Area Under the ROC Curve Maximization for Metric Learning

Bojana Gajić
Vintra, Inc.
Barcelona, Spain
bgajic@vintra.io

Ariel Amato
Vintra, Inc.
Barcelona, Spain
aamato@vintra.io

Ramon Baldrich
Computer Vision Center
Barcelona, Spain
aamato@vintra.io

Joost van de Weijer
Computer Vision Center
Barcelona, Spain
aamato@vintra.io

Carlo Gatta
Vintra, Inc.
Barcelona, Spain
carlo.gatta@gmail.com

1. AUC implementation details

The AUC loss function has three input parameters: (1) list of N features extracted from the backbone architecture, (2) a list of ground truth labels that correspond to the features, and (3) the step size Δs (line 1 in algorithm 1). We first calculate the similarities between all features from the minibatch (line 2 in algorithm 1). The $similarity$ matrix is a square matrix of size $N \times N$, where the value of $similarity[i, j]$ corresponds to the similarity between features i and j.

We get the positive mask as a binary matrix of size $N \times N$ that indicates if the elements in the similarity matrix belong to different images from the same class. Similarly, the negative mask extracts the positions in the similarity matrix that belong to different classes. We mask the $similarity$ matrix by element-wise multiplying it with positive mask and find the minimal similarity per each row (line 3 in algorithm 1). In this way, we obtain a vector of the hardest positive similarities (HPS) for each input feature. Similarly, we mask the $similarity$ matrix with the negative mask and find the maximum similarity per row, resulting in a vector of size N of hardest negative similarities (HNS) for each input feature (line 4 in algorithm 1).

We define a vector of thresholds as a step vector of size $S+1$: $[t_{min}, t_{min} + \Delta s, ..., t_{max}]$ (line 5 in algorithm 1), and get the optimal slope for the given Δs from Table 2 (line 6 in algorithm 1). For each threshold from the step vector and for each hardest positive similarity, we get a value of sigmoid defined in formula 6, and store it in a matrix σ_r^+ of size $N \times (S+1)$ (line 7 in algorithm 1). Similarly, we obtain the σ_r^- matrix based on hardest negative similarities (line 8 in algorithm 1).

We obtain vectors s_1 and s_2 of length S from σ_r^+ and σ_r^- matrices (lines 9-13 in algorithm 1). Although we present the algorithm with a for loop over the samples, this procedure is implemented with matrices and in a parallel way exploiting the GPU parallelization capabilities. We get the estimated area under the ROC curve R based on the samples from mini-batch, as shown in line 14 of algorithm 1. Finally, the AUC loss is calculated as $1 - R$ (line 15 in algorithm 1).

2. Limitations

The main limitation of the AUC loss is its performance on small datasets. In Table 1 we show that AUC performs well on small datasets as well. We trained our model in two scenarios: using original images and using crops of the birds provided by the authors of the dataset. We treated both cases in the same way, resizing the images to 256x256 and using a random 224x224 crop for training, and central for testing. AUC loss achieves results that are comparable with state of the art when trained with original images. Our method outperforms all ensemble-based methods, and shows comparable results with the newest state-of-the-art methods. The only method that performs significantly better is RankMI [4]. Even though RankMI outperforms AUC, it is computationally more expensive, as the model is built out of two networks that are updated alternately, it has two extra hyper parameters; also the authors do not report the input image size. R-Margin model achieves 6.7% higher rank@1 on CUB-200 dataset, while using a bigger mini-batch, distance based tuple mining, and ρ regularization. Additionally, this model has an extra hyper parameter β and the results vary significantly with different initialization values. We believe that AUC loss leads to overfitting due to its strong gradients, when trained on a small size datasets. We improved the performance of AUC by using image crops instead of whole images (see Table 2).

Algorithm 1 AUC_{BH} loss function

Input: $features, labels, \Delta s$

calculate the matrix of $similarities$ based on features

find the lowest similarity per row for features that belong to the same class ($hardest\_positive\_similarity$ or HPS)

find the highest similarity per row for features that do not belong to the same class ($hardest\_negative\_similarity$ or HNS)

$step\_vector \leftarrow [t_{min}, t_{min} + \Delta s, ..., t_{max} - \Delta s]$

for a given Δs, find optimal $slope$ from Table 6

$$\sigma_r^+ \leftarrow \frac{1}{1 + e^{-slope(HPS - step\_vector)}}$$

$$\sigma_r^- \leftarrow \frac{1}{1 + e^{-slope(HNS - step\_vector)}}$$

for j in 1,2,...S **do**

$s \leftarrow step\_vector[j]$

$$s_1[j] \leftarrow \sum_{i=1}^{N} \sigma_r^+[i, s] + \sigma_r^+[i, s + \Delta s]$$

$$s_2[j] \leftarrow \sum_{i=1}^{N} \sigma_r^-[i, s] - \sigma_r^+[i, s + \Delta s]$$

end for

$$R \leftarrow \frac{1}{2N^2} \sum_{j=1}^{S} s_1[j] s_2[j]$$

$AUC \leftarrow 1 - R$

	im	mb	$R@1$	$R@8$
Histogram Loss$_G^{512}$ [12]	256	128	50.3	82.4
N-Pair-Loss$_G^{64}$ [11]	-	120	51.0	83.2
Binomial Deviance$_G^{512}$ [12]	256	128	52.8	83.9
Angular Loss$_G^{512}$ [13]	256	128	54.7	83.9
Clustering$_G^{64}$ [7]	227	128	48.2	81.9
Smart Mining$_G^{64}$ [2]	-	-	49.8	83.3
Margin$_G^{128}$ [6]	224	128	63.8	90.0
HDC$_G^{384}$ [15]	-	100	53.6	85.6
HTL$_G^{128}$ [1]	224	50	57.1	86.5
A-BIER$_G^{512}$ [8]	224	-	57.5	86.2
ABE-8$_G^{512}$ [5]	224	64	60.6	87.7
R-Margin$_{R50}^{128}$ [10]	224	160	64.9	-
RaMBO$_{R50}^{512}$ log log [9]	224	128	64.0	90.6
RankMI$_{R50}^{128}$ [4]	-	120	**66.7**	**91.0**
AUC$_{R50}^{512}$	224	128	58.19	86.36
AUC$_{R50}^{512}$	256	128	62.10	89.45

Table 1. Comparison with the state-of-the-art on the CUB-200-2011 [14] dataset. Embedding dimension is presented as a superscript and the backbone architecture as a subscript. R stands for ResNet, G for GoogLeNet.

	im	mb	$R@1$	$R@8$
PDDM Triplet$_G^{128}$ [3]	-	64	50.9	82.5
PDDM Quadruplet$_G^{128}$ [3]	-	64	58.3	88.4
HDC$_G^{384}$ [15]	-	100	60.7	89.2
Margin$_G^{128}$ [6]	224	128	63.9	90.6
AUC$_{R50}^{512}$	224	128	**68.28**	**92.31**
AUC$_{R50}^{512}$	256	128	**70.81**	**93.53**

Table 2. Comparison with the state of the art on the CUB-200-2011 [14] cropped dataset. Embedding dimension is presented as a superscript and the backbone architecture as a subscript. R stands for ResNet, G for GoogLeNet.

References

[1] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R Scott. Deep metric learning with hierarchical triplet loss. In *Proc. ECCV*, 2018. 2

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

[3] Chen Huang, Chen Change Loy, and Xiaoou Tang. Local similarity-aware deep feature embedding. In *Advances in neural information processing systems*, pages 1262–1270, 2016. 2

[4] Mete Kemertas, Leila Pishdad, Konstantinos G Derpanis, and Afsaneh Fazly. Rankmi: A mutual information maximizing ranking loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14362–14371, 2020. 1, 2

[5] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *Proc. ECCV*, 2018. 2

[6] R. Manmatha, Chao-Yuan Wu, Alexander J. Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *Proc. ICCV*, 2017. 2

[7] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep metric learning via facility location. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5382–5390, 2017. 2

[8] Michael Opitz, Georg Waltner, Horst Possegger, and Horst Bischof. Deep metric learning with bier: Boosting independent embeddings robustly. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2

[9] Michal Rolínek, Vít Musil, Anselm Paulus, Marin Vlastelica, Claudio Michaelis, and Georg Martius. Optimizing rank-based metrics with blackbox differentiation. *arXiv preprint arXiv:1912.03500*, 2019. 2

[10] Karsten Roth, Timo Milbich, Samarth Sinha, Prateek Gupta, Bjoern Ommer, and Joseph Paul Cohen. Revisiting training strategies and generalization performance in deep metric learning. *arXiv preprint arXiv:2002.08473*, 2020. 2

[11] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016. 2

[12] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016. 2

[13] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017. 2

[14] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010. 2

[15] Yuhui Yuan, Kuiyuan Yang, and Chao Zhang. Hard-aware deeply cascaded embedding. In *Proceedings of the IEEE international conference on computer vision*, pages 814–823, 2017. 2