MIC: Mining Interclass Characteristics for Improved Metric Learning

Karsten Roth*, Biagio Brattoli*, Björn Ommer
HCI/IWR, Heidelberg University, Germany
firstname.lastname@iwr.uni-heidelberg.de

Abstract

Metric learning seeks to embed images of objects such that class-defined relations are captured by the embedding space. However, variability in images is not just due to different depicted object classes, but also depends on other latent characteristics such as viewpoint or illumination. In addition to these structured properties, random noise further obstructs the visual relations of interest. The common approach to metric learning is to enforce a representation that is invariant under all factors but the ones of interest. In contrast, we propose to explicitly learn the latent characteristics that are shared by and go across object classes. We can then directly explain away structured visual variability, rather than assuming it to be unknown random noise.

We propose a novel surrogate task to learn visual characteristics shared across classes with a separate encoder. This encoder is trained jointly with the encoder for class information by reducing their mutual information. On five standard image retrieval benchmarks the approach significantly improves upon the state-of-the-art. Code is available at https://github.com/Confusezius/metric-learning-mining-interclass-characteristics.

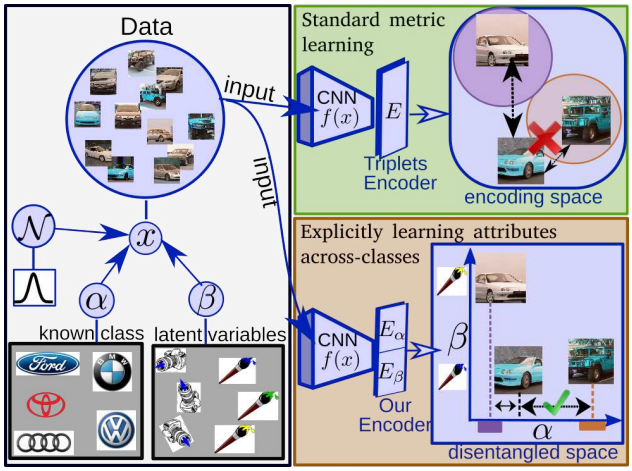

Figure 1. (Left) Images can be described by combinations of latent characteristics and white noise. (Green) Standard metric learning encoders extract class-discriminative information α while disregarding object-specific properties β (e.g. color, orientation). Achieving invariance to such characteristics requires substantial training data. (Brown) Instead, the model can explain them away by learning their structure explicitly. Our novel approach explicitly separates class-specific and shared properties during training to boost the performance of the discriminative encoding.

1. Introduction

Images live in a high dimensional space rich of structured information and unstructured noise. Therefore an image can be described by a finite combination of latent characteristics. The goal of computer vision is then to learn the relevant latent characteristics needed to solve a given task. Particularly in object classification, discriminative characteristics (e.g. car shape) are used to group the images according to predefined classes. To tackle the intra-class variability, modern classifiers can easily learn to be invariant to unstructured noise (e.g. random clutter, occlusion, image brightness). However, a considerable part of the variability is due to structured information shared among classes (e.g. view points and notions of color)

For metric learning this becomes especially important. As metric learning approaches project images into a high-dimensional feature space to measure similarities between images, every learned feature contributes. This means that finding a strong set latent characteristics is crucial. Learning the characteristics shared across classes should therefore benefit the model [20], as it can better explain the object variance within a class. Take for example a model trained only on white cars of a certain category. This model will very likely not be able to recognize a blue car of the same category (Fig.1 top-right). In this example, the encoder ignores the concept of "color" for that particular class, even though it can be learned from the data as a latent variable shared across all cars (Fig.1 bottom-right). This is a typical generalization problem and is traditionally solved by providing more labeled data. However, besides being a costly

*Indicates equal contribution

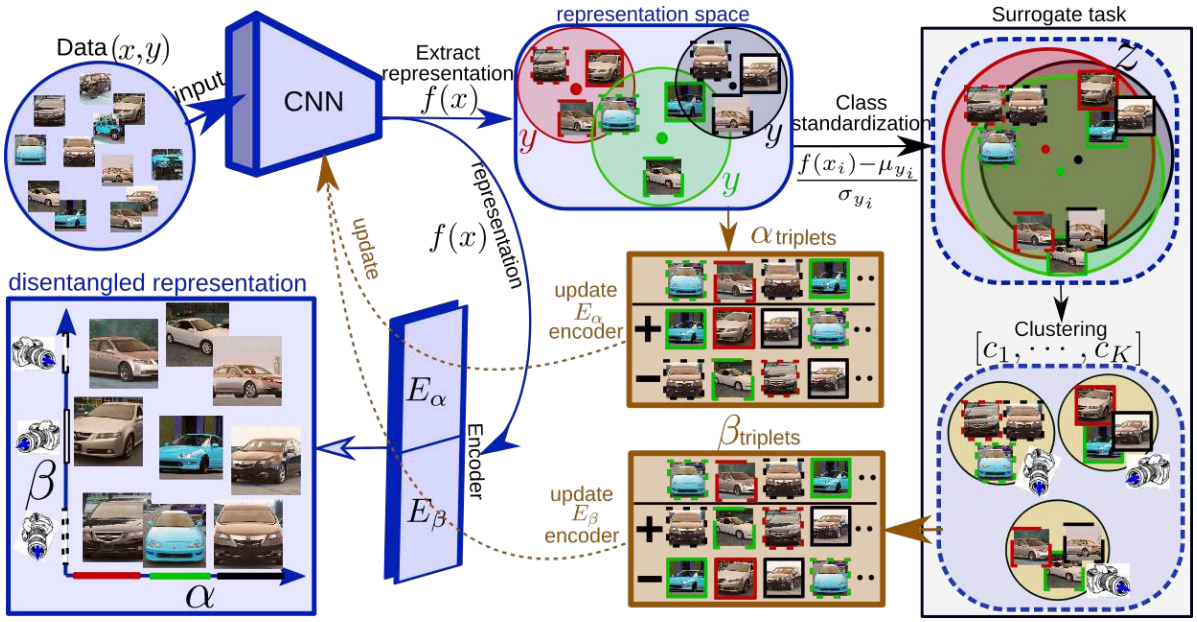

Figure 2. **Overview of our approach**. We aim to learn two separate encoding spaces s.t. class information α extracted by E_α is free from shared properties β by explicitly describing them through an auxiliary encoder E_β. Given a set of image/label pairs (x, y), their CNN feature representation $f(x)$ groups images by both class specific (car model) and shared (orientation, color) characteristics. We separate these by training the class-discriminative encoder E_α with ground-truth labels (*boundary color*). Simultaneously, an auxiliary encoder E_β is trained on labels from a surrogate task (right) to explain away interclass features. The required surrogate labels are generated by standardizing the embedded training data per class and performing clustering. This recovers labels representing the shared structures β (*contour line-styles*). Training both tasks together, E_α learns a robust, β-free encoding, which is now explicitly explained by E_β.

solution, metric learning models need to also generalize to unknown classes, a task which should work independently from the amount of labels provided.

Explicitly modeling intra-class variation has already proven successful[20, 15, 1], such as spatial transformer layers [15], which explicitly learn the possible rotations and translations of an object category.

We therefore propose a model to discriminate between classes while simultaneously learning the shared properties of the objects. To strip intra-class characteristics away from our primary class encoder, thereby facilitating the task of learning good discriminative features, we utilize an auxiliary encoder. While the class encoder can be trained using ground-truth labels, the auxiliary encoder is learned through a novel surrogate task which extracts class-independent information without any additional annotations. Finally, an additional mutual information loss further purifies the class encoder from non-discriminative characteristics by eliminating the information learned from the auxiliary encoder.

This solution can be utilized with any standard metric learning loss, as shown in the result section. Our approach is evaluated on three standard benchmarks for zero-shot learning, CUB200-2011 [37], CARS196 [19] and Stanford Online Products [28], as well as two more recent datasets, In-Shop Clothes [43] and PKU VehicleID [21]. The results show that the proposed approach consistently enhances the performances of existing methods.

2. Related Work

After the success of deep learning in object classification, many researchers have been investigating neural networks for metric learning. A network for classification extracts only the necessary features for discrimination between classes. Instead, metric learning encodes the images into an euclidean space where semantically similar ones are grouped much closer together. This makes metric learning effective in various computer vision applications, such as object retrieval [28, 39], zero-shot learning [39] and face verification [7, 34]. The triplet paradigm [34] is the standard in the field and much work has been done to improve upon the original approach. As an exponential number of possible triplets makes the computation infeasible, many papers propose solutions for mining triplets more efficiently [39, 34, 12, 11, 14]. Recently, Duan *et al.* [8] have proposed a generative model to directly produce hard negatives. ProxyNCA [24] generates a set of class proxies and optimizes the distance of the anchor to said proxies, solving the triplet complexity problem. Others have explored orthogonal directions by extending the triplet paradigm, e.g.

Algorithm 1: Training a model via MIC

Input: data X, full encoder E, inter-/intra class encoders $\{E_\alpha, E_\beta\}$, CNN f, class targets Y_α, batchsize bs, clusternumber C, update frequency T_U, (adversarial) mutual information loss l_d and weight γ, projection network R, gradient reversal op r, metric learning loss functions for $E_{\alpha,\beta}$ $l_{\alpha,\beta}$

$Y_\beta \leftarrow$ `Cluster(Stand(Embed(`X, E, f`))`, C`)`
$epoch \leftarrow 0$
while *Not Converged* **do**
  repeat
    $b_\alpha, b_\beta \leftarrow$ `GetBatch(`X, Y_α, Y_β, bs`)`
    $e_{\alpha,\beta} \leftarrow$ `Embed(`$b_{\alpha,\beta}$, $E_{\alpha,\beta}$, f`)`
    $L^\alpha \leftarrow l_\alpha(e_\alpha, Y_\alpha) + \gamma \cdot l_d(e^r_\alpha, \mathrm{R}(e^r_\beta))$
    $E_\alpha, f \leftarrow$ `Backward(`L^α`)`
    $e_{\alpha,\beta} \leftarrow$ `Embed(`$b_{\alpha,\beta}$, $E_{\alpha,\beta}$, f`)`
    $L^\beta \leftarrow l_\beta(e_\beta, Y_\beta) + \gamma \cdot l_d(e^r_\alpha, \mathrm{R}(e^r_\beta))$
    $E_\beta, f \leftarrow$ `Backward(`L^β`)`
  until *end of epoch*;
  if *epoch* $mod\ T_U == 0$ **then**
    $Y_\beta \leftarrow$ `Cluster(Embed(`X,E_β,f`)`, C`)`
  end
  $epoch \leftarrow epoch + 1$
end

making use of every sample in the (specifically constructed) batch at once [28, 35], enforcing an angular triplet constraint [38], minimizing a cluster quality surrogate [27] or optimizing the overlap between positive and negative similarity histograms [36]. In addition, ensembles have been quite successfully used by combining multiple encoding spaces [29, 30, 41, 9] to maximize their efficiency.

Our work makes use of class-agnostic grouping of our data (see e.g. [2, 3]) and shares similarities with proposals from Liu *et al.* [20], who explicitly decompose images into class-specific and intra-class embeddings using a generative model, as well as Bai *et al.* [1], who, before training, divide each image class into subgroups to find an approximator for intra-class variances that can be included into the loss. However, unlike [1] and [20], we explicitly search for structures shared between classes instead of modelling the intra-class variance per sample [20] or class [1]. In addition, unlike [1], we assume class-independent intra-class variance and iteratively train a second encoder to model intra-class features, thereby purifying the main encoder from non-discriminative features and achieving significantly better results.

Finally, some works have exploited the latent structure of the data as a supervisory signal [25, 26, 6, 4, 5, 33, 32]. In particular, Caron *et al.* [6] learn an unsupervised image representation by clustering the data, starting from a Sobel filter prior initialization. Our approach includes such latent data structures in a similar way, however we use it as auxiliary information to improve upon the metric learning task.

Figure 3. Example of clustering the data based on Z (see Sec3.3) for two datasets: CARS196[19] and SOP[28]. We group the dataset into 5 clusters (rows) and select the first 5 classes (columns) with at least one sample per cluster. For each entry, we selected the sample closest to the centroid per class. On the left is our interpretation of the cluster structure. The results show that subtraction of the class-specific features by standardization helps to group images based on more generic properties, like car orientation and bike parts.

3. Improving Metric Learning

The main idea behind our method is the inclusion of class-shared characteristics into the metric learning process to help the model explain them away. In doing so, we would gain robustness to intrinsic, non-discriminative properties of the data, which is contrary to the common approach of simply forcing invariance towards them. However, three main problems arise with this approach, namely: *(i)* Extracting both class and class-independent characteristics using a single encoder is infeasible and detrimental to the main goal. *(ii)* We lack the labels for extracting these latent properties. *(iii)* We need to explicitly remove unwanted properties from the class embedding. We propose solutions to each of these problems in sections 3.2, 3.3 and 3.4.

3.1. Preliminaries

Metric learning encodes the characteristics that discriminate between classes into an embedding vector, with the goal of training an encoder E such that images x_i from the same class y are nearby in the encoding space and samples from different classes are far apart, given a standard distance in the embedding space.

In deep metric learning, image features are extracted using a neural network $f : \mathbb{R}^{Height \times Width \times 3} \rightarrow \mathbb{R}^F$ producing an image representation vector $f(x)$, which is used as input for the encoder of the embedding $E : \mathbb{R}^F \rightarrow \mathbb{R}^D$. The latter is implemented as a fully connected layer generating

R@k	Dim	1	2	4	NMI
DVML[20]	512	52.7	65.1	75.5	61.4
BIER[29]	512	55.3	67.2	76.9	-
HTL[11]	512	57.1	68.8	78.7	-
A-BIER[30]	512	57.5	68.7	78.3	-
HTG[42]	-	59.5	71.8	81.3	-
DREML[40]	9216	63.9	75.0	83.1	67.8
Semihard[34]	-	42.6	55.0	66.4	55.4
Semihard*	128	57.2	69.4	79.9	63.9
MIC+semih	128	58.8	70.8	81.2	66.0
ProxyNCA[24]	64	49.2	61.9	67.9	64.9
ProxyNCA*	128	57.4	69.2	79.1	62.5
MIC+ProxyNCA	128	60.6	72.2	81.5	64.9
Margin[39]	128	63.6	74.4	83.1	69.0
Margin*	128	62.9	74.1	82.9	66.3
MIC+margin	128	**66.1**	**76.8**	**85.6**	**69.7**

Table 1. Recall@k for k nearest neighbor and NMI on CUB200-2011 [37]. Our model outperforms all previous approaches, even those using a larger number of parameters. (*) indicates our best re-implementation with ResNet50.

R@k	Dim	1	2	4	NMI
HTG[42]	-	76.5	84.7	90.4	-
BIER[29]	512	78.0	85.8	91.1	-
HTL[11]	512	81.4	88.0	92.7	-
DVML[20]	512	82.0	88.4	93.3	67.6
A-BIER[30]	512	82.0	89.0	93.2	-
DREML[40]	9216	86.0	91.7	95.0	76.4
Semihard[34]	-	51.5	63.8	73.5	53.4
Semihard*	128	65.5	76.9	85.2	58.3
MIC+semih	128	70.5	80.5	87.4	61.6
ProxyNCA[24]	64	73.2	82.4	86.4	-
ProxyNCA*	128	73.0	81.3	87.9	59.5
MIC+ProxyNCA	128	75.9	84.1	90.1	60.5
Margin[39]	128	79.6	86.5	90.1	**69.1**
Margin*	128	80.0	87.7	92.3	66.3
MIC+margin	128	**82.6**	**89.1**	**93.2**	68.4

Table 2. Recall@k for k nearest neighbor and NMI on CARS196 [19]. DREML[40] is not comparable given the large embedding dimension. (*) indicates our ResNet50 re-implementation.

an embedding vector of dimension D used for computing similarities. The features f and the encoder E can then be trained jointly by standard back-propagation.

With $d_{ij} = ||E(f(x_i)) - E(f(x_j))||^2$ defining the euclidean distance between the images x_i and x_j, we require that $d_{ij} < d_{ik}$ if $y_j = y_i$ and $y_k \neq y_i$. Given a triplet (x_i, x_j, x_k) with $y_j = y_i$ and $y_k \neq y_i$, the loss is then defined as $l = \max(d_{ij} - d_{ik} + m, 0)$ where m is a margin parameter. Many variants of this loss have been proposed recently, with margin loss[39] (adding an additionally learnable margin β) proving to be best.

3.2. Auxiliary Encoder

To separate the process of extracting both inter- and intra-class (shared) characteristics, we utilize two separate encodings: a class encoder E_α which aims to extract class-discriminative features and an auxiliary encoder E_β to find shared properties. These encoders are trained together (Fig.2). To efficiently train the underlying deep neural network, the two encoders share the same image representation $f(x)$ which is updated by both during training. In the first training task, the class encoder E_α is trained using the provided ground truth labels $y_1, \cdots, y_N$ associated with each image $x_1, \cdots, x_N$ with N the number of samples. A respective, metric-based loss function can be selected arbitrarily (such as a standard triplet loss or the aforementioned margin loss), as this part follows the generic training setup for metric learning problems. Because labels are not provided for the training of our auxiliary encoder, we define an automatic process to mine shared latent structure information from the original data. This information is then used to provide a new set of training labels to train our auxiliary encoder (Fig.2 right). As the training scheme is now equivalent to the primary task, we may choose from the same set of loss functions.

3.3. Extracting Inter-class Characteristics

We seek a task which, without human supervision, spots structured characteristics within the data while ignoring class-specific information. As structured properties are generally defined by characteristics shared among several images, they create homogeneous groups. To find these, clustering offers a well established solution. This algorithm associates images to surrogate labels $c_1, \cdots, c_N$ with $c_i \in [1, \cdots, C]$ and C being the predefined number of clusters. However, applied directly to the data, this method is biased towards class-specific structures since images from the same class share many common properties, like color, context and shape, mainly injected through the data collection process (e.g. a class may be composed of pictures of the same object from multiple angles).

To remove the characteristics shared within the class, we apply normalization guided by the ground truth classes. For each class y we compute the mean μ_y and standard deviation σ_y based on the features $f(x_i), \forall x_i : y_i = y$. Then we obtain the new standardized image representation $Z = [z_1, \cdots, z_N]$with $z_i = \frac{f(x_i) - \mu_{y_i}}{\sigma_{y_i}}$, where the class influence is now reduced. Afterwards, the auxiliary encoder E_β can be trained using the surrogate labels $[c_1, \cdots, c_N]$ produced by clustering the space Z.

For that to work as intended, a strong prior is needed.

R@k	Dim	1	10	100	NMI
DVML[20]	512	70.2	85.2	93.8	**90.8**
BIER[29]	512	72.7	86.5	94.0	-
ProxyNCA[24]	64	73.7	-	-	-
A-BIER[30]	512	74.2	86.9	94.0	-
HTL[11]	512	74.8	88.3	94.8	-
Margin[39]	128	72.7	86.2	93.8	**90.7**
Margin*	128	74.4	87.2	94.0	89.4
MIC+margin	128	**77.2**	**89.4**	**95.6**	90.0

Table 3. Recall@k for k nearest neighbor and NMI on Stanford Online Products [28]. (*) indicates our ResNet50 re-implementation.

R@k	Dim	1	10	30	50
BIER[29]	512	76.9	92.8	96.2	97.1
HTG[42]	-	80.3	93.9	96.6	97.1
HTL[11]	512	80.9	94.3	97.2	97.8
A-BIER[30]	512	83.1	95.1	97.5	98.0
DREML[40]	9216	78.4	93.7	96.7	-
Margin*	128	84.5	95.7	97.6	98.3
MIC+margin	128	**88.2**	**97.0**	**98.0**	**98.8**

Table 4. Recall@k for k nearest neighbor and NMI on In-Shop [43]. (*) indicates our best re-implementation with ResNet50

Test Splits		Small		Large	
R@k	Dim	1	5	1	5
MixDiff+CCL[21]	-	49.0	73.5	38.2	61.6
GS-TRS[1]	-	75.0	83.0	73.2	81.9
BIER[29]	512	82.6	90.6	76.0	86.4
A-BIER[30]	512	86.3	92.7	81.9	88.7
DREML[40]	9216	88.5	94.8	83.1	92.4
Margin*	128	85.1	92.4	80.4	88.9
MIC+margin	128	**86.9**	**93.4**	**82.0**	**91.0**

Table 5. Recall@k for k nearest neighbor and NMI on PKU VehicleID[21]. DREML[40] is not comparable given the large embedding dimension. (*) our best ResNet50 re-implementation

It is standard procedure for deep metric learning to initialize the representation backend f with weights pretrained on ImageNet. This provides a sufficiently good starting point for clustering, which is then reinforced through training E_β.

Fig.3 shows some examples of clusters detected using our surrogate task. This task and the encoder training are summarized in Fig.2.

3.4. Minimizing Mutual Information

The class encoder E_α and auxiliary encoder E_β can then be trained using the respective labels. As we utilize two different learning tasks, E_α and E_β learn distinct characteristics. However, as both share the same input, the image features $f(x)$, a dependency between the encoders can be induced, therefore leading to both encoders learning some similar properties. To reduce this effect and to constrain the discriminative and shared characteristics into their respective encoding space, we introduce a mutual information loss, which we compute through an adversarial setup

$$l_d = -\left(E^r_\alpha(f(x)) \odot R(E^r_\beta(f(x)))\right)^2 \quad (1)$$

with R being a learned, small two-layered fully-connected neural network with normalized output projecting E_β to the encoding space of E_α. $\odot$ stands for an elementwise product, while the r superscript notes a gradient reversal layer [10] which flips the gradient sign s.t. when trying to minimize l_d, i.e. maximizing correlation, the similarity between both encoders is actually decreased. A similar method has been adopted by [30], where shared information is minimized between an ensemble of encoders. In contrast, our goal is to transfer non-discriminate characteristics to an auxiliary encoder. Finally, as l_d scales with R, we avoid trivial solutions (e.g. $R(E_\beta) \to \infty$) by enforcing $R(E_\beta)$ to have unit length, similar to E_α and E_β.

Finally, the total loss L to train our two encoders and the representation f is computed by $L = l_\alpha + l_\beta + \gamma l_d$, where γ weights the contribution of the mutual information loss with respect to the class triplet loss l_α and the auxiliary triplet loss l_β. The full training is described in Alg. 1.

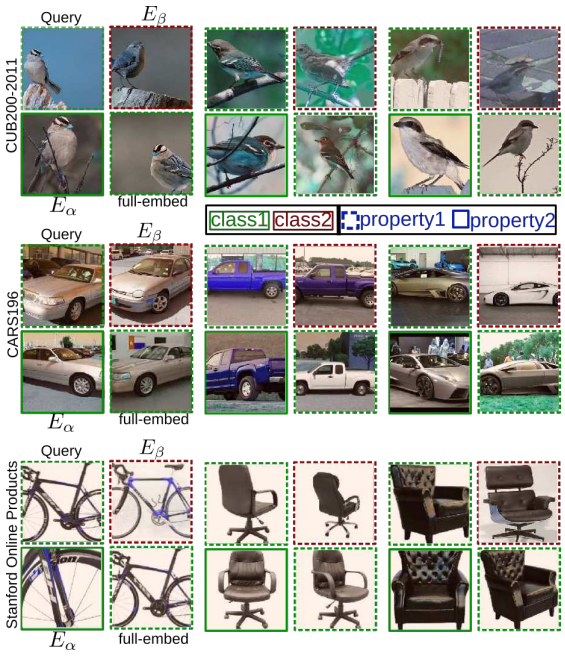

Figure 4. Qualitative nearest neighbor evaluation for CUB200-2011, CARS196 and SOP based on E_α and E_β encodings and their combination. The results show that E_β leverages class-independent information (posture,parts) while E_α becomes independent to those features and focuses on the class detection. The combination of the two reintroduces both.

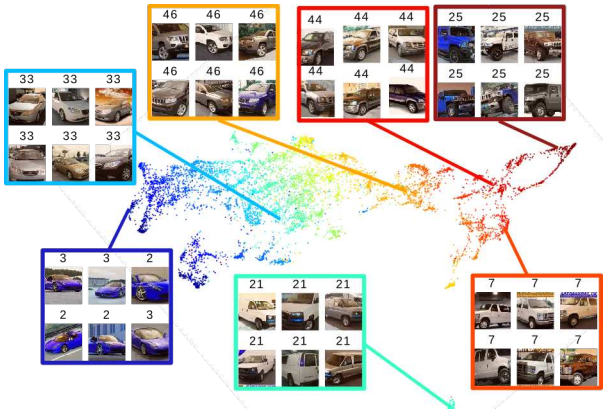

Figure 5. UMAP projection of E_α for CARS196. Seven clusters are selected, showing six images near the centroid and their ground-truth labels. We see that the encoding extracts class-specific information and ignores other (e.g. orientation).

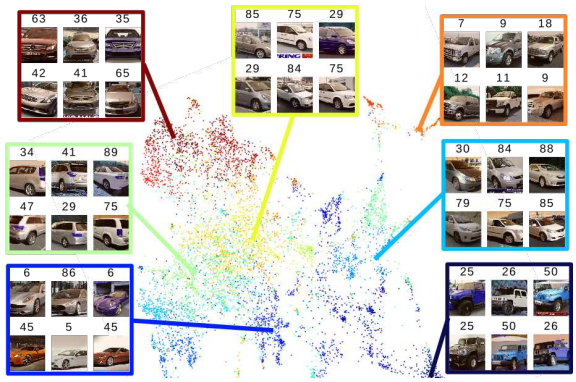

Figure 6. UMAP projection of E_β for CARS196. Seven clusters are selected, showing six images near the centroid and their GT labels. The result shows that the encoding extracts intrinsic characteristics of the object (car) independent from GT classes.

4. Experiments

In this section we offer a quantitative and qualitative analysis of our method, also in comparison to previous work. After providing technical information for reproducing the results of our model, we give some information regarding the standard benchmarks for metric learning and provide comparisons to previous methods. Finally, we offer insights into the model by studying its key components.

4.1. Implementation details

We implement our method using the PyTorch framework [31]. As baseline architecture, we utilize ResNet50 [13] due to its widespread use in recent metric learning work. All experiments use a single NVIDIA GeForce Titan X. Practically, class and auxiliary encoders E_α and E_β use the same training protocol (following [39] with embedding dimensions of 128) with alternating iterations to maximize the usable batch-size. The dimensionality of the auxiliary encoder E_β is fixed (except for ablations in sec. 5) to the dimensionality of E_α to ensure similar computational efficiency compared to previous work. However, due to GPU memory limitations, we use a batchsize of 112 instead of a proposed 128, with no relevant changes in performance.

During training, we randomly crop images of size 224×224 after resizing to 256×256, followed by random horizontal flips. For all experiments, we use the original images without bounding boxes. We train the model using Adam [18] with a learning rate of 10^{-5} and set the other parameters to default. We set the triplet parameters following [39], initializing $\beta = 1.2$ for the margin loss and $\alpha = 0.2$ as fixed triplet margin. Per mini-batch, we sample $m = 4$ images per class for a random set of classes, until the batch size is reached. For γ (Sec. 3.4 eq.) we utilize dataset-dependent values in $[100, 2000]$ determined via cross-validation.

After class standardization, the clustering is performed via standard k-means using the faiss framework [17]. Using the hyperparameters proposed in this paragraph, the computational cost introduced by our approach is 10-20% of total training time. For efficiency, the clustering can be computed on GPU using faiss[17]. The number of clusters is set before training to a fixed, problem-specific value: 30 for CUB200-2011 [37], 200 for CARS196 [19], 50 for Stanford Online Products [28], 150 for In-Shop Clothes [43] and 50 for PKU VehicleID [21]. We update the cluster labels every other epoch. Notably, however, our model is robust to both parameters since a large range of parameters give comparable results. Later in section 5 we study the effect of cluster numbers and cluster label update frequencies for each dataset in more detail to motivate the chosen numbers. Finally, class assignments by clustering, especially in the initial training stages, becomes near arbitrary for samples further away from cluster centers. To ensure that we do not reinforce such a strong initial bias, we found it beneficial to ease the class constraint by randomly switching samples with samples from different cluster classes (with probability $p \leq 0.2$).

4.2. Datasets

Our model is evaluated on five standard benchmarks for image retrieval typically used in deep metric learning. We report the Recall@k metric [16] to evaluate image retrieval and the normalized mutual information score (NMI) [22] for the clustering quality. The training and evaluation procedure follows the standard setup as used in [39].

CARS196[19] with 196 car models over 16,185 images. We use the first 98 classes (8054 images) for training and the remaining 98 (8131 images) for testing.

Stanford Online Products[28] with 120,053 product images in 22,634 classes. 59,551 images (11,318 classes)

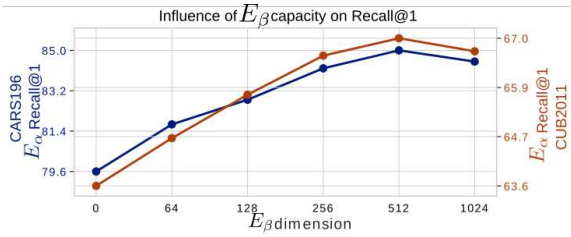

Figure 7. Evaluation of E_α as a function of the E_β capacity. For CARS196 [19] and CUB200-2011 [37], we plot E_α Recall@1 against the E_β dimension during training. The results show that the increase in capacity of E_β and thus the ability to learn properties shared among classes directly benefits the class encoder E_α.

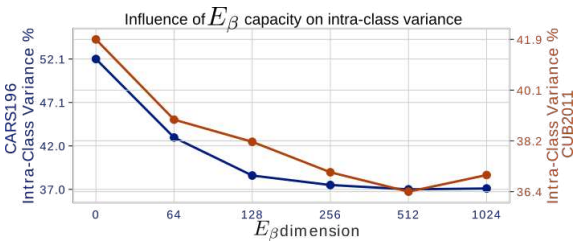

Figure 8. Measure of the intra-class variance in the class embedding E_α as function of the auxiliary encoder E_β dimension. The result shows that the intra-class variance decreases with an increase in E_β capacity. This points towards E_β making it easier for E_α to disregard class-independent information.

are used for training, 60,502 (11,316 classes) for testing.

CUB200-2011[37] with 200 bird species over 11,788 images. Train and Test Sets contain the first and last 100 classes (5,864/5,924 images) respectively.

In-Shop Clothes[43] with 72,712 clothing images in 7,986 classes. 3,997 classes are used for training and 3,985 classes for evaluation. The test set is divided into a query set (14,218 images) and a gallery set (12,612 images).

PKU VehicleID[21] with 221,736 surveillance images of 26,267 vehicles with shared car models. We follow [21] and use 13,134 classes (110,178 images) for training. Testing is done on a predefined small and large testing subset with 7,332 (small) and 20,038 (large) images respectively.

4.3. Quantitative and Qualitative Results

In this section we compare our approach with existing models from recent literature. Our method is applied on three different losses, the standard triplet loss with semi-hard negative mining [34], Proxy-NCA [24] and the state-of-the-art margin loss with weighted sampling [39]. For full transparency, we also provide results with our re-implementation of the baselines.

The results show a consistent gain over the state of the art for all datasets, see tables 1, 2, 3, 4 and 5. In particular, our approach achieves better results than more complex ensembles. On CUB200-2011, we outperform even DREML [40] which trains 48 ResNet models in parallel.

Qualitative results are shown in Fig.4: the class encoder E_α retrieves images sharing class-specific characteristics, while the auxiliary encoder E_β finds intrinsic, class-independent object properties (e.g. posture, context). The combination retrieves images with both characteristics.

5. Ablations

In this section, we investigate the properties of our model and evaluate its components. We qualitatively examine the proposed encoder properties by checking recalled images for both and study the influence of E_β on the recall performance, see Section 5.1. In Section 5 we measure the relation between the intra-class variance and the capacity of our auxiliary encoder E_β. In addition, ablation studies are performed to examine the relevance of each pipeline component and hyper-parameter. We primarily utilize the most common benchmarks CUB200-2011, CARS196 and SOP.

5.1. Embedding Properties

Firstly, we visualize the characteristics of the class encoder E_α (Fig.5) and auxiliary encoder E_β (Fig.6) by projecting the embedded test data to two dimensions using UMAP[23]. The figures show E_α extracting class-discriminative information while E_β encodes characteristics shared across classes (e.g. car orientation).

To evaluate the effect of the auxiliary encoder E_β on the class encoder E_α, we study the properties of the class encoding as function of the capability of E_β to learn shared characteristics. First, we study the performance of E_α on CARS196[19] and CUB200-2011[37] relative to the auxiliary encoder dimension. Utilizing varying E_β dimensionalities, Fig.7 shows a direct relation between E_β capacity and the retrieval capability. E_β with dimension 0 indicates the baseline method [39]. For all other evaluations, the E_β dimension is equal to E_α to keep the computational cost comparable to the baseline [39] (see Sec.4.1).

To examine our initial assumption that learning shared characteristics produces more compact classes, we study the intra-class variance by computing the mean pairwise distances per class, averaged over all classes. These distances are normalized by the average inter-class distance, approximated by the distance between two class centers.Summarized in fig.8 we see higher intra-class variance for basic margin loss (E_β dimension equal to 0). But more importantly, the class compactness is directly related to the capacity of the auxiliary encoder E_β.

We also offer a qualitative evaluation of the surrogate task in Fig.3. After class-standardization, the clustering recognizes latent structures of the data shared across classes.

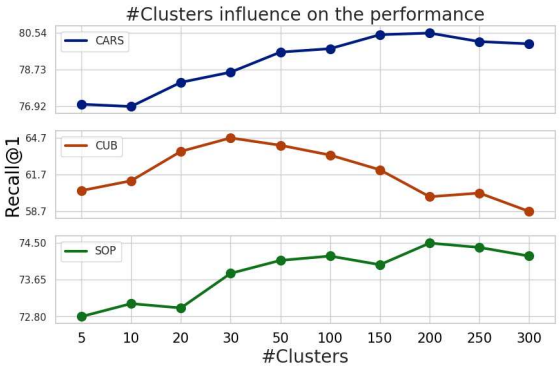

Figure 9. Ablation study: influence of the number of clusters on Recall@1. A fixed cluster label update period of 1 was used with equal learning rate and consistent scheduling.

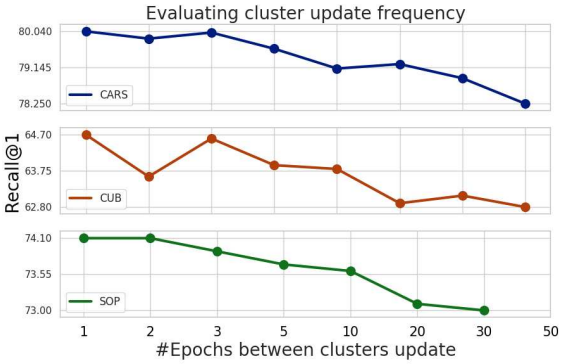

Figure 10. Ablation study: influence of the cluster label update frequency on Recall@1. An optimal number of clusters (see Sec. 4.1) and consistent scheduling was used.

Clust	Stand	MutInfo	CARS	CUB	SOP
-	-	-	80.0	62.9	73.2
+	-	-	79.2	59.1	71.9
+	+	-	81.3	64.9	75.8
+	+	+	**82.6**	**66.1**	**77.2**

Table 6. Ablation study: Relevance of different contributions. Each component is crucial for reaching the best performance. (Clust: E_β training with clusters, Stand: standardization before clustering (Sec3.3), MutInfo: mutual information loss (Sec3.4))

5.2. Testing Components and Parameters

In order to analyze our modules, we evaluate different models, each lacking one of the proposed contribution, see tab. 6. The table shows how each component is needed for the best performance. Comparing to the baseline in the first line, we see that simply introducing an additional task based on clustering the data deteriorates the performance, as we add another class-discriminative training signal that introduces worse or even contradictory information. However, by utilizing standardization, we allow our second encoder to explicitly learn new features to support the class encoder instead of working against it, giving a significant performance boost. A final mutual information loss emphasises the feature separation to improve the results further.

Our approach can be combined with most existing metric learning losses, which we evaluate on ProxyNCA[24] and triplet loss with semihard sampling[34] in Tab.1 and 2. On both CARS196 and CUB200-2011, we see improved image retrieval performance.

To examine the newly introduced hyper-parameters, Fig.9 compares the performances on the three benchmarks using a range of cluster numbers. The plot shows how the number of clusters influences the final performances, meaning the quality of the latent structure extracted by the auxiliary encoder E_β is crucial for a better classification. At the same time, an optimal performance, within a range of $\pm 1\%$ Recall@1, is reached by a large set of cluster values, making the model robust to this hyper-parameter. For these cumulative tests, a higher learning rate and less training epochs were used to both reduce computation time and avoid overfitting to the test set. Based on these examinations, we set a fixed, but dataset-dependent cluster number for all other training runs, see Sec. 4.1.

A similar evaluation has been performed on the update frequency for the auxiliary labels (Fig.10). Updating the cluster frequently clearly provides a boost to our model, suggesting that the auxiliary encoder E_β improves upon the initial clustering. However, within a reasonable range of values (between an update every 1 to 10 epochs) the model has no significant drop in performance. Thus we fix this parameter to update every two epochs for all the experiments.

6. Conclusion

In this paper we have introduced a novel extension for standard metric learning methods to incorporate structured intra-class information into the learning process. We do so by separating the encoding space into two distinct subspaces. One incorporates information about class-dependent characteristics, with the remaining encoder handling shared, class-independent properties. While the former is trained using standard metric learning setups, we propose a new learning task for the second encoder to learn shared characteristics and explain a combined training setup. Experiments on several standard image retrieval datasets show that our method consistently boost standard approaches, outperforming the current state-of-the-art methods and reducing intra-class variance.

Acknowledgements. This work has been supported by Bayer and hardware donations by NVIDIA corporation.

References

[1] Yan Bai, Feng Gao, Yihang Lou, Shiqi Wang, Tiejun Huang, and Ling-Yu Duan. Incorporating intra-class variance to fine-grained visual recognition. *CoRR*, abs/1703.00196, 2017. 2, 3, 5

[2] Miguel Ángel Bautista, Artsiom Sanakoyeu, and Björn Ommer. Deep unsupervised similarity learning using partially ordered sets. *CoRR*, abs/1704.02268, 2017. 3

[3] Miguel A Bautista, Artsiom Sanakoyeu, Ekaterina Tikhoncheva, and Björn Ommer. Cliquecnn: Deep unsupervised exemplar learning. In *Advances in Neural Information Processing Systems*, pages 3846–3854, 2016. 3

[4] Biagio Brattoli, Uta Büchler, Anna-Sophia Wahl, Martin E. Schwab, and Björn Ommer. Lstm self-supervision for detailed behavior analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[5] Uta Büchler, Biagio Brattoli, and Björn Ommer. Improving spatiotemporal self-supervisionby deep reinforcement learning. In *IEEE Conference on European Conference on Computer Vision (ECCV)*, 2018. 3

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. *CoRR*, abs/1807.05520, 2018. 3

[7] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. pages 539–546. IEEE, 2005. 2

[8] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep adversarial metric learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[9] Yoav Freund and Robert 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

[10] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *J. Mach. Learn. Res.*, 17(1):2096–2030, Jan. 2016. 5

[11] Weifeng Ge. Deep metric learning with hierarchical triplet loss. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 269–285, 2018. 2, 4, 5

[12] Ben Harwood, BG Kumar, Gustavo Carneiro, Ian Reid, Tom Drummond, et al. Smart mining for deep metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2821–2829, 2017. 2

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[14] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. Mining on manifolds: Metric learning without labels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7642–7651, 2018. 2

[15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015. 2

[16] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2011. 6

[17] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017. 6

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[19] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 554–561, 2013. 2, 3, 4, 6, 7

[20] Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep variational metric learning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 1, 2, 3, 4, 5

[21] Hongye Liu, Yonghong Tian, Yaowei Wang, Lu Pang, and Tiejun Huang. Deep relative distance learning: Tell the difference between similar vehicles. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2167–2175, 2016. 2, 5, 6, 7

[22] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010. 6

[23] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 7

[24] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 360–368, 2017. 2, 4, 5, 7, 8

[25] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 3

[26] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9359–9367, 2018. 3

[27] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5382–5390, 2017. 3

[28] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 2, 3, 5, 6

[29] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Bier-boosting independent embeddings robustly. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5189–5198, 2017. 3, 4, 5

[30] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 3, 4, 5

[31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 6

[32] Artsiom Sanakoyeu, Vadim Tschernezki, Uta Büchler, and Björn Ommer. Divide and conquer the embedding space for metric learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 3

[33] Nawid Sayed, Biagio Brattoli, and Björn Ommer. Cross and learn: Cross-modal self-supervision. In *German Conference on Pattern Recognition (GCPR)*, 2018. 3

[34] Florian Schroff, Dmitry Kalenichenko, and James 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. 2, 4, 7, 8

[35] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016. 3

[36] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016. 3

[37] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2, 4, 6, 7

[38] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017. 3

[39] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2840–2848, 2017. 2, 4, 5, 6, 7

[40] Hong Xuan, Richard Souvenir, and Robert Pless. Deep randomized ensembles for metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 723–734, 2018. 4, 5, 7

[41] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 814–823, 2017. 3

[42] Yiru Zhao, Zhongming Jin, Guo-jun Qi, Hongtao Lu, and Xian-sheng Hua. An adversarial approach to hard triplet generation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 501–517, 2018. 4, 5

[43] Shi Qiu Xiaogang Wang Ziwei Liu, Ping Luo and Xiaoou 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, 5, 6, 7