

The Group Loss++: A deeper look into group loss for deep metric learning

Ismail Elezi*, Jenny Seidenschwarz*, Laurin Wagner*, Sebastiano Vascon, Alessandro Torcinovich, Marcello Pelillo, *IEEE Fellow*, and Laura Leal-Taixé

Abstract—Deep metric learning has yielded impressive results in tasks such as clustering and image retrieval by leveraging neural networks to obtain highly discriminative feature embeddings, which can be used to group samples into different classes. Much research has been devoted to the design of smart loss functions or data mining strategies for training such networks. Most methods consider only pairs or triplets of samples within a mini-batch to compute the loss function, which is commonly based on the distance between embeddings. We propose *Group Loss*, a loss function based on a differentiable label-propagation method that enforces embedding similarity across *all* samples of a group while promoting, at the same time, low-density regions amongst data points belonging to different groups. Guided by the smoothness assumption that “similar objects should belong to the same group”, the proposed loss trains the neural network for a classification task, enforcing a consistent labelling amongst samples within a class. We design a set of inference strategies tailored towards our algorithm, named *Group Loss++* that further improve the results of our model. We show state-of-the-art results on clustering and image retrieval on four retrieval datasets, and present competitive results on two person re-identification datasets, providing a unified framework for retrieval and re-identification.

Index Terms—deep metric learning, person re-identification, similarity learning, object retrieval, image clustering.

1 INTRODUCTION

Measuring object similarity is at the core of many important machine learning problems like clustering and object retrieval. For visual tasks, this means learning a distance function over images. With the rise of deep neural networks, the focus has rather shifted towards learning a feature embedding that is easily separable using a simple distance function, such as the Euclidean distance. In essence, objects of the same class (similar) should be close by in the learned manifold, while objects of a different class (dissimilar) should be far away.

Historically, the best performing approaches get deep feature embeddings from the so-called siamese networks [1], which are typically trained using the contrastive loss [1] or the triplet loss [2], [3]. A clear drawback of these losses is that they only consider pairs or triplets of data points, missing key information about the relationships between all members of the mini-batch. On a mini-batch of size n , despite that the number of pairwise relations between samples is $\mathcal{O}(n^2)$, contrastive loss uses only $\mathcal{O}(n/2)$ pairwise relations, while triplet loss uses $\mathcal{O}(2n/3)$ relations. Additionally, these methods consider only the relations between objects of the same class (positives) and objects of other classes (negatives), without making any distinction that negatives belong to different classes. This leads to not taking into consideration the global structure of the embedding space, and consequently results in lower clustering and retrieval performance. To compensate for that, researchers

rely on other tricks to train neural networks for deep metric learning: intelligent sampling [4], multi-task learning [5] or hard-negative mining [6]. Recently, researchers have been increasingly working towards exploiting in a principled way the global structure of the embedding space [7], [8], [9], [10], typically by designing ranking loss functions instead of following the classic triplet formulations.

In a similar spirit, we propose *Group Loss*, a novel loss function for deep metric learning that considers the similarity between all samples in a mini-batch. To create the mini-batch, we sample from a fixed number of classes, with samples coming from a class forming a *group*. Thus, each mini-batch consists of several randomly chosen groups, and each group has a fixed number of samples. An iterative, fully-differentiable label propagation algorithm is then used to build feature embeddings which are similar for samples belonging to the same group, and dissimilar otherwise.

At the core of our method lies an iterative process called replicator dynamics [11], [12], that refines the local information, given by the softmax layer of a neural network, with the global information of the mini-batch given by the similarity between embeddings. The driving rationale is that the more similar two samples are, the more they affect each other in choosing their final label and tend to be grouped together in the same group, while dissimilar samples do not affect each other on their choices.

We then study the embedding space generated by networks trained with our Group Loss, resulting in a few observations that we exploit by introducing a set of inference strategies. We call this model, the *Group Loss++* and show that reaches significantly better results than the *Group Loss*, making clustering and image retrieval easier. Finally, we show that our proposed model can be used to train networks in the field of person re-identification,

- I.E, J.S, L.W and L.L.T are with *Dynamic Vision and Learning Group* at the *Technical University of Munich*, S.V, A.T and M.P are at *Ca’ Foscari University of Venice*
Corresponding authors: Ismail Elezi (ismail.elezi@tum.de), Laura Leal-Taixé (leal.taixe@tum.de)
- * denotes equal contribution

thus providing a similarity learning unified framework that works both for retrieval and re-identification.

Our **contribution** in this work is five-fold:

- We propose the *Group Loss*, a novel loss function to train neural networks for deep metric embedding that takes into account the local information of the samples, as well as their similarity.
- We propose a differentiable label-propagation iterative model to embed the similarity computation within backpropagation, allowing end-to-end training with our new loss function.
- We introduce a set of inference strategies, resulting in *Group Loss++* that improve the results of our model.
- We show state-of-the-art qualitative and quantitative results in four standard clustering and retrieval datasets.
- We show competitive results on two person re-identification datasets, thus providing a unified framework for similarity learning.

2 RELATED WORK

Classical metric learning losses. The first attempt at using a neural network for feature embedding was done in the seminal work of Siamese Networks [1]. A cost function called *contrastive loss* was designed in such a way as to minimize the distance between pairs of images belonging to the same cluster and maximize the distance between pairs of images coming from different clusters. In [13], researchers used the principle to successfully address the problem of face verification. Another line of research on convex approaches for metric learning led to the triplet loss [2], [3], which was later combined with the expressive power of neural networks [6]. The main difference from the original Siamese network is that the loss is computed using triplets (an anchor, a positive, and a negative data point). The loss is defined to make the distance between features of the anchor and the positive sample smaller than the distance between the anchor and the negative sample. The approach was so successful in the field of face recognition and clustering, that soon many works followed. The majority of works on the Siamese architecture consist of finding better cost functions, resulting in better performances on clustering and retrieval. In [14], the authors generalized the concept of triplet by allowing a joint comparison among $N - 1$ negative examples instead of just one. [15] designed an algorithm for taking advantage of the mini-batches during the training process by lifting the vector of pairwise distances within the batch to the matrix of pairwise distances, thus enabling the algorithm to learn feature embedding by optimizing a novel structured prediction objective on the lifted problem. The work was later extended in [16], proposing a new metric learning scheme based on structured prediction that is designed to optimize a clustering quality metric, i.e., the normalized mutual information [17]. Better results were achieved on [18], where the authors proposed a novel angular loss, which takes angle relationship into account. A very different problem formulation was given by [19], where the authors used a spectral clustering-inspired approach to achieve deep embedding. Similar to that is [20] where the authors design a network to directly learn to cluster objects.

A recent work presents several extensions of the triplet loss that reduce the bias in triplet selection by correcting the distribution shift on the selected triplets [21]. Our work differs from all the mentioned works. While the goal is the same, finding good feature embeddings for every image, we do not use triplets but instead use all the relations in a minibatch, thus learning the global manifold of the dataset.

Classification-based losses. Recently, the work of [22] showed that a carefully tuned normalized softmax cross-entropy loss function combined with a balanced sampling strategy can achieve competitive results. A similar line of research is that of [23], where the authors use a combination of normalized-scale layers and Gram-Schmidt optimization to achieve efficient usage of the softmax cross-entropy loss for metric learning. CenterLoss [24] modifies the cross-entropy loss by simultaneously learning a center for deep features of each class and penalizing the distances between the deep features and their corresponding class centers. SoftTriple loss [25] goes a step further by taking into consideration the similarity between classes. Furthermore, it uses multiple centers for class, allowing them to reach state-of-the-art results, at a cost of significantly increasing the number of parameters of the model. Our work significantly differs from the mentioned works. Instead of learning one or more centers, or considering the similarity between classes, we instead refine the features of each image based on the similarity with the other images in the dataset. By doing so, we achieve state-of-the-art results without increasing the number of parameters of the model.

Proxy-losses. Related to SoftTriple loss [25] are the proxy-losses. The authors of [26], [27], [28] proposed to optimize the triplet loss on a different space of triplets than the original samples, consisting of an anchor data point and similar and dissimilar learned proxy data points. These proxies approximate the original data points so that a triplet loss over the proxies is a tight upper bound of the original loss. The final formulation of the loss is shown to be similar to that of softmax cross-entropy loss, challenging the long-held belief that classification losses are not suitable for the task of metric learning. Proxy-GML [29] goes a step further by using reverse label propagation to adjust the neighbor relationships of the proxies based on their ground truth. While, like the mentioned works, our work modifies cross-entropy loss, we do not use any proxy representation. Furthermore, unlike Proxy-GML, at the heart of our work is a label propagation between the samples in the mini-batch, thus directly optimizing the embedding space.

Person re-identification. Similar to deep metric learning, Siamese Networks [1] are historically utilized to generate feature embeddings since the the earliest deep learning-based works for person re-identification [30], [31]. Typically, works on person re-identification use versions of the triplet loss together with sampling strategies [32], [33], [34], [35], [36], [37], [38]. Several works [38], [39], [40], [41], [42], propose to subdivide the image of the whole person into several parts and process them separately to enforce the architectures to learn local, more fine-grained features [39]. While many works rely on basic classification backbones such as ResNets [43] or DenseNets [44], several works introduce their own backbone specifically tailored to the task of person re-identification. For example, [36] proposed

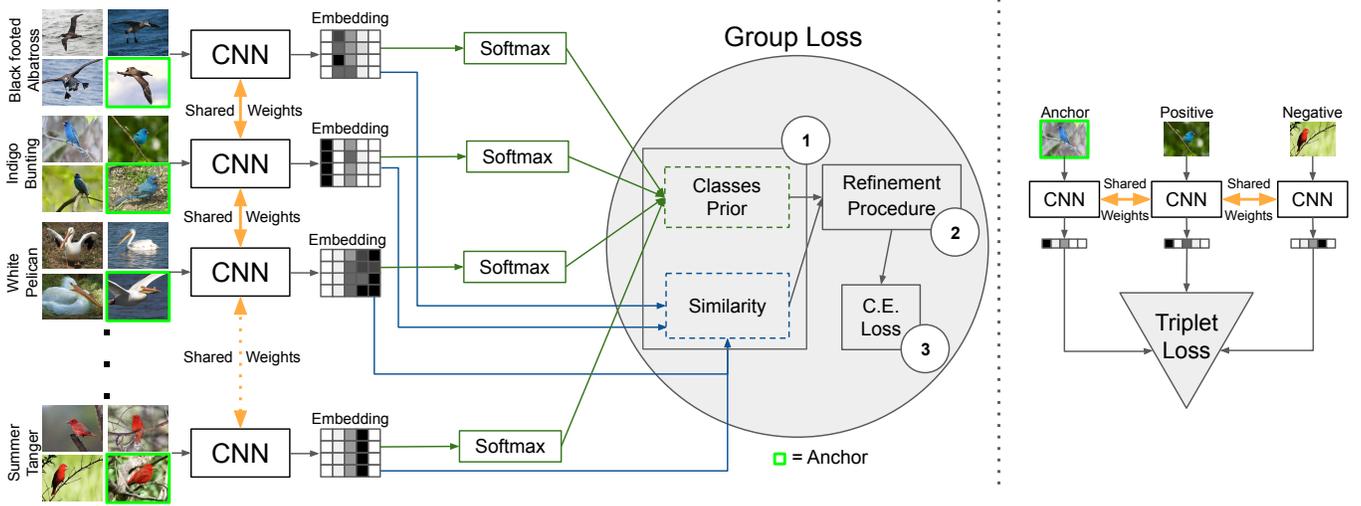


Fig. 1. A comparison between a neural model trained with the Group Loss (left) and the triplet loss (right). Given a mini-batch of images belonging to different classes, their embeddings are computed through a convolutional neural network. Such embeddings are then used to generate a similarity matrix that is fed to the Group Loss along with prior distributions of the images on the possible classes. The green contours around some mini-batch images refer to *anchors*. It is worth noting that, differently from the triplet loss, the Group Loss considers multiple classes and the pairwise relations between all the samples. Numbers from ① to ③ refer to the Group Loss steps, see Sec 3.2 for the details.

a neural architecture search approach inspired by [45] that aims to find an optimal architecture by searching a search space of operations to construct and concatenate neural cells in an automated way. Probably the closest to our work is that of [46] where the authors target the task of person re-identification from a graph-theoretical point of view. Given query and gallery images form an undirected graph and the images belonging to the same identity should form a cluster. These clusters can be found by making use of constrained dominant sets [47] in order to control the global shape of the energy landscape. The optimization of the clustering objective is done by a similar iterative procedure that uses replicator dynamics [11]. Unlike [46], we use the iterative procedure to spread the label preferences from images to their neighbors.

3 GROUP LOSS

3.1 Preliminaries

Given a bunch of samples I_1, I_2, \dots, I_i , the task of metric learning is to learn a function that maps them to a compact representation (an embedding) $\phi(I_1), \phi(I_2), \dots, \phi(I_i)$ such that the representation of the samples coming from the same class should be close, while those coming from different classes should be further away. In inference, this function is used to find representations of samples that come from different classes, and then this representation is used to do retrieval (e.g. KNN) or clustering (e.g. K-Means [48]). For the purpose of this work, we will consider that samples are images, although in principle, they can be any type of data.

The classical way of doing metric learning is to use the **contrastive** loss [1] that minimizes the distance between pairs of images belonging to the same class and maximizes the distance between pairs of images coming from different classes. Given two images A and B , the loss is described by the following equation:

$$L(A, B) = y^* \|\phi(A) - \phi(B)\|^2 + (1 - y^*) \max(0, m^2 - \|\phi(A) - \phi(B)\|^2) \quad (1)$$

where y^* is an indicator variable that is 1 if A and B belong to the same class and 0 otherwise, while m is a margin.

An extension of the contrastive loss is the **triplet loss** [6], which takes as input an anchor A , a positive P , and a negative N , where the loss optimizes the distance between A and P to be smaller (up to a margin m) than the distance between A and N :

$$L(A, P, N) = \max(0, \|\phi(A) - \phi(P)\|^2 - \|\phi(A) - \phi(N)\|^2 + m) \quad (2)$$

Most recent metric learning loss functions are extensions of the triplet loss [4], [6], [10], [14], [15], [16], [18], [19], [33], [49], often combined with some type of intelligent sampling. Consequently, they are not based on a classification loss function, e.g., cross-entropy, but rather a loss function based on embedding distances. The rationale behind it is that what matters for a classification network is that the output is correct, which does not necessarily mean that the embeddings of samples belonging to the same class are similar. Since each sample is classified independently, it is entirely possible that two images of the same class have two distant embeddings that both allow for a correct classification. We argue that a classification loss can still be used for deep metric learning if the decisions do not happen independently for each sample, but rather jointly for a whole *group*, i.e., the set of images of the same class in a mini-batch. In this way, the method pushes for images belonging to the same class to have similar embeddings.

Towards this end, we propose *Group Loss* [50], an iterative procedure that uses the global information of the mini-batch to refine the local information provided by the

softmax layer of a neural network. This iterative procedure categorizes samples into different *groups*, and enforces consistent labelling among the samples of a group. While softmax cross-entropy loss judges each sample in isolation, the Group Loss allows us to judge the overall class separation for *all* samples. In section 3.4, we show the differences between the softmax cross-entropy loss and Group Loss and highlight the mathematical properties of our new loss.

3.2 Overview of Group Loss

Given a mini-batch B consisting of n images, consider the problem of assigning a class label $\lambda \in \Lambda = \{1, \dots, m\}$ to each image in B . In the remainder of the manuscript, $\mathbf{X} = (x_{i\lambda})$ represents a $n \times m$ (non-negative) matrix of image-label soft assignments. In other words, each row of \mathbf{X} represents a probability distribution over the label set Λ ($\sum_{\lambda} x_{i\lambda} = 1$ for all $i = 1 \dots n$).

Our model consists of the following steps (see also Fig. 1 and Algorithm 1):

- 1) **Initialization:** Initialize \mathbf{X} , the image-label assignment using the softmax outputs of the neural network. Compute the $n \times n$ pairwise similarity matrix \mathbf{W} using the neural network embedding.
- 2) **Refinement:** Iteratively, refine \mathbf{X} considering the similarities between all the mini-batch images, as encoded in \mathbf{W} , as well as their labeling preferences.
- 3) **Loss computation:** Compute the cross-entropy loss of the refined probabilities and update the weights of the neural network using backpropagation.

3.3 Initialization

Image-label assignment matrix. The initial assignment matrix denoted $\mathbf{X}(0)$, comes from the softmax output of the neural network. We can replace some of the initial assignments in matrix \mathbf{X} with one-hot labelings of those samples. We call these randomly chosen samples *anchors*, as their assignments do not change during the iterative refine process and consequently do not directly affect the loss function. However, by using their correct label instead of the predicted label (coming from the softmax output of the CNN), they guide the remaining samples towards their correct label.

Similarity matrix. A measure of similarity is computed among all pairs of embeddings (computed via a CNN) in B to generate a similarity matrix $\mathbf{W} \in \mathbf{R}^{n \times n}$. In this work, we compute the similarity measure using the Pearson’s correlation coefficient [51]:

$$\omega(i, j) = \frac{\text{Cov}[\phi(\mathbf{I}_i), \phi(\mathbf{I}_j)]}{\sqrt{\text{Var}[\phi(\mathbf{I}_i)]\text{Var}[\phi(\mathbf{I}_j)]}} \quad (3)$$

for $i \neq j$, and set $\omega(i, i)$ to 0. The choice of this measure over other options such as cosine layer, Gaussian kernels, or learned similarities, is motivated by the observation that the correlation coefficient uses data standardization, thus providing invariance to scaling and translation – unlike the cosine similarity, which is invariant to scaling only – and it does not require additional hyperparameters, unlike Gaussian kernels [52]. The fact that a measure of the linear relationship among features provides a good similarity measure

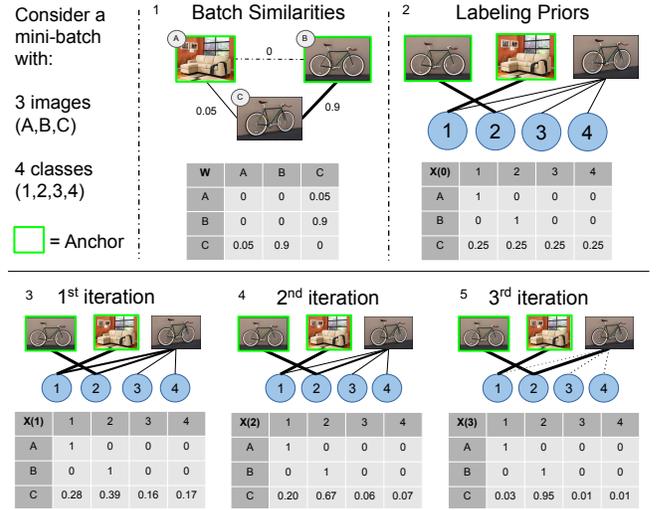


Fig. 2. A toy example of the refinement procedure, where the goal is to classify sample C based on the similarity with samples A and B. (1) The Affinity matrix used to update the soft assignments. (2) The initial labeling of the matrix. (3-4) The process iteratively refines the soft assignment of the unlabeled sample C. (5) At the end, sample C gets the same label of A, (A, C) being more similar than (B, C).

can be explained by the fact that the computed features are actually a highly non-linear function of the inputs. Thus, the linear correlation among the embeddings actually captures a non-linear relationship among the original images.

3.4 Refinement

In this core step of the proposed algorithm, the initial assignment matrix $\mathbf{X}(0)$ is refined in an iterative manner, taking into account the similarity information provided by matrix \mathbf{W} . \mathbf{X} is updated in accordance with the *smoothness assumption*, which prescribes that similar objects should share the same label.

To this end, let us define the *support* matrix $\mathbf{\Pi} = (\pi_{i\lambda}) \in \mathbf{R}^{n \times m}$ as

$$\mathbf{\Pi} = \mathbf{W} \mathbf{X} \quad (4)$$

whose (i, λ) -component

$$\pi_{i\lambda} = \sum_{j=1}^n w_{ij} x_{j\lambda} \quad (5)$$

represents the *support* that the current mini-batch gives to the hypothesis that the i -th image in B belongs to class λ . Intuitively, in obedience to the smoothness principle, $\pi_{i\lambda}$ is expected to be high if images similar to i are likely to belong to class λ .

Given the initial assignment matrix $\mathbf{X}(0)$, our algorithm refines it using the following update rule:

$$x_{i\lambda}(t+1) = \frac{x_{i\lambda}(t)\pi_{i\lambda}(t)}{\sum_{\mu=1}^m x_{i\mu}(t)\pi_{i\mu}(t)} \quad (6)$$

where the denominator represents a normalization factor which guarantees that the rows of the updated matrix sum up to one. This is known as multi-population replicator dynamics in evolutionary game theory [11] and is equivalent to nonlinear relaxation labeling processes [53], [54].

In matrix notation, the update rule (6) can be written as:

$$\mathbf{X}(t+1) = \mathbf{Q}^{-1}(t) [\mathbf{X}(t) \odot \mathbf{\Pi}(t)] \quad (7)$$

where

$$\mathbf{Q}(t) = \text{diag}([\mathbf{X}(t) \odot \mathbf{\Pi}(t)] \mathbf{1}) \quad (8)$$

and $\mathbf{1}$ is the all-one m -dimensional vector. $\mathbf{\Pi}(t) = \mathbf{W}\mathbf{X}(t)$ as defined in (4), and \odot denotes the Hadamard (element-wise) matrix product. In other words, the diagonal elements of $\mathbf{Q}(t)$ represent the normalization factors in (6), which can also be interpreted as the average support that object i obtains from the current mini-batch at iteration t . Intuitively, the motivation behind our update rule is that at each step of the refinement process, for each image i , a label λ will increase its probability $x_{i\lambda}$ if and only if its support $\pi_{i\lambda}$ is higher than the average support among all the competing label hypothesis Q_{ii} . This can be motivated by a Darwinian survival-of-the-fittest selection principle, see e.g. [11].

Thanks to the Baum-Eagon inequality [54], which has recently gained a renewed interest as an instance of Multiplicative Weights Update (MWU) rule [55], [56], it is easy to show that the dynamical system defined by (6) has very nice convergence properties. In particular, it strictly increases at each step the following functional:

$$\mathcal{F}(\mathbf{X}) = \sum_{i=1}^n \sum_{j=1}^n \sum_{\lambda=1}^m w_{ij} x_{i\lambda} x_{j\lambda} \quad (9)$$

provided that the weights w_{ij} are non-negative. The functional (9) represents a measure of "consistency" [57] of the assignment matrix \mathbf{X} , in accordance to the smoothness assumption (\mathcal{F} rewards assignments where highly similar objects are likely to be assigned the same label). In other words:

$$\mathcal{F}(\mathbf{X}(t+1)) \geq \mathcal{F}(\mathbf{X}(t)) \quad (10)$$

with equality if and only if $\mathbf{X}(t)$ is a stationary point. Hence, our update rule (6) is, in fact, an algorithm for maximizing the functional \mathcal{F} over the space of row-stochastic matrices. Baum and Eagon [58] further prove a more general result for non-homogeneous polynomials, with non-negative coefficients by proving that such mappings increase *homotopically*, which in our case means that for $0 \leq \eta \leq 1$:

$$\mathcal{F}(\eta\mathbf{X}(t+1) + (1-\eta)\mathbf{X}(t)) \geq \mathcal{F}(\mathbf{X}(t)) \quad (11)$$

Note, that this contrasts with classical gradient methods, for which an increase in the objective function is guaranteed only when infinitesimal steps are taken, and determining the optimal step size entails computing higher-order derivatives. Here, instead, the step size is implicit and yet, at each step, the value of the functional increases.

3.4.1 Dealing with negative similarities

Equation (6) assumes that the matrix of similarity is non-negative. However, for similarity computation, we use a correlation metric (see Equation (3)) which produces values in the range $[-1, 1]$. In similar situations, different authors propose different methods to deal with the negative outputs. The most common approach is to shift the matrix of similarity towards the positive regime by subtracting the biggest negative value from every entry in the matrix [12].

Nonetheless, this shift has a side effect: if a sample of class k_1 has very low similarities to the elements of a large group of samples of class k_2 , these similarity values (which after being shifted are all positive) will be summed up. If the cardinality of class k_2 is very large, then summing up all these small values lead to a large value, and consequently affect the solution of the algorithm. What we want instead, is to ignore these negative similarities, hence we propose *clamping*. More concretely, we use a *ReLU* activation function over the output of Equation (3).

3.5 Loss computation

Once the labeling assignments converge (or in practice, a maximum number of iterations is reached), we apply the cross-entropy loss to quantify the classification error and backpropagate the gradients. Recall, the refinement procedure is optimized via *replicator dynamics*, as shown in the previous section. By studying Equation (7), it is straightforward to see that it is composed of fully differentiable operations (matrix-vector and scalar products), and so it can be easily integrated within backpropagation. Although the refining procedure has no parameters to be learned, its gradients can be backpropagated to the previous layers of the neural network, producing, in turn, better embeddings for similarity computation.

3.6 Summary of Group Loss

In this section, we proposed the Group Loss function for deep metric learning. During training, the Group Loss works by grouping together similar samples based on both the similarity between the samples in the mini-batch and the local information of the samples. The similarity between samples is computed by the correlation between the embeddings obtained from a CNN, while the local information is computed with a softmax layer on the same CNN embeddings. Using an iterative procedure, we combine both sources of information and effectively bring together embeddings of samples that belong to the same class.

During inference, we simply forward pass the images through the neural network to compute their embeddings, which are directly used for image retrieval within a nearest neighbor search scheme. The iterative procedure is not used during inference, thus making the feature extraction as fast as that of any other competing method.

4 GROUP LOSS++: INFERENCE STRATEGIES

In this section, we study the geometry of the embedding space that is built by the neural network trained with Group Loss. For each observation we make, we propose an inference strategy to further exploit such embedding geometry, increasing the results of retrieval without retraining.

4.1 β -Normalization

The embedding vector of an image is characterized by its length and direction. The softmax loss is mostly determined by the direction of an embedding pointing in the same direction as the weights of the correct class. Due to this property, the Group Loss embeddings are mainly trained to

Algorithm 1: The Group Loss

-
- Input:** input : Set of pre-processed images in the mini-batch \mathbf{B} , set of labels \mathbf{y} , neural network ϕ with learnable parameters θ , similarity function ω , number of iterations T
- 1 Compute feature embeddings $\phi(\mathbf{B}, \theta)$ via the forward pass
 - 2 Compute the similarity matrix $\mathbf{W} = [\omega(i, j)]_{ij}$
 - 3 Initialize the matrix of priors $\mathbf{X}(0)$ from the softmax layer
 - 4 **for** $t = 0, \dots, T-1$ **do**
 - 5 $\mathbf{Q}(t) = \text{diag}([\mathbf{X}(t) \odot \mathbf{\Pi}(t)] \mathbf{1})$
 - 6 $\mathbf{X}(t+1) = \mathbf{Q}^{-1}(t) [\mathbf{X}(t) \odot \mathbf{\Pi}(t)]$
 - 7 Compute the cross-entropy $\mathcal{J}(\mathbf{X}(T), \mathbf{y})$
 - 8 Compute the derivatives $\partial \mathcal{J} / \partial \theta$ via backpropagation, and update the weights θ
-

have a high inter-class and low intra-class cosine similarity. No further constraints are put on the embedding space by the loss. This results in embeddings that have a low loss only because they have high cosine similarity (point in the same direction) with the weights of the correct class despite having very different lengths.

By analyzing the embedding space, we observe that there is a large variance in the length of the embedding vectors, while the mean embedding length per class is very similar for all the classes. This means that feature vectors in the embedding space are often close due to being of similar length and not because they point in similar directions. Most state-of-the-art metric learning methods, including the original formulation of Group Loss [50], address this issue by applying Euclidean normalization. Using Euclidean normalization on inference assumes that all of the discriminatory information of the embedding vector is stored in its direction. However, by doing so, all the information that the softmax layer stores in the length of the vector is lost. Since longer vectors get classified with more confidence by the softmax loss, the network learned to store discriminatory information in the length of the embeddings too.

In this work, we propose β -Normalization, a weighted normalization in which the information stored in the length of the vector is included but controlled by a parameter β . Using the embedding vector $\phi(\mathbf{I}_i)$, to control the degree of vector length information to the final embedding by modifying the scalar β in the following equation:

$$\phi(\mathbf{I}_i)_\beta = \frac{\phi(\mathbf{I}_i)}{\|\phi(\mathbf{I}_i)\|_2} + \beta \phi(\mathbf{I}_i). \quad (12)$$

One can quickly observe that 0-Normalization is equivalent to euclidean normalization and 1-Normalization barely has any influence on the original embeddings since we observed that the euclidean length of the embedding vectors is orders of magnitude larger than 1. Unlike euclidean normalization, β -normalization enables us to make a compromise on the informativeness of length and direction which, in practice, leads to better performance.

4.2 Mixed pooling

Most metric learning works use Inception [59] with batch-norm [60], ResNets [43] or DenseNets [44]. All these networks use average pooling to transform the final feature volume into the embedding vector. Any pooling operation used to aggregate features will incur a loss of information. In the case of average pooling, the aggregation is performed averaging all the activations which nicely summarizes the global context of the feature map. Nonetheless, this also means that very large or very small values will be diluted in the average operation, potentially causing a loss of important information. One can argue that we can simply replace average pooling with max pooling. However, max-pooling comes with its own issues as it assumes that most of the information is present in the maximum value of a feature map. We propose to use mixed pooling [61], an operation that has been widely used in CNNs (e.g. person Re-ID), but with the exception of [27] has not been used in the field of metric learning. We use both max and average pooling, combining them to capture the global context of average pooling as well as the emphasis on the important information contained in large activation values that comes from max pooling.

4.3 The effect of the final activation functions

Most CNNs use the rectified linear unit (ReLU) as an activation function, which implies all the entries in the final feature volume before the final pooling contain positive values. Since max and average pooling only select and aggregate these positive values, the resulting embeddings are positive for every entry. From a geometric point of view, enforcing positive values heavily constrains the embedding space. For an n -dimensional embedding space, we only use $\frac{1}{2^n}$ of all the available space. To circumvent this issue and leverage the discriminatory information contained in negative activations, we argue that we should replace the final ReLU activation function with a leaky ReLU [62] on inference. This way we can control the influence that negative activations have on the final embedding by varying the slope in the leaky ReLU. Therefore possible discriminatory information contained in negative activation's can be translated into the embedding in a controlled fashion and is not lost.

4.4 Re-ranking

In person re-identification, a commonly used post-processing technique to make results more robust at test time is to perform k-reciprocal re-ranking [63]. The intuition of it is to filter out false matches from the nearest neighbours (NN) of a given query image of class y_i . Furthermore, a gallery sample of class y_i might be contained in the NN of some NN of a query image but not in the NN of query image itself. For example, if a query image is taken from the front, an image taken from diagonally behind might not be contained in the NN of the query itself but an image taken from the side might lie very close to both images in the embedding space. To also take such images into account, the NN of the query are extended.

To this end, we generate the reciprocal nearest neighbour of size k_1 (k_1 -RNN) set of a given query image. This set

only contains the gallery images from the query’s k_1 -nearest neighbour (k_1 -NN) set that also contain the query image in their own k_1 -NN set. Then, we expand the k_1 -RNN set of the query image by $\frac{1}{2}k_1$ -RNN sets of the gallery images that are contained in the query’s k_1 -RNN set and whose $\frac{1}{2}k_1$ -RNN sets are sufficiently similar to the query’s k_1 -RNN sets, i.e., if their intersection is larger than some margin. To ensure that the expanded k_1 -RNN set does not get too large, its size gets restricted to k_2 . Finally, we compute the Jaccard distance between the updated k_1 -RNN sets of the query and gallery images. We use the weighted sum between the initial distance and the Jaccard distance with the weighting parameter λ to rank the gallery images.

While k-reciprocal re-ranking is omnipresent in person re-identification, and has shown impressive performance improvements, it was originally developed in the field of image retrieval [64]. However, in the deep learning era, the vast majority of works do not utilize it. In this work, we apply re-ranking in the task of image retrieval, showing that it consistently improves results, and thus should be a part of the retrieval toolbox.

4.5 Flip inference

We flip every image and feed to the network both the original and the flipped image. We take the two resulting embeddings and average them. This makes the embedding invariant to flips in the input image. Introducing this invariance leads to superior embeddings because the discriminating features that were learned during training are included more optimally in the final embedding. In other words, if we had only seen birds from the left side in training and we encounter birds photographed from the right side during testing, the learned features that are strongly dependent on the side from which we look at the bird have little to no discriminatory power when Flip inference is not used. This intuition is further strengthened by the observation that for the Datasets where the images are fairly symmetric (In-Shop [65], Stanford Online Products [15]) we gain much less than on datasets where the images are less symmetric (CUBS-200-2011 [66], CARS196 [67]).

5 EXPERIMENTAL SETUP

5.1 Implementation details

We use the PyTorch [68] library for the implementation of Group Loss and Group Loss++.

Retrieval. We use GoogleNet [59] with batch-normalization [60] as the backbone feature extraction. We pretrain the network on the *ILSVRC 2012-CLS* dataset [69]. For pre-processing, in order to get a fair comparison, we follow the implementation details of [16]. The inputs are resized to 256×256 pixels, and then randomly cropped to 227×227 . Like other methods except for [14], we use only a center crop during testing time. We train all networks for the classification task for 10 epochs. We then train the network for the Group Loss task for 60 epochs using RAdam optimizer [70]. After 30 epochs, we lower the learning rate by multiplying it by 0.1. Following the works of [22], [71], [72], in order to calibrate the network, we use temperature scaling. We give the β hyperparameter for normalization,

and the hyperparameter for leaky relu in the appendix. We find the hyperparameters using random search [73]. We use small mini-batches of size 30–100. As sampling strategy, for each mini-batch, we first randomly sample a fixed number of classes, and then for each of the chosen classes, we sample a fixed number of samples.

Re-identification. We use DenseNet161 [44] as backbone for the Group Loss. For fair comparison, we use the same pre-processing as in [37]. To be specific, we resize all images to 256×128 , pad the result with 10 zero-valued pixels on each side and, finally, randomly crop the images again to 256×128 . We train each network for 100 epochs and reduce the learning rate by 10 after 30 and 60 epochs using RAdam optimizer [74]. We find all hyperparameters using random search [73]. As commonly done in the field of person re-identification, we use random erasing [75], where a random region of random size is erased from an image. This imitates occlusion, a common challenge in person re-identification, and therefore leads to a more robust performance of the model. We apply label smoothing as proposed in [76] to the auxiliary softmax cross-entropy loss right after the backbone CNN. Finally, inspired by [37], we add a batch normalization layer right before the final fully-connected layer that computes class probability scores. This is to ensure the feature vectors are balanced in all dimensions. Further, the features are normally distributed near the surface which, on the one hand, eases the convergence of the auxiliary softmax cross-entropy loss, and on the other hand, constrains the features belonging to the same person to be more compactly distributed [37]. The latter also impacts the similarity computation of the Group Loss itself. As for the sampling strategy, for each mini-batch, we first sample a fixed number of classes with a low average distance between each other, and then for each class, we sample a fixed number of samples.

5.2 Benchmark datasets

For image retrieval, we perform experiments on four publicly available datasets, evaluating our algorithm on both clustering and retrieval metrics. For training and testing, we follow the conventional splitting procedure [15].

- **CUB-200-2011** [66] is a dataset containing 200 species of birds with 11,788 images, where the first 100 species (5,864 images) are used for training and the remaining 100 species (5,924 images) are used for testing.
- **Cars 196** [67] dataset is composed of 16,185 images belonging to 196 classes. We use the first 98 classes (8,054 images) for training and the other 98 classes (8,131 images) for testing.
- **Stanford Online Products** dataset [15], contains 22,634 classes with 120,053 product images in total, where 11,318 classes (59,551 images) are used for training and the remaining 11,316 classes (60,502 images) are used for testing.
- **In-Shop Clothes** dataset [65] contains 7,982 classes of clothing items and 4 images on average in each class. 3,997 classes are used for training, while the test set is split into a query set and a gallery set from 3,985 classes.

For person re-identification we use the following datasets:

- **CUHK03 New Protocol** [30]: The dataset comprises of 14,096 images of 14,097 identities, where each identity is captured from two different cameras. We use the new protocol [63] where the identities are divided in 767 identities for training and 700 identities for testing.
- **Market1501** [77]: This dataset contains bounding boxes of 1,501 identities. Each identity is captured from at least two different and at most six different cameras. They are split into 751 identities containing 12,936 images and 750 identities containing 19,732 images for training and testing, respectively. All bounding boxes were automatically generated.

5.3 Evaluation metrics

Based on the experimental protocol detailed above, we evaluate the retrieval performance and clustering quality on data from unseen classes of the 4 aforementioned datasets. For the retrieval task, we calculate the percentage of the testing examples whose K nearest neighbors contain at least one example of the same class. This quantity is also known as **Recall@K** [78] and is the most used metric for image retrieval evaluation. The chosen value K follows other works in the literature [15], and typically a larger K is used for datasets that have a higher number of classes. Similar to all other approaches, we perform clustering using K-means algorithm [48] on the embedded features. Like in other works, we evaluate the clustering quality using the Normalized Mutual Information measure (**NMI**) [17]. The choice of NMI measure is motivated by the fact that it is invariant to label permutation, a desirable property for cluster evaluation.

For person re-identification, we use the two commonly used metrics: **CMC** that represents the percentage of query images whose top- k ranked query images contain at least one image of the same identity (similar to Recall@K for retrieval); and **mAP** that represents the mean average precision of all query images.

6 RESULTS IN IMAGE RETRIEVAL

In this section, we will compare our baseline model, namely *Group Loss*, with the model with all inference tricks, *Group Loss++*. Furthermore, we compare our method with In this section, we compare the *Group Loss* with state-of-the-art models on both image retrieval and clustering tasks.

6.1 Comparison to SOTA

In Tab. 1, we present the results of *Group Loss++* and compare with the results of *Group Loss* and those of the other approaches. On the *CUB-200-2011* dataset, our *Group Loss++* outperforms the other approaches by a large margin, with the second-best model, Proxy-Anchor [28] reaching 4.2 percentage points(*pp*) lower absolute accuracy in Recall@1 metric. On NMI metric, *Group Loss++* achieves a score of 71.9 which is 0.6*pp* better than previous SOTA, the Proxy-NCA++ [27]. Compared to *Group Loss* [50] we reach 7.1*pp* higher accuracy in Recall@1 and 2.9*pp* higher score in NMI



Fig. 3. The effect of the number of refinement steps. 0 represents the case where only cross-entropy is used, so there are no refinement steps.

metric. Similarly, on *Cars 196*, *Group Loss++* achieves the best results on Recall@1, with Proxy-Anchor [28] coming second with a 4.3*pp* lower score in Recall@1. We improve compared to *Group Loss* by 4.8*pp* in Recall@1. On the same dataset, we reach the highest results in NMI metric, a 1.5*pp* improvement over Proxy-GML method [29]. On *Stanford Online Products*, *Group Loss++* reaches competitive results in Recall@1, 0.9*pp* worse than HORDE (which uses larger images), improving over *Group Loss* by 4*pp*. On the same dataset, when evaluated on the NMI score, our *Group Loss++* outperforms all the other methods. Finally, on *In-Shop Clothes*, *Group Loss++* reaches the second best results when evaluated in Recall@1, 0.6*pp* behind Proxy-Anchor, but improving over *Group Loss* by 4.1*pp*.

6.2 The effect of the refinement procedure

We validate our *Group Loss* method, by checking the effect of the refinement procedure, and its stability with respect to the number of refinement steps. We show the results in Fig. 3. In *CUB-200-2011* dataset, we only see a marginal improvement compared to cross-entropy loss, with the best results being when we use only one refinement step, reaching 0.6*pp* better than when using cross-entropy. However, on the three other datasets, we reach significantly better results than cross-entropy, showing that the refinement procedure is the key to the success of *Group Loss*. More precisely, in *Cars 196* dataset, we improve over cross-entropy by up to 3.6*pp* where we use four refinement steps; in *Stanford Online Products*, we improve by 8.9*pp* where we use four refinement steps, and in *In-Shop Clothes* we improve by up to 7.3 – 7.4*pp* where we use three, or four refinement steps.

An interesting observation, is that the method performs best where there is a larger number of refinement steps (three or four), but the performance degrades where the number of refinement steps reaches five. The reasoning behind this is that for each image to be influenced not only by the images which are very similar to it (neighbours in context of nodes in the graph), but also by the images that are similar to its neighbors, it needs more than one refinement iteration. On the other hand, too many iterations, seem to make the performance worse, because every embedding starts becoming similar to each other. If too many iterations are done, every node starts effecting every other node, which is analogous to the problem of oversmoothing [86] in graph neural networks [87].

Method	BB	CUB-200-2011					CARS196					Stanford Online Products			
		R@1	R@2	R@4	R@8	NMI	R@1	R@2	R@4	R@8	NMI	R@1	R@10	R@100	NMI
Triplet [6] CVPR15	G	42.5	55	66.4	77.2	55.3	51.5	63.8	73.5	82.4	53.4	66.7	82.4	91.9	89.5
Npairs [14] NeurIPS16	G	51.9	64.3	74.9	83.2	60.2	68.9	78.9	85.8	90.9	62.7	66.4	82.9	92.1	87.9
Deep Spectral [19] ICML17	BNI	53.2	66.1	76.7	85.2	59.2	73.1	82.2	89.0	93.0	64.3	67.6	83.7	93.3	89.4
Angular Loss [18] ICCV17	G	54.7	66.3	76	83.9	61.1	71.4	81.4	87.5	92.1	63.2	70.9	85.0	93.5	88.6
Proxy-NCA [26] ICCV17	BNI	49.2	61.9	67.9	72.4	59.5	73.2	82.4	86.4	88.7	64.9	73.7	-	-	90.6
Margin Loss [4] ICCV17	R50	63.6	74.4	83.1	90.0	69.0	79.6	86.5	91.9	95.1	69.1	72.7	86.2	93.8	90.7
Hierarchical triplet [33] ECCV18	BNI	57.1	68.8	78.7	86.5	-	81.4	88.0	92.7	95.7	-	74.8	88.3	94.8	-
ABE [79] ECCV18	G	60.6	71.5	79.8	87.4	-	85.2	90.5	94.0	96.1	-	76.3	88.4	94.8	-
Normalized Softmax [22] BMVC19	R50	59.6	72.0	81.2	88.4	66.2	81.7	88.9	93.4	96.0	70.5	73.8	88.1	95	89.8
RLH [10] CVPR19	BNI	57.4	69.7	79.2	86.9	63.6	74	83.6	90.1	94.1	65.4	76.1	89.1	95.4	89.7
Multi-similarity [49] CVPR19	BNI	65.7	77.0	86.3	91.2	-	84.1	90.4	94.0	96.5	-	78.2	90.5	96.0	-
Relational Knowledge [80] CVPR19	G	61.4	73.0	81.9	89.0	-	82.3	89.8	94.2	96.6	-	75.1	88.3	95.2	-
Divide and Conquer [81] CVPR19	R50	65.9	76.6	84.4	90.6	69.6	84.6	90.7	94.1	96.5	70.3	75.9	88.4	94.9	90.2
SoftTriple Loss [25] ICCV19	BNI	65.4	76.4	84.5	90.4	69.3	84.5	90.7	94.5	96.9	70.1	78.3	90.3	95.9	92.0
HORDE [82] ICCV19	BNI	66.3	76.7	84.7	90.6	-	83.9	90.3	94.1	96.3	-	80.1	91.3	96.2	-
MIC [83] ICCV19	R50	66.1	76.8	85.6	-	69.7	82.6	89.1	93.2	-	68.4	77.2	89.4	95.6	90.0
Easy triplet mining [84] WACV20	R50	64.9	75.3	83.5	-	-	82.7	89.3	93.0	-	-	78.3	90.7	96.3	-
Proxy NCA++ [27] ECCV20	R50	66.3	77.8	87.7	91.3	71.3	84.9	90.6	94.9	97.2	71.5	79.8	91.4	96.4	-
Proxy Anchor [28] CVPR20	BNI	68.4	79.2	86.8	91.6	-	86.1	91.7	95.0	97.3	-	79.1	90.8	96.2	-
Proxy-GML [29] NeurIPS20	BNI	66.6	77.6	86.4	-	69.8	85.5	91.8	95.3	-	72.4	78.0	90.6	96.2	90.2
Group Loss [50] ECCV20	BNI	65.5	77.0	85.0	91.3	69.0	85.6	91.2	94.9	97.0	72.7	75.1	87.5	94.2	90.8
Group Loss++	BNI	72.6	80.5	86.2	91.2	71.9	90.4	93.8	96.0	97.5	73.9	79.2	90.1	95.8	92.0

TABLE 1

Retrieval and Clustering performance on CUB-200-2011, CARS196 and Stanford Online Products datasets. Bold indicates best, red second best, and blue third best results. BB=backbone, G=GoogLeNet, BNI=BN-Inception, and R50=ResNet50.

Method	BB	R@1	R@10	R@20	R@40
FashionNet [65] CVPR16	V	53.0	73.0	76.0	79.0
A-BIER [85] PAMI20	G	83.1	95.1	96.9	97.8
ABE [79] ECCV18	G	87.3	96.7	97.9	98.5
Multi-similarity [49] CVPR19	BNI	89.7	97.9	98.5	99.1
Learning to Rank [8]	R50	90.9	97.7	98.5	98.9
HORDE [82] ICCV19	BNI	90.4	97.8	98.4	98.9
MIC [83] ICCV19	R50	88.2	97.0	98.0	98.8
Proxy NCA++ [27] ECCV20	R50	90.4	98.1	98.8	99.2
Proxy Anchor [28] CVPR20	BNI	91.5	98.1	98.8	99.1
GroupLoss [50] ECCV20	BNI	86.8	96.4	97.5	98.4
GroupLoss++	BNI	90.9	97.6	98.4	98.9

TABLE 2

Retrieval performance on In Shop Clothes. Bold indicates best, red second best, and blue third best results. BB=backbone, G=GoogLeNet, BNI=BN-Inception, V=VGG16, and R50=ResNet50

Inference	CUB-200-2011		CARS196		SOP		In-Shop
	R@1	NMI	R@1	NMI	R@1	NMI	R@1
GL	65.5	69.0	85.6	72.7	75.1	90.8	86.8
mixed	67.5	69.5	88.2	72.9	78.1	91.2	89.1
β	66.8	69.0	87.1	72.2	75.9	91.3	87.1
L	61.3	66.6	85.6	72.7	76.8	91.5	87.6
R	68.3	69.1	87.4	71.8	75.7		87.8
F	68.0	70.9	88.1	73.2	76.1	91.3	87.4
GL++	72.6	71.9	90.4	73.9	79.2	92.0	90.9

TABLE 3

The influence of different inference configurations on isolation. The last row represents the Results of Group Loss++. mixed=mixed pooling, β = β -normalization, L=LeakyReLU, R=re-ranking, F=Flip inference

6.3 Inference ablation study

We now validate the inference choices described in the previous section. Our baseline is the Group Loss [50], where we do not use any inference strategies, and that is denoted in Tab. 3-4 as GL. In Tab. 3 we present the individual effect of each inference strategy. We show that replacing average pooling with mixed pooling, gives an improvement of 2 – 3pp in the four datasets. We also see that beta-normalization in isolation gives a significant improvement, as do the re-ranking and inference flipping. On the other hand, replacing relu activation function with leaky-relu gives a moderate improvement in all datasets except CUB, where it reduces the performance by 4.2pp.

Inference	CUB-200-2011		CARS196		SOP		In-Shop
	R@1	NMI	R@1	NMI	R@1	NMI	R@1
GL	65.5	69.0	85.6	72.7	75.1	90.8	86.8
mixed	67.5	69.3	88.5	72.4	78.1	91.4	89.1
mixed + β	68.0	70.2	88.5	72.8	78.0	91.8	88.9
mixed + β + L	68.8	69.6	88.6	71.7	78.5	92.0	89.5
mixed + β + L + R	70.2	70.2	89.3	72.1	78.9	92.0	90.4
mixed + β + L + R + F	72.6	71.9	90.4	73.9	79.2	92.0	90.9

TABLE 4

The influence of different inference configurations, cumulatively. The last row represents the Results of Group Loss++. mixed=mixed pooling, β = β -normalization, L=LeakyReLU, R=re-ranking, F=Flip inference

We present the cumulative effect of the inference strategies in Tab. 4. We see that adding each module comes with a significant improvement, showing that the strategies are complementary, hence we get the best results when we combine all of them.

6.4 Implicit regularization and less overfitting.

In Figures 4 and 5, we compare the results of training vs. testing on Cars 196 [67] and Stanford Online Products [15] datasets. We see that the difference between Recall@1 at train and test time is small, especially on Stanford Online Products dataset. On Cars 196 the best results we get for the training set are circa 94% in the Recall@1 measure, only 7.5 percentage points (pp) higher than what we reach with Group Loss in the testing set (4pp higher than with the Group Loss++). From the works we compared with, the only one which reports the results on the training set is [19]. They reported results of over 90% in all three datasets (for the training sets), much above the test set accuracy which lies at 73.1% on Cars 196 and 67.6% on Stanford Online Products dataset. Unfortunately, they do not show the results after each epoch. [88] also provides results, but it uses a different network thus making the comparison impossible.

We further implement the P-NCA [26] loss function and perform a similar experiment, in order to be able to compare training and test accuracy directly with our method. In Figure 4, we show the training and testing curves of P-NCA on the Cars 196 [67] dataset. We see that while in the training set, P-NCA reaches higher results than our method,

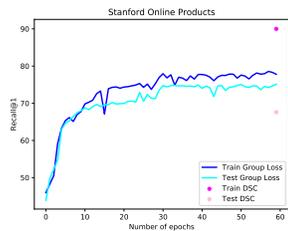
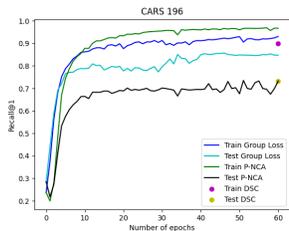


Fig. 4. Group Loss: Training vs test- Recall@1 curves on *Cars 196* dataset. Fig. 5. Group Loss: Training vs test- Recall@1 curves on *Stanford Online Products* dataset.

in the testing set, our method outperforms P-NCA by almost 20pp. Unfortunately, we were unable to reproduce the results of the paper [26] on *Stanford Online Products* dataset. Furthermore, even when we turn off L_2 -regularization, the generalization performance of our method does not drop at all. Our intuition is that by taking into account the structure of the entire manifold of the dataset, our method introduces a form of regularization. We can clearly see a smaller gap between training and test results when compared to competing methods, indicating less overfitting.

7 RESULTS IN PERSON RE-IDENTIFICATION

Despite that the field of person re-identification is related to image retrieval, they have been studied separately. In this work, we provide a unified framework for image retrieval and person re-identification showing that our method works well for both problems.

7.1 Comparison to SOTA

We compare the results of our method with state-of-the-art approaches on three re-identification datasets. In contrast to our approach, most of the approaches we compare to either take person parts into account [36], [38], [40], [89], [90], [91], [92] or utilize features at different stages of the network [93], [94], [95], [95], [96] to get more fine-grained features.

As can be seen in Table 5, we report the best results in mAP score, reaching 1pp higher score than the previous SOTA on the CUHK03 dataset. In the rank-1 score, we reach the second-best results. We reach competitive results on the Market-1501 dataset, reaching 0.9pp worse results in rank-1 metric. Note, we reach these results without any specific adaption of the architecture on the person re-identification datasets, compared to methods like [95] that uses three training stages.

8 DISCUSSION AND LIMITATIONS

8.1 The connections with Graph Neural Networks

Our method is connected to graph neural networks [87], [107], however, an advantage of our label-propagator is that it has guaranteed convergence properties, and eventually reaches a stationary point (Eq. 10, 11). Closest to this work, is our recently published method [108] where the samples exchange feature preferences (instead of label preferences) based on a message-passing network [109] implemented via graph attention networks [110] which can be seen as

Method	BB	CUHK03	
		rank-1	mAP
HA-Net [97] CVPR18	HA-CNN	41.7	38.6
MLFN [98] CVPR18	MLFN	52.8	47.8
CAMA [89] CVPR19	R50	64.2	66.6
MHN [99] ICCV19	R50	71.1	65.4
OSNet [100] ICCV19	OSNet	72.3	67.8
Auto-ReID [†] [36] ICCV19	Auto-ReID	73.3	69.9
ISP [101] ECCV20	HRNet-W32	75.2	71.4
BAT-net [90] ICCV19	BNI	76.2	73.2
BDB [38] ICCV19	R50	76.4	73.5
RGA-SC [94] CVPR20	R50	79.6	74.5
PyrNet [†] [91] CVPR19	DN201	80.8	82.7
SCSN (3 stages) [95] CVPR20	R50	84.1	80.2
Group Loss	DN161	62.0	58.4
Group Loss++	DN161	82.5	83.7

TABLE 5

Comparison to state-of-the-art results on the new test protocol and the detected bounding boxes of CUHK03 dataset [63]. Bold indicates best, red second best, and blue third best results. BB=backbone, DN201=DenseNet-201, DN161=DenseNet-161, BNI=BN-Inception and R50=ResNet50. All other backbones are specifically designed to person re-identification. † means that either k-reciprocal re-ranking or some other re-ranking approach is used.

Method	Market-1501		
	BB	rank-1	mAP
TriNet [†] [35]	R50	91.8	87.2
SGGNN [†] [102] ECCV18	R50	92.3	82.2
DGSRW [†] [103] CVPR18	R50	92.7	82.5
GCSL [96] CVPR18	R50	93.5	81.6
HA-Net [97] CVPR18	HA-CNN	93.8	82.2
IANet [104] CVPR19	R50	94.4	83.1
HSP [†] [76] CVPR18	BNI	94.6	90.9
CAMA [89] CVPR19	R50	94.7	84.5
OSNet [100] ICCV19	OSNet	94.8	84.9
DG-Net [105] CVPR19	R50	94.8	86.0
PCB [40] ECCV18	R50	95.1	91.9
BAT-net [90] ICCV19	BNI	95.1	87.4
MHN [99] ICCV19	R50	95.1	85.0
BDB [38] ICCV19	R50	95.3	86.7
ISP [101] ECCV20	HRNet-W32	95.3	88.6
Auto-ReID [†] [36] ICCV19	AutoReID	95.4	94.2
BoT [†] [37] CVPR19	R50	95.4	94.2
DCDS [†] [46] ICCV19	R101	95.4	93.3
ABDNet [106] ICCV19	R50	95.6	88.3
SCSN (3 stages) [95] CVPR20	R50	95.7	88.5
PyrNet [†] [91] CVPR19	DN201	96.1	94.0
RGA-SC [94] CVPR20	R50	96.1	88.4
Group Loss	DN161	93.7	78.3
Group Loss++	DN161	95.2	92.1

TABLE 6

Comparison to state-of-the-art results on Market-1501 [77]. Bold indicates best, red second best, and blue third best results.

BB=backbone, DN201=DenseNet-201, DN161=DenseNet-161, BNI=BN-Inception and R50=ResNet50. All other backbones are specifically designed to person re-identification. † means that either k-reciprocal re-ranking or some other re-ranking approach is used.

Transformers [111]. Considering that graph neural networks have been very successful in the field of semi-supervised learning, it would be interesting to extend our method towards semi-supervised tasks.

8.2 Dealing with Relative Labels

The proposed method assumes that the dataset consists of classes with (absolute) labels and a set of samples in each class. This is the case for the datasets used in metric learning [15], [66], [67] technique evaluations. However, deep metric learning can be applied to more general problems where the absolute class label is not available but a relative label is available. For example, the data might be given as pairs that are similar or dissimilar. Similarly, the data may be given as triplets consisting of anchor (A), positive (P) and negative (N) images, such that A is semantically closer to P than N. For example, in [112] a triplet network has been used to learn good visual representation where only relative labels are used as self-supervision (two tracked patches from the same video form a “similar” pair and the patch in the first frame and a patch sampled from another random video forms a “dissimilar” pair). Similarly, in [113], relative labels are used as self-supervision for learning good spatio-temporal representation.

Our method, similar to other classification-based losses [22], [25], [26] for deep metric learning, or triple loss improvements like Hierarchical Triplet Loss [33] assumes the presence of absolute labels. Unlike traditional losses [1], [6], in cases where only relative labels are present, all the mentioned methods do not work. However, in the presence of both relative and absolute labels, then in our method, we could use relative labels to initialize the matrix of similarities, potentially further improving the performance of networks trained with The Group Loss.

8.3 Dealing with a Large Number of Classes

In all classification-based methods, the number of outputs in the last layer linearly increases with the number of classes in the dataset. This can become a problem for learning on a dataset with a large number of classes (say $N > 1000000$) where metric learning methods like pairwise/triplet losses/methods can still work. In our method, the similarity matrix is square on the number of samples per mini-batch, so its dimensions are the same regardless if there are 5 or 5 million classes. However, the matrix of probabilities is linear in the number of classes. If the number of classes grows, computing the softmax probabilities and the iterative process becomes indeed computationally expensive. An alternative, which reaches similar results, is to sparsify the matrix. Considering that in any mini-batch, we use only a small number of classes (< 10), all the entries not belonging to these classes may be safely set to 0, followed by a normalization of the probability matrix. This allows both saving storage (e.g. using sparse tensors in PyTorch) and an efficient tensor-tensor multiplication. It also needs to be said, that in retrieval, the number of classes is typically not very large, and many other methods [22], [25], [26], [33] face the same problem. However, in related fields, e.g., face recognition, there could be millions of classes (identities), in which case we can use the proposed solution.

9 CONCLUSION

In this work, we proposed the Group Loss, a new loss function for deep metric learning that goes beyond triplets. By considering the content of a mini-batch, it promotes embedding similarity across all samples of the same class, while enforcing dissimilarity for elements of different classes. This is achieved with a fully-differentiable layer that is used to train a CNN in an end-to-end fashion. We then propose the Group Loss++, by modifying the original formulation to include inference strategies that significantly increase the performance of our model without any training changes. Finally, we show that our method can be used without changes for the task of person re-identification reaching competitive results. Our method reaches state-of-the-art results in several metric learning datasets.

ACKNOWLEDGMENTS

This research was partially funded by the Humboldt Foundation through the Sofja Kovalevskaja Award and a Humboldt Research Fellowship.

dataset	β	LeakyReLU	α	λ	k_1	k_2
CUB-200-2011	0.004	0.75	0.5	0.85	50	20
Cars196	0.004	0.4	0.6	0.9	50	20
In-Shop	0.002	1.0	0.9	0.7	4	2
Stanford Online Products	0.0005	1.0	0.6	0.94	50	20

TABLE 7

Hyperparameters of inference strategies for all four datasets.

APPENDIX

MORE IMPLEMENTATION DETAILS

We first pre-train all networks in the classification task for 10 epochs. We then train our networks on all three datasets [15], [66], [67] for 60 epochs. During training, we use a simple learning rate scheduling in which we divide the learning rate by 10 after the first 30 epochs.

We find all training hyperparameters using random search [73]. For the weight decay ($L2$ -regularization) parameter, we search over the interval $[0.1, 10^{-16}]$, while for learning rate we search over the interval $[0.1, 10^{-5}]$, choosing 0.0002 as the learning rate for all networks and all datasets.

We achieve the best results with a regularization parameter set to 10^{-6} for *CUB-200-2011*, 10^{-7} for *Cars 196* dataset, and 10^{-12} for *Stanford Online Products* dataset. This further strengthens our intuition that the method is implicitly regularized and it does not require strong regularization.

Apart from the training hyperparameters, we find the inference hyperparameters by manual search for each dataset individually as the hyperparameters are highly dependent on each other. The final values can be found in Tab. 7.

ROBUSTNESS ANALYSIS WITH RESPECT TO HYPERPARAMETERS

Number of anchors. In Fig. 6 and 7, we show the effect of the number of anchors with respect to the number of samples per class. We do the analysis on *CUB-200-2011* [66] and *Cars 196* [67] datasets. The results reported are the percentage point differences in terms of Recall@1 with respect to the best performing set of parameters. The number of anchors ranges from 0 to 4, while the number of samples per class varies from 5 to 10. It is worth noting that on *CUB-200-2011* [66], our best setting considers 1 or 2 anchors over 9 samples. Moreover, even when we do not use any anchor, the difference in Recall@1 is no more than 2pp. Similarly, the method shows the same robustness as on *Cars 196* [67], with the best result being only 2.1 percentage points better at the Recall@1 metric than the worst result. Note that the results decrease mainly when we do not have any labeled sample, i.e., when we use zero anchors.

Number of classes per mini-batch. In Fig. 8, we present the change in Recall@1 on the *CUB-200-2011* [66] dataset if we increase the number of classes we sample at each iteration. The best results are reached when the number of classes is not too large. This is a welcome property, as we can train on small mini-batches, known to achieve better generalization performance [114].

SENSITIVITY WITH RESPECT TO INFERENCE HYPERPARAMETERS

In this section, we analyze the sensitivity of the performance of GL++ with respect to the inference hyperparameters. We

Relative difference w.r.t. Best Recall@1

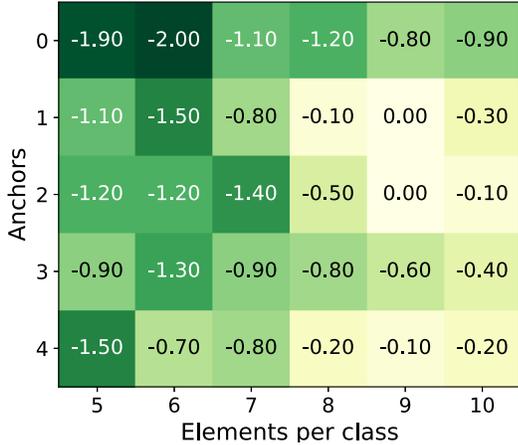


Fig. 6. The effect of the number of anchors and the number of samples per class in CUB-200-2011.

Relative difference w.r.t. Best Recall@1

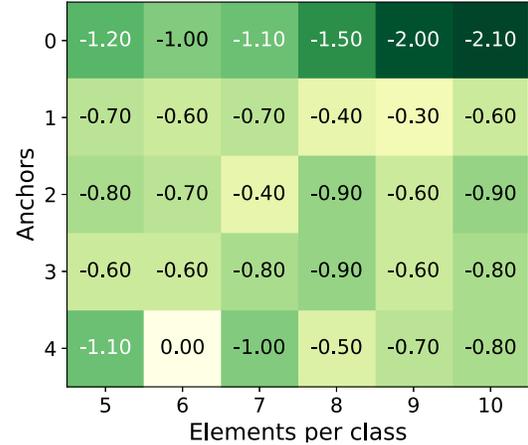


Fig. 7. The effect of the number of anchors and the number of samples per class in Cars 196.

Effect of the number of classes per batch

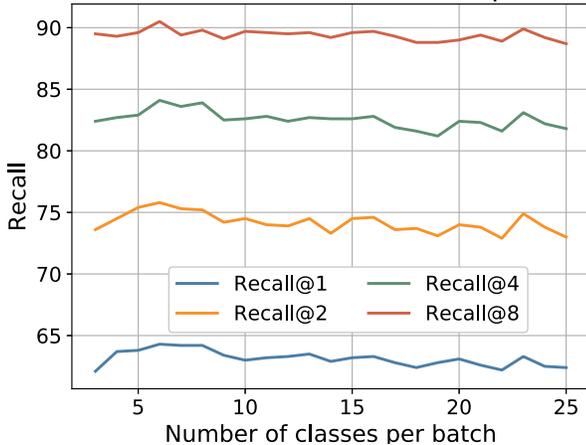


Fig. 8. The effect of the number of classes per mini-batch.

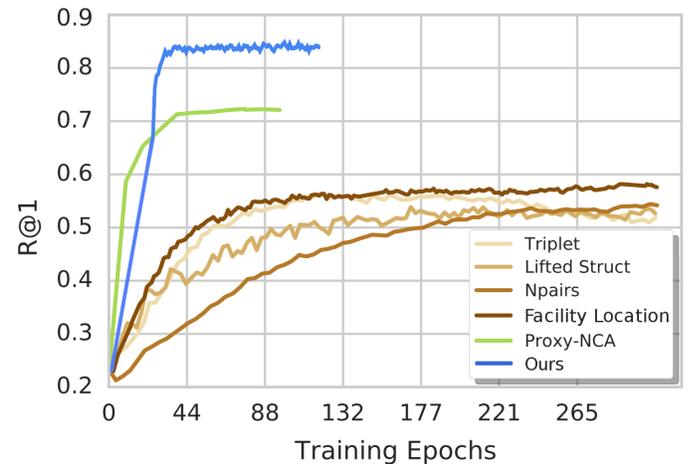


Fig. 9. Recall@1 as a function of training epochs on Cars 196 dataset. Figure adapted from [26].

do the analysis on the smaller *CUB-200-2011* and *Cars 196* datasets.

In Fig. 10, we show that the performance on *CUB-200-2011* is way more sensitive towards different β -values than the performance on *Cars 196* mirroring the difference in the datasets. While the same holds for the negative slope of the LeakyReLU (see Fig. 11), different combinations of maximum pooling and average pooling are similarly sensitive for both datasets (see Fig. 12) where the α -value represents the weighting between max and average pooling:

$$f_{flat} = \alpha \maxpool(f) + (1 - \alpha) \text{avgpool}(f) \quad (13)$$

$f \in \mathcal{R}^{C \times H \times W}$ is a feature map with C channels of height H and width W , \maxpool and avgpool represent maximum pooling and average pooling, and $f_{flat} \in \mathcal{R}^C$ is the flattened feature vector. As can be seen from the sensitivity analysis of the pooling operation, only very large and very low α -values lead to a more significant performance drop.

We also investigate the three re-ranking hyperparameters k_1 , k_2 , and λ with respect to the performance sensitivity. As can be seen from Fig. 13, the performance gets worse for large and small λ -values while being highly stable for values between 0.85 and 0.95 for both datasets. As can be seen from Fig. 14, the performance on *Cars 196* is hardly influenced by the size of the k_1 -NN set while on *CUB-200-2011*, we see a slight decrease in performance. Similarly, for k_2 the performance on *CUB-200-2011* is more influenced than the performance on *Cars* dataset (see Fig. 15).

Overall, the performance is highly robust towards the choice of inference hyperparameters such that the performance only drops drastically for extreme parameter choices.

CONVERGENCE RATE.

In Fig. 9, we present the convergence rate of the model on the *Cars 196* dataset. Within the first 30 epochs, our model reaches higher results than the other models, making our model significantly faster than other approaches. The other

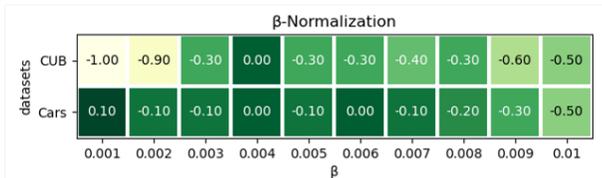


Fig. 10. Sensitivity of R@1 with respect to β -normalization (equidistant).

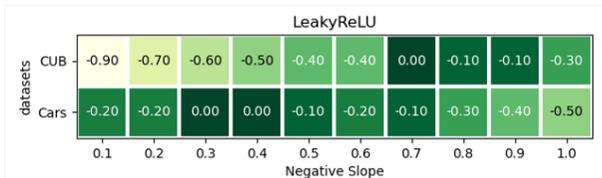


Fig. 11. Sensitivity of R@1 with respect to negative slope of LeakyReLU.

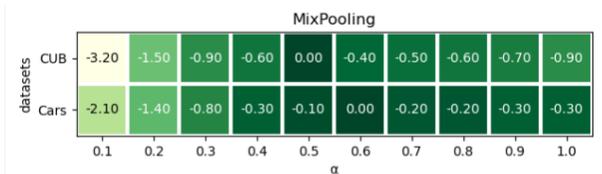


Fig. 12. Sensitivity of R@1 with respect to different combinations of average and maximum pooling.

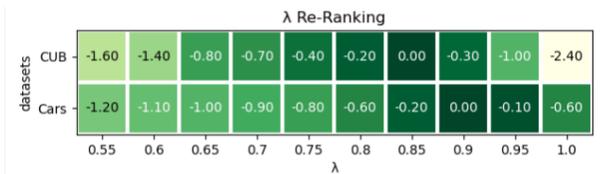


Fig. 13. Sensitivity of R@1 with respect to different λ -values of re-ranking.

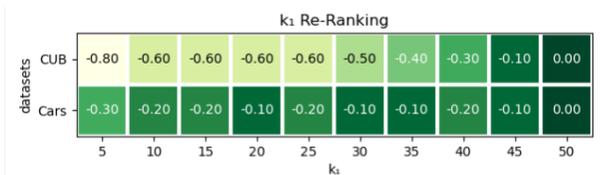


Fig. 14. Sensitivity of R@1 with respect to different k_1 -values of re-ranking.

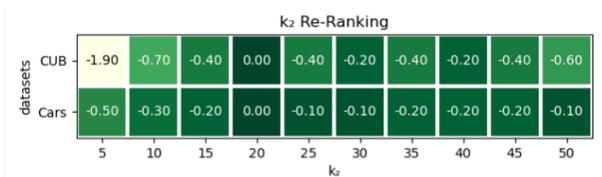


Fig. 15. Sensitivity of R@1 with respect to different k_2 -values of re-ranking.

models except Proxy-NCA [26], need hundreds of epochs to converge. Additionally, we compare the training time with Proxy-NCA [26]. On a single Volta V100 GPU, the average running time of our method per epoch is 23.59 seconds on *CUB-200-2011* and 39.35 seconds on *Cars 196*, compared to 27.43 and 42.56 seconds of Proxy-NCA [26]. Hence, our method is faster than one of the fastest methods in the literature. Note, the inference time of every method is the same because the network is only used for feature embedding extraction.

DEALING WITH NEGATIVE SIMILARITIES

Equation (6) in the main paper assumes that the matrix of similarity is non-negative. However, for similarity computation, we use a correlation metric (see Equation (3) in the main paper) which produces values in the range $[-1, 1]$. In similar situations, different authors propose different methods to deal with the negative outputs. The most common approach is to shift the matrix of similarity towards the positive regime by subtracting the biggest negative value from every entry in the matrix [12]. Nonetheless, this shift has a side effect: If a sample of class k_1 has very low similarities to the elements of a large group of samples of class k_2 , these similarity values, which after being shifted are all positive, will be summed up. If the cardinality of class k_2 is very large, then summing up all these small values leads to a large value, and consequently affects the solution of the algorithm. What we want instead, is to ignore these negative similarities, hence we propose *clamping*. More concretely, we use a *ReLU* activation function over the output of Equation (1).

We compare the results of shifting vs clamping using the Group Loss (GL). On the *Cars 196* dataset, we do not see a significant difference between the two approaches. However, on the *CUB-200-2011* dataset, the Recall@1 metric is 51 with shifting, much below the 65.5 obtained when using clamping. We investigate the matrix of similarities for the two datasets, and we see that the number of entries with negative values for the *CUB-200-2011* dataset is higher than for the *Cars 196* dataset. This explains the difference in behavior, and also verifies our hypothesis that clamping is a better strategy to use within *Group Loss*.

OTHER BACKBONES

In the main paper, we perform all experiments using a GoogleNet backbone with batch normalization. This choice is motivated by the fact that most methods use this backbone, making comparisons fair. In this section, we explore the performance of our method for other backbone architectures, to show the generality of our proposed loss formulation. We choose to train a few networks from Densenet family [44]. Densenets are a modern CNN architecture that show similar classification accuracy to BN-Inception in most tasks (so they are a similarly strong classification baseline¹). Furthermore, by training multiple networks of the same family, we can study the effect of the capacity of the network, i.e., how much can we gain from using a larger network? Finally, we are interested in studying if the choice of hyperparameters can be transferred from one backbone to another.

We present the results of our method using Densenet backbones in Tab. 8. We use the same hyperparameters as the ones used for the BN-Inception experiments, reaching state-of-the-art results on both *Cars 196* [67] and *Stanford Online Products* [15] datasets, even compared to ensemble and sampling methods. The results in *Stanford Online Products* [15] are particularly impressive considering that Group Loss

1. The classification accuracy of different backbones can be found in the following link: <https://pytorch.org/docs/stable/torchvision/models.html>. BN-Inception's top 1/top 5 error is 7.8%/25.2%, very similar to those of Densenet121 (7.8%/25.4%).

was the first time any method in the literature has broken the 80 point barrier in Recall@1 metric. We also reach state-of-the-art results on the *CUB-200-2011* [66] dataset when we consider only methods that do not use ensembles (with the Group Loss ensemble reaching the highest results in this dataset). We observe a clear trend when increasing the number of parameters (weights), with the best results on both *Cars 196* [67] and *Stanford Online Products* [15] datasets being achieved by the largest network, Densenet161 (whom has a lower number of convolutional layers than Densenet169 and Densenet201, but it has a higher number of weights/parameters).

Finally, we study the effects of hyperparameter optimization using the Group Loss (GL). Despite that the networks reached state-of-the-art results even without any hyperparameter tuning, we expect a minimum amount of hyperparameter tuning to help. To this end, we used random search [73] to optimize the hyperparameters of our best network on the *Cars 196* [67] dataset. We reach a 90.7 score (2pp higher score than the network with default hyperparameters) in Recall@1, and 77.6 score (3pp higher score than the network with default hyperparameters) in NMI metric, showing that individual hyperparameter optimization can boost the performance. The score of 90.7 in Recall@1 is not only by far the highest score ever achieved, but also the first time any method has broken the 90 point barrier in Recall@1 metric when evaluated on the *Cars 196* [67] dataset.

ENSEMBLES

We do an experiment using the Group Loss (GL), where we build an ensemble of networks trained by our model. Unlike other methods in the literature, we simply train independently k networks and then concatenate their features during inference. In Table 9 we present the results of our ensemble, and compare them with the results of other ensemble approaches. Our ensemble method (using 5 neural networks) is the highest performing model in *CUB-200-2011*, outperforming the second-best method (Divide and Conquer [81]) by 1pp in Recall@1 and by 0.4pp in NMI. In *Cars 196* our method outperforms the second best method (ABE 8 [79]) by 2.8pp in Recall@1. The second best method in NMI metric is the ensemble version of RLL [10] which gets outperformed by 2.4pp from the Group Loss. In *Stanford Online Products*, our ensemble reaches the third-highest result on the Recall@1 metric (after RLL [10] and GPW [49]) while increasing the gap with the other methods in NMI metric.

TEMPERATURE SCALING

We mentioned in the main paper that as input to the Group Loss (step 3 of the algorithm) we initialize the matrix of priors $X(0)$ from the softmax layer of the neural network. Following the works of [22], [71], [72], we apply a sharpening function to reduce the entropy of the softmax distribution. We use the common approach of adjusting the *temperature* of this categorical distribution, known as temperature scaling. Intuitively, this procedure calibrates our network and in turn, provides a more informative prior to the dynamical system. Additionally, this calibration allows the dynamical

system to be more effective in adjusting the predictions, i.e, it is easier to change the probability of a class if its initial value is 0.6 rather than 0.95. The function is implemented using the following equation:

$$T_{softmax}(z_i) = \frac{e^{z_i/T}}{\sum_i e^{z_i/T}}, \quad (14)$$

which can be efficiently implemented by simply dividing the prediction logits by a constant T .

Recent works in supervised learning [71] and semi-supervised learning [72] have found that temperature calibration improves the accuracy for the image classification task. We arrive at similar conclusions for the task of metric learning, obtaining 2.5pp better Recall@1 scores on *CUB-200-2011* [66] and 2pp better scores on *Cars 196* [67]. Note, the methods of Table 1 (main paper) that use a classification loss, use also temperature scaling.

QUALITATIVE RESULTS

In Fig. 16 we present qualitative results on the retrieval task in *CUB-200-2011*, *Cars 196* and *Stanford Online Products* datasets. In all cases, the query image is given on the left, with the four nearest neighbors given on the right. Green boxes indicate the cases where the retrieved image is of the same class as the query image, and red boxes indicate a different class. As we can see, our model is able to perform well even in cases where the images suffer from occlusion and rotation. On the *Cars 196* dataset, we see a successful retrieval even when the query image is taken indoors and the retrieved image outdoors, and vice-versa. The first example of *Cars 196* dataset is of particular interest. Despite that the query image contains 2 cars, its four nearest neighbors have the same class as the query image, showing the robustness of the algorithm to uncommon input image configurations.

In Fig. 17, we visualize the t-distributed stochastic neighbor embedding (t-SNE) [117] of the embedding vectors obtained by our method on the *CUB-200-2011* [66] dataset. The plot is best viewed on a high-resolution monitor when zoomed in. We highlight several representative groups by enlarging the corresponding regions in the corners. Despite the large pose and appearance variation, our method efficiently generates a compact feature mapping that preserves semantic similarity.

REFERENCES

- [1] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," in *NIPS*, 1994.
- [2] M. Schultz and T. Joachims, "Learning a distance metric from relative comparisons," in *NIPS*, 2003.
- [3] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *JMLR*, 2009.
- [4] R. Manmatha, C. Wu, A. J. Smola, and P. Krähenbühl, "Sampling matters in deep embedding learning," in *ICCV*, 2017.
- [5] X. Zhang, F. Zhou, Y. Lin, and S. Zhang, "Embedding label structures for fine-grained feature representation," in *CVPR*, 2016.
- [6] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015.
- [7] J. Revaud, J. Almazán, R. S. Rezende, and C. R. de Souza, "Learning with average precision: Training image retrieval with a listwise loss," in *ICCV*, 2019.
- [8] F. Çakır, K. He, X. Xia, B. Kulis, and S. Sclaroff, "Deep metric learning to rank," in *CVPR*, 2019.

Model	CUB			CARS			SOP		
	Params	R@1	NMI	Params	R@1	NMI	Params	R@1	NMI
GL Densenet121	7056356	65.5	69.4	7054306	88.1	74.2	18554806	78.2	91.5
GL Densenet161	26692900	64.7	68.7	26688482	88.7	74.6	51473462	80.3	92.3
GL Densenet169	12650980	65.4	69.5	12647650	88.4	75.2	31328950	79.4	92.0
GL Densenet201	18285028	63.7	68.4	18281186	88.6	75.8	39834806	79.8	92.1
GL Inception v2	10845216	65.5	69.0	10846240	85.6	72.7	16589856	75.7	91.1

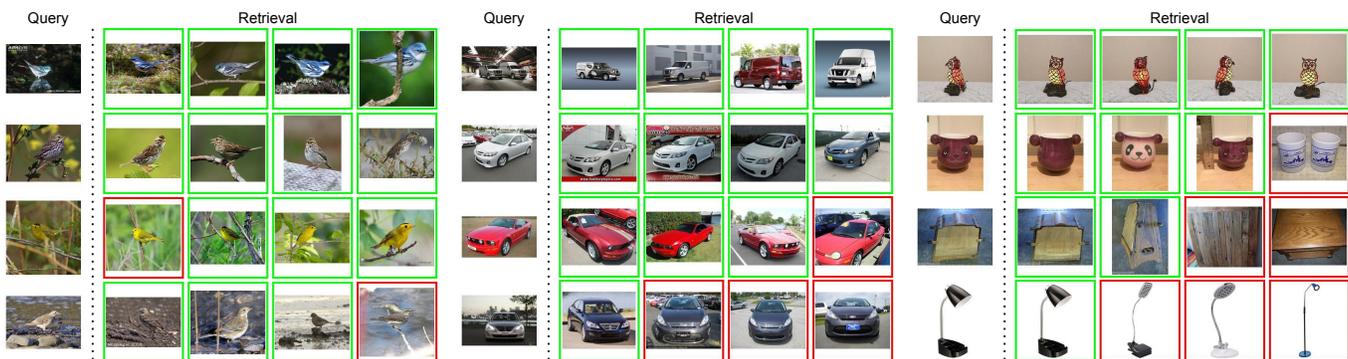
TABLE 8

The results of Group Loss in Densenet backbones.

Loss+Ensembles	CUB-200-2011					CARS 196					Stanford Online Products			
	R@1	R@2	R@4	R@8	NMI	R@1	R@2	R@4	R@8	NMI	R@1	R@10	R@100	NMI
BIER 6 [115]	55.3	67.2	76.9	85.1	-	75.0	83.9	90.3	94.3	-	72.7	86.5	94.0	-
HDC 3 [116]	54.6	66.8	77.6	85.9	-	78.0	85.8	91.1	95.1	-	70.1	84.9	93.2	-
ABE 2 [79]	55.7	67.9	78.3	85.5	-	76.8	84.9	90.2	94.0	-	75.4	88.0	94.7	-
ABE 8 [79]	60.6	71.5	79.8	87.4	-	85.2	90.5	94.0	96.1	-	76.3	88.4	94.8	-
A-BIER 6 [85]	57.5	68.7	78.3	86.2	-	82.0	89.0	93.2	96.1	-	74.2	86.9	94.0	-
D and C 8 [81]	65.9	76.6	84.4	90.6	69.6	84.6	90.7	94.1	96.5	70.3	75.9	88.4	94.9	90.2
RLL 3 [10]	61.3	72.7	82.7	89.4	66.1	82.1	89.3	93.7	96.7	71.8	79.8	91.3	96.3	90.4
Group Loss 2-ensemble	65.8	76.7	85.2	91.2	68.5	86.2	91.6	95.0	97.1	72.6	75.9	88.0	94.5	91.1
Group Loss++ 5-ensemble	66.9	77.1	85.4	91.5	70.0	88.0	92.5	95.7	97.5	74.2	76.3	88.3	94.6	91.1

TABLE 9

Retrieval and Clustering performance of our ensemble compared with ensemble methods. Bold indicates best results.

Fig. 16. Retrieval results on a set of images from the *CUB-200-2011* (left), *Cars 196* (middle), and *Stanford Online Products* (right) datasets using our Group Loss model. The left column contains query images. The results are ranked by distance. The green square indicates that the retrieved image is from the same class as the query image, while the red box indicates that the retrieved image is from a different class.

- [9] K. He, F. Çakir, S. A. Bargal, and S. Sclaroff, "Hashing as tie-aware learning to rank," in *CVPR*, 2018.
- [10] X. Wang, Y. Hua, E. Kodirov, G. Hu, R. Garnier, and N. M. Robertson, "Ranked list loss for deep metric learning," in *CVPR*, 2019.
- [11] J. Weibull, *Evolutionary Game Theory*. MIT Press, 1997.
- [12] A. Erdem and M. Pelillo, "Graph transduction as a noncooperative game," *Neural Computation*, 2012.
- [13] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, 2005.
- [14] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *NIPS*, 2016.
- [15] H. O. Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *CVPR*, 2016.
- [16] H. O. Song, S. Jegelka, V. Rathod, and K. Murphy, "Deep metric learning via facility location," in *CVPR*, 2017.
- [17] A. F. McDaid, D. Greene, and N. J. Hurley, "Normalized mutual information to evaluate overlapping community finding algorithms," *CoRR*, vol. abs/1110.2515, 2011.
- [18] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin, "Deep metric learning with angular loss," in *ICCV*, 2017.
- [19] M. T. Law, R. Urtasun, and R. S. Zemel, "Deep spectral clustering learning," in *ICML*, 2017.
- [20] B. B. Meier, I. Elezi, M. Amirian, O. Dürr, and T. Stadelmann, "Learning neural models for end-to-end clustering," in *ANNPR*, 2018.
- [21] B. Yu, T. Liu, M. Gong, C. Ding, and D. Tao, "Correcting the triplet selection bias for triplet loss," in *ECCV*, 2018.
- [22] A. Zhai and H. Wu, "Classification is a strong baseline for deep metric learning," in *BMVC*, 2019.
- [23] X. Zheng, R. Ji, X. Sun, B. Zhang, Y. Wu, and F. Huang, "Towards optimal fine grained retrieval via decorrelated centralized loss with normalize-scale layer," in *AAAI*, 2019.
- [24] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*, 2016.
- [25] Q. Qian, L. Shang, B. Sun, J. Hu, T. Tacoma, H. Li, and R. Jin, "Softtriple loss: Deep metric learning without triplet sampling," in *ICCV*, 2019.
- [26] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *ICCV*, 2017.
- [27] E. W. Teh, T. DeVries, and G. W. Taylor, "Proxynca++: Revisiting and revitalizing proxy neighborhood component analysis," in *ECCV*, 2020.
- [28] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *CVPR*, 2020.
- [29] Y. Zhu, M. Yang, C. Deng, and W. Liu, "Fewer is more: A deep graph metric learning perspective using fewer proxies," in *NeurIPS*, 2020.
- [30] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter

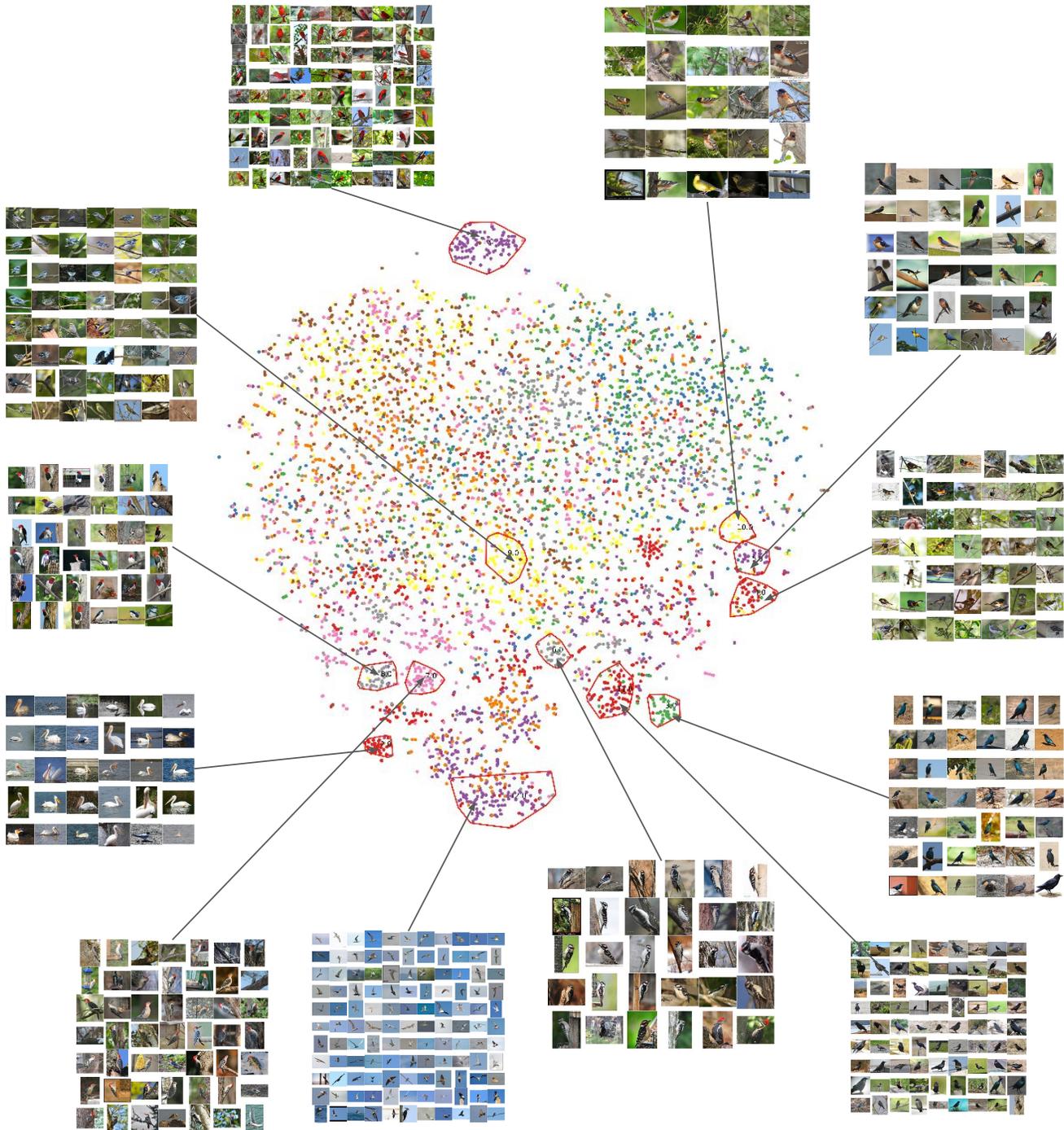


Fig. 17. t-SNE [117] visualization of our embedding on the CUB-200-2011 [66] dataset, with some clusters highlighted. Best viewed on a monitor when zoomed in.

- pairing neural network for person re-identification,” in *CVPR*, 2014.
- [31] E. Ahmed, M. J. Jones, and T. K. Marks, “An improved deep learning architecture for person re-identification,” in *CVPR*, 2015.
- [32] W. Chen, X. Chen, J. Zhang, and K. Huang, “Beyond triplet loss: A deep quadruplet network for person re-identification,” in *CVPR*, 2017.
- [33] W. Ge, W. Huang, D. Dong, and M. R. Scott, “Deep metric learning with hierarchical triplet loss,” in *ECCV*, 2018.
- [34] X. Zhang, H. Luo, X. Fan, W. Xiang, Y. Sun, Q. Xiao, W. Jiang, C. Zhang, and J. Sun, “Alignedreid: Surpassing human-level performance in person re-identification,” *CoRR*, vol. abs/1711.08184, 2017.
- [35] A. Hermans, L. Beyer, and B. Leibe, “In defense of the triplet loss for person re-identification,” *CoRR*, vol. abs/1703.07737, 2017.
- [36] R. Quan, X. Dong, Y. Wu, L. Zhu, and Y. Yang, “Auto-reid: Searching for a part-aware convnet for person re-identification,” in *ICCV*, 2019.
- [37] H. Luo, Y. Gu, X. Liao, S. Lai, and W. Jiang, “Bag of tricks and a strong baseline for deep person re-identification,” in *CVPR Workshops*, 2019.
- [38] Z. Dai, M. Chen, X. Gu, S. Zhu, and P. Tan, “Batch dropblock network for person re-identification and beyond,” in *ICCV*, 2019.
- [39] 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.

- [40] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang, "Beyond part models: Person retrieval with refined part pooling (and A strong convolutional baseline)," in *ECCV*, 2018.
- [41] R. R. Viorator, M. Haloi, and G. Wang, "Gated siamese convolutional neural network architecture for human re-identification," in *ECCV*, 2016.
- [42] Z. Zhang, C. Lan, W. Zeng, and Z. Chen, "Densely semantically aligned person re-identification," in *CVPR*, 2019.
- [43] K. He, X. Zhang, S. Ren, and J. ian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
- [44] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.
- [45] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *ICLR*, 2019.
- [46] L. T. Alemu, M. Shah, and M. Pelillo, "Deep constrained dominant sets for person re-identification," in *ICCV*, 2019.
- [47] E. Zemene, L. T. Alemu, and M. Pelillo, "Constrained dominant sets for retrieval," in *ICPR*, 2016.
- [48] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob., Vol. 1*, 1967.
- [49] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *CVPR*, 2019.
- [50] I. Elezi, S. Vascon, A. Torcinovich, M. Pelillo, and L. Leal-Taixé, "The group loss for deep metric learning," in *ECCV*, 2020.
- [51] K. Pearson, "Notes on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, 1895.
- [52] I. Elezi, A. Torcinovich, S. Vascon, and M. Pelillo, "Transductive label augmentation for improved deep network learning," in *ICPR*, 2018.
- [53] A. Rosenfeld, R. A. Hummel, and S. W. Zucker, "Scene labeling by relaxation operations," *IEEE Trans. Syst. Man Cybern.*, 1976.
- [54] M. Pelillo, "The dynamics of nonlinear relaxation labeling processes," *Journal of Mathematical Imaging and Vision*, 1997.
- [55] G. Palaioupanos, I. Panageas, and G. Piliouras, "Multiplicative weights update with constant step-size in congestion games: Convergence, limit cycles and chaos," in *NIPS*, 2017.
- [56] I. Panageas, G. Piliouras, and X. Wang, "Multiplicative weights updates as a distributed constrained optimization algorithm: Convergence to second-order stationary points almost always," in *ICML*, 2019.
- [57] R. A. Hummel and S. W. Zucker, "On the foundations of relaxation labeling processes," *IPAMI*, 1983.
- [58] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology," *Bulletin of the American Mathematical Society*, 1967.
- [59] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.
- [60] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.
- [61] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *International Conference on Rough Sets and Knowledge Technology*, 2014.
- [62] A. L. Maas and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *ICML*, 2013.
- [63] Z. Zhong, L. Zheng, D. Cao, and S. Li, "Re-ranking person re-identification with k-reciprocal encoding," in *CVPR*, 2017.
- [64] D. Qin, L. Bossard, T. Quack, and L. v. Gool, "Hello neighbor: accurate object retrieval with k-reciprocal nearest neighbors," in *ICCV*, 2011.
- [65] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, "Deepfashion: Powering robust clothes recognition and retrieval with rich annotations," in *CVPR*, 2016.
- [66] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," California Institute of Technology, Tech. Rep., 2011.
- [67] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, 2013.
- [68] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *NIPS Workshops*, 2017.
- [69] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *IJCV*, 2015.
- [70] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *ICLR*, 2020.
- [71] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *ICML*, 2017.
- [72] D. Berthelot, N. Carlini, I. J. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," in *NeurIPS*, 2019.
- [73] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *JMLR*, 2012.
- [74] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," *CoRR*, vol. abs/1908.03265, 2019.
- [75] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *AAAI*, 2020.
- [76] M. M. Kalayeh, E. Basaran, M. Gökmen, M. E. Kamasak, and M. Shah, "Human semantic parsing for person re-identification," in *CVPR*, 2018.
- [77] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian, "Scalable person re-identification: A benchmark," in *ICCV*, 2015.
- [78] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IPAMI*, 2011.
- [79] W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon, "Attention-based ensemble for deep metric learning," in *ECCV*, 2018.
- [80] W. Park, D. Kim, Y. Lu, and M. Cho, "Relational knowledge distillation," in *CVPR*, 2019.
- [81] A. Sanakoyeu, V. Tschernezki, U. Büchler, and B. Ommer, "Divide and conquer the embedding space for metric learning," in *CVPR*, 2019.
- [82] P. Jacob, D. Picard, A. Histace, and E. Klein, "Metric learning with HORDE: high-order regularizer for deep embeddings," in *ICCV*, 2019.
- [83] B. Brattoli, K. Roth, and B. Ommer, "MIC: mining interclass characteristics for improved metric learning," in *ICCV*, 2019.
- [84] H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," in *WACV*, 2020.
- [85] M. Opitz, G. Waltner, H. Possegger, and H. Bischof, "Deep metric learning with BIER: boosting independent embeddings robustly," *IPAMI*, 2020.
- [86] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *AAAI*, 2018.
- [87] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.
- [88] N. Vo and J. Hays, "Generalization in metric learning: Should the embedding layer be embedding layer?" in *WACV*, 2019.
- [89] W. Yang, H. Huang, Z. Zhang, X. Chen, K. Huang, and S. Zhang, "Towards rich feature discovery with class activation maps augmentation for person re-identification," in *CVPR*, 2019.
- [90] P. Fang, J. Zhou, S. K. Roy, L. Petersson, and M. Harandi, "Bilinear attention networks for person retrieval," in *ICCV*, 2019.
- [91] N. Martinel, G. L. Foresti, and C. Micheloni, "Aggregating deep pyramidal representations for person re-identification," in *CVPR Workshops*, 2019.
- [92] Y. Suh, J. Wang, S. Tang, T. Mei, and K. M. Lee, "Part-aligned bilinear representations for person re-identification," in *ECCV*, 2018.
- [93] L. Qi, J. Huo, L. Wang, Y. Shi, and Y. Gao, "Maskreid: A mask based deep ranking neural network for person re-identification," *CoRR*, vol. abs/1804.03864, 2018.
- [94] Z. Zhang, C. Lan, W. Zeng, X. Jin, and Z. Chen, "Relation-aware global attention for person re-identification," in *CVPR*, 2020.
- [95] X. Chen, C. Fu, Y. Zhao, F. Zheng, J. Song, R. Ji, and Y. Yang, "Salience-guided cascaded suppression network for person re-identification," in *CVPR*, 2020.
- [96] D. Chen, D. Xu, H. Li, N. Sebe, and X. Wang, "Group consistent similarity learning via deep CRF for person re-identification," in *CVPR*, 2018.
- [97] W. Li, X. Zhu, and S. Gong, "Harmonious attention network for person re-identification," in *CVPR*, 2018.
- [98] X. Chang, T. M. Hospedales, and T. Xiang, "Multi-level factorisation net for person re-identification," in *CVPR*, 2018.
- [99] B. Chen, W. Deng, and J. Hu, "Mixed high-order attention network for person re-identification," in *ICCV*, 2019.

- [100] K. Zhou, Y. Yang, A. Cavallaro, and T. Xiang, "Omni-scale feature learning for person re-identification," in *ICCV*, 2019.
- [101] K. Zhu, H. Guo, Z. Liu, M. Tang, and J. Wang, "Identity-guided human semantic parsing for person re-identification," in *ECCV*, 2020.
- [102] Y. Shen, H. Li, S. Yi, D. Chen, and X. Wang, "Person re-identification with deep similarity-guided graph neural network," in *ECCV*, 2018.
- [103] Y. Shen, H. Li, T. Xiao, S. Yi, D. Chen, and X. Wang, "Deep group-shuffling random walk for person re-identification," in *CVPR*, 2018.
- [104] R. Hou, B. Ma, H. Chang, X. Gu, S. Shan, and X. Chen, "Interaction-and-aggregation network for person re-identification," in *CVPR*, 2019.
- [105] Z. Zheng, X. Yang, Z. Yu, L. Zheng, Y. Yang, and J. Kautz, "Joint discriminative and generative learning for person re-identification," in *CVPR*, 2019.
- [106] T. Chen, S. Ding, J. Xie, Y. Yuan, W. Chen, Y. Yang, Z. Ren, and Z. Wang, "Abd-net: Attentive but diverse person re-identification," in *ICCV*, 2019.
- [107] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Networks*.
- [108] J. Seidenschwarz, I. Elezi, and L. Leal-Taixé, "Learning intra-batch connections for deep metric learning," in *ICML*, 2021.
- [109] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *ICML*, 2017.
- [110] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.
- [111] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.
- [112] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *ICCV*, 2015.
- [113] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and learn: Unsupervised learning using temporal order verification," in *ECCV*, 2016.
- [114] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," in *ICLR*, 2017.
- [115] M. Opitz, G. Waltner, H. Possegger, and H. Bischof, "BIER - boosting independent embeddings robustly," in *ICCV*, 2017.
- [116] Y. Yuan, K. Yang, and C. Zhang, "Hard-aware deeply cascaded embedding," in *CVPR*, 2017.
- [117] L. van der Maaten and G. E. Hinton, "Visualizing non-metric similarities in multiple maps," *Machine Learning*, 2012.



Dr. Ismail Elezi obtained his Ph.D. at Ca' Foscari University of Venice under the supervision of Prof. Marcello Pelillo and Prof. Thilo Stadelmann. In 2019, he was a visiting Ph.D. student at the Dynamic Vision and Learning (DVL) Group headed by Prof. Laura Leal-Taixé working on metric learning and generative adversarial networks. He then did an internship at Nvidia (Santa Clara) working on active learning. Since October 2020, he is a postdoctoral researcher at DVL, working on topics such as metric learning,

generative adversarial networks, active learning, and semi-supervised learning. He recently received a Humboldt Research Fellowship.



Jenny Seidenschwarz obtained her bachelor's degree in Mechanical Engineering at the Technical University of Munich, and her master's degree in Robotics, Cognition, and Intelligence also at the Technical University of Munich. In her Master's, she specialized in Deep Learning and Machine Learning and wrote her thesis on Deep Metric Learning at the Dynamic Vision and Learning group under the supervision of Dr. Ismail Elezi and Prof. Dr. Laura Leal-Taixé.

Following her thesis, she started a Ph.D. position at the beginning of 2021 also at the Dynamic Vision and Learning group.



Laurin Wagner obtained his bachelor's degree in Mathematics at the Technical University of Munich, and his Master's degree in Data Science at the Technical University of Munich and the Royal Institute of Technology in Stockholm. In his Master's, he focused on Machine Learning and specialized in Computer Vision. He wrote his thesis in the area of Deep Metric Learning at the Dynamic Vision and Learning group under the supervision of Dr. Ismail Elezi and Prof. Dr. Laura Leal-Taixé.



Dr. Sebastiano Vascon is an assistant professor at Ca' Foscari University of Venice. In 2016 he obtained the title of Ph.D. from the Italian Institute of Technology under the supervision of prof. Vittorio Murino. He then moved to University of Venice working as a PostDoc within the group of prof. Marcello Pelillo. His research activities are in the fields of computer vision, game theory, machine, and deep learning.



Dr. Alessandro Torcinovich is a postdoc at Ca' Foscari University of Venice. In 2021, he obtained his doctoral degree under the supervision of prof. Marcello Pelillo. During his Ph.D. program, he worked on the integration of game-theoretic techniques and deep learning methods for addressing weakly supervised tasks. In addition, he brought on a technology transfer project in collaboration with the ESA on anomaly detection tasks.



Prof. Marcello Pelillo is a full professor of Computer Science at Ca' Foscari University in Venice, Italy, where he directs the Computer Vision and Pattern Recognition group. He held visiting research positions at Yale University, McGill University, the University of Vienna, York University (UK), the University College London, the National ICT Australia (NICTA). He has been General Chair for ICCV 2017, Program Chair for ICPR 2020, and has founded many conferences and workshops (e.g., EMMCVPR, SIMBAD, IWCV). He serves (has served) on the Editorial Boards of the journals *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *Pattern Recognition*, *IET Computer Vision*, *Frontiers in Computer Image Analysis*, *Brain Informatics*, and serves on the Advisory Board of the *International Journal of Machine Learning and Cybernetics*. He has been elected a Fellow of the IEEE and a Fellow of the IAPR, and has recently been appointed IEEE SMC Distinguished Lecturer. His Erdos number is 2.



Prof. Dr. Laura Leal-Taixé is a tenure-track professor (W2) at the Technical University of Munich leading the Dynamic Vision and Learning group and a Principal research scientist at Argo AI. Before that, she spent two years as a postdoctoral researcher at ETH Zurich, Switzerland, and a year as a senior postdoctoral researcher in the Computer Vision Group at the Technical University in Munich. She obtained her Ph.D. from the Leibniz University of Hannover in Germany, spending a year as a visiting scholar at

the University of Michigan, Ann Arbor, USA. She pursued B.Sc. and M.Sc. in Telecommunications Engineering at the Technical University of Catalonia (UPC) in her native city of Barcelona. She is a recipient of the Sofja Kovalevskaja Award of 1.65 million euros for her project socialMaps.