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

Abstract

Most popular metric learning losses have no direct relation with the evaluation metrics that are subsequently applied to evaluate their performance. We hypothesize that training a metric learning model by maximizing the area under the ROC curve (which is a typical performance measure of recognition systems) can induce an implicit ranking suitable for retrieval problems. This hypothesis is supported by previous work that proved that a curve dominates in ROC space if and only if it dominates in Precision-Recall space. To test this hypothesis, we design and maximize an approximated, derivable relaxation of the area under the ROC curve. The proposed AUC loss achieves state-of-the-art results on two large scale retrieval benchmark datasets (Stanford Online Products and DeepFashion In-Shop). Moreover, the AUC loss achieves comparable performance to more complex, domain specific, state-of-the-art methods for vehicle re-identification.

1. Introduction

The main objective of metric learning systems is to embed high dimensional data (such as images, videos, or audio signals) into a lower dimensional space, while ensuring that the data that comes from the same class or identity is embedded within a cluster, which is separated from the clusters of data that belong to other classes.

The traditional approaches (such as SIFT, SURF or bag-of-words) were replaced by newer deep learning methods that compute the embedding by processing input data by deep neural networks, and use this embedding for comparison of images. The neural networks are trained by minimizing a loss function that models the desired structure of the embedding space. Even though the most widely used losses for metric learning, such as constrastive loss [5], triplet loss [33], quadruplet loss [4], classification loss [44], etc, train models to provide locally optimal solutions, there is no direct relation between the optimized loss and the commonly used evaluation criteria (such as recall@1 and mAP).

One notable exception is the loss presented in [30], which proposes a direct maximization of the mean Average Precision (mAP) for solving the retrieval task, which perfectly mimics the final metric learning goal. However, this loss requires obtaining the vector representations of all training images by a full forward pass through a deep neural network several times, in order to provide a single gradient. The authors demonstrate the performance of the loss on training data up to 43k images, which is significantly less than the size of commonly used datasets nowadays.

The area under the ROC curve is a well known way of evaluating *recognition* systems [3, 11, 13]. As the amount of available data, as well as computational power, in the past were limited, the ROC curve based on the available samples was not smooth, and the area below such an empirical curve was not accurate. Therefore, in [13] the authors proposed two ways to approximate the real area under the ROC curve: Gaussian based approximation and Wilcoxon Statistics. Maximization of the area under the ROC curve for *classification* task was proposed in [3], using the Wilcoxon statistics. Even though this approach was appropriate in the past, we show in Section 4.3.4 that our proposal is clearly superior when evaluated on metric learning datasets.

In this paper, we propose to directly minimize a loss which has a clear relation with the metrics typically used to evaluate retrieval problems. This hypothesis is supported by the fact that "a curve dominates in ROC space if and only if it dominates in PR space" [6]. In this paper we propose the AUC loss, a new metric learning loss which explicitly maximizes an underestimate of the area under the ROC curve at

the mini-batch level. We show how the area under the ROC curve can be approximated by its differentiable relaxation, without the need for extra hyper-parameter search. *AUC loss is effective and computationally inexpensive.* We tested AUC loss on four benchmark datasets, and showed that it achieved state-of-the-art performance for both retrieval (which is measured by mAP and rank@N) and recognition (measured by the area under the ROC curve).

2. Related Work

Losses that are designed for metric learning can be separated into two groups: pairwise losses and listwise losses.

Pairwise loss functions. This group of losses includes some of the most popular loss functions used for metric learning, instance retrieval, face recognition or re-identification. All these losses have one thing in common: they optimize absolute or relative distances of pairs of images; *they minimize distances between descriptors that represent images from the same class/identity, and maximize the ones that are coming from different classes*.

The pioneer from this group was the contrastive loss [12]. The authors of this approach use a Siamese architecture of two streams. If the two input samples are from the same class, the loss pushes their descriptors closer, and separates their descriptors if they belong to different classes.

The Triplet loss [33] requires a Siamese architecture with three streams, and it is fed with three images: an anchor image I_a, a positive image from the same class I_p and a negative image from any other class I_n. All three images are embedded into their respective descriptors r_a, r_p and r_n. The triplet loss pushes the descriptors from the same class closer to each other while separating the descriptors from different classes if the difference between the distance of anchor and negative ($d^- = ||r_a - r_n||^2$) and anchor-positive ($d^+ = ||r_a - r_p||^2$) is smaller than a margin m.

In [4] the authors propose adding an additional, fourth stream to the Siamese architecture, which embeds an additional negative image. The final objective is having an anchor-positive distance smaller than the anchor-negative, while making sure that the anchor-negative distance is greater than the negative-negative distance.

Many approaches, inspired by the triplet loss, proposed various ways to optimize training time and quality of the results by using more information from the data points that are available in a mini-batch. In [16] the authors propose creating mini-batches of P classes and K images per class. In each training step they do a forward pass of all $P \times K$ images and get their descriptors. They propose two ways of optimizing the main objective: *Batch hard* and *Batch all*. The *batch hard* triplet loss treats all images from the mini-batch as anchors, and for each one of them selects the one from the same class that is furthest away as a positive, and the one from a different class, closest to the anchor, as a negative. *Batch all* strategy calculates the loss based on all positive and negative pairs from the mini-batch.

Similarly to the *batch all* triplet loss, the structured loss [28] and the n-pair loss [34] take advantage of all positive and negative pairs from the mini-batch. In [34] the authors propose creating a mini-batch of N pairs $\{(x_1, x_1^+), (x_2, x_2^+), ..., (x_N, x_N^+)\}$ from N different classes. For each positive pair they sample $N-1$ negative samples from all different classes, and they use them for calculating the loss. In [28] the negative pairs are sampled inside of a mini-batch, so that the negative is one of the closest samples to either anchor or positive for each anchor-positive pair in the mini-batch.

Finally, one group of methods attempts to optimize the area under the ROC curve [7, 17, 45]. [45] and [7] have the goal to optimize for *classification*, while [17] trains the model for *cross-modal metric learning*. They all apply a loose approximation of indicator function $\mathbb{1}(x < y)$ as $y - x$. This approximation leads to a loss that is very similar to the Triplet loss with two minor differences: 1) the anchor of positive and negative pair is not necessarily the same sample, and 2) the margin m is set to 1.

Listwise loss functions Even though the pairwise loss functions optimize distances between positive and negative pairs, they do not explicitly optimize a ranking measure. The typical ranking measure is mean average precision (mAP) which cannot easily be optimized, as it requires ranking, which is not a differentiable operation.

In [37] the authors propose the Histogram loss. The method is designed to separate the histograms that approximate the distributions of positive and negative similarities inside of a mini-batch. This objective does not directly optimize the ranking task, but indirectly, it forces all positive pairs to have higher similarity than all negative pairs. [37] is the inspiration of several listwise losses [14, 15, 30] that directly optimize the average precision. The pioneer in this line is a differentiable approximation of average precision (AP) for retrieval in Hamming space, that focuses especially on tie scenarios (where both positive and negative samples belong to the same histogram bin) [14]. In [15] the authors apply the same strategy on retrieval and patch matching tasks. In [30] the authors propose a solution for training a very deep CNN with large images while optimizing mAP loss on the whole train set. Once all gradients are accumulated, it backpropagates the errors through the network. This method cannot easily scale to larger datasets, due to its high computational cost per weights update.

3. Method

In this section we first introduce the area under the ROC formula, which we relax to obtain its differentiable version, which is the base of the proposed new AUC loss.

3.1. Area under the ROC curve

Given a threshold value t, $T(t)$ the True Positive Ratio (TPR) and $F(t)$ the False Positive Ratio (FPR), the ROC curve is the parametrized curve $t \rightarrow (F(t), T(t))$, as shown by the red line in Fig. 1. The area under the ROC curve can be written as follows:

$$A = \int_{t=t_{min}}^{t_{max}} T(t) \frac{dF(t)}{dt} dt. \tag{1}$$

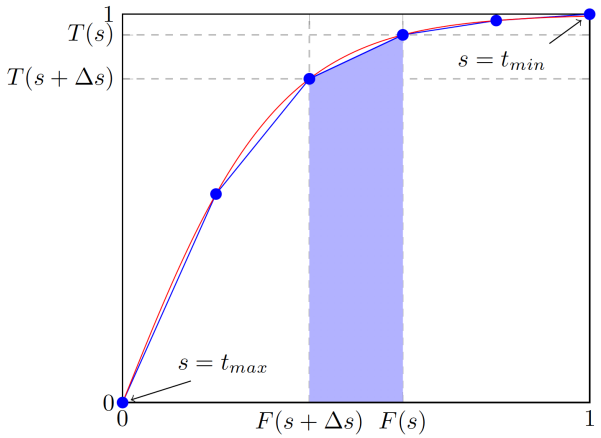

Figure 1. The ROC curve (red line) and its approximation based on a set of thresholds s (blue line). The area under the approximated curve is calculated using the Trapezoidal rule.

Given a similarity function $f(d_1, d_2) \in [t_{min}, ..., t_{max}]$, for a pair of data points d_1 and d_2 and for the set of all positive pairs $P = \{(a_1, p_1), (a_2, p_2), \ldots, (a_{N_P}, p_{N_P})\}$ in the training set, the TPR can be written as:

$$T(t) = \frac{1}{N_P} \sum_{i=1}^{N_P} H(f(a_i, p_i) - t), \tag{2}$$

where $H(\cdot)$ is the Heaviside function. For the set of all negative pairs $N = \{(a_1, n_1), (a_2, n_2), \ldots, (a_{N_N}, n_{N_N})\}$, the FPR can be written as:

$$F(t) = \frac{1}{N_N} \sum_{j=1}^{N_N} H(f(a_j, n_j) - t). \tag{3}$$

Plugging Equations 2 and 3 into 1 we have:

$$A = \int_{t=t_{min}}^{t_{max}} \frac{\sum_i^{N_P} H(f(a_i, p_i) - t)}{N_P} \frac{d}{dt} \left(\frac{\sum_j^{N_N} H(f(a_j, n_j) - t)}{N_N} \right) dt. \tag{4}$$

This formula cannot be used for gradient based optimization: (1) the integral cannot be directly computed and (2) the Heaviside function has zero gradient almost everywhere.

3.2. Differentiable relaxation of AUC

3.2.1 Integral to series (Riemann sum)

Firstly, we approximate the continuous integral from Equation 4 with its discrete representation. We apply the Trapezoidal rule and obtain the numerical approximation of Equation 1:

$$A^* = \sum_{s=t_{min}}^{t_{max}-\Delta s} \frac{T(s + \Delta s) + T(s)}{2} \left(F(s) - F(s + \Delta s) \right). \tag{5}$$

where s spans the interval $[t_{min}, t_{max}]$ in S discrete steps of size $\Delta s = (t_{max} - t_{min})/S$. This approximation corresponds to the area below the piece-wise linear blue curve from Fig. 1. The number of steps is a relevant parameter since more steps provide a better approximation of the integral. Taking into account that $T(s)$ and $F(s)$ depend only on the parameter s, they can be calculated in parallel for a set of values $s \in \{t_{min}, t_{min} + \Delta s, ..., t_{max} - \Delta s\}$, allowing for an efficient implementation on GPUs.

3.2.2 Heaviside to sigmoid

The second step involves using a derivable approximation of the Heaviside function; we use the following parametrized sigmoidal-like function:

$$\sigma_r(x, t) = \frac{1}{1 + e^{-r(x-t)}}. \tag{6}$$

This choice has three main rationales: (1) for large values of r this function becomes a good approximation of the Heaviside function discontinuity, (2) it provides very small gradients far from the discontinuity and, (3) it is symmetric around t thus producing an unbiased approximation error.

These characteristics allow having relevant gradients in the area close to the discontinuity and, at the same time, keeping the properties of the Heaviside function and almost completely ignoring sample pairs that have similarity very different from the considered threshold t. The tuning of the parameter r is strictly related to the size of the steps Δs and will be addressed in section 3.4.

Differently from [3], we choose to approximate multiple Heaviside functions (one for each threshold) with the same number of sigmoidal functions. In this way our loss provides abundant gradients for all relevant positive and negative pairs. Using the approximation from Equation 6, we

can re-write Equations 2 and 3 as follows:

$$T^*(t) = \frac{1}{N_P}\sum_{i}^{N_P}\sigma_r(f(a_i,p_i),t),$$
$$F^*(t) = \frac{1}{N_N}\sum_{j}^{N_N}\sigma_r(f(a_j,n_j),t). \quad (7)$$

Finally, we can substitute $T(\cdot)$ and $F(\cdot)$ from Equation 5 with their respective approximations $T^*(\cdot)$ and $F^*(\cdot)$ (Equations 7), and for sake of simplicity use shorter notation f_{p_i} instead of $f(a_i,p_i)$, and f_{n_i} instead of $f(a_i,n_i)$, we obtain the following differentiable AUC formula:

$$A^{**} = \sum_{s=t_{min}}^{t_{max}-\Delta s}\frac{1}{2N_P}\sum_{i=1}^{N_P}(\sigma_r(f_{p_i},s)+\sigma_r(f_{p_i},s+\Delta s))$$
$$\frac{1}{N_N}\sum_{i=1}^{N_N}(\sigma_r(f_{n_i},s)-\sigma_r(f_{n_i},s+\Delta s)). \quad (8)$$

3.2.3 Theoretical analysis of the accuracy of the AUC approximation

The sigmoidal approximation introduces a zero mean error, being perfectly symmetric around zero. The absolute error is the integral of the absolute difference between the sigmoidal function and the heaviside function. With simple calculus, the absolute error is $err_s(r) = \frac{2\log(2)}{r}$, thus inversely proportional to the slope r. Using $r = 42.2$, the total absolute error is merely $err_s(42.2) = 0.001427$ which, when compared with the Heaviside function in the range [-1, 1], in terms of area, corresponds to an absolute relative error of 0.1427%.

The trapezoidal approximation has an error that depends on the number of integration points and the second order derivative of the continuous function. The domain of the AUC function f is [0, 1]; in this case there exists a value $\epsilon \in [0,1]$, such that the integration error is bounded by the maximum $err_i = -\frac{f''(\epsilon)}{12N_p^2}$, where N_p is the number of points used in the integration. Thus the error decreases quadratically with N_p and linearly increases proportional to the second order derivative of the continuous unknown AUC. Using $\Delta s = 0.05$ (see section 4.3.1), we have $N_p = 41$ points, thus $err_i = 0,0000495 f''(\epsilon)$. Such an error is relevant only if the AUC curve is extremely non-smooth, with large values for the second order derivative, which is very unlikely for a real-case scenario AUC curve.

3.3. AUC loss function

Equation 8 is a derivable approximation of the area under the ROC curve on a set of positive and negative pairs. Ideally, this equation is applied on the whole dataset to calculate a very tight approximation of the real area under the ROC curve. However, accessing the whole dataset in one training step is computationally expensive. Therefore, we calculate the approximated area under the ROC curve based on the samples available at the mini-batch level.

Inspired by [16], we create mini-batches out of k samples from each of l classes, and explore two different strategies for calculating the loss: *batch all* and *batch hard*. In *batch all* strategy we use all positive and all negative pairs when calculating the loss. The *batch hard* strategy calculates the loss based only on the similarities of the hardest positive and negative samples for each sample from the mini batch. If $N = kl$ is the mini-batch size, we can write the *batch all* and *batch hard* AUC losses as following:

$$\mathcal{L}_{AUC_{BH}} = 1 - \frac{1}{2N^2}\sum_{s=t_{min}}^{t_{max}-\Delta s}\sum_{i=1}^{N}(\sigma_r(f_{p_i},s)+\sigma_r(f_{p_i},s+\Delta s))$$
$$\sum_{i=1}^{N}(\sigma_r(f_{n_i},s)-\sigma_r(f_{n_i},s+\Delta s)). \quad (9)$$

Even though the *batch all* strategy takes into account all pairs from the mini-batch, it leads to a much weaker underestimate of the AUC w.r.t the the *batch hard* strategy. Additionally, the best scenario of training a model with AUC_{BA} would require batch creation where the number of all positive pairs would be the same as the number of all negative pairs, which is impossible using *batch all*. AUC_{BH} maximizes an underestimation of the area under the full ROC curve on a mini-batch level. We show the experimental comparison of the two strategies in section 4.3.2.

The AUC_{BH} loss defined in Formula 9 can be seen as a pairwise loss, as it is calculated based on similarities of image pairs. However, what makes AUC_{BH} different from the other pairwise losses is that it does not directly optimize the relations between positive and negative pairs, but rather maximizes the approximated area under the ROC curve based on pair similarities. The full algorithm and its detailed explanation can be found in the supplementary materials.

3.4. AUC metaparameters

The AUC loss function, as defined in Equation 9 has two metaparameters: 1) step size Δs, and 2) slope of the sigmoid function r. The step size is a relevant parameter, and the smaller the step, the more accurate the approximation of the integral.

The setting of r parameter in Equation 6 is of vital importance for the proposed approach. The value of r should be large enough to ensure a good approximation of the Heaviside function while providing useful and well-balanced gradients for a gradient-based optimization strategy. Fig. 2

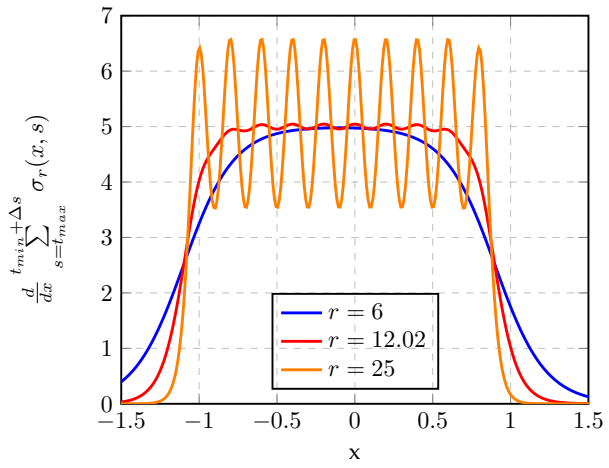

Figure 2. First order derivative of sum of sigmoids for $\Delta s = 0.2$.

Table 1. Optimal r for a set of Δs parameters.

Δs	0.01	0.02	0.05	0.1	0.2
r	201.0	101.0	42.2	22.47	12.02

shows the first order derivative of the sum of sigmoidal functions over x for different values of r on Fig. 2. Small r leads to flat gradient magnitudes around the middle of the range, while significantly decreasing the magnitude close to the edge of the range (blue line). On the other hand, having a large r introduces oscillations of the magnitude of the gradients on the whole input range (orange line). The approximation of the integral over t with a discrete summation can generate larger gradients for thresholds t that are close to the points of the grid if the slope of the sigmoidal-like function is too large. For this reason we would like to find the parameter r for which the square of the second order derivative of the summation of all sigmoidal-like functions over x is minimal[1]:

$$r = \arg\min_{s} \int_{t_{min}}^{t_{max}} \left(\frac{d^2 \sum_{s=t_{max}}^{t_{min}+\Delta s} \sigma_r(x,s)}{dx^2} \right)^2 dx. \quad (10)$$

In such a way, we force the magnitudes of the gradients generated for all values of x to be almost independent of the relative position of x to the grid point s. We find a non-degenerate local minimum[2] of Equation 10 numerically for a set of Δs parameters (see Table 1). This setting, for $\Delta s = 0.2$, is presented by the red line in Fig. 2.

[1]The second order derivative of sigmoidal function is not always positive, so we use its square for calculating the minimal oscillations.

[2]The trivial solution of this equation is $r = 0$, which is degenerate and thus unacceptable.

4. Emperical evidence

In this section we analyze the performance of the AUC loss under different settings and training parameters. We perform experiments with different step size Δs, image size, and we compare the two AUC strategies: *batch all* and *batch hard*. We evaluate the performance of the AUC loss on four publicly available datasets and compare the results with current state-of-the-art approaches.

4.1. Datasets

Stanford Online Products (SOP) [28] is a widely used metric learning dataset containing 120k images of 22.6k products. The dataset is split into two partitions: the training one containing 59.5k images of 11.3k products, and the testing one with 60.5k images of 11.3k products.

DeepFashion - In-Shop Clothes Retrieval [22] is part of DeepFashion which is designed for instance image retrieval. It contains 54.6k images of 11.7k clothing items.

Caltech-UCSD Birds 200 (CUB-200) [40] is a small dataset that is commonly used for image retrieval. It has 6033 images of 200 categories of birds. Following the common practice for the retrieval task, we use the first 100 categories for training, and the rest for testing. Additionally, we use bounding boxes that are provided by the authors during both training and testing.

VERI-Wild [23] is a re-identification dataset of vehicles in the wild. The images are captured by 174 surveillance cameras during one month, resulting in 277,797 images of 30,671 training identities, and three testing partitions.

4.2. Implementation details

In all the experiments we use ResNet50 as a backbone architecture, and we initialize it with the ImageNet pre-trained weights, as it is common practice for metric learning training [26]. We take the output of the last convolutional layer and apply global max pooling to obtain a feature vector for each input image. We reduce the size of the feature vector to 512 by an orthogonally initialized fully connected layer. Finally, we l_2 normalize the vector. This normlization projects all vectors to a hypersphere which allows using the dot product for calculating vector to vector similarities.

We train our models on large scale datasets (all except CUB-200) by using the ADAM optimizer with initial learning rate 10^{-4}, with a decay of 0.9 every $10,000$ steps. When training a model on a small dataset, such as CUB-200, using the ADAM optimizer is not appropriate, as it could lead to overfitting. Therefore, we use the SGD optimizer with initial learning rate 10^{-3} which is decayed by 0.1 each $3,000$ steps.

We create each mini-batch out of 128 images, 2 images for each of 64 classes/identities, if not stated differently. All images in a mini-batch are resized to either 224×224 or

Table 2. Validation r@1 as a function of Δs tested on SOP dataset.

Δs	0.01	0.02	0.05	0.1	0.2
$r@1[\%]$	79.11	79.06	78.97	77.46	74.67

Table 3. Comparison of *batch all* and *batch hard* strategies on Stanford Online Products [28] dataset.

	im size	mb size	$R@1$	$R@10$
AUC BA^{512}_{R50}	224x224	190	64.70	80.45
AUC BH^{512}_{R50}	224x224	190	75.72	89.13

256×256. In all the experiments we augment one of the two images per class in a mini-batch. We use horizontal flipping, cutout, zoom-in/out, color shift and motion blur as augmentation techniques.

4.3. AUC loss analysis

4.3.1 Δs parameter

Δs parameter. Here we analyse the impact of Δs parameter from Equation 9 on a model trained on Stanford Online Products dataset, following the implementation details in Section 4.2.

We compare the retrieval performances, represented by rank@1 of models trained by optimizing the AUC loss function for different step sizes Δs and show the results in Table 2. The smaller the Δs, the more precise the approximation of the integral, which directly implies better performance with negligible memory overhead. When Δs is equal to 0.2, the ROC curve is approximated with 11 points, which is a poor approximation. On the contrary, by setting Δs to 0.01 we obtain a better approximation of the curve based on 201 points. On the other hand, Δs of 0.05 already represents the curve by 41 points, which is precise enough, and therefore we set Δs to 0.05 in our experiments.

The *memory requirements of the AUC loss* are negligible with respect to the memory needed for storing the backbone architecture. As an example, having a mini-batch of 128, and embedding size 512, the memory required for calculating the loss function for $\Delta s = 0.01$ is $\approx$ 1MB.

4.3.2 *Batch all* vs *batch hard* strategies.

We trained two models on Stanford Online Products dataset, under the same conditions (as described in section 4.2), and we created mini-batches of 190 images. We randomly sampled a set of 10 different classes, and from each one of them we chose 19 images randomly, which results in having $N_N = 16242$ negative, and $N_P = 1710$ positive pairs. If a chosen class has less than 19 images, we sampled some images more than once. As shown in Table 3, the AUC_{BH} optimizes a stronger underestimation of the area, thus providing stronger and better gradients during the training.

4.3.3 Computational cost

We report the numerical results of direct time measurement of both triplet loss and AUC loss for 1000 training steps (including forward and backward propagation) on a TITAN X with 12GB of RAM. $Triplet_{BH}$ takes 888.5 second while AUC_{BH} 887.7 seconds, which makes the computational cost of the new AUC loss the same as the computational cost of the triplet loss when employing the BH strategy.

4.3.4 AUC loss evaluation

In this section we compare the AUC loss with the triplet *batch hard* loss, as this loss is a milestone for metric learning, and it was also found to obtain still competitive results with state-of-the-art in a recent survey paper [26]. We followed the same training parameters as described in 4.2 and used images resized to 224x224, and compared the results on four publicly available datasets, as shown in Tables 4 and 5. Taking into account that the AUC loss does not have extra hyper parameters for tuning, and that the influence of the margin when training a model with the triplet *batch hard* loss is minimal [16], we set the margin for the triplet loss to 0.3.

The AUC loss outperforms the triplet *batch hard* by a large margin on SOP, inShop, CUB-200 and VERI-Wild datasets without additional hyper parameter tuning or any increase in the computational cost w.r.t. the triplet loss. In addition to comparing the numerical results of ranking in terms of mAP, rank1 and rank10, we compared the area under the ROC curves for all models. This measure evaluates the separability between different classes, and confirms that the AUC loss provides a more robust model with better class separability.

In order to provide a fair comparison with the method introduced in [3], we implemented a variation of the loss that is originally proposed and used for *classification* based on the Wilcoxon statistic. We employ the *batch hard* strategy by using only the hardest positive and hardest negative pairs for each input image when calculating the loss:

$$\mathcal{L}_{Wilcoxon} = 1 - \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma(f_{p_i} - f_{n_j}). \quad (11)$$

We compare the Wilcoxon loss with AUC under the same experimental settings on four metric learning datasets, and present results in Tables 4 and 5. The AUC loss provides significantly better results on all datasets. We believe that the main advantage of AUC with respect to Wilcoxon statistics is that AUC loss relies on *a family of sigmoidal functions* (one for each threshold in the curve), while Wilcoxon

Table 4. Comparison of the AUC, Wilcoxon and the triplet *batch hard* loss functions on the Stanford Online Products [28], In-shop Clothes [22] and CUB-200-2011 [40] datasets.

	SOP			inShop			CUB-200-2011			CUB-200-2011_crop		
	$R@1$	$R@10$	AUC	$R@1$	$R@10$	AUC	$R@1$	$R@8$	AUC	$R@1$	$R@8$	AUC
Triplet-BH	74.90	87.98	98.98	89.81	97.24	98.62	53.79	84.26	87.51	62.57	90.27	90.34
Wilcoxon-BH	70.22	84.38	98.07	82.63	94.18	96.28	44.59	76.77	79.10	51.20	84.59	83.75
AUC	**78.97**	**91.11**	**99.00**	**91.04**	**97.84**	**98.88**	**58.19**	**86.36**	**88.89**	**68.28**	**92.31**	**92.76**

Table 5. Comparison of the AUC, Wilcoxon and the triplet *batch hard* loss functions on the VERI-Wild [23] dataset.

	VERI-Wild_small			VERI-Wild_medium			VERI-Wild_large		
	mAP	$R@1$	AUC	mAP	$R@1$	AUC	mAP	$R@1$	AUC
Triplet-BH	70.79	84.13	99.88	62.40	77.78	99.88	51.12	70.00	99.89
Wilcoxon-BH	47.76	70.5	99.57	40.06	63.48	99.62	30.65	54.56	99.59
AUC	**75.66**	**89.26**	**99.89**	**68.49**	**84.76**	**99.89**	**58.81**	**79.49**	**99.89**

statistics approximates the area under the ROC curve based on the results of a single sigmoidal function.

Table 6. Comparison with the state-of-the-art on the Stanford Online Products [28] dataset. Embedding dimension is presented as a superscript and the backbone architecture as a subscript. R stands for ResNet, G for GoogLeNet, I for Inception.

	im	mb	$R@1$	$R@10$
Histogram Loss$^{512}_{G}$ [37]	256	128	63.9	81.7
Binomial Deviance$^{512}_{G}$ [37]	256	128	65.5	82.3
N-Pair-Loss$^{512}_{G}$ [34]	-	120	67.7	83.8
Clustering$^{64}_{G}$ [27]	227	128	67.0	83.7
Angular Loss$^{512}_{G}$ [38]	256	128	70.9	85.0
HDC$^{384}_{G}$ [42]	-	100	69.5	84.4
Margin$^{128}_{R50}$ [24]	224	80	72.7	86.2
A-BIER$^{512}_{G}$ [29]	224	-	74.2	86.9
HTL$^{128}_{G}$ [10]	224	50	74.8	88.3
ABE-8$^{512}_{G}$ [21]	224	64	76.3	88.4
FastAP$^{512}_{R50}$ [2]	224	256	76.4	89.1
RaMBO$^{512}_{R50}$ log log [31]	224	128	78.6	90.5
RankMI$^{128}_{R50}$ [19]	-	120	74.3	87.9
R-Margin$^{128}_{R50}$ [32]	224	160	78.5	-
MS$^{512}_{I}$ [39]	224	640	78.2	90.5
AUC$^{512}_{R50}$	224	128	**78.97**	**91.11**
AUC$^{512}_{R50}$	256	128	**80.32**	**91.89**

4.4. SOTA comparison

Tables 6 - 8 present extensive comparison with state-of-the-art methods on four publicly available retrieval and re-identification datasets in terms of recall@k and mean average precision (mAP), as well as image size, mini-batch size, backbone architecture and embedding size.

Table 7. Comparison with the state-of-the-art methods on the In-shop Clothes [22] dataset. Embedding dimension is presented as a superscript and the backbone architecture as a subscript. R stands for ResNet, G for GoogLeNet, V for VGG.

	im	mb	$R@1$	$R@10$
FashionNet$_V$ [22]	-	-	53.0	73.0
HDC$^{384}_{G}$ [42]	224	100	62.1	84.9
DREML$^{48}_{R18}$ [41]	256	128	78.4	93.7
HTL$^{128}_{G}$ [10]	224	650	80.9	94.3
A-BIER$^{512}_{G}$ [29]	224	-	83.1	95.1
ABE-8$^{512}_{G}$ [21]	224	64	87.3	96.7
FastAP$^{512}_{R50}$ [2]	224	256	90.9	97.7
RaMBO$^{512}_{R50}$ log log [31]	224	128	86.3	96.2
MS$^{512}_{I}$ [39]	224	640	89.7	97.9
AUC$^{512}_{R50}$	224	128	**91.04**	**97.84**
AUC$^{512}_{R50}$	256	128	**91.01**	**97.90**

We show the results on the Stanford Online Products dataset in table 6. Due to diversity of product categories (bicycle, chair, lamp etc), the methods trained on this dataset are not domain specific. Our method achieves state-of-the-art performance when trained with images that are resized and cropped to 224x224 pixels, which is comparable with the image size used in the majority of state-of-the-art methods. AUC outperforms all state-of-the-art methods, including the ensemble methods such as A-BIER [29], ABE-8 [21] and HDC [42]. AUC achieves better performance than FastAP [2], even though FastAP uses information about category when sampling images for batches, while splitting them into smaller chunks for training with effectively bigger batch size. We further improve the performance of our method by using images of size 256x256.

Another dataset appropriate for image retrieval is Deep-

Table 8. Comparison with the state-of-the-art methods on the VERI-Wild [23] dataset. Embedding dimension is presented as a superscript and the backbone architecture as a subscript. R stands for ResNet, A for ad-hoc, M for MobileNet

			VERI-Wild_small		VERI-Wild_medium		VERI-Wild_large	
	im size	mb size	mAP	$R@1$	mAP	$R@1$	mAP	$R@1$
Veri-wild$^{1024}_{A}$ [23]	224	-	35.11	64.03	29.80	57.82	22.78	49.43
MLSL$^{1024}_{M}$ [1]	224	24	46.32	86.03	42.37	83.00	36.61	77.51
PGAN$^{512}_{R50}$ [43]	224	64	-	-	-	-	**74.10**	**93.80**
GLAMOR$^{512}_{R18}$ [35]	208	36	77.15	92.13	-	-	-	-
AUC$^{512}_{R50}$	224	128	75.66	89.26	68.49	84.76	58.81	79.49
AUC-BoN$^{512}_{R50}$	224	128	**80.31**	**93.50**	**74.55**	**91.22**	66.47	88.36
SAVER$^{2048}_{R50}$ [20]	256	-	**83.40**	**96.90**	**78.70**	**96.00**	71.30	94.10
SAFR$^{2048}_{R50}$ [36]	350	72	-	-	-	-	**77.90**	**92.10**
UMTS$^{512}_{R50}$ [18]	256	64	72.70	84.50	66.10	79.30	54.20	72.80
PVEN$^{2048}_{R50}$ [25]	256	8	82.50	-	77.00	-	69.70	-
AUC-BoN$^{512}_{R50}$	256	128	82.14	94.43	76.68	92.18	68.87	89.15

Fashion In-Shop. We trained models with images resized to 224x224 and 256x256, and they achieved comparable results. We believe that the bigger image size does not provide relevant benefits on this dataset because there is not much room for improvement even when trained with small images. In Table 7 we show that we achieve state-of-the-art results on this dataset. The only method that achieves results comparabale to AUC is FastAP, while using a mini-batch twice as big as ours.

Finally, we tested our method on VERI-Wild dataset, which is used for vehicle re-identification. The majority of state-of-the-art models combine several loss functions for achieving better results (e.g. [1, 20, 23, 25, 35, 36, 43]). Additionally, several methods that are evaluated on this dataset use some domain specific information during training, such as position of the mirrors and wheels, color of the vehicle, side and front view etc. Even though our method is simpler and not domain-specific, its performance is comparable with domain specific state-of-the-art approaches, as shown in Table 8. Taking into account that all images in this dataset have the same characteristics (they are all vehicles) and that the number of classes is much greater than the mini-batch size, we combine AUC with BoN, a state-of-the-art method for hard negative sampling [8, 9]. This combination significantly improves the performance of AUC alone. Finally, we train a model with AUC-BoN using bigger input images, which further improve the results.

The model that achieves state-of-the-art results is SAFR [36], and it uses significantly bigger input images than the ones that we use in the experiments (350x350 pixels) and embedding size 2048 and its implementation exceeds the hardware limitations that we have. Additionally, this model uses three loss functions: smoothed softmax, triplet and center loss, as well as an unsupervised attention network. SAVER [20] is another method that performs slightly better than AUC-BoN. However, this model uses a more complicated network architecture that contains variational autoencoder, together with ResNet50 backbone, and a combination of cross entropy and triplet losses.

In conclusion, the AUC loss outperforms state-of-the-art methods on large scale retrieval datasets, and is comparable with more complex models used for vehicle re-identification.

5. Conclusion and Future Works

In this paper we confirmed the hypothesis that training a model by maximizing the area under the ROC curve can be beneficial for retrieval. We presented a new loss that maximizes a derivable variant of the area under the ROC curve. The AUC loss achieves better or comparable results to more complex state-of-the-art methods on large datasets, without the need of hyper parameter tuning. The main limitation of the AUC loss is that it achieves lower accuracy on small datasets, as discussed in the supplementary material. We presented the performance of the AUC loss on metric learning problems, but our future work will include variants of this loss which could potentially bring benefits to other problems of computer vision, such as image classification and object detection. Additionally, we will explore the usage of this loss for other types of data, such as audio and video.

Ethical concerns: As any other general metric learning algorithm, the AUC loss can be used for face recognition and re-identification, that can possibly lead to mass surveillance.

References

[1] Saghir Alfasly, Yongjian Hu, Haoliang Li, Tiancai Liang, Xiaofeng Jin, Beibei Liu, and Qingli Zhao. Multi-label-based

similarity learning for vehicle re-identification. *IEEE Access*, 7:162605–162616, 2019. 8

[2] Fatih Cakir, Kun He, Xide Xia, Brian Kulis, and Stan Sclaroff. Deep metric learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1861–1870, 2019. 7

[3] Toon Calders and Szymon Jaroszewicz. Efficient auc optimization for classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 42–53. Springer, 2007. 1, 3, 6

[4] Weihua Chen, Xiaotang Chen, Jianguo Zhang, and Kaiqi Huang. Beyond triplet loss: a deep quadruplet network for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2017. 1, 2

[5] Sumit Chopra, Raia Hadsell, Yann LeCun, et al. Learning a similarity metric discriminatively, with application to face verification. In *CVPR (1)*, pages 539–546, 2005. 1

[6] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006. 1

[7] Yi Ding, Peilin Zhao, Steven Hoi, and Yew-Soon Ong. An adaptive gradient method for online auc maximization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015. 2

[8] Bojana Gajic, Ariel Amato, Ramon Baldrich, and Carlo Gatta. Bag of negatives for siamese architectures. In *Proc. BMVC*, 2019. 8

[9] Bojana Gajic, Ariel Amato, and Carlo Gatta. Fast hard negative mining for deep metric learning. In *PR*, 2020. 8

[10] Weifeng Ge, Weilin Huang, Dengke Dong, and Matthew R Scott. Deep metric learning with hierarchical triplet loss. In *Proc. ECCV*, 2018. 7

[11] David Marvin Green, John A Swets, et al. *Signal detection theory and psychophysics*, volume 1. Wiley New York, 1966. 1

[12] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006. 2

[13] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982. 1

[14] Kun He, Fatih Cakir, Sarah Adel Bargal, and Stan Sclaroff. Hashing as tie-aware learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4023–4032, 2018. 2

[15] Kun He, Yan Lu, and Stan Sclaroff. Local descriptors optimized for average precision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[16] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 2, 4, 6

[17] Jing Huo, Yang Gao, Yinghuan Shi, and Hujun Yin. Cross-modal metric learning for auc optimization. *IEEE transactions on neural networks and learning systems*, 29(10):4844–4856, 2018. 2

[18] Xin Jin, Cuiling Lan, Wenjun Zeng, and Zhibo Chen. Uncertainty-aware multi-shot knowledge distillation for image-based object re-identification. *arXiv preprint arXiv:2001.05197*, 2020. 8

[19] 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. 7

[20] Pirazh Khorramshahi, Neehar Peri, Jun-cheng Chen, and Rama Chellappa. The devil is in the details: Self-supervised attention for vehicle re-identification. *arXiv preprint arXiv:2004.06271*, 2020. 8

[21] Wonsik Kim, Bhavya Goyal, Kunal Chawla, Jungmin Lee, and Keunjoo Kwon. Attention-based ensemble for deep metric learning. In *Proc. ECCV*, 2018. 7

[22] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proc. CVPR*, 2016. 5, 7

[23] Yihang Lou, Yan Bai, Jun Liu, Shiqi Wang, and Lingyu Duan. Veri-wild: A large dataset and a new method for vehicle re-identification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3235–3243, 2019. 5, 7, 8

[24] R. Manmatha, Chao-Yuan Wu, Alexander J. Smola, and Philipp Krähenbühl. Sampling matters in deep embedding learning. In *Proc. ICCV*, 2017. 7

[25] Dechao Meng, Liang Li, Xuejing Liu, Yadong Li, Shijie Yang, Zheng-Jun Zha, Xingyu Gao, Shuhui Wang, and Qingming Huang. Parsing-based view-aware embedding network for vehicle re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7103–7112, 2020. 8

[26] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020. 5, 6

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

[28] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proc. CVPR*, 2016. 2, 5, 6, 7

[29] 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. 7

[30] Jerome Revaud, Jon Almazan, Rafael S. Rezende, and Cesar Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1, 2

[31] 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. 7

[32] 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. 7

[33] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proc. CVPR*, 2015. 1, 2

[34] 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, 7

[35] Abhijit Suprem and Calton Pu. Looking glamorous: Vehicle re-id in heterogeneous cameras networks with global and local attention. *arXiv preprint arXiv:2002.02256*, 2020. 8

[36] Abhijit Suprem, Calton Pu, and Joao Eduardo Ferreira. Small, accurate, and fast vehicle re-id on the edge: the safr approach. *arXiv preprint arXiv:2001.08895*, 2020. 8

[37] Evgeniya Ustinova and Victor Lempitsky. Learning deep embeddings with histogram loss. In *Advances in Neural Information Processing Systems*, pages 4170–4178, 2016. 2, 7

[38] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017. 7

[39] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R Scott. Multi-similarity loss with general pair weighting for deep metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5022–5030, 2019. 7

[40] 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. 5, 7

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

[42] 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. 7

[43] Xinyu Zhang, Rufeng Zhang, Jiewei Cao, Dong Gong, Mingyu You, and Chunhua Shen. Part-guided attention learning for vehicle re-identification. *arXiv preprint arXiv:1909.06023*, 2019. 8

[44] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Advances in neural information processing systems*, pages 8778–8788, 2018. 1

[45] Peilin Zhao, Steven CH Hoi, Rong Jin, and Tianbo YANG. Online auc maximization. 2011. 2