

Smooth-AP: Smoothing the Path Towards Large-Scale Image Retrieval

Supplementary Material

Andrew Brown^[0000-0002-9556-2633], Weidi Xie^[0000-0003-3804-2639],
Vicky Kalogeiton^[0000-0002-7368-6993], and
Andrew Zisserman^[0000-0002-8945-8573]

Visual Geometry Group, University of Oxford
{abrown, weidi, vicky, az}@robots.ox.ac.uk
<https://www.robots.ox.ac.uk/~vgg/research/smooth-ap/>

Table of Contents

1	Further qualitative results	1
2	Source code	8
3	Details on the effects of increasing the mini-batch size on Smooth-AP	8
4	Choice of hyper-parameters for the compared-to methods for the INaturalist experiments	10
5	Complexity of the proposed loss	11

1 Further qualitative results

In this Section, we provide additional qualitative results for four of the datasets used in our experiments (VGGFace2 Test set, Stanford Online Products, and INaturalist). For each dataset, we display the top retrieved instances for various queries from the baseline model, both with and without appending the Smooth-AP loss. In all cases, the query example is shown in blue. The retrieved instances belonging to the same class as the query (*i.e.* positive set) are shown in green, while the ones belonging to a different class from the query (*i.e.* negative set) are shown in red. For all retrieval examples we have shown the corresponding precision-recall curves below, in which the baseline model is represented in blue, and the Smooth-AP model for the same query instance is represented overlaid in green. For Figures S1, S2 the retrieval set is ranked from left to right starting in the top row next to the query. For Figures S3, S4, S5, S6 the retrieval set is ranked from top to bottom. In each case, the Average Precision (AP) computed over the whole retrieval set is provided either below or alongside the ranked instances.



Fig. S1: Qualitative results from the VGGFace2 Test set for the SENet-50 [2] baseline model. We show a query instance (blue) and the first 79 ranked instances in the retrieval set for the baseline model both before and after Smooth-AP was appended (ranked from left to right, starting next to the query). As shown by the precision-recall curves, Smooth-AP causes the Average Precision to jump by an impressive 52.9%, and the number of false positives (shown in red) in the top ranked retrieved instances drops considerably. The size of the positive set for each instance in the VGGFace2 test set, $|P| \approx 338$.

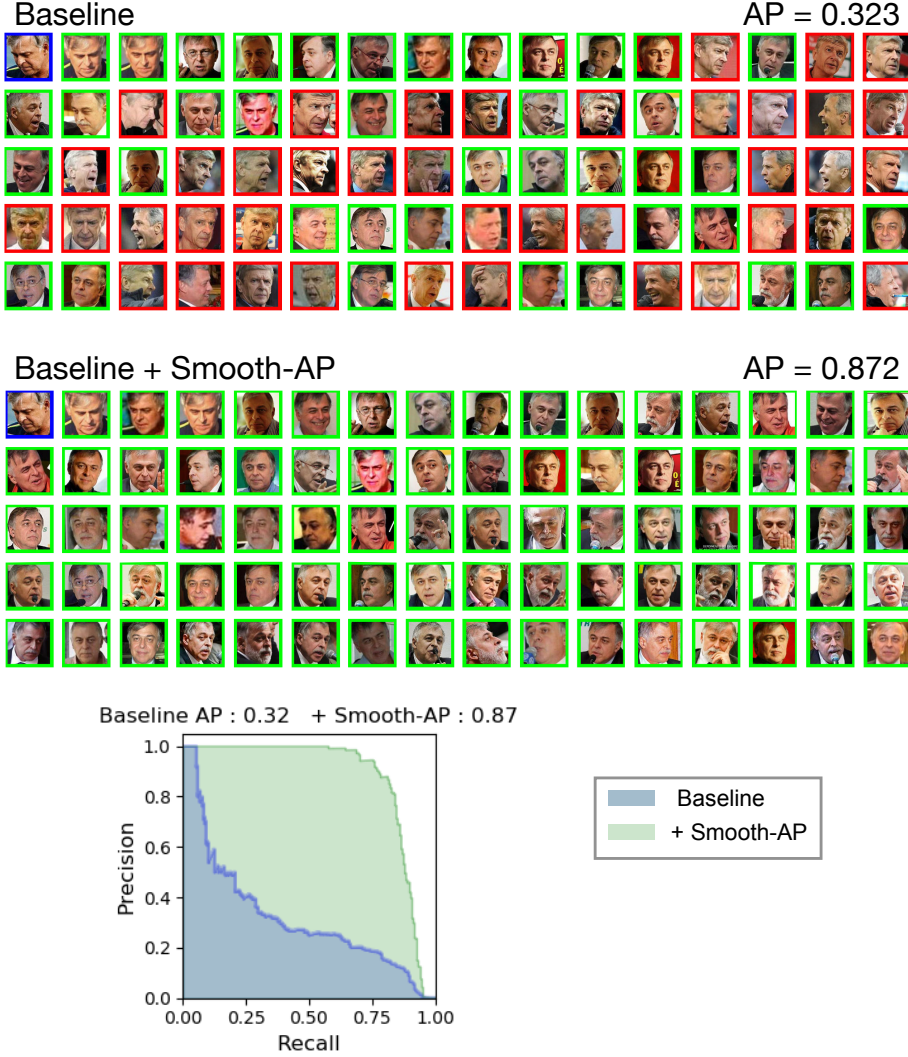


Fig. S2: Here, we show the qualitative results for a second query instance for the SENet-50 [2] baseline model on the VGGFace2 test set, both before and after appending the Smooth-AP loss. Here, we see another large increase in Average Precision of 54.9% caused by the addition of the Smooth-AP loss. We see that all false positives (shown in red) are removed from the top ranked retrieved instances after adding the Smooth-AP loss. Take the situation where each row of top-ranked instances corresponds to pages of retrieved results that a user is presented with when using a retrieval system. With the baseline model, the user comes across many false positives in the first few pages. After appending the Smooth-AP loss, the user encounters no false positives in at least the first five pages. This demonstrates the benefit to *user experience* in appending Smooth-AP to a retrieval network.

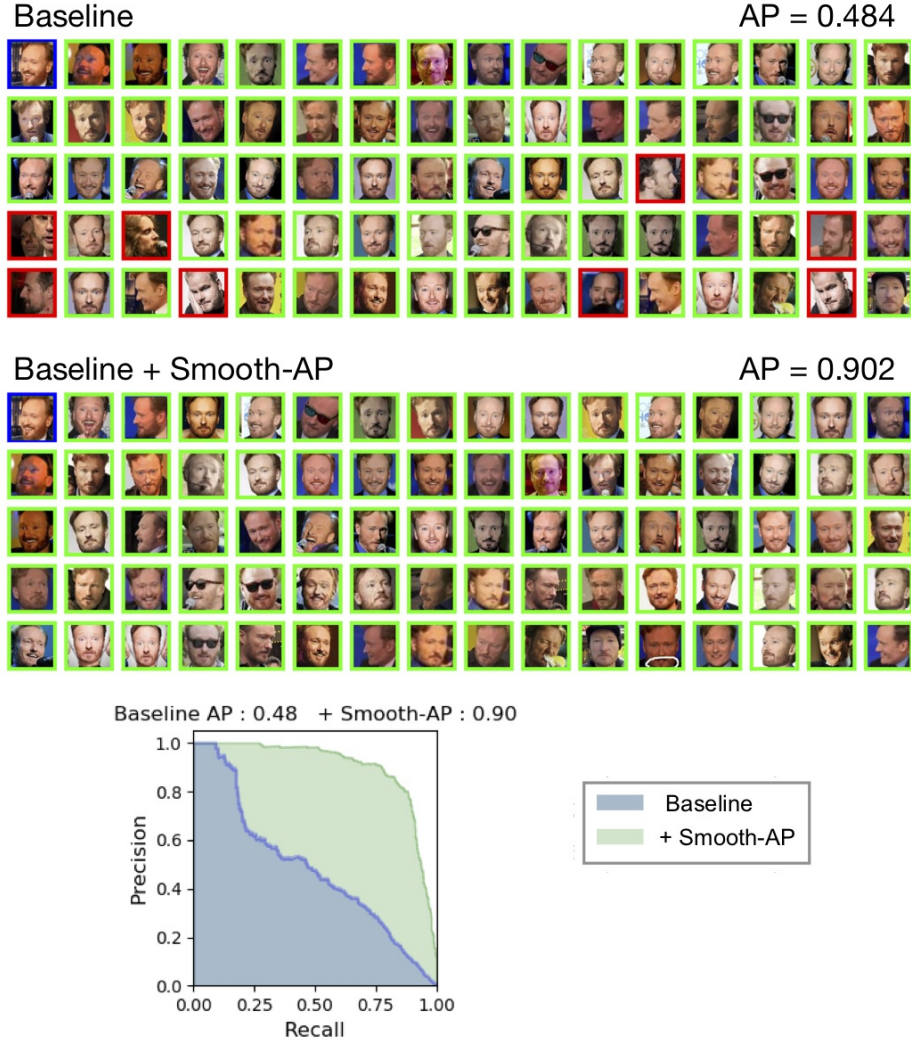


Fig. S3: Here, we show the qualitative results for a third query instance for the SENet-50 [2] baseline model on the VGGFace2 test set, both before and after appending the Smooth-AP loss. We see a large improvement in Average Precision of 41.8% after adding the Smooth-AP loss and the removal of all false positives from the top ranked retrieval results. These results confirm that Smooth-AP is indeed tailored to addressing the ranking issue.

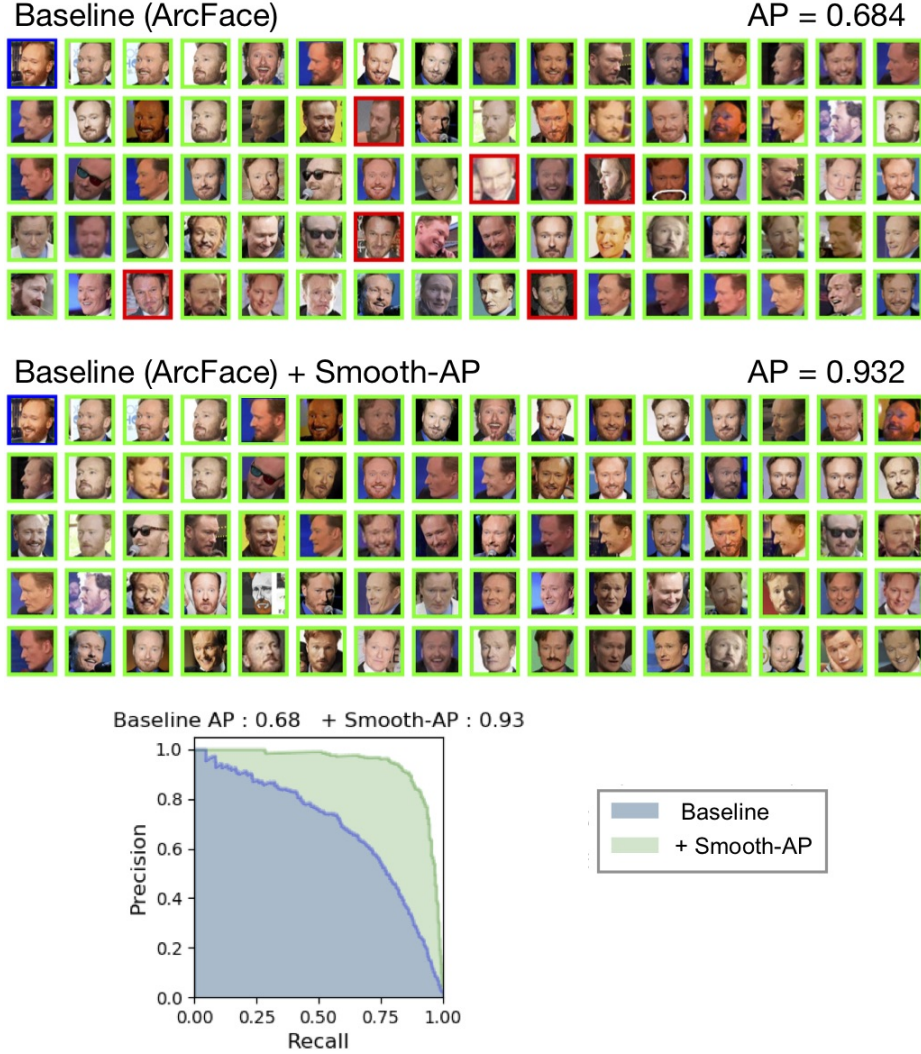


Fig. S4: Here, we show the qualitative results for a query instance from the VGGFace2 test set (same as in Figure S3) for the state-of-the-art ArcFace (ResNet-50) [3] baseline, both before and after appending the Smooth-AP loss. Appending the Smooth-AP loss to this impressive baseline leads to a large gain in Average Precision (24.8%), and again to the removal of all false positives from the top ranked retrieval results. This demonstrates that state-of-the-art face retrieval networks are far from saturated on the Average Precision metric, and through appending the simple Smooth-AP loss, this metric and the resulting user experience when using the face retrieval system can be greatly improved.









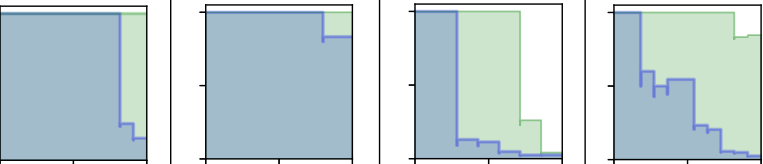
Queries				
Top 13 ranked retrieval results				
	<div>Baseline</div>			
	<div>+ Smooth-AP</div>			
	<div></div>			
	<div></div>			
	<div></div>			
	<div></div>			
	<div></div>			
	<div></div>			
	<div></div>			
	<div></div>			
	<div></div>			
	<div></div>			
AP	0.8541.000	0.9671.000	0.3350.758	0.4330.971
Precision-Recall curves				
	(a)	(b)	(c)	(d)

Fig.S5: This Figure shows four separate query instances from the online products dataset and the top ranked instances from the retrieval set when using the baseline model (ImageNet pre-trained weights), and after appending the Smooth-AP loss. It is noted that for this dataset, the size of the positive set ($|P|$) in the retrieval set is very small ($|P| = 11, 5, 7, 11$ for (a),(b),(c),(d) respectively), and so for cases (a) and (b) all positive instances are shown correctly retrieved above all false positives (also indicated by the AP=1.00) for the Smooth-AP model. Particularly *impressive* are the examples (b) and (c), where instances from the positive set which depict a far different pose from the query are retrieved above false positives that are very visually similar to the query.
























































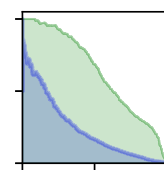
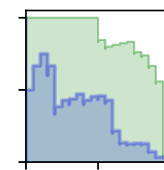
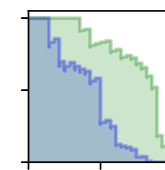
Queries						
Top 9 ranked retrieval results						
						
						
						
						
						
						
						
						
AP	0.240	0.637	0.348	0.844	0.435	0.774
Precision-Recall curves						
	(a)	(b)	(c)			

Fig. S6: This Figure shows three separate query instances from the INaturalist dataset and the top ranked instances from the retrieval set when using the baseline model (ImageNet pre-trained weights), and after appending the Smooth-AP loss. As can be seen by the false positive retrieved instances for the baseline model, this is a highly challenging, fine-grained dataset; yet in all cases shown, appending the Smooth-AP loss leads to large gains in Average Precision. $|P| = 985, 20, 28$ for (a),(b),(c) respectively.

2 Source code

Here, we provide pseudocode for the Smooth-AP loss in Table S1 written in PyTorch style. The simplicity of the method is demonstrated by the short implementation.

Algorithm: Pseudocode for Smooth-AP in Pytorch-style

```
# scores: predicted relevance scores (1 x m)
# gt: groundtruth relevance scores (1 x m)

def t_sigmoid(tensor, tau=1.0):
    # tau is the temperature.
    exponent = -tensor / tau
    y = 1.0 / (1.0 + exp(exponent))
    return y

def smooth_ap(scores, gt):
    # repeat the number row-wise.
    s1 = scores.repeat(m, 1) # s1: m x m
    # repeat the number column-wise.
    s2 = s1.transpose # s2: m x m
    # compute difference matrix
    D = s1 - s2
    # approximating heaviside
    D_ = t_sigmoid(D, tau=0.01)
    # ranking of each instance
    R = 1 + sum(D_ * (1-eye(m)), 1)
    # compute positive ranking
    R_pos = gt.T * R
    # compute AP
    AP = (1 / sum(gt)) * sum(R_pos / R)
    return 1-AP
```

Table S1: pseudocode for the proposed Smooth-AP loss in PyTorch style. The method is very simple to implement and takes only a few lines of code.

3 Details on the effects of increasing the mini-batch size on Smooth-AP

In this Section, we provide a further quantitative validation of the claims made in Section 6.4 (in the main manuscript) about the effects of mini-batch size on the Smooth-AP loss. We conjecture that a large mini-batch size increases the likelihood of relevance scores in the mini-batch being close to each other, and hence elements of the difference matrix (Equation 4 in main manuscript) falling into the narrow operating region of the sigmoid (we define the the operating region of

the sigmoid as the narrow region with non-negligible gradients, see Figure S7b), meaning that non-negligible gradients are fed backwards from Smooth-AP. This conjecture can be verified by increasing the mini-batch size during training and logging the proportion of elements of the difference matrix that fall into the operating region of the sigmoid. For each mini-batch during training, a difference matrix D is constructed of size $(m * m)$ where m is the mini-batch size. The proportion of elements of D that fall into the operating region of the sigmoid used in Smooth-AP, which we denote as P , can be computed using Equation 1 (we use a value of 0.005 to represent a non-negligible gradient). While keeping all parameters equal except mini-batch size, the average P is computed across all mini-batches in one epoch of training on the Online Products dataset for several different mini-batch sizes, with the results plotted in Figure S7a. As expected, P increases with mini-batch size due to the fact that more instances in a mini-batch means that more instances are close enough together in terms of similarity score to lie within the operating zone of the sigmoid. This in turn leads to more non-negligible gradients being fed backwards to the network weights, and hence a higher evaluation performance, as was shown in the ablation Table 5 in the main manuscript.

$$P = \frac{\sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (|\frac{d\mathcal{G}(D_{ij})}{dD_{ij}}| > 0.005)}{m^2} \quad (1)$$

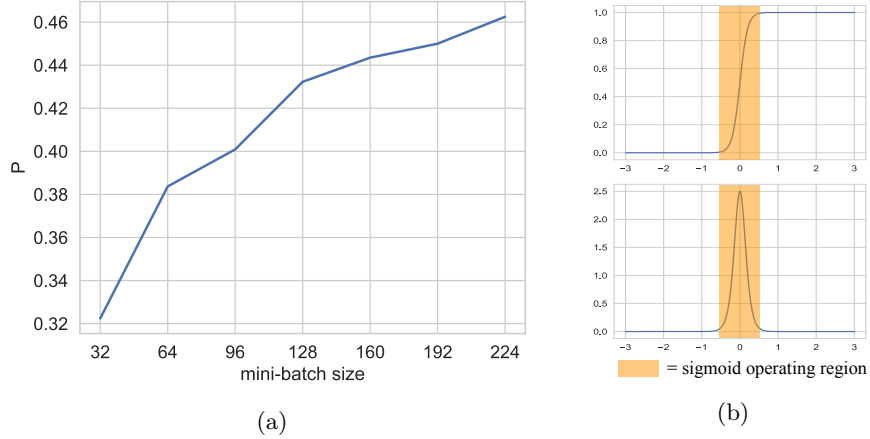


Fig.S7: (a): P , the proportion of elements of the difference matrix that fall into the operating region of the sigmoid (shown in (b)), and hence receive non negligible gradients, for several different mini-batch sizes. This explains why Smooth-AP benefits from large mini-batch sizes.

4 Choice of hyper-parameters for the compared-to methods for the INaturalist experiments

The two AP-optimising methods that we compare to for the INaturalist experiments (Table 3) have several hyper-parameters associated with them. For FastAP [1], there is the number of histogram bins, L , and for Blackbox AP [5], there is the value of λ and the margin. For both methods we choose the hyper-parameters that are recommended in the respective publications for the largest dataset that was experimented on, which would be closest to INaturalist in terms of number of training images. For FastAP the number of histogram bins L is set to 20, and for Blackbox AP, λ is set to 4 and the margin is set to 0.02. We note that the evaluated Recall@K scores might be increased by varying these parameters. For all experiments on the INaturalist dataset, we multiply the learning rate on the last linear layer by a factor of 2.

5 Complexity of the proposed loss

Table S2 shows the time complexities of Smooth-AP, and also the other AP-optimising loss functions that we compare to. We also measure the times of the forward and backward passes for a single training iteration when each of the different loss functions are appended onto a ResNet50 [4] backbone architecture. More specifically, we measure the forwards and backwards pass time for the backbone network, *backbone time*, and the appended loss function, *loss time*. These values for timings are averaged over all iterations in one training epoch for the Online Products dataset. The relevant notation: \mathcal{M} is the number of instances in the retrieval set, which during mini-batch training is equal to the size of the mini-batch (with $p + n = \mathcal{M}$ and p, n the number of positive and negative instances, respectively). L refers to the number of bins used in the Histogram Binning technique [1,6]. Even though the complexity of the proposed Smooth-AP loss is higher, Table S2 shows that this leads to a very small overhead in computational cost on top of the ResNet50 backbone architecture ($< 3ms$ for every iteration compared to previous methods where the backbone takes 705 ms), and hence in practice has a minor impact on the *usability* of the proposed loss function. In all experiments here, all training parameters are equal ($|P| = 4$, and mini-batch size \mathcal{M} of 112).

<i>Method</i>	<i>complexity</i>	<i>backbone time (ms)</i>	<i>loss time (ms)</i>
Blackbox AP [5]	$\mathcal{O}(n \log(n))$	705.0	3.7
FastAP [1]	$\mathcal{O}(\mathcal{M}L)$	705.0	4.2
Smooth-AP	$\mathcal{O}(\mathcal{M}^2)$	705.0	6.6

Table S2: The time complexities of different AP-optimising methods that we compare to, as well as the time taken for the forwards and backwards pass through the backbone for one iteration, *backbone time*, and the time taken for the computation of the loss, *loss time*. The slightly increased time complexity of Smooth-AP leads to a negligible increase in training time.

References

1. Cakir, F., He, K., Xia, X., Kulis, B., Sclaroff, S.: Deep metric learning to rank. In: Proc. CVPR (2019)
2. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: VGGFace2: A dataset for recognising faces across pose and age. In: Proc. Int. Conf. Autom. Face and Gesture Recog. (2018)
3. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: Proc. CVPR (2019)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. CVPR (2016)
5. Rolínek, M., Musil, V., Paulus, A., Vlastelica, M., Michaelis, C., Martius, G.: Optimizing rank-based metrics with blackbox differentiation. In: Proc. CVPR (2020)
6. Ustinova, E., Lempitsky, V.: Learning deep embeddings with histogram loss. In: NeurIPS (2016)