Visual Geo-Localization

Matteo Gambino s287572 | Michele Pierro s287846 | Fabio Grillo s287873

Abstract

In order to predict the location of a query image by retrieving annotated photographs with similar descriptors needs an efficient and reliable generation of those descriptors. In order to accomplish that objective, is fundamental that the network focuses on portion of the various images that contains useful information and at the same time ignore not informative areas like the ones containing elements like cars or pedestrians. For that reason attention layers are fundamental in the proposed network. In addition to that we are comparing state of the art techniques for the visual geo-localization task like GeM [1], NetVLAD [2] and CRN [3]. The code used is publicly available here

1. Introduction

2. Related works

3. Methods

Like [1], [2], [3] we have casted the problem of place recognition as the task of image retrieval. We have implemented 3 different networks all based on the ResNet-18 [4] backbone without the fully connected layers and the last convolution layer. On the top of this backbone we have inserted 4 different heads, inspired by the works of [1], [2], [3], in order to generate the image descriptors.

3.1. Base Head

This is the simplest head we have used and it's necessary in order to have a baseline to compare the other results. After the last convolution layer of ResNet-18 we have normalized the feature map and used average pooling in order to generate the descriptors. This simple head tries to extract from the query the spatial information by comparing the average value of the features in a given area and represent the traditional way to extract those descriptors.

3.2. GeM head

Following the work of [1], we have used a Generalized Mean approach in order to extract better descriptors for the query image. The generalized mean we are using is defined as:

$$f_k = (\frac{1}{X_k} \sum_{x \in X_k} x^{p_k})^{\frac{1}{p_k}} \tag{1}$$

where X_k represent one of the normalized features map and p_k is the pooling parameter. This pooling parameter is expressing how much is localized the zone of the image the network is focusing on. The p_k parameter, although it can be learned and inserted into back propagation, it has been fixed and a single value is used for each activation map as suggested by [1]. We have inserted a fully connected layer that takes as input the pooled features in order to whiten the image descriptors since it has been shown by [1] that this approach is providing better results than using other strategies like PCA.

3.3. NetVLAD head

Inspired by the work presented in [2], we have implemented also a NetVLAD head that solves in an elegant way the problem of computing Vectors of Locally Aggregated Descriptors (VLAD), as described in [5], in CNN. This network, in order to compute those VLAD descriptors, is using two different parts. The first is called soft-assignment branch that is replacing the hard assignment of a descriptor to a single cluster with a soft assignment of the descriptor to every cluster. This is performed using a soft-max operation on top of the output of a $1x1$ convolution layer that produces the probabilities s_k that a given descriptor x_i belongs to a cluster k by:

$$s_k(x_i) = \frac{e^{W_k^T x_i + b_k}}{\sum_{k'} e^{W_{k'}^T x_i + b_{k'}}} \tag{2}$$

The second part, denominated VLAD core, is effectively computing the VLAD representation of the image given an image descriptor x, the cluster centers c and the computed soft assignments s following the equation:

$$V(j,k) = \sum_{i=1}^{N} s_k(x_i)(x_i(j) - c_k(j)) \tag{3}$$

where $x_i(j)$ and $c_k(j)$ represent respectively the j-th dimension of the i-th descriptor and the k-th cluster center.

The obtained descriptors are then intra-normalized (using a column-wise L-2 norm), flattened into a vector and then a final L-2 norm operation is applied. For this network is fundamental to initialize the cluster centers in order to obtain good performances. This initialization is preformed in a preliminary step using a small subset of the training data available and consists in computing the features representations, using the pre-trained ResNet-18 backbone, and then extracting the descriptors by randomly selecting some of the obtained features. Then, in order to obtain the cluster's centroids, k-means is used over the computed descriptors and the weights in the convolution layer for soft-assignment are initialized, to reproduce the results that would have been obtained with VLAD described in [5], using:

$$W = \alpha(\frac{c}{||c||_2} \cdot d) \tag{4}$$

where c and d are respectively the computed clusters centers and descriptors, α is instead selected to be large in order to better mimic the traditional VLAD.

3.4. CRN head

Seen the results provided from the previous implemented heads and the success of the attention layers to make a network focus on relevant only parts of an image, we have decided to add an attention layer at the NetVLAD head following the approach proposed by [3]. This is perfectly integrated in the NetVLAD architecture and it has the duty to produce a map that rescales the weights produced by the soft assignment step of the NetVLAD layer. This layer is composed by an initial average pooling sub-layer that has the duty to reduce the dimensionality of the feature maps produced by the backbone. Differently from what specified in [3], it is not reducing the features maps to a fixed size but it is simply reducing by a half the dimensions of the features. In order to capture features at different spatial resolutions, 3 convolution filters (with kernel sizes respectively of 3, 5, 7) are applied to the pooled features. The output of those filter is concatenated and an additional $1x1$ convolution filter is used in order to accumulate the features produced. The resulting mask is then upsampled, in order to restore the original features map dimensionality, by using a bilinear interpolator.This results into a mask that is used as to re-weight the features produced by the soft assignment specified into the NetVLAD description as showed in figure 1 into the context modulation layer. This layer is performing the product of the mask and the soft-assignment scores. The output of the context modulation layer is then used in the standard NetVLAD core instead of the soft-assignment scores. Also this head requires the initialization of centroids as the NetVLAD head and the initialization adopted is the same described in the section 3.3.

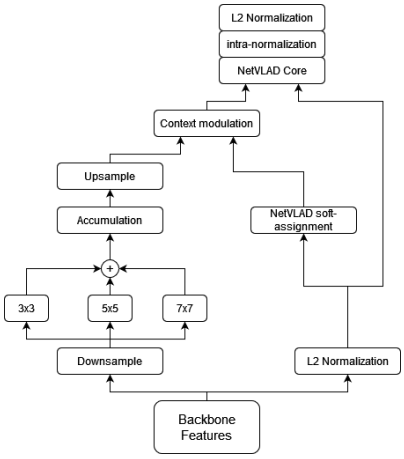


Figure 1. The architecture of the CRN head.

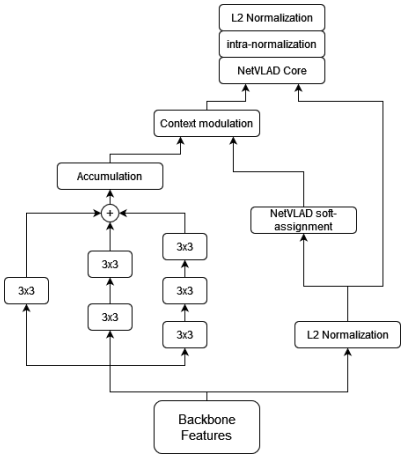
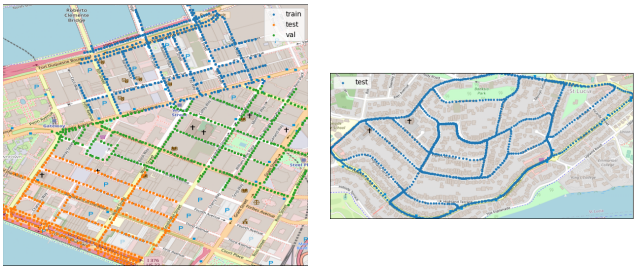


Figure 2. The architecture of the CRN2 head.

3.5. CRN2 head

In order to reduce the number of parameters to be learned by the CRN head, we have implemented a second version of this head, called CRN2, that exploits the ideas of concatenating multiple $3x3$ filters in order to obtain the same receptive field of a bigger filter but using less parameters as suggested in [6]. In particular we have replaced the $5x5$ and the $7x7$ filters showed in figure 1 with respectively 2 and 3 stacked $3x3$ filters as showed in figure 2. In addition to that, we have used dilated convolution in order to remove the pooling and upsampling layers by generating the mask directly at the desired resolution inspired by the work proposed in [7]. This, although requires more computation, will produce more accurate masks that may help the network to better focus on the relevant parts of the images. Also this head requires the initialization of centroids as the CRN and the NetVLAD heads and the initialization adopted is the same described in the section 3.3.



(a) Pitts30k (b) St. Lucia

Figure 3. The location of the images contained in both the pitts30k 3a dataset and the St. Lucia 3b dataset.

4. Experiments

In this section we evaluate the results obtained by the implemented heads and the selection of best hyper-parameters will be discussed.

Datasets The experiments have been run on the pitts30k dataset [2] that is containing 3 different predefined splits, respectively for training, validation and testing, composed by $10k$ images each. Those are images taken in the city of Pittsburgh from the Google street view images. In addition, also the St. Lucia dataset [8] has been used only for testing the models trained on the pitts30k dataset. The location of the images contained in both datasets can be seen in figure 3.

Mining procedure In order to train our models we need to generate triplets in the form $\{I_q, I^+, I^-\}$: for each query image I_q we are looking for positive examples I^+ and negative ones (I^-). In order to do so, for each query image we retrieve the images into the database that are in a range specified by the positive distance threshold. We use 2 possibly different values for this threshold at train and test time. Among all images in this range, we select as best positive the one that has descriptors more similar to the query image and we use this image as I^+. All the images not in the positive distance threshold range are considered as negative samples. We select by default the 10 images, from the negative set, that have the as similar as possible descriptors as the query image. We call those images hard negatives and we are using those instead of randomly picking from the negative set in order to make the task for the network more challenging resulting in a more robust model. Since we have to generate the descriptors that vary over the training procedure, we are periodically recalculating the triplets within the epochs.

Loss function We have set the problem of visual geolocalization in approximating the location of a query image by retrieving the nearest database images in the descriptors space. For that reason, the objective of our training procedure is to make images geographically close have a similar representation in descriptor space and instead make as far as possible the representation of geographically far images. This is possible by exploiting contrastive learning and the loss function that we are using is defined by

$$L(q,p,n) = \sum_i max(||q-p||_2 - ||q-n_i||_2 + m, 0) \quad (5)$$

where p, q and n_i represent the descriptors extracted by the query image, the positive image and the negative ones (by default we extract 10 negatives examples for each query image as described in Mining procedure). The parameter m is instead specifying a margin between the positive and negative descriptor representation of the images. In fact, if a negative sample has a distance in the descriptors space higher than the margin m the resulting loss will be 0 and, instead, if the distance between the positive and negative descriptors is lower than the margin the loss will be proportional to the margin violation.

Metric adopted Those results have been obtained by evaluating the models by using a standard evaluation procedure for place recognition. A given query image is said to be correctly localized if at least one of the N retrieved images is placed at a distance lower or equal to TTD from the ground truth position. This distance is set, if not differently specified, to 25 meters. After that we are calculating the percentage of correctly classified images for different values of N (indicated with $R@N$).

4.1. Comparison among the proposed heads

The results of comparison between the various proposed heads are reported in table 1. Those results have been obtained on the pitts30k test set. As it's possible to notice the head that is giving best results is the CRN and that shows that adding attention is essential for improving the quality of generated descriptors. It's important to notice also how the results are influenced by the number of produced descriptors. In fact both the NetVLAD and the CRN head are generating much more descriptors with respect to the GeM and the base heads and this seems correlated to higher recalls. Since the CRN and the NetVLAD heads are outperforming the other ones, we will focus more on those 2 during the rest of this section.

4.2. CRN and CRN2 models

As it's possible to notice from table 1, the performances of the CRN and CRN2 networks are very close one to each other as expected from the definition of the two networks. It's important to notice that the CRN2 network is using less parameters with respect to the CRN network. In fact, the CRN2 network is using only 245K parameters compared

	Descs.	$R@1$	$R@5$	$R@10$	$R@20$
Base	256	60.1	80.6	87.4	91.7
GeM	256	71.6	87.0	91.0	94.0
NetVLAD	16384	79.1	89.3	92.3	94.4
CRN*	16384	81.7	**90.7**	**93.4**	**95.3**
CRN2	16384	**81.8**	**90.7**	93.2	95.2

Table 1. Results on the pitts30k test set obtained with the various heads compared with the base head. The number of generated descriptors is also shown in the column Descs.

Network	time (s)
NetVLAD	0.0206
CRN	0.0215
CRN2	0.0285

Table 2. The running time for a single image for the various networks computed on a GPU NVIDIA Tesla K80.

to the 529k parameters used by the CRN network for the generation of the mask. As shown in the table 2, the time required to extract descriptors for a single image is slightly higher for the CRN2 network but the differences are not significative. For those reasons, the CRN2 network has to be preferred to the CRN network.

5. Ablation study

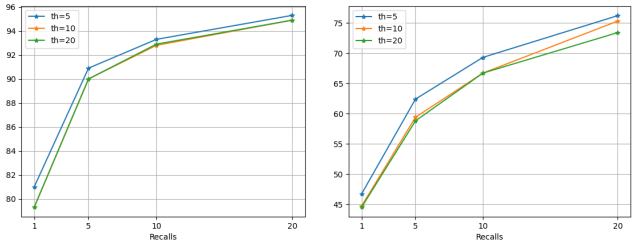
In this section we discuss the effect of changing one by one some parameters of the NetVLAD network, we especially focused on trying different learning rates, modifying the distance at which positives are taken.
We also changed the input images of the dataset by implementing some data augmentation techniques and by changing the resolution of the images.

5.1. Comparison between different learning rates

As first ablation study we tried different learning rates, from the table 3 we can see that the best results in calculating the percentage of correctly classified images are obtained with a learning rate of 10^{-5} , with this learning rate the network is superior to the other configuration of learning rate in each case. We also noticed that by decreasing the learning rate we increment the number of epochs needed to end the training.

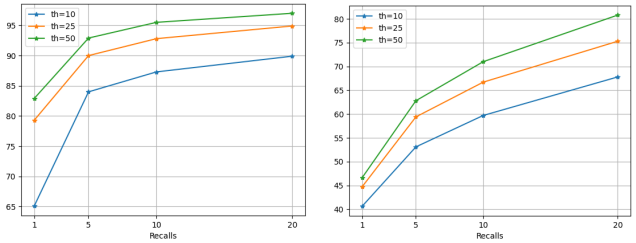
	$R@1$	$R@5$	$R@10$	$R@20$
lr = 10^{-3}	78.6	89.4	92.5	94.7
lr = 10^{-4}	79.1	89.3	92.3	94.4
lr = 10^{-5}	**79.3**	**90.0**	**92.8**	**94.9**

Table 3. Results obtained with the NetVLAD head on the pitts30k test set with different learning rates



(a) Pitts30k (b) St. Lucia

Figure 4. Graph showing the recalls obtained with different train positive distance threshold on both the pitts30k 4a dataset and the St. Lucia 4b dataset.



(a) Pitts30k (b) St. Lucia

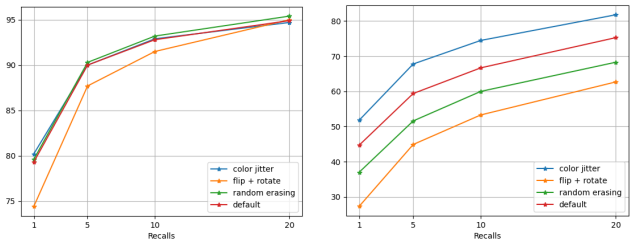
Figure 5. Graph showing the recalls obtained with different test positive distance threshold on both the pitts30k 5a dataset and the St. Lucia 5b dataset.

5.2. Comparison between different positive distance threshold

Initially the distance at which positive are taken was set at 25 meters, we tried to change the parameter $train\_positives\_dist\_threshold$ with different values. The graph contained in figure 4 shows that, during training, the positive distance threshold set at 5 meter outperforms all the other thresholds in every recall in both the Pitts30k and St. Lucia datasets.

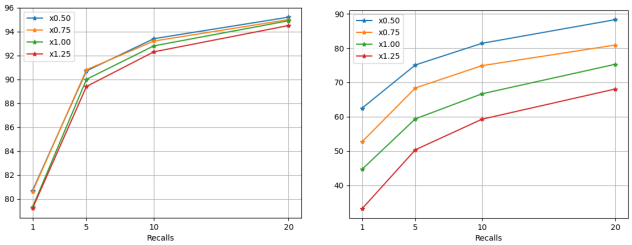
We also tried different test positive distance threshold and in this case there is a big difference between the different distances, greater distances perform in a better way than smaller ones.

In the Pitts30k dataset we can see that the recall on one image is 65.1 with the positive threshold distance set at 10 meters while the recall with the distance set at 50 meters is 82.9, there is a margin of 17.8%
The same trend is maintained in all the different recalls on both the Pitts30k and St. Lucia datasets, but while in the Pitts30k the margin between the distance threshold set at 10 meters and the one set at 50 meters stabilizes, in the St Lucia Datasets keeps incrementing, initially it's 6% at R1 and becomes 13% at R20.



(a) Pitts30k (b) St. Lucia

Figure 6. Graph showing the recalls obtained with different augmentation techniques on both the pitts30k 6a dataset and the St. Lucia 6b dataset.



(a) Pitts30k (b) St. Lucia

Figure 7. Graph showing the recalls obtained with different input image size on both the pitts30k 7a dataset and the St. Lucia 7b dataset.

5.3. Comparison between different data augmentation techniques

In order to see how the results change we tried some data augmentation techniques, we decided to use 3 of them: Color Jitter, Horizontal Flip and Random Erasing. With Color Jitter we randomized the contrast, brightness and color saturation of the image, with Horizontal Flip we flipped the image over the vertical axis and with the last transformation we erase with random values the pixel contained inside a rectangle region. For the Pitts30k dataset we can see in figure 6a that all the data augmentation techniques have a similar performance compared to the default configuration of the dataset; Only the flip and rotate technique gets sightly worse results in the first recalls.
For the St. Lucia dataset we can see in figure 6b that all the data augmentation techniques have very different performance, in this case only the Color Jitter transformation gets better results than the default configurations, while Random Erasing and Flip + Rotate get sightly worse results in all the recalls.

5.4. Comparison between different images sizes

The last ablation study has been the variation of the images sizes by scaling their dimensions by a scaling factor. We decided to try 4 scaling factors (including the standard one 1.00), the value selected are: 0.50, 0.75, 1.00, and 1.25. The results of the scaling factor are shown in figure 7, for both the Pitts30k and St. Lucia dataset it's clear that a reduction of the image size leads to an increment in the recall's value; This increment is way more visible in the St. Lucia dataset (figure 7b).

6. Conclusions

References

[1] F. Radenovic, G. Tolias, and O. Chum, "Fine-tuning cnn image retrieval with no human annotation," *TPAMI*, 2018. 1

[2] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," *TPAMI*, 2018. 1, 3

[3] H. Jin Kim, E. Dunn, and J.-M. Frahm, "Learned contextual feature reweighting for image geo-localization," *CVPR*, 2017. 1, 2

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CVPR*, 2016. 1

[5] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3304–3311. 1, 2

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. 2

[7] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017. 2

[8] M. Warren, D. McKinnon, H. He, and B. Upcroft, "Unaided stereo vision based pose estimation," in *Australasian Conference on Robotics and Automation*, G. Wyeth and B. Upcroft, Eds. Brisbane: Australian Robotics and Automation Association, 2010. [Online]. Available: http://eprints.qut.edu.au/39881/ 3