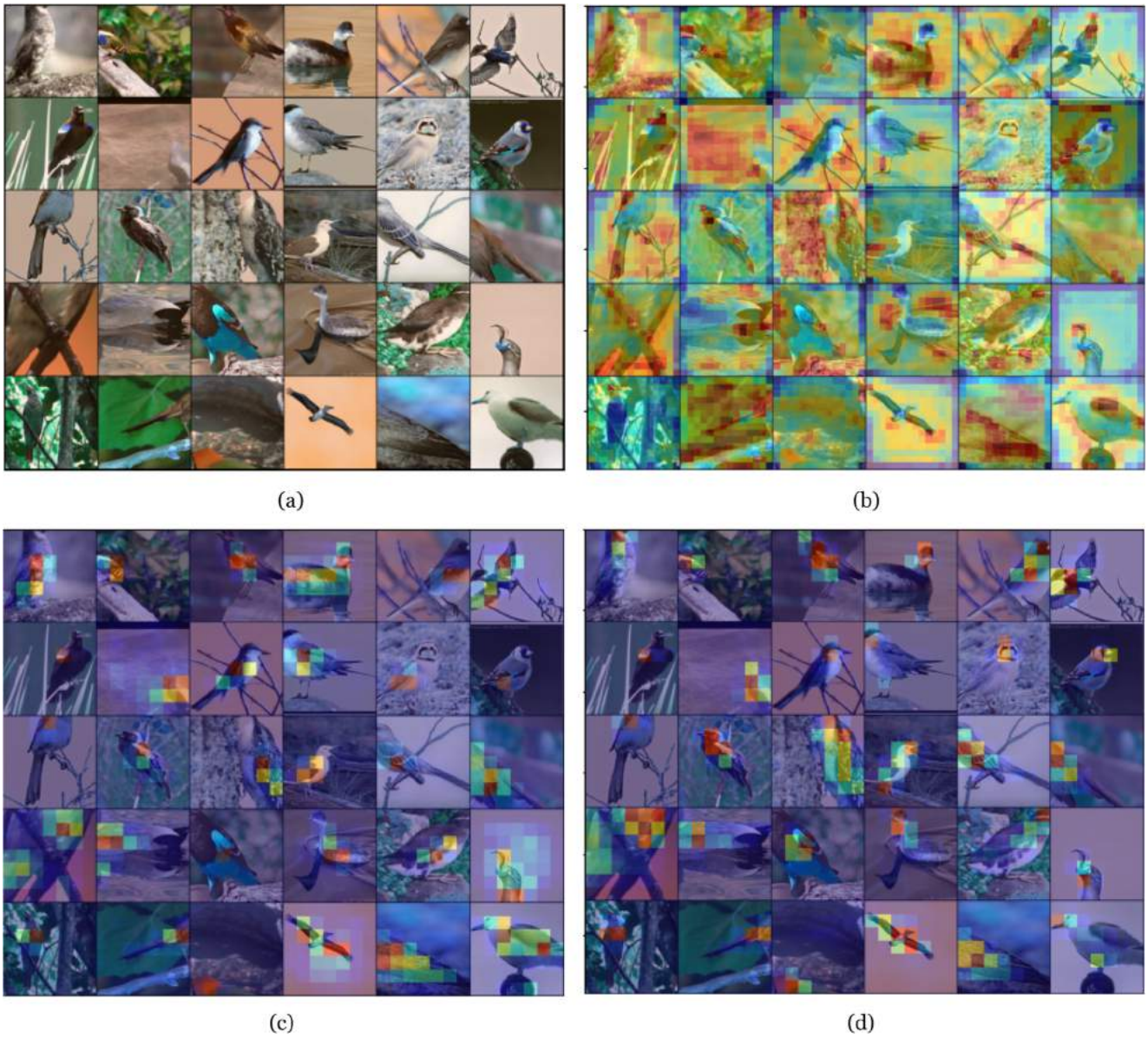


Fig. 15 Attended Feature maps of some training images in the CUB-200-2011 [9] dataset

to [29], the embedding layer learning rate is set to 10^{-3} (10 times faster than other layers).

Like [13], the learned embedding size is set to 64, and we adopt the same transformations on the input images. The images are reshaped to 256 × 256, and then randomly are cropped to 224 × 224. The batch size is also set to 64 for all methods.

As our work concentrates improving of DML on unseen classes, we almost keep the default values of hyper-parameters in DML methods for all datasets and perform small adjustments. To adjust the hyper-parameters, we split the training information into 80/20 train/validation randomly and perform a grid search. For triplet-hinge DML, we adopt the *'semi-hard'* sampling strategy. We also find out setting *margin* = 0.01 is appropriate on all the evaluated datasets. In Angular loss, we keep the default value of $\alpha = 45^{\circ}$ for all datasets. The learning rate of proxy centers in Proxy-NCA is selected from the range {0.01, 0.15, 0.02}. The proxy generation ratio (μ) and mix up interpolation factor (α) in Proxy-synthesis are adjusted from {0.5, 1, 2} and {0.2, 0.4, 0.6} (respectively)

Finally, in our method, we use *dropout-rate*=0.1, and l_{thresh} = 1.5 in all experiment. Also, the value of λ_0 is selected from {0.1, 0.5, 1}. Table 2 summarizes the hyper-parameters of evaluated DML methods along with their adjustments.

5.4 Results and analysis

In the first experiment, we evaluate the proposed methods in the CUB-200-2011, Cars-196, and Oxford Flowers-102 datasets in the ZSL setting as described in Table 1. We repeat

each experiment five times and report the mean of the results in Tables 3, 4 and 5. Also, we plot the *NMI* and Recall@k scores of the evaluated methods versus epochs on the test classes in CUB-200-2011 and Cars-196 datasets in Figs. 12 and 13, respectively.

As the results indicate, the proposed method obtains a considerable improvement over the baseline DML methods. It confirms that our approach (general and discriminative feature learning + adversarial classification module) is indeed beneficial for the generalization of DML methods in a ZSL setting. It learns a more generalizable and discriminative representation from the input image and effectively prevents overfitting on the observed classes. Meanwhile, Proxy-Synthesis shows minor improvement over the baseline loss. Note that Proxy-Synthesis generates a larger batch from the input batch by generating synthetic proxies and samples. However, it is less effective compared with our method. For example, Adapt-Adv achieves the Recall@1 = 69.94% and kNN-Acc = 71.76% on the Cars dataset, while Proxy-Synthesis reaches Recall@1 = 66.88% and kNN-Acc = 68.48%.

Moreover, we did not observe a significant advantage from the Energy confusion approach over the baseline methods. The main drawback of the approach is that the confusion coefficient λ has been fixed during the training phase. Hence, it prevents the baseline DML to learn the discriminative features efficiently during the initial stages of training. Besides, the energy confusion term does not consider the structure of the embedding and randomly selects instances from opposite classes. In contrast, our method captures the global structure of the embedding by employing a classification module.

5.4.1 Ablation study

In this experiment, we provide ablation studies of both *general discriminative feature learning* and the *class adversarial module* and investigate the contribution of each component individually. We choose the Triplet loss as a baseline and derive two variants of the proposed *Adapt-Adv* as follows:

1. Tri + GDFL[12]: indicates triplet DML using the *general discriminative feature vector*. Here, we omit the class adversarial module.
2. Tri + Adv: shows triplet DML along with the *class adversarial module*. Here, we discard the *general discriminative features* and use the output of the last hidden layer as the feature vector.

Table 6 reports the results. As the results show, we can derive the following conclusions:

I. Both the *general discriminative feature learning* and *class adversarial module* are effective and improve the performance of the baseline DML.
II. The *general discriminative feature learning* is more effective in improving the baseline method.

5.4.2 Hyper parameters analysis

We evaluate the effect of the adversarial coefficient λ_0 in this experiment. To this end, we select the Triplet loss as the baseline DML and train the Adapt-Adv DML on the Oxford 102 Flowers dataset at different λ_0 values. The ZSL setting as specified in Table 1 is adopted in the experiment. Fig. 14 depicts the NMI and Recall@1 of the Adapt-Adv DML vs λ_0 values on unseen classes.

The results indicate the effectiveness of the class adversarial module. by choosing $\lambda_0 = 0$, our method reduces to Triplet DML with only *general discriminative feature learning* and the performance is unsatisfactory in comparison with the other λ_0 values. As λ_0 increases, the quality of the learned embedding peaks around the range (0.3, 0.7) and our method surpasses the baseline by a large margin.

5.4.3 Visualization of attended feature maps

In the next experiment, we illustrate the attended feature maps of the proposed feature extractor in the CUB-200-2011 dataset. Figure 15 shows some preprocessed training images along with attended feature maps of selected layers. As seen, the attended features cover the most important parts of the images and each layer attends to different regions of the input images. The *inception_4d* feature maps capture more general and visual patterns of the images whereas *inception_4e* and inception_5a focused on more discriminative parts of the birds. Thus, the concatenated feature vector contains weighted useful patterns of the input image that is useful to discriminate both seen and unseen classes during the test stage.

6 Conclusion and future work

This paper presented a novel framework to enhance the generalization of DML methods in a ZSL setting. To this end, we developed a generalized and discriminative feature learning approach and also utilized an adaptive class adversarial module. Our work can be applied to many baseline DML algorithms improving their performance on unseen classes by a large margin. The proposed methods learn a general representation covering the most discriminative parts of the input image, which is useful for both the seen and unseen categories.

[12] General Discriminative Feature Learning

We evaluated the proposed methods on some challenging datasets in machine vision domains in a ZSL setting. The obtained results were very encouraging, and the proposed methods consistently outperformed the baselines on all evaluated datasets. It confirmed the necessity and significance of our idea that adopting some general yet discriminative feature learning as well as an adaptive confusion mechanism, is indeed helpful for most DML applications.

In future work, we intend to examine our feature learning approach on different popular deep neural network architectures and utilize the proposed framework in other applications of DML. Besides, we aim to extend the work for semi-supervised learning.

Acknowledgments We would like to acknowledge the Machine Learning Lab in the Engineering Faculty of FUM for their kind and technical support.

Availability of data Datasets used in the experiments are publicly available and can be downloaded from the following links:

1. Oxford 102 Flowers: https://www.robots.ox.ac.uk/~vgg/data/flowers/102/
2. CUB-200-2011: https://github.com/cyizhuo/CUB-200-2011-dataset
3. CARS-196: https://ai.stanford.edu/~jkrause/cars/car_dataset.html

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Chopra S, et al. (2005) Learning a similarity metric discriminatively, with application to face verification. 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), IEEE
2. Hoffer E, Ailon N (2015). Deep metric learning using triplet network. International Workshop on Similarity-Based Pattern Recognition, Springer
3. Wang J, et al. (2014) Learning fine-grained image similarity with deep ranking. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition
4. Ni J, et al. (2017) Fine-grained patient similarity measuring using deep metric learning. Proceedings of the 2017 ACM on conference on information and knowledge management
5. Sohn K (2016) Improved deep metric learning with multi-class n-pair loss objective. Advances in Neural Information Processing Systems
6. Kaya M, Bilge HŞ (2019) Deep metric learning: a survey. Symmetry 11(9):1066
7. Simo-Serra E, et al. (2015) Discriminative learning of deep convolutional feature point descriptors. Proceedings of the IEEE International Conference on Computer Vision
8. Schroff F, et al. (2015) Facenet: a unified embedding for face recognition and clustering. Proceedings of the IEEE conference on computer vision and pattern recognition
9. Wah C, et al. (2011) The Caltech-UCSD Birds-200-2011 dataset, Computation & Neural Systems, technical report, CNS-TR-2011-001
10. Wu C-Y, et al. (2017) Sampling matters in deep embedding learning. Proceedings of the IEEE International Conference on Computer Vision
11. Zheng W, et al. (2019) Hardness-aware deep metric learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition
12. Kim M, et al. (2022) Variational continual proxy-anchor for deep metric learning. International Conference on Artificial Intelligence and Statistics, PMLR
13. Movshovitz-Attias Y, et al. (2017) No fuss distance metric learning using proxies. Proceedings of the IEEE International Conference on Computer Vision
14. Oh Song H, et al. (2017) Deep metric learning via facility location. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition
15. Qian Q, et al. (2019) Softtriple loss: deep metric learning without triplet sampling. Proceedings of the IEEE International Conference on Computer Vision
16. Ge W (2018) Deep metric learning with hierarchical triplet loss. Proceedings of the European conference on computer vision (ECCV)
17. Wang J, et al. (2017) Deep metric learning with angular loss. Proceedings of the IEEE International Conference on Computer Vision
18. Ustinova E, Lempitsky V (2016) Learning deep embeddings with histogram loss. Adv Neural Inf Proces Syst 29:4170–4178
19. Wang X, et al. (2019) Multi-similarity loss with general pair weighting for deep metric learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition
20. Yao H, Zhang S, Hong R, Zhang Y, Xu C, Tian Q (2019) Deep representation learning with part loss for person re-identification. IEEE Trans Image Process 28(6):2860–2871
21. Oh Song H, et al. (2016) Deep metric learning via lifted structured feature embedding. Proceedings of the IEEE conference on computer vision and pattern recognition
22. Jiang W, Huang K, Geng J, Deng X (2020) Multi-scale metric learning for few-shot learning. IEEE Trans Circ Syst Video Technol 31:1091–1102
23. Li X, Yu L, Fu CW, Fang M, Heng PA (2020) Revisiting metric learning for few-shot image classification. Neurocomputing 406: 49–58
24. Duan Y, et al. (2018) Deep adversarial metric learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition
25. Wang Z, et al. (2020) Adaptive margin based deep adversarial metric learning. 2020 IEEE 6th Intl conference on big data security on cloud (BigDataSecurity), IEEE Intl conference on high performance and smart computing,(HPSC) and IEEE Intl conference on intelligent data and security (IDS), IEEE
26. Xu X, He L, Lu H, Gao L, Ji Y (2019) Deep adversarial metric learning for cross-modal retrieval. World Wide Web 22(2):657–672
27. Gu G, Ko B (2021) Proxy synthesis: learning with synthetic classes for deep metric learning. Proc AAAI Conference on Artificial Intelligence (AAAI), 34, 10853, 10860
28. Zhang H et al (2017) mixup: beyond empirical risk minimization. arXiv preprint arXiv:1710.09412
29. Chen B, Deng W (2019) Energy confused adversarial metric learning for zero-shot image retrieval and clustering. Proceedings of the AAAI Conference on Artificial Intelligence.
30. Zhu J, Zhong D, Luo K (2020) Boosting unconstrained Palmprint recognition with adversarial metric learning. IEEE Trans Biom Behav Identity Sci 2(4):388–398

31. Shen T, et al. (2018) Disan: directional self-attention network for rnn/cnn-free language understanding. Proceedings of the AAAI Conference on Artificial Intelligence
32. Szegedy C, et al. (2016) Rethinking the inception architecture for computer vision. Proceedings of the IEEE conference on computer vision and pattern recognition
33. Krause J, et al. (2013) 3d object representations for fine-grained categorization. Proceedings of the IEEE International Conference on Computer Vision Workshops
34. Nilsback M-E, Zisserman A (2008). Automated flower classification over a large number of classes. 2008 Sixth Indian conference on computer vision, Graphics & Image Processing, IEEE

Karrar Alkaabi born in Najaf year 1986 in Iraq. He received his bachelor degree in Computer Science from Al-Qadisiyah University, 2009, in Al-Diwaniyah, Iraq, and M.Sc. degree in Computer Engineering, Department of Artificial Intelligence and Robotics from Ferdowsi University of Mashhad, 2017. He works as an Assistant Lecturer at the University of Kufa, Najaf, Iraq. He is interested in Artificial Intelligence, Machine Learning, and Image Processing.

Reza Monsefi born in Ahwaz year 1956 in south of Iran. He has received his Honors Degree in Electrical and Electronic Engineering from Manchester University, 1978, Manchester, UK, M.Sc. in Control Engineering, Salford University, 1981, and Ph.D. in Data Communication and Supervisory Control, 1983, Salford University, Manchester, UK. He is a fellow member of IEE, and was a senior lecturer and full professor, retired in June 2022 from Ferdowsi University of Mashhad, Mashhad, Iran. Wireless Networks, Machine Learning and Artificial Intelligence are among his professional interests.

Davood Zabihzadeh currently is an assistant professor in computer engineering department of Hakim Sabzevari University (HSU). He received his Ph.D. in AI from Ferdowsi University of Mashhad, 2017. His current research interests include deep metric learning, zero and few shot learning, machine vision, and deep neural networks.