Hard-Aware Deeply Cascaded Embedding

Yuhui Yuan[1,3] Kuiyuan Yang[2] Chao Zhang[1,4*]
[1]Key Laboratory of Machine Perception(MOE),Peking University
[2]DeepMotion [3]Microsoft Research
[4]Cooperative Medianet Innovation Center, Shanghai Jiao Tong University
yhyuan@pku.edu.cn, kuiyuanyang@deepmotion.ai, chzhang@cis.pku.edu.cn

Abstract

Riding on the waves of deep neural networks, deep metric learning has achieved promising results in various tasks by using triplet network or Siamese network. Though the basic goal of making images from the same category closer than the ones from different categories is intuitive, it is hard to optimize the objective directly due to the quadratic or cubic sample size. Hard example mining is widely used to solve the problem, which spends the expensive computation on a subset of samples that are considered hard. However, hard is defined relative to a specific model. Then complex models will treat most samples as easy ones and vice versa for simple models, both of which are not good for training. It is difficult to define a model with the just right complexity and choose hard examples adequately as different samples are of diverse hard levels. This motivates us to propose the novel framework named Hard-Aware Deeply Cascaded Embedding(HDC) to ensemble a set of models with different complexities in cascaded manner to mine hard examples at multiple levels. A sample is judged by a series of models with increasing complexities and only updates models that consider the sample as a hard case. The HDC is evaluated on CARS196, CUB-200-2011, Stanford Online Products, VehicleID and DeepFashion datasets, and outperforms state-of-the-art methods by a large margin.

1. Introduction

Deep metric embedding has attracted increasing attention for various tasks , such as visual product search [6, 18, 23, 27, 35], face recognition [21, 32, 41, 4], local image descriptor learning [9, 2, 12, 24], person/vehicle re-identification [22, 7, 43, 17], zero-shot image classification [19, 45, 5], fine-grained image classification [8, 40, 44] and object tracking [14, 33]. Although deep metric embedding is modified into different forms for various tasks, it shares the same objective to learn an embedding space that pulls similar images closer and pushes dissimilar images far away. Typically, the target embedding space is learned with a convolutional neural network equipped with contrastive/triplet loss.

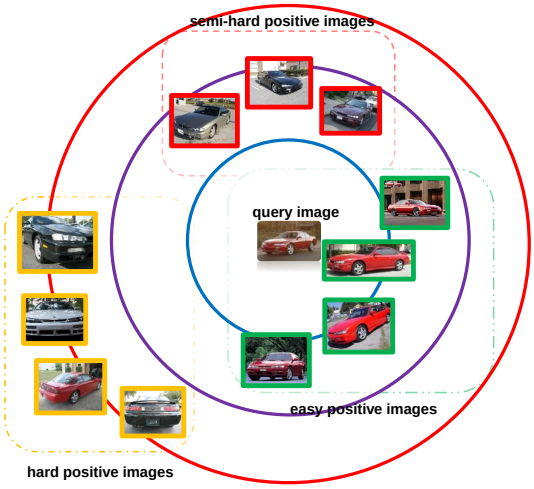


Figure 1. **Illustration of samples with different hard levels**: A query image is shown at the center, while other images from the same category (Nissan 240X Coupe 1998 from CARS196 [11]) are used to form positive pairs with the query image.

Different from the traditional classification based models, the models of deep metric embedding consider two images (a pair) or three images (a triplet) as a training sample. Thus N images can generate $\mathcal{O}(N^2)$ or $\mathcal{O}(N^3)$ samples. It becomes impossible to consider all samples even for a moderate number of images. Fortunately, not all samples are equally informative to train a model, which inspires many recent works to mine hard examples for training [8, 24, 39].

However, the hard level of a sample is defined relative to a model. Then samples can be divided into different hard levels as illustrated in Figure 1. For a complex model, most samples will be treated as easy ones, and the model converges fast but is prone to overfitting. While for a simple model, most samples will be treated as hard ones and cannot

*Corresponding author : Chao Zhang.

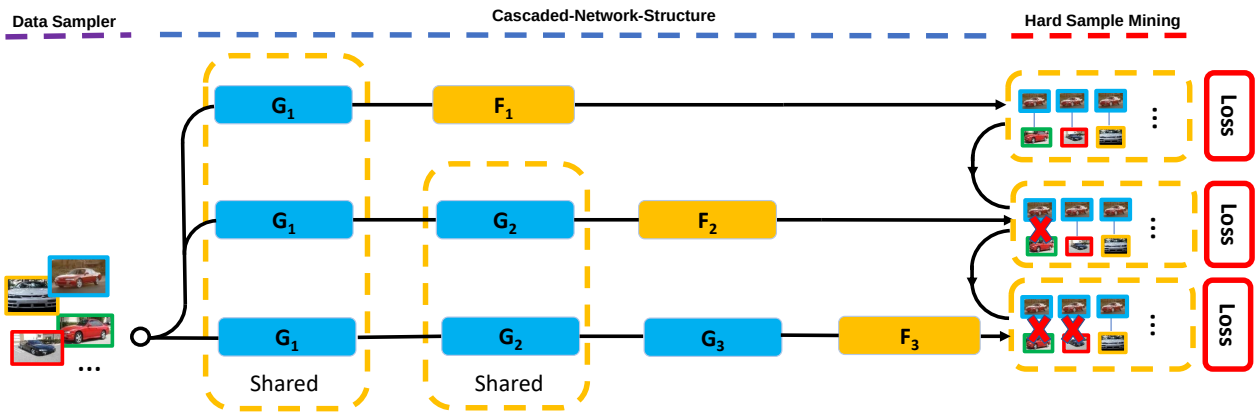


Figure 2. **Hard-Aware Deeply Cascaded Embedding** : We will train the first model with all the pairs, the second model with the semi-hard samples which are selected by the above model, the third model with the remained hard samples selected by the second model. Our framework support any K cascaded models. We plot the case=3 for convenience. $G_1, G_2, G_3, F_1, F_2, F_3$ are the computation blocks in Convolutional Neural Networks.

fully benefit from hard example mining. It would be ideal to define a model with the just right complexity to mine hard examples adequately, which is an open problem itself.

To alleviate the above problem, we ensemble a set of models with different complexities in a cascaded manner and mine hard examples adaptively, which is schematically illustrated in Figure 2. The most simple model is implemented by a shallow network, while complex models are implemented by cascading more layers following the simple ones. During the training phase, a sample will be considered by a series of models with increasing depth. Specifically, a sample firstly makes its forward pass through the simple model, the pass will stop if the simple model considers the sample as an easy one, otherwise the forward pass continues until a model considers the sample as an easy one or the deepest model is reached. Then the errors will be back-propagated to models that consider the sample as a hard case. We empirically show that the HDC achieves state-of-the-art results on five benchmarks.

In summary, we make the following contributions:

- We propose the Hard-Aware Deeply Cascaded Embedding to solve the under-fitting and over-fitting problem when mining the hard samples during training. To the best of our knowledge, this is the first attempt to investigate and solve this problem.
- We conduct extensive experiments on five various datasets and all achieve state-of-the-art results. The promising results on different datasets demonstrate that the proposed method has good generalization capability.

2. Related Work

Deep metric learning attracts great attention in recent years, and hard negative mining is becoming a common practice to effectively train deep metric networks [8, 24, 39]. Wang *et al*. [39] sample triplets during the first 10 training epoches randomly, and mine hard triplets in each mini-batch after 10 epoches. Cui *et al*. [8] leverage human to label hard negative images from images assigned high confidence scores by the model during each round. Simo-Serra *et al*. [24] analyze the influence of both of hard positive mining and hard negative mining, and find that the combination of aggressive mining for both positive and negative pairs improves the discrimination. However, these methods mine the hard images only based on a single model, which cannot adequately leverage samples with different hard levels.

Our method of ensembling a set of models of different complexities in a cascaded manner shares the same spirit as the acceleration technique used in object/face detection [1, 16, 30, 42]. In the detection task, an image may contain several positive patches and a large number of negative patches. To reduce the computational cost, the model is broken down into a set of cascaded computation blocks, where computation blocks at early stages reject most easy background patches, while computation blocks at latter stages focus more on object-like patches.

Our method also shares similar form with deeply-supervised network (DSN) proposed for image classification [15], of which loss functions are added to the output layers and several middle layers. DSN improves the directness and transparency of the hidden layer learning process and tries to alleviate the "gradient vanishing" problem. Similar idea is adopted in GoogLeNet [31]. BranchyNet [34] attempts to speed up image classification by taking advantage of the DSN framework during test phase, where an image will be predicted using features learned at an early layer if high confidence score can be achieved. During the training phase of DSN, all samples are used and intermediate losses are only used to assist the training of the deepest model. While in our framework, samples of different hard

levels are assigned to models with adequate complexities, and all models are ensembled together as a whole model. To be noted, ensemble is also a useful technique that has been widely used in model design to boost performance. Hinton *et al*. [29] add dropout into fully-connected layers, which implicitly ensembles an exponential number of sub-networks in a single network. He *et al*. [10] propose ResNet by adding residual connections into a network and win the *ILSVRC 2015* competition, which is latterly proved by Veit *et al*. [36] that ResNet is actually exponential ensembles of relatively shallow networks.

In addition, there are several works focused on designing new loss functions for deep metric embedding recently. Rippel *et al*. [19] design a Nearest Class Multiple Centroids (NCMC) like loss which encourages images from the same category to form sub-clusters in the embedding space. Huang *et al*. [6] propose position-dependent deep metric to solve the problem that intra-class distance in a high-density region may be larger than the inter-class distance in low-density regions. Ustinova *et al*. [35] propose a histogram loss, which aims to make the similarity distributions of positive and negative pairs less overlapping. Unlike other losses used for deep embedding, histogram loss comes with virtually no parameters that need to be tuned. K. Sohn *et al*. [26] proposed multi-class N-pair loss by generalizing triplet loss by allowing joint comparison among more than one negative examples. Different from our work, these works improve deep embedding by designing new loss functions within a single model. They can benefit from our method by mining hard examples adaptively using multiple cascaded models.

3. Hard-Aware Deeply Cascaded Embedding

Hard-Aware Deeply Cascaded embedding(HDC) is based on a straightforward intuition: handling samples of different hard levels with models of different complexities. Based on deep neural networks, models with different complexities can be naturally derived from sub-networks of different depths. For clarity, we will first formulate the general framework of HDC and then analyze the concrete case for the contrastive loss.

3.1. Model Formulation

Here are some notations that will be used to describe our method:

- $\mathcal{P} = \{I_i^+, I_j^+\}$: all the positive image pairs constructed from training set, where I_i^+ and I_j^+ are supposed to be similar or share the same label.
- $\mathcal{N} = \{I_i^-, I_j^-\}$: all the negative image pairs constructed from training set, where I_i^- and I_j^- are supposed to be irrelevant or from different labels.
- G_k : the k^{th} computation block including several convolutional layers, pooling layers, and other possible operations in a network. Suppose there are K blocks in total, G_1 takes an image as input, and $G_k, k > 1$ takes the outputs of its previous block as input, then all K blocks are cascaded together as a feed-forward network.
- $\{o_{i,k}^+, o_{j,k}^+\}$: the output of the k^{th} computation block G_k for the positive pairs $\{I_i^+, I_j^+\}$.
- $\{o_{i,k}^-, o_{j,k}^-\}$: the output of the k^{th} computation block G_k for the negative pairs $\{I_i^-, I_j^-\}$.
- F_k : the k^{th} transform function that transforms o_k to a low dimensional feature vector f_k for distance calculation.
- $\{f_{i,k}^+, f_{j,k}^+\}$: the k^{th} computed feature vector after F_k for the positive pairs $\{I_i^+, I_j^+\}$.
- $\{f_{i,k}^-, f_{j,k}^-\}$: the k^{th} computed feature vector after F_k for the negative pairs $\{I_i^-, I_j^-\}$.

Accordingly, there are K models corresponding to K sub-networks of different depths. The first model is the simplest one which uses the first block G_1 and generates features for the pairs $\{I_i, I_j\}$ by:

$$\{o_{i,1}, o_{j,1}\} = G_1 \circ \{I_i, I_j\} \tag{1}$$
$$\{f_{i,1}, f_{j,1}\} = F_1 \circ \{o_{i,1}, o_{j,1}\} \tag{2}$$

If the pair is considered easy by the current model, it will not be passed to more complex models. Otherwise, the pair will continue its forward pass until the k^{th} model considers it as an easy case or the final K^{th} model is reached. We can calculate the features of k^{th} model by:

$$\{o_{i,k}, o_{j,k}\} = G_k \circ \{o_{i,k-1}, o_{j,k-1}\} \tag{3}$$
$$\{f_{i,k}, f_{j,k}\} = F_k \circ \{o_{i,k}, o_{j,k}\} \tag{4}$$

Then the loss of k^{th} model is defined as:

$$\mathcal{L}_k = \sum_{(i,j)\in\mathcal{P}_k} \mathcal{L}_k^+(i,j) + \sum_{(i,j)\in\mathcal{N}_k} \mathcal{L}_k^-(i,j) \tag{5}$$

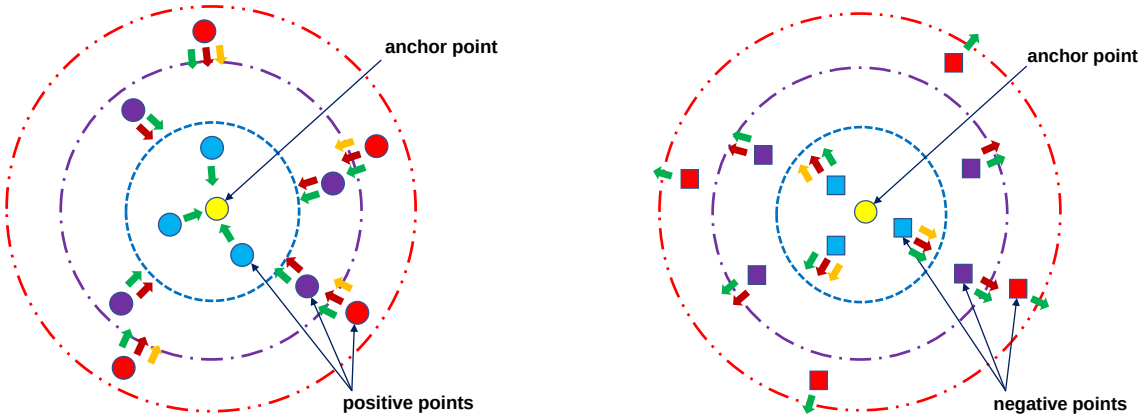
where $\mathcal{P}_k$ denotes all positive pairs that are considered as hard examples by previous models and $\mathcal{N}_k$ indicate negative ones. The definition of hard will be concretely given in Section 3.2.

Therefore, the final loss of the HDC is defined as:

$$\mathcal{L} = \sum_{k=1}^{\mathrm{K}} \lambda_k \mathcal{L}_k \tag{6}$$

where λ_k is the weight for model k.

The HDC is different from previous deep metric embedding, where only a single model (i.e., model K) is used to mine hard samples. As samples of a dataset are with diverse hard levels, it is difficult to find a single model with



(a) (b)

Figure 3. **Data Distribution:** (a) Positive Pairs Distribution: Based on the anchor point in the center, $\mathcal{P}_0$ contains all the points. $\mathcal{P}_1$ contains red, purple points. $\mathcal{P}_2$ only contains red points. (b) Negative Pairs Distribution: $\mathcal{N}_0$ contains all the points. $\mathcal{N}_1$ contains red, purple points. $\mathcal{N}_2$ only contains red points. Green arrows denote loss from Cascade-Model-1, red arrows denote loss from Cascade-Model-2 and yellow arrow denote loss from Cascade-Model-3.

the just right complexity to mine hard samples. In contrast, the HDC framework cascades multiple models with increasing complexities and mines samples of different hard levels in a seamless way.

The model parameters are distributed in G_k, F_k, $1 \leq k \leq \mathrm{K}$. They can be optimized by the standard SGD, the gradient of G_k is:

$$\frac{\partial \mathcal{L}}{\partial G_k} = \sum_{l=k}^{\mathrm{K}} \lambda_l \frac{\partial \mathcal{L}_l}{\partial G_k} \tag{7}$$

where the gradient of G_k is calculated by all the models that are built on G_k. The gradient of F_k is:

$$\frac{\partial \mathcal{L}}{\partial F_k} = \lambda_k \frac{\partial \mathcal{L}_k}{\partial F_k} \tag{8}$$

where the gradient of F_k is only calculated by model k since F_k is only used by model k for feature transformation.

The HDC is general for deep metric embedding with hard example mining. Here we take contrastive loss as an example to give the specific loss function. We first introduce the original contrastive loss which penalizes large distance between positive pairs and negative pairs with distance smaller than a margin, i.e.,

$$\mathcal{L}^+(i,j) = \mathcal{D}(f_i^+, f_j^+) \tag{9}$$

$$\mathcal{L}^-(i,j) = \max\{0, \mathcal{M} - \mathcal{D}(f_i^-, f_j^-)\} \tag{10}$$

where $\mathcal{D}(f_i, f_j)$ is the Euclidean distance between the two L2-normalized feature vectors of f_i and f_j, $\mathcal{M}$ is the margin. By applying the contrastive loss to Eq.(5), we get the HDC based contrastive loss, i.e.,

$$\mathcal{L}_k = \sum_{(i,j)\in\mathcal{P}_k} \mathcal{D}(f_{i,k}^+, f_{j,k}^+) + \sum_{(i,j)\in\mathcal{N}_k} \max\{0, \mathcal{M} - \mathcal{D}(f_{i,k}^-, f_{j,k}^-)\} \tag{11}$$

3.2. Definition of Hard Example

Given the defined loss function, we follow conventional hard example mining to define the samples of large loss values as hard examples except that multiple losses will be used to mine hard examples for each sample. Because the loss distributions are different for different models and keep changing during training, it is difficult to predefine thresholds for each model when mining hard samples. Instead, we simply rank losses of all positive pairs in a mini-batch in descending order and take top h^k percent samples in the ranking list as hard positive set for the model k. Similar strategies are adopted for hard negative example mining. Then the selected hard samples are forwarded to the later cascaded models.

Here, we use a toy dataset with positive pairs as illustrated in Figure 3(a) and negative pairs as illustrated in Figure 3(b), together with the model with K = 3 illustrated in Figure 2 to schematically the process of hard example mining. **Cascade Model-1** will forward all pairs in $\mathcal{P}_0$ and $\mathcal{N}_0$, and try to push all positive points towards the anchor point while pushing all negative points away from the anchor point, and form $\mathcal{P}_1$, $\mathcal{N}_1$ (points in the 2^{nd} and 3^{rd} tier) by selecting hard samples according to its loss. Similarly, $\mathcal{P}_2$ and $\mathcal{N}_2$ (points in the 3^{rd} tier) are formed by **Cascade Model-2**.

From the illustrated model in Figure 2, ensembling models in a cascaded manner brings an additional advantage of computational efficiency, since lots of computations are shared during forward pass which is efficient for both training and testing.

3.3. Implementation Details

We use mini-batch SGD to optimize the loss function (6), and adopt multi-batch [32] to use all the possible pairs in a mini-batch for stable estimation of the gradient. Algorithm 1 details the framework of our implementation for the HDC. Specifically, sampling strategy from [28] is to construct a mini-batch of images as input, e.g., a mini-batch of 100 images are randomly sampled evenly from 10 different categories. To leverage more training samples, we further take multi-batch method [32] to construct all image pairs in a mini-batch to calculate the training loss, e.g., $100^2 - 100 = 9900$ pairs are constructed for 100 images. With the cascaded models, an image is represented by concatenating features from all models.

Algorithm 1 Hard-Aware Deeply Cascaded Embedding.

Given training images set $\{I_i\}_{i=1}^{N}$.
for $t = 1; t < T; t++$ **do**
  Sample a mini-batch of training images, following the method in [28] and initialize the $\mathcal{P}_0$ and $\mathcal{N}_0$ within the mini-batch following the method in [32].
  for $k = 1; k \leq \mathrm{K}; k++$ **do**
    Forward all the images in set $\mathcal{P}_{k-1}$ and $\mathcal{N}_{k-1}$ to k^{th} model to compute the features according to Eq.(3) and Eq.(4).
    Compute the losses for the all pairs constructed in the mini-batch according to Eq.(9) and Eq.(10).
    Get the $\mathcal{P}_k$ and $\mathcal{N}_k$ by choosing the hard pairs following the method described in Section 3.2.
    Backward and update the gradients according to corresponding parts in Eq.(7) and Eq.(8) for all the pairs in $\mathcal{P}_k$ and $\mathcal{N}_k$.
  end for
end for

4. Experimental Evaluation

The proposed HDC is verified on image retrieval tasks and evaluated by two standard performance metrics, i.e., MAP and Recall@K. MAP [17] is the mean of average precision scores for all query images over the all the returned images. Recall@K is the average recall scores over all the query images in testing set following the definition in [27]. Specifically, for each query image, top K nearest images will be returned based on some algorithm, the recall score will be 1 if at least one positive image appears in the returned K images and 0 otherwise.

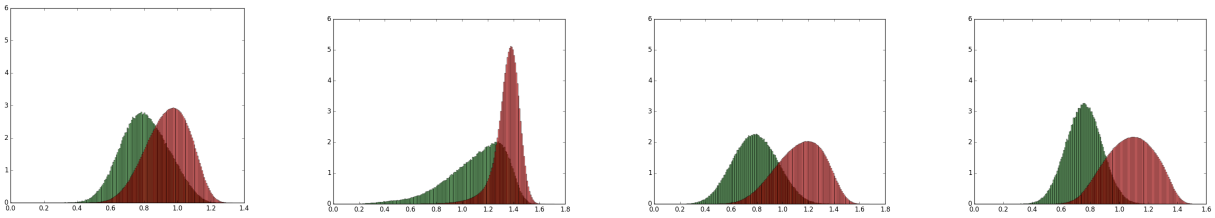
4.1. Datasets

Five datasets that are commonly chosen in deep metric embedding are used in our experiments. For fair comparison with the existing methods, we follow the standard protocol of train/test split.

- *CARS196* dataset [11], which has 196 classes of cars with 16,185 images, where the first 98 classes are for training (8,054 images) and the other 98 classes are for testing (8,131 images). Both query set and database set are the test set.
- *CUB-200-2011* dataset [37], which has 200 species of birds with 11,788 images, where the first 100 classes are for training (5,864 images) and the rest of classes are for testing (5,924 images). Both query set and database set are the test set.
- *Stanford Online Products* dataset [28], which has 22,634 classes with 120,053 products images, where 11,318 classes are for training (59,551 images) and 11,316 classes are for testing (60,502 images). Both query set and database set are the test set.
- *In-shop Clothes Retrieval* dataset [18], which contains 11,735 classes of clothing items with 54,642 images. Following the settings in [18], only 7,982 classes of clothing items with 52,712 images are used for training and testing. 3,997 classes are for training (25,882 images) and 3,985 classes are for testing (28,760 images). The test set are partitioned to query set and database set, where query set contains 14,218 images of 3,985 classes and database set contains 12,612 images of 3,985 classes.
- *VehicleID* dataset [17] is a large-scale vehicle dataset that contains 221,763 images of 26,267 vehicles, where the training set contains 110,178 images of 13,134 vehicles and the testing set contains 111,585 images of 13,133 vehicles. Following the settings in [17], we use 3 test splits of different sizes constructed from the testing set. The small test set contains 7,332 images of 800 vehicles. The medium test set contains 12,995 images of 1,600 vehicles. The large test set contains 20,038 images of 2,400 vehicles.

4.2. Experiment Setup

We choose GoogLeNet [31] as our model for retrieval tasks. Since GoogLeNet has three output classifiers(two auxiliary classifiers from intermediate layers), HDC adopts them as three cascaded sub-networks corresponding to the three rows illustrated in Figure 2. i.e., G1 contains the layers from the input to "Inception(4a)" inclusively before the first classifier. We initialize the weights from the network pretrained on ImageNet ILSVRC-2012 [20]. We use the same hyper parameters in all experiments without specifically tuning. Specifically, K is set to 3, λ_1=λ_2=λ_3=1,



(a) LDA score = 0.54 (b) LDA score = 0.91 (c) LDA score = 1.99 (d) LDA score = 2.50

Figure 4. **Histograms for positive and negative distance distribution on the CARS196 test set:** (a) GoogLeNet/pool5^{1024} (b) Contrastive$^{\dagger 128}$ (c) Hard + Contrastive$^{\dagger 128}$ (d) HDC + Contrastive$^{\dagger 384}$. We can see that the overlap area between the 2 distributions decreases from left to right. You can check the LDA score of these methods on table 1 increases from left to right.

Table 1. **Comparisons of the Statistics of Histograms and Recall@K on CARS196 test set**. The mean and variance under the column named Positive Pairs correspond to m^+ and v^+. The mean and variance under the column named Negative Pairs correspond to m^- and v^-.

	Recall@K(%)						Positive Pairs		Negative Pairs		
	1	2	4	8	16	32	mean	variance	mean	variance	LDA score
GoogLeNet/pool5^{1024}	40.5	53.0	65.0	76.3	86.0	93.1	0.804	0.019	0.941	0.016	0.54
Contrastive$^{\dagger 128}$	56.0	67.6	77.0	84.8	90.5	94.5	1.110	0.052	1.350	0.011	0.91
Hard + Contrastive$^{\dagger 128}$	67.6	77.9	85.6	91.2	95.0	97.3	0.786	0.029	1.140	0.034	1.99
HDC + Contrastive-1$^{\dagger 128}$	41.9	55.5	67.6	78.3	86.9	93.2	0.741	0.045	1.200	0.074	1.77
HDC + Contrastive-2$^{\dagger 128}$	58.0	70.4	80.2	87.5	92.9	96.1	0.660	0.023	1.050	0.046	2.20
HDC + Contrastive-3$^{\dagger 128}$	71.4	81.8	88.5	93.4	**96.6**	98.2	0.792	0.014	1.070	0.020	2.27
HDC + Contrastive$^{\dagger 384}$	**73.7**	**83.2**	**89.5**	**93.8**	96.5	**98.4**	0.756	0.015	1.080	0.027	**2.50**

Table 2. **Comparisons of the Statistics of Histograms and Recall@K on CUB-200-2011 test set**.

	Recall@K(%)						Positive Pairs		Negative Pairs		
	1	2	4	8	16	32	mean	variance	mean	variance	LDA score
HDC + Contrastive-1$^{\dagger 128}_{\square}$	43.4	55.8	69.1	80.4	88.1	93.9	0.709	0.023	1.000	0.026	1.73
HDC + Contrastive-2$^{\dagger 128}_{\square}$	51.9	63.8	75.1	84.3	91.2	95.3	0.637	0.016	0.919	0.021	2.15
HDC + Contrastive-3$^{\dagger 128}_{\square}$	58.5	71.1	80.8	88.5	93.5	96.5	0.770	0.012	1.000	0.013	2.12
HDC + Contrastive$^{\dagger 384}_{\square}$	**60.7**	**72.4**	**81.9**	**89.2**	**93.7**	**96.8**	0.741	0.012	0.989	0.014	**2.37**

$\{h^1, h^2, h^3\} = \{100, 50, 20\}$, mini-batch size is 100, margin parameter $\mathcal{M}$ is set to 1, the initial learning rate starts from 0.01 and is divided by 10 every 3-5 epoches, and we train models for at most 15 epoches. The other settings follow the same protocol in [28]. The embedding dimensions of all the cascade models in our HDC are 128, so the embedding dimension of the ensembled model is 384. The code is publicly available at `https://github.com/PkuRainBow/Hard-Aware-Deeply-Cascaded-Embedding_release`

4.3. Comparison with Baseline

We name different methods with superscript and subscript to denote their specific settings, the number in superscript denotes the dimension used by the method, the subscript $\square$ denotes bounding boxes are used during training and testing. Different from the original Contrastive^{128} [3], we use the Contrastive$^{\dagger 128}$ to denote the contrastive loss computed with multi-batch [32] method.

To directly verify the effectiveness of HDC, we first design several baseline methods including: (1) **GoogLeNet/pool5**^{1024} uses the feature vector directly outputted from pool5 of the pre-trained GoogLeNet, (2) **Contrastive**$^{\dagger 128}$ uses contrastive loss without hard example mining, (3) **Hard + Contrastive**$^{\dagger 128}$ combines the contrastive loss and hard example mining. In addition to report our method named as **HDC + Contrastive**$^{\dagger 384}$, we also report the performance of sub models learned in our method, i.e., **HDC + Contrastive-1**$^{\dagger 128}$, **HDC + Contrastive-2**$^{\dagger 128}$ and **HDC + Contrastive-3**$^{\dagger 128}$. Hard+Contrastive uses the same network architecture as HDC+Contrastive-3, i.e., {G1,G2,G3,F3}. Only top 50 percent examples with larger loss are chosen as hard examples to update the model. The results of these methods on CARS196 are summarized in Table 1. Obviously, training on the target dataset brings significant performance improvement comparing with **GoogLeNet/pool5**^{1024}, hard example mining further brings more performance gain, while the hard aware method achieves the best performance. **HDC + Contrastive-3**$^{\dagger 128}$ is much better than the traditional **Hard + Contrastive**$^{\dagger 128}$ as the shallow modules of the model are also trained by hard samples mined by shallow models. Our method in the last row achieves the best result comparing with all baselines, which verifies the effectiveness of the hard-aware sample mining.

Table 3. **Recall@K(%) on CARS196 and CUB-200-2011**.

	CARS196						CUB-200-2011					
K	1	2	4	8	16	32	1	2	4	8	16	32
Contrastive^{128} [3]	21.7	32.3	46.1	58.9	72.2	83.4	26.4	37.7	49.8	62.3	76.4	85.3
Triplet^{128} [21, 38]	39.1	50.4	63.3	74.5	84.1	89.8	36.1	48.6	59.3	70.0	80.2	88.4
LiftedStruct^{128} [27]	49.0	60.3	72.1	81.5	89.2	92.8	47.2	58.9	70.2	80.2	89.3	93.2
Binomial Deviance^{512} [35]	-	-	-	-	-	-	52.8	64.4	74.7	83.9	90.4	94.3
Histogram Loss^{512} [35]	-	-	-	-	-	-	50.3	61.9	72.6	82.4	88.8	93.7
HDC + Contrastive$^{\dagger 384}$	**73.7**	**83.2**	**89.5**	**93.8**	**96.7**	**98.4**	**53.6**	**65.7**	**77.0**	**85.6**	**91.5**	**95.5**
PDDM + Triplet$^{128}_{\square}$ [6]	46.4	58.2	70.3	80.1	88.6	92.6	50.9	62.1	73.2	82.5	91.1	94.4
PDDM + Quadruplet$^{128}_{\square}$ [6]	57.4	68.6	80.1	89.4	92.3	94.9	58.3	69.2	79.0	88.4	93.1	95.7
HDC + Contrastive$^{\dagger 384}_{\square}$	**83.8**	**89.8**	**93.6**	**96.2**	**97.8**	**98.9**	**60.7**	**72.4**	**81.9**	**89.2**	**93.7**	**96.8**
Npairs$_{\boxplus}$ [26]	71.1	79.7	86.5	91.6	-	-	50.9	63.3	74.3	83.2	-	-
HDC + Contrastive$^{\dagger 384}_{\boxplus}$	**75.0**	**83.9**	**90.3**	**94.3**	**96.8**	**98.4**	**54.6**	**66.8**	**77.6**	**85.9**	**91.7**	**95.6**

Figure 4 shows the distance distributions of positive pairs and negative pairs following [35], where green area represents the distance distributions of positive pairs while red area for negative pairs. Our method has the smallest overlapping area, and better separates positive pairs and negative pairs in the embedding space. We also calculate the LDA score which measures the distance between two distributions to quantitatively compare the difference, i.e.,

$$score = \frac{|m^- - m^+|^2}{v^+ + v^-} \tag{12}$$

where m^+ and m^- are the mean distance of positive pairs and negative pairs, v^+ and v^- are the variance of the distances of positive pairs and negative pairs. The results on CARS196 are reported in the right part of Table 1. It can be observed that the retrieval performance measured by Recall@K positively correlates with LDA score, and our method achieves the highest LDA score 2.50. We also conduct experiment on CUB-200-2011 and report the results in Table 2, where the conclusion is the same on CARS196.

4.4. Comparison with state-of-the-art

We compare our method with state-of-the-art methods on the five datasets. On the CARS196, CUB-200-2011 and Stanford Online Products datasets: (1) **LiftedStruct**^{128} [27] uses a novel structured prediction objective on the lifted dense pairwise distance matrix. (2) **PDDM + Triplet**$^{128}_{\square}$ [6] combines Position-Dependent Deep Metric units (PDDM) and Triplet Loss. (3) **PDDM + Quadruplet**$^{128}_{\square}$ [6] combines the PDDM with Quadruplet Losses proposed in [13]. (4) **Histogram Loss**^{512} [35] is penalizing the overlap between distributions of distances of positive pairs and negative pairs.(5) **Binomial Deviance**^{512} [35] is used to evaluate the cost between similarities and labels, which is proved robust to outliers. (6)**Npairs**$_{\boxplus}$ [26] uses multi-class N-pair loss by generalizing triplet loss by allowing joint comparison among more than one negative examples. The subscript $\boxplus$ means using multiple crops when testing, while all the other methods use single crop except **Npairs**$_{\boxplus}$. All these methods use GoogLeNet as the base model, which is the same as our method.

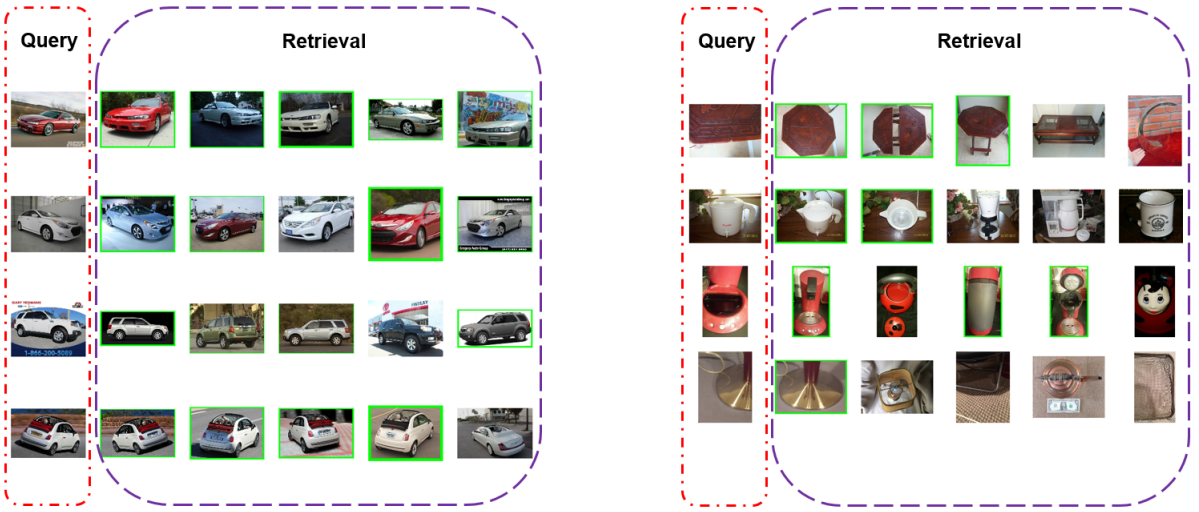
Table 4. **Recall@K(%) on In-shop Clothes Retrieval Dataset**.

K	1	10	20	30	40	50
FashionNet + Joints^{4096} [18]	41.0	64.0	68.0	71.0	73.0	73.5
FashionNet + Poselets^{4096} [18]	42.0	65.0	70.0	72.0	72.0	75.0
FashionNet^{4096} [18]	53.0	73.0	76.0	77.0	79.0	80.0
HDC + Contrastive$^{\dagger 384}$	**62.1**	**84.9**	**89.0**	**91.2**	**92.3**	**93.1**

On VehicleID and In-shop Clothes Retrieval datasets : (1) **CCL + Mixed Diff**^{1024} [17] uses the Coupled Cluster Loss and Mixed Difference Network Structure. (2) **FashionNet** [18] simultaneously learns the landmarks and attributes of the images using *VGG-16* [25]. To test the generality of our method, we use the same hyper-parameters without specifically tuning on these datasets.

Table 3 quantifies the advantages of our method on both CARS196 and CUB-200-2011. We conduct three groups of experiments to ensure fairness as different methods adopt different settings, i.e., with/without bounding boxes and with multiple crops when testing. **PDDM + Triplet**$^{128}_{\square}$ and **PDDM + Quadruplet**$^{128}_{\square}$ both use the images cropped with the annotated bounding boxes as training set and test set. With bounding boxes, cluttered backgrounds are removed and better performance is expected. HDC + Contrastive$^{\dagger 384}_{\square}$ shows significant performance gain both on CARS196 and CUB-200-2011. On CARS196, we improve the Recall@1 score from 57.4% to 83.8%. CUB-200-2011 is more challenging than CARS196 as the car is rigid while birds have more variations. We get 2.4% absolute improvement on CUB-200-2011. **Histogram Loss**^{512} and **Binomial Deviance**^{512} are trained without bounding boxes, for fair comparison, we also validate our method without using bounding boxes. HDC + Contrastive$^{\dagger 384}$ outperforms all methods without using bounding boxes on both datasets, and has even better results than methods using bounding boexes on CARS196. Besides, we test our method when using multiple crops for test. HDC + Contrastive$^{\dagger 384}_{\boxplus}$ also achieves state-of-the-art performance compared with **Npairs**$_{\boxplus}$.

Table 5 reports the results on Stanford Online Products.



(a) (b)

Figure 5. **Retrieval Results on CARS196 and Stanford Online Products:** (a) CARS196. (b) Stanford Online Products.

Table 5. **Recall@K(%) on Stanford Online Products.**

K	1	10	100	1000
Contrastive^{128} [3]	42.0	58.2	73.8	89.1
Triplet^{128} [21, 38]	42.1	63.5	82.5	94.8
LiftedStruct^{128} [27]	60.8	79.2	91.0	97.3
LiftedStruct^{512} [27]	62.1	79.8	91.3	97.4
Binomial Deviance^{512} [35]	65.5	82.3	92.3	97.6
Histogram Loss^{512} [35]	63.9	81.7	92.2	97.7
HDC + Contrastive$^{\dagger 384}$	**69.5**	**84.4**	**92.8**	97.7
Npairs$_{\boxplus}$ [26]	67.7	83.8	92.9	97.8
HDC + Contrastive$^{\dagger 384}_{\boxplus}$	**70.1**	**84.9**	**93.2**	97.8

Table 6. **MAP of Vehicle Retrieval Task.**

MAP	Small	Medium	Large
VGG + Triplet Loss^{1024}	0.444	0.391	0.373
VGG + CCL^{1024} ([17])	0.492	0.448	0.386
Mixed Diff + CCL^{1024} ([17])	0.546	0.481	0.455
GoogLeNet/pool5^{1024}	0.418	0.392	0.347
HDC + Contrastive$^{\dagger 384}$	**0.655**	**0.631**	**0.575**

Stanford Online Products suffers the problem of large number of categories and few images per category, which is very different from CARS196 and CUB-200-2011. Our method achieves 4% absolute improvements over previous state-of-the-art methods measured by Recall@1. When testing with multiple crops, HDC + Contrastive$^{\dagger 384}_{\boxplus}$ further improves the Recall@1 from 67.7% to 70.1%. Figure 5(a) shows some retrieval results on Stanford Online Products with features learned by HDC + Contrastive$^{\dagger 384}$.

Similar to the Stanford Online Products, DeepFashion In-shop Clothes and VehicleID also suffer the problem of limited images in each class and large number of classes. Table 4 and 6 compare the results on the two datasets, where our method outperforms state-of-the-art methods by a large margin.

Through extensive empirical comparisons on various datasets under different settings, we show that our method is general and can achieve better performance.

5. Conclusions

In this paper, we propose a novel Hard-Aware Deeply Cascaded Embedding to consider both hard levels of samples and the complexities of models. Different from training three separated models, our design ensembles a set of models with increasing complexities in a cascaded manner and shares most of the computation among models. Samples with different hard levels are mined accordingly using the models with adequate complexities. Controlled experimental results demonstrate the advantages by the hard-aware design, and extensive comparisons on five benchmarks further verify the effectiveness of the proposed method in learning deep metric embedding.

Currently, the method is verified by three cascaded models with increasing complexities, in the future, we would further improve the method by cascading more models and increasing complexities in a smoother way. And we would also try to combine our method with other loss functions in the future work.

Acknowledgements

This work is partially supported by the National Key Basic Research Project of China (973 Program) under Grant 2015CB352303 and the National Nature Science Foundation of China under Grant 61671027.

References

[1] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson. Real-time pedestrian detection with deep network cascades. 2015.

[2] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk. Pn-net: Conjoined triple deep network for learning local image descriptors. *arXiv:1601.05030*, 2016.

[3] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *TOG*, 2015.

[4] B. Bhattarai, G. Sharma, and F. Jurie. Cp-mtml: Coupled projection multi-task metric learning for large scale face retrieval. *arXiv:1604.02975*, 2016.

[5] M. Bucher, S. Herbin, and F. Jurie. Improving semantic embedding consistency by metric learning for zero-shot classiffication. In *ECCV*, 2016.

[6] C. C. L. Chen Huang and X. Tang. Local similarity-aware deep feature embedding. In *NIPS*, 2016.

[7] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *CVPR*, 2016.

[8] Y. Cui, F. Zhou, Y. Lin, and S. Belongie. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. *arXiv:1512.05227*, 2015.

[9] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, 2015.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CVPR*, 2016.

[11] J. D. L. F.-F. Jonathan Krause, Michael Stark. 3d object representations for fine-grained categorization. In *ICCV*, 2013.

[12] V. Kumar B G, G. Carneiro, and I. Reid. Learning local image descriptors with deep siamese and triplet convolutional networks by minimising global loss functions. In *CVPR*, 2016.

[13] M. T. Law, N. Thome, and M. Cord. Quadruplet-wise image similarity learning. In *ICCV*, 2013.

[14] L. Leal-Taixé, C. Canton-Ferrer, and K. Schindler. Learning by tracking: Siamese cnn for robust target association. *arXiv:1604.07866*, 2016.

[15] C.-Y. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *AISTATS*, 2015.

[16] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, 2015.

[17] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *CVPR*, 2016.

[18] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, 2016.

[19] O. Rippel, M. Paluri, P. Dollar, and L. Bourdev. Metric learning with adaptive density discrimination. *arXiv:1511.05939*, 2015.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[21] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015.

[22] H. Shi, Y. Yang, X. Zhu, S. Liao, Z. Lei, W. Zheng, and S. Z. Li. Embedding deep metric for person re-identification: A study against large variations. In *ECCV*. Springer, 2016.

[23] E. Simo-Serra and H. Ishikawa. Fashion Style in 128 Floats: Joint Ranking and Classification using Weak Data for Feature Extraction. In *CVPR*, 2016.

[24] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015.

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 2014.

[26] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016.

[27] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. *arXiv:1511.06452*, 2015.

[28] H. O. Song, X. Yu, J. Stefanie, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *CVPR*, 2016.

[29] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

[30] C. Sun, M. Paluri, R. Collobert, R. Nevatia, and L. Bourdev. Pronet: Learning to propose object-specific boxes for cascaded neural networks. *arXiv:1511.03776*, 2015.

[31] C. Szegedy, W. Liu, Y. Jia, and P. Sermanet. Going deeper with convolutions. 2015.

[32] O. Tadmor, Y. Wexler, T. Rosenwein, S. Shalev-Shwartz, and A. Shashua. Learning a metric embedding for face recognition using the multibatch method. *arXiv:1605.07270*, 2016.

[33] R. Tao, E. Gavves, and A. W. Smeulders. Siamese instance search for tracking. *arXiv:1605.05863*, 2016.

[34] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks. ICPR, 2016.

[35] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *NIPS*, 2016.

[36] A. Veit, M. Wilber, and S. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *NIPS*, 2016.

[37] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[38] J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, and Y. Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014.

[39] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[40] Y. Wang, J. Choi, V. I. Morariu, and L. S. Davis. Mining discriminative triplets of patches for fine-grained classification. *arXiv:1605.01130*, 2016.

[41] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*. Springer, 2016.

[42] F. Yang, W. Choi, and Y. Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *CVPR*, 2016.

[43] J. You, A. Wu, X. Li, and W.-S. Zheng. Top-push video-based person re-identification. *arXiv:1604.08683*, 2016.

[44] X. Zhang, F. Zhou, Y. Lin, and S. Zhang. Embedding label structures for fine-grained feature representation. In *CVPR*, 2016.

[45] Z. Zhang and V. Saligrama. Zero-shot learning via joint latent similarity embedding. In *CVPR*, 2016.