Supplementary material: The Group Loss for Deep Metric Learning

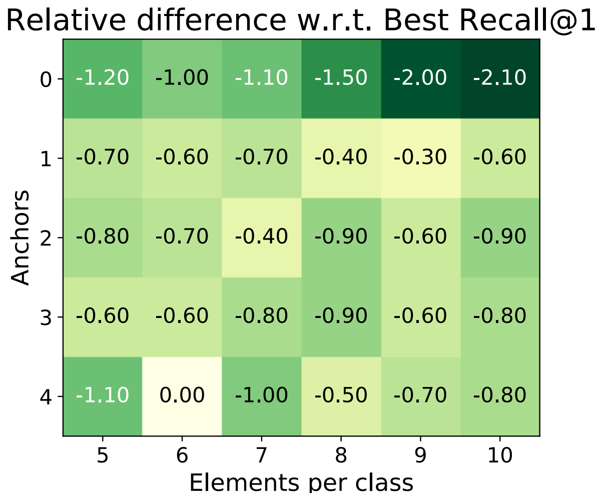
Ismail Elezi[1], Sebastiano Vascon[1], Alessandro Torcinovich[1], Marcello Pelillo[1], and Laura Leal-Taixé[2]

[1] Ca' Foscari University of Venice
[2] Technical University of Munich

1 Robustness Analysis of CARS 196 Dataset

In the main work, we showed the robustness analysis on the *CUB-200-2011* [16] dataset (see Figure 4 in the main paper). Here, we report the same analysis for the *Cars 196* [9] dataset. This leads us to the same conclusions as shown in the main paper.

We do a grid search over the total number of elements per class versus the number of anchors, as we did for the experiment in the main paper. We increase the number of elements per class from 5 to 10, and in each case, we vary the number of anchors from 0 to 4. We show the results in Fig. 1. Note, the results decrease mainly when we do not have any labeled sample, i.e., when we use zero anchors. The method shows the same robustness as on the *CUB-200-2011* [16] dataset, with the best result being only 2.1 percentage points better at the Recall@1 metric than the worst result.



Anchors	5	6	7	8	9	10
0	-1.20	-1.00	-1.10	-1.50	-2.00	-2.10
1	-0.70	-0.60	-0.70	-0.40	-0.30	-0.60
2	-0.80	-0.70	-0.40	-0.90	-0.60	-0.90
3	-0.60	-0.60	-0.80	-0.90	-0.60	-0.80
4	-1.10	0.00	-1.00	-0.50	-0.70	-0.80

Fig. 1: The effect of the number of anchors and the number of samples per class.

2 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 [16,9,15] 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 hyperparameters using random search [1]. 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. During inference we normalize the features using the L2-norm.

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.

3 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 [7,6,2,18], 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 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:

$$T_{softmax}(z_i) = \frac{e^{z_i/T}}{\sum_i e^{z_i/T}}, \tag{1}$$

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

Recent works in supervised learning [6] and semi-supervised learning [2] 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 2.5*pp* better Recall@1 scores on *CUB-200-2011* [16] and 2*pp* better scores on *Cars 196* [9]. Note, the methods of Table 1 (main paper) that use a classification loss, use also temperature scaling.

4 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 [8]. Densenets are a modern CNN architecture which show similar classification accuracy to GoogleNet in most tasks (so they are a similarly strong classification baseline [3]). 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. 1. We use the same hyperparameters as the ones used for the GoogleNet experiments, reaching state-of-the-art results on both *CARS 196* [9] 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 this is 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* [16] 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* [9] 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. Despite that the networks reached state-of-the-art results even without any hyperparameter tuning, we expect a minimum amount of hyperparameters tuning to help. To this end, we used random search [1] to optimize the hyperparameters of our best network on the *CARS 196* [9] dataset. We reach a 90.7 score (2*pp* higher score than the network with default hyperparameters) in Recall@1, and 77.6 score (3*pp* 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* [9] dataset.

5 Comparisons with SoftTriple Loss [13]

A recent paper (SoftTriple loss [13], ICCV 2019) explores another type of classification loss for the problem of metric learning. The main difference between our method and [13] is that our method checks the similarity between samples, and then refines the predicted probabilities (via a dynamical system) based on

[3] 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%).

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
SofTriple [13]	11307040	65.4	69.3	11296800	84.5	70.1	68743200	78.3	92

Table 1: The results of Group Loss in Densenet backbones and comparisons with SoftTriple loss [13]

that information. [13] instead deals with the intra-class variability, but does not explicitly take into account the similarity between the samples in the mini-batch. They propose to add a new layer with 10 units per class.

We compare the results of [13] with our method in Tab. 1. SoftTriple loss [13] reaches a higher result than our method in all three datasets in Recall@1 metric, and higher results than the Group Loss on the *CUB-200-2011* and *Stanford Online Products* datasets in NMI metric. However, this comes at a cost of significantly increasing the number of parameters. On the *Stanford Online Products* dataset in particular, the number of parameters of [13] is 68.7 million. In comparison, we (and the other methods we compare the results with in the main paper) use only 16.6 million parameters. In effect, their increase in performance comes at the cost of using a neural network which is 4 times larger as ours, making results not directly comparable. Furthermore, using multiple centres is crucial for the performance of [13]. Fig. 4 of the work [13] shows that when they use only 1 centre per class, the performance drops by $3pp$, effectively making [13] perform worse than the Group Loss by $2pp$.

We further used the official code implementation to train their network using only one center on the *CARS 196* [9] dataset, reaching 83.1 score in Recall@1, and 70.1 score in NMI metric, with each score being $0.6pp$ lower than the score of The Group Loss. Essentially, when using the same backbone, SoftTriple loss [13] reaches lower results than our method.

As we have shown in the previous section, increasing the number of parameters improves the performances of the network, but it is not a property of the loss function. In fact, a similarly sized network to theirs (Densenet 169) consistently outperforms SoftTriple loss, as can be seen in Tab. 1. For this reason, we keep this comparison in the supplementary material, while we leave for the main paper the comparisons with more than 20 methods that use the same backbone.

6 Alternative Loss Formulation

In the main paper, we formulated the loss as an iterative dynamical system, followed by the cross-entropy loss function. In this way, we encourage the network to predict the same label for samples coming from the same class. One might argue that this is not necessarily the best loss for metric learning, in the end, we are interested in bringing similar samples closer together in the embedding space, without the need of having them classified correctly. Even though several works have shown that a classification loss can be used for metric learning [12,18,13], we test whether this is also the best formulation for our loss function.

We therefore experiment with a different loss function which encourages the network to produce similar label distributions (soft labels) for the samples coming from the same class. We first define Kullback-Leibler divergence for two distributions P and Q as:

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) log \frac{P(x)}{Q(x)}. \tag{2}$$

We then minimize the divergence between the predicted probability (after the iterative procedure) of samples coming from the same class. Unfortunately, this loss formulation results in lower performances on both *CUB-200-2011* [16] (3*pp*) and *Cars 196* [9] (1.5*pp*). Thus, we report the experiments in the main paper only with the original loss formulation.

7 Dealing with Negative Similarities

Equation (4) in the main paper assumes that the matrix of similarity is non-negative. However, for similarity computation, we use a correlation metric (see Equation (1) 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 [4]. 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 (1).

We compare the results of shifting vs clamping. On the *CARS 196* dataset, we do not see a significant difference between the two approaches. However, on the *CUBS-200-2011* dataset, the Recall@1 metric is 51 with shifting, much below the 64.3 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 *CUBS-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*.

8 Dealing with Relative Labels

The proposed method assumes that the dataset consists of classes with (absolute) labels and set of samples in each class. This is the case for the datasets used in metric learning [16,9,15]) technique evaluations. However, deep metric learning can be applied to more general problems where the absolute class label is not available but only 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 [17] 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 [11], relative labels are used as self-supervision for learning good spatio-temporal representation.

Our method, similar to other classification-based losses [18,12,13] for deep metric learning, or triple loss improvements like Hierarchical Triplet Loss [5] assumes the presence of absolute labels. Unlike traditional losses [3,14], 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.

9 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 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 we have tried, reaching similar results to those presented in the paper) 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 would allow 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 [18,12,13,5] face the same problem. However, in related fields (for example, face recognition), there could be millions of classes (identities), in which case we can use the proposed solution.

10 t-SNE on CUB-200-2011 Dataset

Fig. 2 visualizes the t-distributed stochastic neighbor embedding (t-SNE) [10] of the embedding vectors obtained by our method on the *CUB-200-2011* [16] 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.

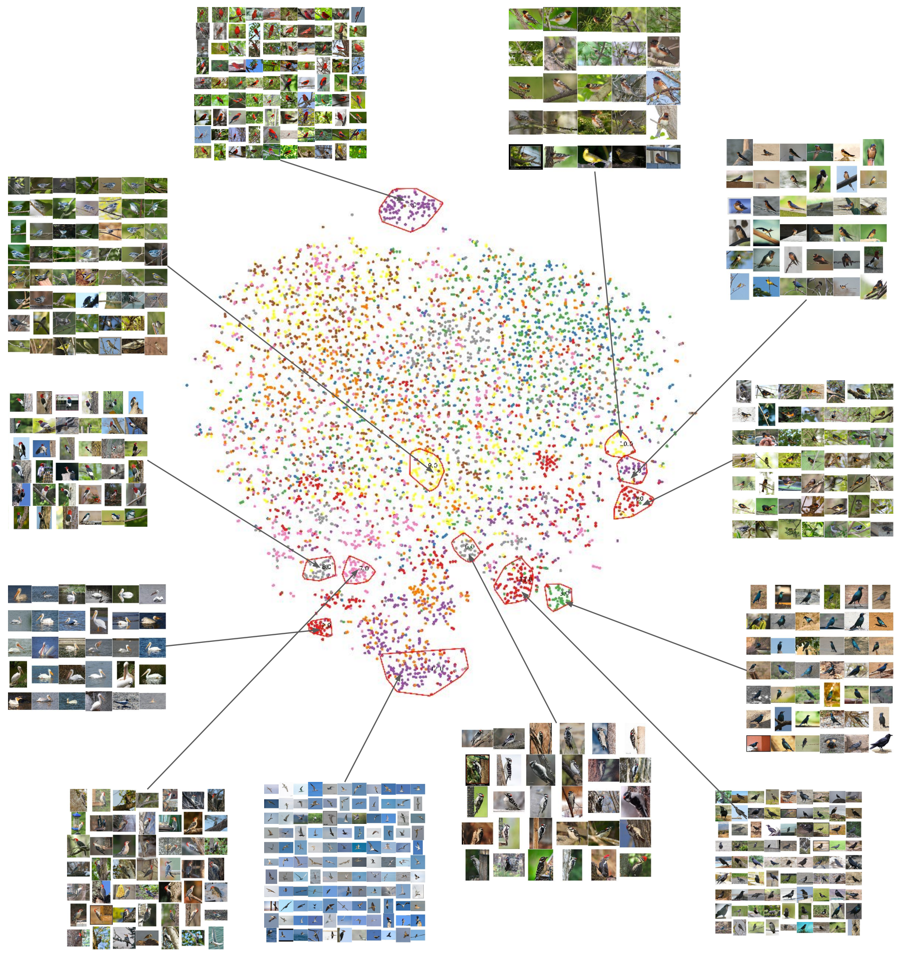


Fig. 2: t-SNE [10] visualization of our embedding on the CUB-200-2011 [16] dataset, with some clusters highlighted. Best viewed on a monitor when zoomed in.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**, 281–305 (2012)
2. Berthelot, D., Carlini, N., Goodfellow, I.J., Papernot, N., Oliver, A., Raffel, C.: Mixmatch: A holistic approach to semi-supervised learning. In: Advances in Neural Information Processing Systems, NeurIPS. pp. 5050–5060 (2019)
3. Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., Shah, R.: Signature verification using a" siamese" time delay neural network. In: Advances in Neural Information Processing Systems, NIPS. pp. 737–744 (1994)
4. Erdem, A., Pelillo, M.: Graph transduction as a noncooperative game. Neural Computation **24**(3), 700–723 (2012)
5. Ge, W., Huang, W., Dong, D., Scott, M.R.: Deep metric learning with hierarchical triplet loss. In: European Conference in Computer Vision, ECCV. pp. 272–288 (2018)
6. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: Precup, D., Teh, Y.W. (eds.) International Conference on Machine Learning, ICML (2017)
7. Hinton, G.E., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. CoRR (2015)
8. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR. pp. 2261–2269 (2017)
9. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: International IEEE Workshop on 3D Representation and Recognition (3dRR-13). Sydney, Australia (2013)
10. van der Maaten, L., Hinton, G.E.: Visualizing non-metric similarities in multiple maps. Machine Learning **87**(1), 33–55 (2012)
11. Misra, I., Zitnick, C.L., Hebert, M.: Shuffle and learn: Unsupervised learning using temporal order verification. In: European Conference on Computer Vision ECCV (2016)
12. Movshovitz-Attias, Y., Toshev, A., Leung, T.K., Ioffe, S., Singh, S.: No fuss distance metric learning using proxies. In: IEEE International Conference on Computer Vision, ICCV. pp. 360–368 (2017)
13. Qian, Q., Shang, L., Sun, B., Hu, J., Tacoma, T., Li, H., Jin, R.: Softtriple loss: Deep metric learning without triplet sampling. In: IEEE/CVF International Conference on Computer Vision, ICCV. pp. 6449–6457 (2019)
14. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR. pp. 815–823 (2015)
15. Song, H.O., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR. pp. 4004–4012 (2016)
16. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The Caltech-UCSD Birds-200-2011 Dataset. Tech. Rep. CNS-TR-2011-001, California Institute of Technology (2011)

17. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: IEEE International Conference on Computer Vision, ICCV. pp. 2794–2802 (2015)
18. Zhai, A., Wu, H.: Classification is a strong baseline for deep metric learning. In: British Machine Vision Conference BMVC. p. 91 (2019)