Divide and Conquer the Embedding Space for Metric Learning

Artsiom Sanakoyeu* Vadim Tschernezki* Uta Büchler Björn Ommer
Heidelberg Collaboratory for Image Processing, IWR, Heidelberg University

Abstract

Learning the embedding space, where semantically similar objects are located close together and dissimilar objects far apart, is a cornerstone of many computer vision applications. Existing approaches usually learn a single metric in the embedding space for all available data points, which may have a very complex non-uniform distribution with different notions of similarity between objects, e.g. appearance, shape, color or semantic meaning. Approaches for learning a single distance metric often struggle to encode all different types of relationships and do not generalize well. In this work, we propose a novel easy-to-implement divide and conquer approach for deep metric learning, which significantly improves the state-of-the-art performance of metric learning. Our approach utilizes the embedding space more efficiently by jointly splitting the embedding space and data into K smaller sub-problems. It divides both, the data and the embedding space into K subsets and learns K separate distance metrics in the non-overlapping subspaces of the embedding space, defined by groups of neurons in the embedding layer of the neural network. The proposed approach increases the convergence speed and improves generalization since the complexity of each sub-problem is reduced compared to the original one. We show that our approach outperforms the state-of-the-art by a large margin in retrieval, clustering and re-identification tasks on CUB200-2011, CARS196, Stanford Online Products, In-shop Clothes and PKU VehicleID datasets. Source code: https://bit.ly/dcesml.

1. Introduction

Deep metric learning methods learn to measure similarities or distances between arbitrary groups of data points, which is a task of paramount importance for a number of computer vision applications. Deep metric learning has been successfully applied to image search [3, 19, 32, 45], person/vehicle re-identification [5, 29, 51], fine-grained retrieval [30], near duplicate detection [54], clustering [16] and zero-shot learning [32, 43, 2, 37, 4].

*Both authors contributed equally to this work.

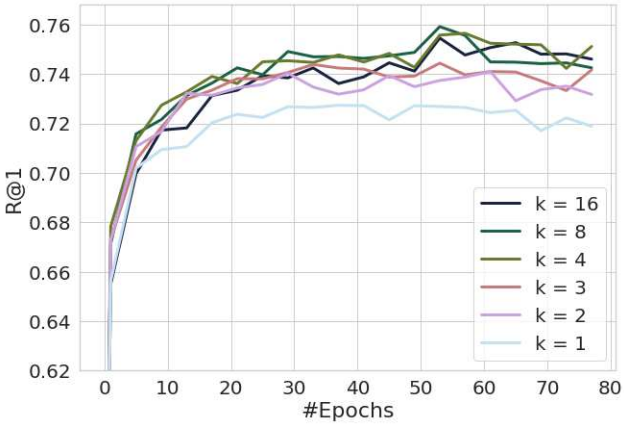

Figure 1: Evaluation of different numbers of learners. We train our model with $K = 1, 2, 3, 4, 8$ and 16 learners on the Stanford Online Products dataset [31] and report the change of the Recall@1 score during training. An increase in the number of learners leads to higher Recall@1. The best performance is achieved with $K = 8$.

The core idea of deep metric learning is to pull together samples with the same class label and to push apart samples coming from different classes in the learned embedding space. An embedding space with the desired properties is learned by optimizing loss functions based on pairs of images from the same or different class [13, 3], triplets of images [38, 45, 17] or tuples of larger number of images [20, 43, 41, 1], which express positive or negative relationships in the dataset.

Existing deep metric learning approaches usually learn a single distance metric for all samples from the given data distribution. The ultimate goal for the learned metric is to resolve all conflicting relationships and pull similar images closer while pushing dissimilar images further away. However, visual data is, commonly, not uniformly distributed, but has a complex structure, where different regions of the data distribution have different densities [20]. Data points in different regions of the distribution are often related based on different types of similarity such as shape, color, identity or semantic meaning. While, theoretically, a deep neural

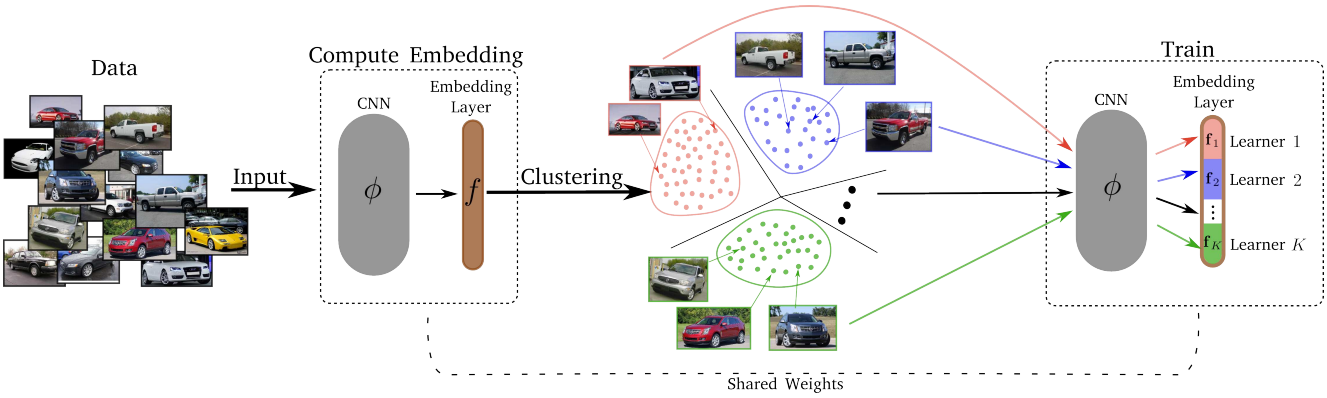

Figure 2: Pipeline of our approach. We first cluster the data in the embedding space in K groups and assign a separate subspace (learner) of the embedding layer to every cluster. During training, every learner only sees the samples assigned to the corresponding cluster.

network representation is powerful enough to approximate arbitrary continuous functions [18], in practice this often leads to poor local minima and overfitting. This is partially due to an inefficient usage of the embedding space [32, 34] and an attempt to directly fit a single distance metric to all available data [36, 26, 27].

The problems stated above motivate an approach which will use the embedding space in a more profound way by learning a separate distance metric for different regions of the data distribution. We propose a novel deep metric learning approach, inspired by the well-known divide and conquer algorithm. We explicitly split the embedding space and the data distribution into multiple parts given the network representation and learn a separate distance metric per subspace and its respective part of the distribution. Each distance metric is learned on its own subspace of the embedding space, but is based on a shared feature representation. The final embedding space is seamlessly composed by concatenating the solutions on each of the non-overlapping subspaces. See Fig. 2 for an illustration.

Our approach can be utilized as an efficient drop-in replacement for the final linear layer commonly used for learning embeddings in the existing deep metric learning approaches, regardless of the loss function used for training. We demonstrate a consistent performance boost when applying our approach to the widely-used triplet loss [38] and more complex state-of-the-art metric learning losses such as Proxy-NCA [30] and Margin loss [49]. By using the proposed approach, we achieve new state-of-the-art performance on five benchmark datasets for retrieval, clustering and re-identification: CUB200-2011 [44], CARS196 [25], Stanford Online Products [31], In-shop Clothes [55], and PKU VehicleID [29].

2. Related work

Metric learning has been of major interest for the vision community since its early beginnings, due to its broad applications including object retrieval [32, 43, 48], zero-shot and single-shot learning [43, 32], keypoint descriptor learning [40], face verification [5] and clustering [16]. With the advent of CNNs, several approaches have been proposed for supervised distance metric learning. Some methods use pairs [52] or triplets [46, 38] of images. Others use quadruplets [41, 43] or impose constraints on tuples of larger sizes like Lifted Structure [32], n-pairs [41] or poset loss [1].

Using a tuple of images as training samples yields a huge amount of training data. However, only a small portion of the samples among all N^p possible tuples of size p is meaningful and provides a learning signal. A number of recent works tackle the problem of hard and semi-hard negative mining which provides the largest training signal by designing sampling strategies [49, 14, 10, 53, 21]. Existing sampling techniques, however, require either running an expensive, quadratic on the number of data points preprocessing step for the entire dataset and for every epoch [14, 10], or lack global information while having a local view on the data based on a single randomly-drawn mini-batch of images [38, 49, 41]. On the contrary, our approach efficiently alleviates the problem of the abundance of easy samples, since it jointly splits the embedding space and clusters the data using the distance metric learned so far. Hence, samples inside one cluster will have smaller distances to one another than to samples from another cluster, which serves as a proxy to the mining of more meaningful relationships [38, 14]. For further details of our approach see Sec. 3.

Recently, a lot of research efforts have been devoted to designing new loss functions [42, 43, 41, 47, 30, 35]. For example, Facility Location [42] optimizes a cluster quality metric, Histogram loss [43] minimizes the overlap between

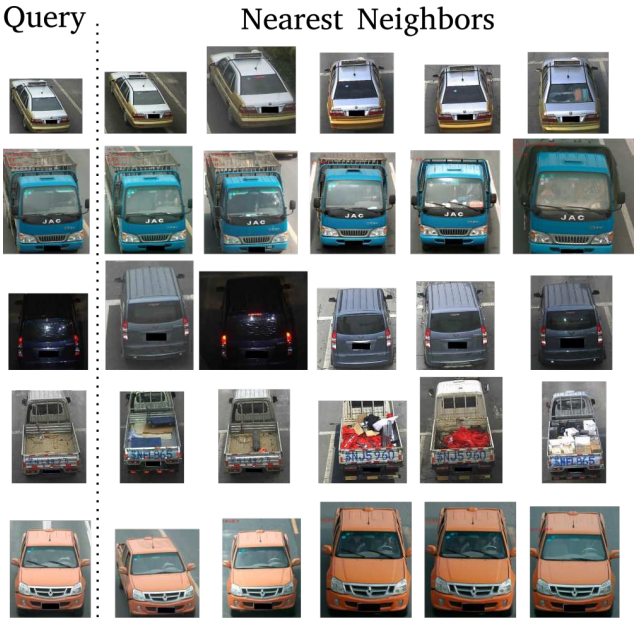

Figure 3: Qualitative image retrieval results on PKU VehicleID [29]. We show 5 nearest neighbors per randomly chosen query image given our trained features. The queries and retrieved images are taken from the test set of the dataset.

Figure 4: Qualitative image retrieval results on Stanford Online Products [31]. We randomly choose 5 query images from the test set of the Stanford Online Products dataset and show 5 nearest neighbors per query image given the features of our trained model. The retrieved images originate also from the test set.

the distribution of positive and negative distances. Kihyuk Sohn proposed in [41] the N-pairs loss which enforces a softmax cross-entropy loss among pairwise similarity values in the batch. The Proxy-NCA loss, presented in [30] computes proxies for the original points in the dataset and optimizes the distances to these proxies using NCA [35]. Our work is orthogonal to these approaches and provides a framework for learning a distance metric independent on the choice of a particular loss function.

Another line of work in deep metric learning which is more related to our approach is ensemble learning [51, 34, 8, 12]. Previous works [51, 34] employ a sequence of "learners" with increasing complexity and mine samples of different complexity levels for the next learners using the outputs of the previous learners. HDC [51] uses a cascade of multiple models of a specific architecture, A-BIER [34] applies a gradient boosting learning algorithm to train several learners inside a single network in combination with an adversarial loss [9, 11]. The key difference of the aforementioned approaches to our approach is that we split the embedding space and cluster the data jointly, so each "learner" will be assigned to the specific subspace and corresponding portion of the data. The "learners" are independently trained on non-overlapping chunks of the data which reduces the training complexity of each individual learner, facilitates the learning of decorrelated representations and can be easily parallelized. Moreover, our approach does not introduce extra parameters during training and works in a single model. It does not require any additional loss functions and can be applied to any existing network architecture.

3. Approach

The main intuition behind our approach is the following: Solving bigger problems is usually harder than solving a set of smaller ones. We propose an effective and easily adaptive divide and conquer algorithm for deep metric learning. We divide the data into multiple groups (sub-problems) to reduce the complexity and solve the metric learning problem on each sub-problem separately. Since we want the data partitioning to be coupled with the current state of the embedding space, we cluster the data in the embedding space learned so far. Then we split the embedding layer of the network into slices. Each slice of the embedding layer represents an individual learner. Each learner is assigned to one cluster and operates in a certain subspace of the original embedding space. At the conquering stage we merge the solutions of the sub-problems, obtained by the individual learners, to get the final solution. We describe each step of our approach in details in Sec. 3.2 and 3.3.

3.1. Preliminaries

We denote the training set as $X = \{x_1, \dots, x_n\} \subset \mathcal{X}$, where $\mathcal{X}$ is the original RGB space, and the corresponding class labels as $Y = \{y_1, \dots, y_n\}$. A Convolutional Neural Network (CNN) learns a non-linear transformation

Figure 5: Qualitative image retrieval results on In-shop clothes [55]. We randomly choose 5 query images from the query set of the In-shop clothes dataset and show 5 nearest neighbors per query image given our trained features. The retrieved images are taken from the gallery set.

Figure 6: Qualitative image retrieval results on CARS196 [25]. We randomly choose 5 query images from the test set of the CARS196 dataset and show 5 nearest neighbors per query image given our trained features. The retrieved images are taken from the test set.

of the image into an m-dimensional deep feature space $\phi(\cdot;\theta_\phi) : \mathcal{X} \to \mathbb{R}^m$, where θ_ϕ is the set of the CNN parameters. For brevity, we will use the notations $\phi(x_i;\theta_\phi)$ and ϕ_i interchangeably.

To learn a mapping into the embedding space, a linear layer $f(\cdot;\theta_f) : \mathbb{R}^m \to \mathbb{R}^d$ with d neurons is typically appended to the CNN, where θ_f denotes the parameters of this layer. $f(\cdot;\theta_f)$ is often normalized to have a unit length for training stability [38]. The goal of metric learning is to jointly learn ϕ and f in such a way that $(f \circ \phi)(x;\theta_\phi,\theta_f)$ maps similar images close to one another and dissimilar ones far apart in the embedding space. Formally, we define a distance between two data points in the embedding space as

$$d_f(x_i, x_j) = ||f(\phi_i) - f(\phi_j)||_2. \quad (1)$$

To learn the distance metric, one can use any loss function with options such as [38, 43, 41, 30, 49, 32]. Our framework is independent of the choice of the loss function. In this paper we experiment with three different losses: Triplet loss [38], Proxy-NCA loss [30] and Margin loss [49]. For simplicity, we will demonstrate our approach in this section on the example of triplet loss, which is defined as

$$l_{\text{triplet}}(a,p,n;\theta_\phi,\theta_f) = \left[d_f(a,p)^2 - d_f(a,n)^2 + \alpha\right]_+, \quad (2)$$

where $[\cdot]_+$ denotes the positive part and α is the margin. The triplet loss strives to keep the positive data point p closer to the anchor point a than any other negative point n. For brevity we omit the definitions of other losses, but we refer the interested reader to the original works [30, 49].

3.2. Division of the embedding space

We begin with the division stage of our approach. To reduce the complexity of the problem and to utilize the entire embedding space more efficiently we split the embedding dimensions and the data into multiple groups. Each learner will learn a separate distance metric using only a subspace of the original embedding space and a part of the data.

Splitting the data: Let K be the number of sub-problems. We group all data points $\{x_1,\dots,x_n\}$ according to their pairwise distances in the embedding space into K clusters $\{C_k | 1 \le k \le K\}$ with K-means.

Splitting the embedding: Next, we define K individual learners within the embedding space by splitting the embedding layer of the network into K consecutive slices. Formally, we decompose the embedding function $f(\cdot;\theta_f)$ into K functions $\{\mathbf{f}_1,\dots,\mathbf{f}_K\}$, where each $\mathbf{f}_k$ maps the input into the d/K-dimensional subspace of the original d-dimensional embedding space: $\mathbf{f}_k(\cdot;\theta_{\mathbf{f}_k}) : \mathbb{R}^m \to \mathbb{R}^{d/K}$. $\mathbf{f}_1$ will map into the first d/K dimensions of the original embedding space, $\mathbf{f}_2$ into the second d/K dimensions and so on. Please see Fig. 2 for an illustration. Note that the number of the model parameters stays constant after we perform the splitting of the embedding layer, since the learners share the underlying representation.

3.3. Conquering stage

In this section, we first describe the step of solving individual problems. Then, we outline the merging step, where the solutions of sub-problems are combined to form the final solution.

Training: After the division stage, every cluster C_k is assigned to a learner $\mathbf{f}_k$, $1 \le k \le K$. Since all learners reside within a single linear embedding layer and share the underlying feature representation, we train them jointly in

an alternating manner. In each training iteration only one of the learners is updated. We uniformly sample a cluster $C_k, 1 \leq k \leq K$ and draw a random mini-batch B from it. Then, a learner $\mathbf{f}_k$ minimizes its own loss defined as follows:

$$\mathcal{L}_k^{\theta_\phi, \theta_{\mathbf{f}_k}} = \sum_{(a,p,n)\sim B} \left[d_{\mathbf{f}_k}(a,p)^2 - d_{\mathbf{f}_k}(a,n)^2 + \alpha \right], \quad (3)$$

where triplet $(a, p, n) \in B \subset C_k$ denotes the triplets sampled from the current mini-batch, and $d_{\mathbf{f}_k}$ is the distance function defined in the subspace of the k-th learner. As described in Eq. 3 each backward pass will update only the parameters of the shared representation θ_ϕ and the parameters of the current learner $\theta_{\mathbf{f}_k}$. Motivated by the fact that the learned embedding space is improving during the time, we update the data partitioning by re-clustering every T epochs using the full embedding space. The full embedding space is composed by simply concatenating the embeddings produced by the individual learners.

Merging the solutions: Finally, following the divide and conquer paradigm, after individual learners converge, we merge their solutions to get the full embedding space. Merging is done by joining the embedding layer slices, corresponding to the K learners, back together. After this, we fine-tune the embedding layer on the entire dataset to achieve the consistency between the embeddings of the individual learners. An overview of the full training process of our approach can be found in Algorithm 1.

Algorithm 1 Training a model with our approach

Input: X,f,θ_ϕ, θ_f,K,T ▷ data, linear layer, CNN weights, weights of f, # clusters, re-cluster freq.

▷ cluster affiliation $\forall x_i \in X$

$\{\mathbf{f}_1, \dots \mathbf{f}_K\} \leftarrow$ SplitEmbedding(f) ▷ set of Learners

$epoch \leftarrow 0$

while Not Converged **do**

  if $epoch$ mod $T == 0$ **then**

    $f \leftarrow$ ConcatEmbedding$(\{\mathbf{f}_1, \dots \mathbf{f}_K\})$

    $emb \leftarrow$ ComputeEmbedding$(X, \theta_\phi, \theta_f)$

    $\{C_1, \dots, C_K\} \leftarrow$ ClusterData(emb, K)

    $\{\mathbf{f}_1, \dots \mathbf{f}_K\} \leftarrow$ SplitEmbedding(f)

  end if

  repeat

    $C_k \sim \{C_1, \dots, C_K\}$ ▷ sample cluster

    $b \leftarrow$ GetBatch(C_k) ▷ draw mini-batch

    $\mathcal{L}_k \leftarrow$ FPass$(b, \theta_\phi, \theta_{\mathbf{f}_k})$ ▷ Compute Loss of Learner $\mathbf{f}_k$ (Eq. 3)

    $\theta_\phi, \theta_{\mathbf{f}_k} \leftarrow$ BPass$(L, \theta_\phi, \theta_{\mathbf{f}_k})$ ▷ Update weights

  until Epoch completed

  $epoch \leftarrow epoch + 1$

end while

$f \leftarrow$ ConcatEmbedding$(\{\mathbf{f}_1, \dots \mathbf{f}_K\})$

$\theta_\phi, \theta_f \leftarrow$ Finetune$(X, \theta_\phi, \theta_f, f)$

Output: θ_ϕ, θ_f

4. Experiments

In this section, we first introduce the datasets we use for evaluating our approach and provide afterwards additional details regarding the training and testing of our framework. We then show qualitative and quantitative results which we compare with the state-of-the-art by measuring the image retrieval quality and clustering performance. The ablation study in subsection 4.4 provides then some inside into our metric learning approach.

4.1. Datasets

We evaluate the proposed approach by comparing it with the state-of-the-art on two small benchmark datasets (CARS196 [25], CUB200-2011 [44]), and on three large-scale datasets (Stanford Online Products [31], In-shop Clothes [55], and PKU VehicleID [29]). For assessing the clustering performance we utilize the normalized mutual information score [39] NMI$(\Omega, \mathbb{C}) = \frac{2 \cdot I(\Omega, \mathbb{C})}{H(\Omega) + H(\mathbb{C})}$, where Ω denotes the ground truth clustering and $\mathbb{C}$ the set of clusters obtained by K-means. Here I represents the mutual information and H the entropy. For the retrieval task we report the Recall@k metric [22].

Stanford Online Products [31] is one of the largest publicly available image collections for evaluating metric learning methods. It consists of $120,053$ images divided into $22,634$ classes of online products, where $11,318$ classes ($59,551$ images) are used for training and $11,316$ classes ($60,502$ images) for testing. We follow the same evaluation protocol as in [32]. We calculate Recall@k score for $k = 1, 10, 100, 1000$ for evaluating the image retrieval quality and the NMI metric for appraising the clustering performance, respectively.

CARS196 [25] contains 196 different types of cars distributed over 16,185 images. The first 98 classes ($8,054$ images) are used for training and the other 98 classes ($8,131$ images) for testing. We train and test on the entire images without using bounding box annotations.

CUB200-2011 [44] is an extended version of the CUB200 dataset which consolidates images of 200 different bird species with 11,788 images in total. The first 100 classes ($5,864$ images) are used for training and the second 100 classes ($5,924$ images) for testing. We train and test on the entire images without using bounding box annotations.

In-shop Clothes Retrieval [55] contains $11,735$ classes of clothing items with $54,642$ images. We follow the evaluation protocol of [55] and use a subset of $7,986$ classes with $52,712$ images. $3,997$ classes are used for training and $3,985$ classes for testing. The test set is partitioned into query set and gallery set, containing $14,218$ and $12,612$

R@k	1	10	100	1000	NMI
Histogram [43]	63.9	81.7	92.2	97.7	-
Bin. Deviance [43]	65.5	82.3	92.3	97.6	-
Triplet Semihard [42]	66.7	82.4	91.9	-	89.5
LiftedStruct [32]	63.0	80.5	91.7	97.5	87.4
FacilityLoc [42]	67.0	83.7	93.2	-	-
N-pairs [41]	67.7	83.7	93.0	97.8	88.1
Angular [47]	70.9	85.0	93.5	98.0	88.6
DAML (N-p) [6]	68.4	83.5	92.3	-	89.4
HDC [51]	69.5	84.4	92.8	97.7	-
DVML [28]	70.2	85.2	93.8	-	90.8
BIER [33]	72.7	86.5	94.0	98.0	-
ProxyNCA [30]	73.7	-	-	-	-
A-BIER [34]	74.2	86.9	94	97.8	-
HTL [10]	74.8	88.3	94.8	98.4	-
Margin baseline [49]	72.7	86.2	93.8	98.0	**90.7**
Ours (Margin)	**75.9**	**88.4**	**94.9**	**98.1**	90.2

Table 1: Recall@k for $k = 1, 10, 100, 100$ and NMI on Stanford Online Products [31]

R@k	1	2	4	8	NMI
Triplet Semihard [42]	51.5	63.8	73.5	82.4	53.4
LiftedStruct [32]	48.3	61.1	71.8	81.1	55.1
FacilityLoc [42]	58.1	70.6	80.3	87.8	59.0
SmartMining [14]	64.7	76.2	84.2	90.2	-
N-pairs [41]	71.1	79.7	86.5	91.6	64.0
Angular [47]	71.4	81.4	87.5	92.1	63.2
ProxyNCA [30]	73.2	82.4	86.4	88.7	64.9
HDC [51]	73.7	83.2	89.5	93.8	-
DAML (N-pairs) [6]	75.1	83.8	89.7	93.5	66.0
HTG [53]	76.5	84.7	90.4	94	-
BIER [33]	78.0	85.8	91.1	95.1	-
HTL [10]	81.4	88.0	92.7	95.7	-
DVML [28]	82.0	88.4	93.3	96.3	67.6
A-BIER [34]	82.0	89.0	93.2	96.1	-
Margin baseline [49]	79.6	86.5	91.9	95.1	69.1
Ours (Margin)	**84.6**	**90.7**	**94.1**	**96.5**	**70.3**
DREML [50]	86.0	91.7	95.0	97.2	76.4

Table 2: Recall@k for $k = 1, 2, 4, 8$ and NMI on CARS196 [25]

images, respectively.

PKU VehicleID [29] is a large-scale vehicle dataset that contains $221,736$ images of $26,267$ vehicles captured by surveillance cameras. The training set contains $110,178$ images of $13,134$ vehicles and the testing set comprises $111,585$ images of $13,133$ vehicles. We evaluate on 3 test sets of different sizes as defined in [29]. The small test set contains $7,332$ images of 800 vehicles, the medium test set contains $12,995$ images of 1600 vehicles, and the large test set contains $20,038$ images of 2400 vehicles. This dataset has smaller intra-class variation, but it is more challenging than CARS196, because different identities of vehicles are considered as different classes, even if they share the same car model.

4.2. Implementation Details

We implement our approach by closely following the implementation of Wu et al. [49] based on ResNet-50 [15]. We use an embedding of size $d = 128$ and an input image size of 224×224 [15] for all our experiments. The embedding layer is randomly initialized. All models are trained using Adam [24] optimizer with the batch size of 80 for Stanford Online Products and In-shop Clothes datasets, and 128 for the other datasets. We resize the images to 256 and apply random crops and horizontal flips for data augmentation. For training our models we set the number of learners $K = 4$ for CUB200-2011 and CARS196 due to their small size, and $K = 8$ for all the other datasets. We have noticed that our approach is not sensitive to the values of T in the range between 1 and 10. We set $T = 2$ for all experiment, since the value alteration did not lead to significant changes in the experimental results.

Similar to [49, 38] we initialize Margin loss with $\beta = 1.2$ and Triplet loss with $\alpha = 0.2$. Mini-batches are sampled following the procedure defined in [38, 49] with $m = 4$ images per class per mini-batch for Margin loss [49] and Triplet loss [38], and uniformly for Proxy-NCA [30]. During the clustering (Sec. 3.2) and test phase, an image embedding is composed by concatenating the embeddings of individual learners.

4.3. Results

We now compare our approach to the state-of-the-art. From Tables 1, 2, 3, 4 and 5 we can see that our method with Margin loss [49] outperforms existing state-of-the-art methods on all 5 datasets, proving its wide applicability. Note that we use a smaller embedding size of $d = 128$ instead of 512 employed by runner-up approaches HTL [10], A-BIER [34], BIER [33], DVML [28], DAML [6], and Angular loss [47]; HDC [51] uses a 384-dimensional embedding layer. Moreover, we compare our results to the deep ensembles approach DREML [50], which trains an ensemble of 48 ResNet-18 [15] networks with a total number of 537M trainable parameters. Our model has only 25.5M trainable parameters and still outperforms DREML [50] on CUB200-2011 and In-shop Clothes datasets by a large margin.

We demonstrate the results of our approach with three different losses on CUB200-2011: Triplet [38], Proxy-NCA [30] and Margin loss [49]. Our approach improves the Recall@1 performance by at least 2.1% in each of the experiments (see Tab. 3). This confirms that our approach is universal and can be applied to a variety of metric learning

R@k	1	2	4	8	NMI
LiftedStruct [32]	46.6	58.1	69.8	80.2	56.2
FacilityLoc [42]	48.2	61.4	71.8	81.9	59.2
SmartMining [14]	49.8	62.3	74.1	83.3	-
Bin. Deviance [43]	52.8	64.4	74.7	83.9	-
N-pairs [41]	51.0	63.3	74.3	83.2	60.4
DVML [28]	52.7	65.1	75.5	84.3	61.4
DAML (N-pairs) [6]	52.7	65.4	75.5	84.3	61.3
Histogram [43]	50.3	61.9	72.6	82.4	-
Angular [47]	54.7	66.3	76.0	83.9	61.1
HDC [51]	53.6	65.7	77.0	85.6	-
BIER [33]	55.3	67.2	76.9	85.1	-
HTL [10]	57.1	68.8	78.7	86.5	-
A-BIER [34]	57.5	68.7	78.3	86.2	-
HTG [53]	59.5	71.8	81.3	88.2	-
Triplet $_{\text{semihard}}$ [42]	42.6	55.0	66.4	77.2	55.4
Triplet $_{\text{semihard}}$ baseline*	53.1	65.9	76.8	85.3	60.3
Ours (Triplet $_{\text{semihard}}$)	55.4	66.9	77.5	86.5	61.9
ProxyNCA [30]	49.2	61.9	67.9	72.4	64.9
ProxyNCA baseline*	58.7	70.0	79.1	87.0	62.5
Ours (ProxyNCA)	61.8	73.1	81.8	88.2	65.7
Margin baseline [49]	63.6	74.4	83.1	90.0	69.0
Ours (Margin)	**65.9**	**76.6**	**84.4**	**90.6**	**69.6**
DREML [50]	63.9	75.0	83.1	89.7	67.8

Table 3: Recall@k for $k = 1, 2, 4, 6, 8$ and NMI on CUB200-2011 [44]. * denotes our own implementation based on ResNet-50 with $d = 128$.

R@k	1	10	20	30	50	NMI
FashionNet [55]	53.0	73.0	76.0	77.0	80.0	-
HDC [51]	62.1	84.9	89.0	91.2	93.1	-
BIER [33]	76.9	92.8	95.2	96.2	97.1	-
HTG [53]	80.3	93.9	95.8	96.6	97.1	-
HTL [10]	80.9	94.3	95.8	97.2	97.8	-
A-BIER [34]	83.1	95.1	96.9	97.5	98.0	-
Margin baseline* [49]	82.6	94.8	96.2	97.0	97.7	87.8
Ours (margin)	**85.7**	**95.5**	**96.9**	**97.5**	**98.0**	**88.6**
DREML [50]	78.4	93.7	95.8	96.7	-	-

Table 4: Recall@k for $k = 1, 10, 20, 30, 50$ and NMI on In-shop Clothes [55]. * denotes our own implementation based on ResNet-50 with $d = 128$.

Split Size	Small		Medium		Large	
R@k	1	5	1	5	1	5
Mixed Diff+CCL [29]	49.0	73.5	42.8	66.8	38.2	61.6
GS-TRS loss [7]	75.0	83.0	74.1	82.6	73.2	81.9
BIER [33]	82.6	90.6	79.3	88.3	76.0	86.4
A-BIER [34]	86.3	92.7	83.3	88.7	81.9	88.7
Margin baseline* [49]	85.1	91.4	82.9	88.9	79.2	88.4
Ours (margin)	**87.7**	**92.9**	**85.7**	**90.4**	**82.9**	**90.2**
DREML [50]	88.5	94.8	87.2	94.2	83.1	92.4

Table 5: Recall@k for $k = 1, 5$ on the small, medium and large PKU VehicleID [29] dataset. * denotes our own implementation based on ResNet-50 with $d = 128$.

loss functions. We noticed that it shows especially large improvements on large-scale datasets such as on PKU VehicleID, where we improve by 3.6% over the baseline with Margin loss [49] and surpass the state-of-the-art by 1% in terms of Recall@1 score on the large test set. We attribute this success on such a challenging dataset to the more efficient exploitation of large amounts of data due to dividing it between different learners which operate on non-overlapping subspaces of the entire embedding space.

In addition to the quantitative results, we show in Figure 3,4,5 and 6 qualitative image retrieval results on CUB200-2011, Stanford Online Products, In-shop clothes, and Cars196. Note that our model is invariant to viewpoint and daylight changes.

4.4. Ablation Study

We perform several ablation experiments to demonstrate the effectiveness of the proposed method and evaluate the different components of our contribution. We use the Stanford Online Products dataset and train all models with Margin loss [49] for 80 epochs.

First, we analyze the choice of the number of learners K. As can be seen in Fig. 1, Recall@1 significantly increased already with $K = 2$. The best result is achieved with $K = 8$, where each learner operates in a 16-dimensional embedding subspace. Increasing the number of learners from $K > 1$ on, results in faster convergence and better local optima.

Next, we study the effect of clustering the data. In Tab. 6 we see that substituting K-means clustering in the embedding space with random data partitioning significantly degrades the performance. On the other hand, what happens if we use K-means clustering in the embedding space, but do not split the embedding f into K subspaces $\mathbf{f}_1, \ldots, \mathbf{f}_K$ during training? I.e., we perform regular training but with sampling from clusters. From Tab. 6 we see that it leads to a performance drop compared to the proposed approach, however it is already better than the baseline. This is due to the fact that drawing mini-batches from the clusters yields harder training samples compared to drawing mini-batches from the entire dataset. The expectation of the distance between a negative pair within the cluster is lower than the expectation of the distance between a negative pair randomly sampled from the entire dataset, as visually depicted on Fig. 7 and Fig. 8. This shows that: a) sampling from clusters provides a stronger learning signal than regular sampling from the entire dataset, b) to be able to efficiently learn from harder

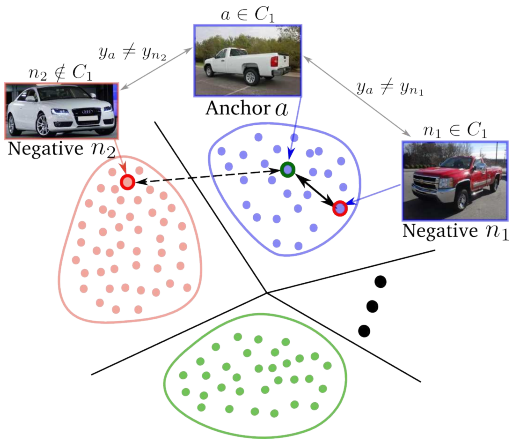

Figure 7: Natural hard negative mining. During Training, we only sample tuples (e.g., pairs or triplets) from the same cluster. The expected value of the distance between a negative sample and an anchor within a cluster is lower than the expected value when the data points belong to different clusters. Our approach naturally finds hard negatives without explicitly performing a hard negative mining procedure.

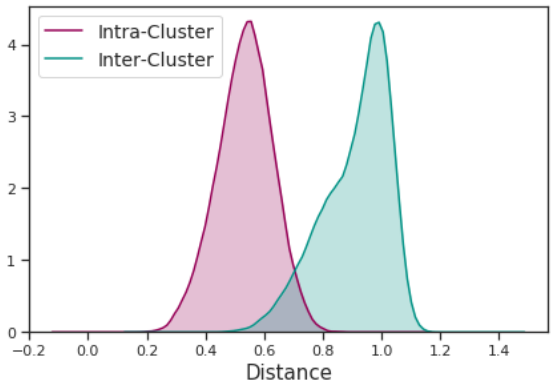

Figure 8: Intra-cluster and inter-cluster distributions of distances for negative pairs. Red histogram shows the distribution of the pairwise distances of samples having different class labels but from the same cluster (intra-cluster); green histogram shows the distribution of the pairwise distances of samples having different class labels and drawn from different clusters (inter-cluster). Negative pairs within one cluster have lower distances and are harder on average.

samples we need an individual learner for each cluster, which significantly reduces the complexity of the metric learning task. We also substitute K-means clustering with the fixed data partitioning, based on the ground truth labels, which are manually grouped according to semantic similarity (see "GT labels grouping" in Tab. 6). We recognize that the use of a flexible clustering scheme, which depends on the data distribution in the embedding space, leads to better performance than using class labels.

Runtime complexity: Splitting the embedding space into subspaces and training K independent learners reduces the time required for a single forward and backward pass, since we only use a d/K-dimensional embedding instead of the full embedding. We perform K-means clustering every T epochs. We use the K-means implementation from the Faiss library [23] which has an average complexity of $O(Kni)$, where n is the number of samples, and i is the number of iterations. This adds a neglectable overhead compared to the time required for a full forward and backward pass of all images in the dataset. For example, in case of $T = 2$ the clustering will add $\approx 25\%$ overhead and in case of $T = 8$ only $\approx 6.25\%$.

R@k	1	10	100	1000
Baseline [49]	72.7	86.2	93.8	98.0
K-means in the emb. space, no embedding splitting	75.0	87.6	94.2	97.8
Random data partition	73.2	85.8	93.4	97.6
GT labels grouping	74.5	87.1	93.8	97.6
K-means in the emb. space	**75.9**	**88.4**	**94.9**	**98.1**

Table 6: Evaluation of different data grouping methods on Stanford Online Products [31] with $K = 8$ and Margin loss [49].

5. Conclusion

We introduced a simple and efficient divide and conquer approach for deep metric learning, which divides the data in K clusters and assigns them to individual learners, constructed by splitting the network embedding layer into K non-overlapping slices. We described the procedure for joint training of multiple learners within one neural network and for combining partial solutions into the final deep embedding. The proposed approach is easy to implement and can be used as an efficient drop-in replacement for a linear embedding layer commonly used in the existing deep metric learning approaches independent on the choice of the loss function. The experimental results on CUB200-2011 [44], CARS196 [25] and Stanford Online Products [31], In-shop Clothes [55] and PKU VehicleID [29] show that our approach significantly outperforms the state-of-the-art on all the datasets.

This work has been supported by a DFG grant OM81/1-1 and a hardware donation by NVIDIA corporation.

References

[1] M. Á. Bautista, A. Sanakoyeu, and B. Ommer. Deep unsupervised similarity learning using partially ordered sets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1923–1932, 2017. 1, 2

[2] M. A. Bautista, A. Sanakoyeu, E. Tikhoncheva, and B. Ommer. Cliquecnn: Deep unsupervised exemplar learning. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 3846–3854, 2016. 1

[3] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 34(4):98, 2015. 1

[4] U. Büchler, B. Brattoli, and B. Ommer. Improving spatiotemporal self-supervision by deep reinforcement learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 770–786, 2018. 1

[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005. 1, 2

[6] Y. Duan, W. Zheng, X. Lin, J. Lu, and J. Zhou. Deep adversarial metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2018. 6, 7

[7] Y. Em, F. Gag, Y. Lou, S. Wang, T. Huang, and L.-Y. Duan. Incorporating intra-class variance to fine-grained visual recognition. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 1452–1457. IEEE, 2017. 7

[8] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. 3

[9] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 3

[10] W. Ge, W. Huang, D. Dong, and M. R. Scott. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–285, 2018. 2, 6, 7

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 3

[12] J. Guo and S. Gould. Deep cnn ensemble with data augmentation for object detection. *arXiv preprint arXiv:1506.07224*, 2015. 3

[13] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *null*, pages 1735–1742. IEEE, 2006. 1

[14] B. Harwood, V. Kumar, G. Carneiro, I. Reid, and T. Drummond. Smart mining for deep metric learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 6, 7

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[16] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 31–35. IEEE, 2016. 1, 2

[17] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015. 1

[18] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991. 2

[19] C. Huang, C. Change Loy, and X. Tang. Unsupervised learning of discriminative attributes and visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5175–5184, 2016. 1

[20] C. Huang, C. C. Loy, and X. Tang. Local similarity-aware deep feature embedding. In *Advances in Neural Information Processing Systems*, pages 1262–1270, 2016. 1

[21] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Mining on manifolds: Metric learning without labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 2

[22] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2011. 5

[23] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017. 8

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[25] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 2, 4, 5, 6, 8

[26] Z. Li and J. Tang. Weakly supervised deep metric learning for community-contributed image retrieval. *IEEE Transactions on Multimedia*, 17(11):1989–1999, 2015. 2

[27] Z. Li, J. Tang, and T. Mei. Deep collaborative embedding for social image understanding. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2

[28] X. Lin, Y. Duan, Q. Dong, J. Lu, and J. Zhou. Deep variational metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 689–704, 2018. 6, 7

[29] H. Liu, Y. Tian, Y. Wang, L. Pang, and T. Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2167–2175, 2016. 1, 2, 3, 5, 6, 7, 8

[30] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh. No fuss distance metric learning using proxies. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 2, 3, 4, 6, 7

[31] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4004–4012, 2016. 1, 2, 3, 5, 6, 8

[32] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 1, 2, 4, 5, 6, 7

[33] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Bier-boosting independent embeddings robustly. In *International Conference on Computer Vision (ICCV)*, 2017. 6, 7

[34] M. Opitz, G. Waltner, H. Possegger, and H. Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *arXiv preprint arXiv:1801.04815*, 2018. 2, 3, 6, 7

[35] S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. *Advances in Neural Information Processing Systems (NIPS)*, 17:513–520, 2005. 2, 3

[36] J. C. Rubio, A. Eigenstetter, and B. Ommer. Generative regularization with latent topics for discriminative object recognition. *Pattern Recognition*, 48(12):3871–3880, 2015. 2

[37] A. Sanakoyeu, M. A. Bautista, and B. Ommer. Deep unsupervised learning of visual similarities. *Pattern Recognition*, 78:331–343, 2018. 1

[38] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 1, 2, 4, 6

[39] H. Schütze, C. D. Manning, and P. Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press, 2008. 5

[40] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 118–126, 2015. 2

[41] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016. 1, 2, 3, 4, 6, 7

[42] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy. Deep metric learning via facility location. In *Computer Vision and Pattern Recognition (CVPR)*, volume 8, 2017. 2, 6, 7

[43] E. Ustinova and V. Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016. 1, 2, 4, 6, 7

[44] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 2, 5, 7, 8

[45] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014. 1

[46] 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 *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1386–1393, 2014. 2

[47] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2612–2620. IEEE, 2017. 2, 6, 7

[48] P. Wohlhart and V. Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015. 2

[49] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Sampling matters in deep embedding learning. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 4, 6, 7, 8

[50] H. Xuan, R. Souvenir, and R. Pless. Deep randomized ensembles for metric learning. *arXiv preprint arXiv:1808.04469*, 2018. 6, 7

[51] Y. Yuan, K. Yang, and C. Zhang. Hard-aware deeply cascaded embedding. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017. 1, 3, 6, 7

[52] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[53] Y. Zhao, Z. Jin, G.-j. Qi, H. Lu, and X.-s. Hua. An adversarial approach to hard triplet generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 501–517, 2018. 2, 6, 7

[54] S. Zheng, Y. Song, T. Leung, and I. Goodfellow. Improving the robustness of deep neural networks via stability training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 4480–4488, 2016. 1

[55] S. Q. X. W. Ziwei Liu, Ping Luo and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 4, 5, 7, 8