Visual Geo-Localization: a deep analysis of the task

Fersino Davide
s292602

Rigoli Alessandro
s269841

Vitali Giacomo
s281290

Abstract

In this work, we focus on the problem of Visual Geo-Localization, where the objective is to accurately find the position of a place depicted in photography. First, we present an overview of the task and how researchers faced the problem in the past years. Second, we provide some details about the solutions adopted in our work. Third, we provide an implementation of these methods, together with different studies on the possible variations of the architecture that address the task and its detail. All our experiments are performed on two datasets: Pitts30K and St.Lucia. Our code is publicly available here:
https://github.com/GiaVit97/project_vg

1. Introduction

The task of *visual place recognition* (VPR) is to find associations between one or multiple query images with a database of images of known places. In the two last decades, this task has seen a drastic acceleration of the research caused to the mass diffusion of smartphones, the large availability of publicly shared pictures on online platforms, and the rise of mobile robots operating in the open world. In this paper, we analyze the task of *visual geo-localization* (VG), i.e., the task of coarsely finding the geographical position where a given photo was taken as an image retrieval problem. The known places are collected in a *database* and a new image to be localized is called *query*.

Over the years, this problem remains very challenging. Images of places in general present multiple visual elements, and not only one single identifying object. Moreover, the appearance of a place can change over time due to natural variations like illumination, weather, and seasons or due to construction sites, new buildings, and so on.

In the last years, convolutional neural networks (CNNs) have emerged as a powerful image representation for various tasks such as object classification. In particular, since Krizhevsky *et al*. [4] demonstrated that deep CNNs can reach excellent performance in visual tasks, it has been recognized that these architectures can act as powerful generators of image representations. Moreover, it has been shown that CNNs can learn generic features that are transferable to other visual tasks. These are some of the reasons that bring the deep learned representations to image retrieval to surpass the performance achieved with hand-crafted methods.

2. Related Works

As described in a survey of deep place recognition [5], the problem of visual place recognition has a lot of studies on the image representations used in this problem and how they have evolved from using hand-crafted to deep-learned features, on how metric learning techniques are used to get more discriminative representations, as well as techniques for dealing with distractors and shifts in the visual domain of the images.

As we said previously, the CNNs bring a lot of performances with respect to the hand-crafted methods. The first attempt at using CNNs as representation generators for image and place recognition demonstrated that the vector of activations in a fully connected layer of a classification pre-trained on ImageNet [8] could be effectively used for retrieval. The problem is that the fully connected layer is not robust to the presence of distractors or occlusions. Moreover, fully connected representations are limited by fixed input size and by requiring large numbers of parameters. For this reason, researchers start investigating the generation of image representations directly from the output of convolutional layers. Also, they tried to use convolutional feature maps, but the performances of this simple method is not far off from the results obtained with fully connected representations. Simply flattening the feature maps of a convolutional layer does not take full advantage of the spatial information contained therein. This consideration guided the development of the current state-of-the-art representations for place retrieval. We can categorize them into two families: aggregation of the convolutional features using methods derived from hand-crafted representations of local descriptors and pooling methods that summarize the convolutional features.

For the first one, researchers have proposed aggregation modules that can be plugged on top of a CNN and allow

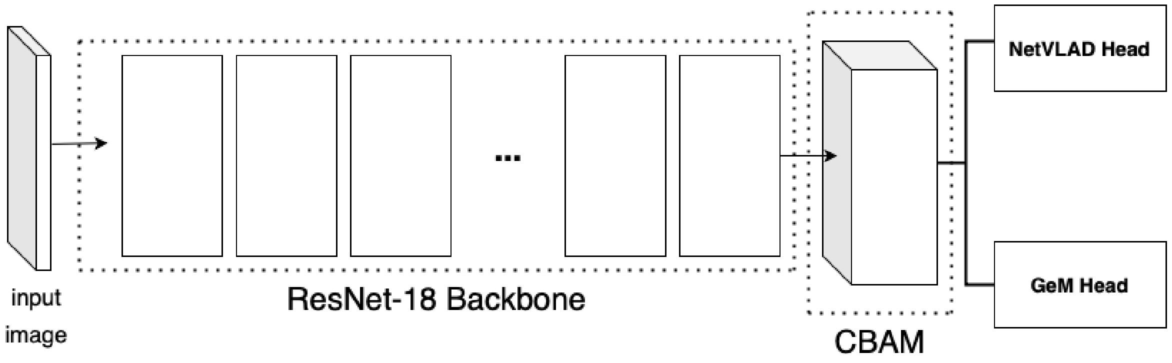

Figure 1. CNN architecture with different final heads and the optional CBAM attention layer.

end-to-end learning. An example can be NetVLAD [1], a layer that implements the VLAD embedding and aggregation with differentiable operations. Also, NetVLAD provides more trainable parameters than VLAD and more flexibility.

For the second one, researchers have shown that convolutional features from mid/late layers, unlike shallow non-learned features, can be successfully aggregated and compared without embedding. The simplest pooled representation is achieved by max-pooling the features maps of a convolutional layer. Another popular representation is obtained by parameterless sum-pooling of convolutional feature maps. The current state-of-the-art pooling method is the generalized-mean aggregation layer GeM [7] that generalizes both max and average pooling.

Besides the aggregation or pooling method used to build these representations, their effectiveness depends also on how they are learned from the data. The selection of positive and negative examples is crucial when training a model with contrastive and triplet losses. If they are too easy, the network will not learn to properly discriminate the images. On the other hand, forcing the network to learn extremely hard examples could lead to overfitting and bad local minima. One of the most popular methods is triplet loss. Also, second-order information can be enforced both in the spatial context and the abstract feature dimensions. For example, second-order similarity (SOS) loss [6] is used to enhance the triplet loss with hard-negative mining.

Moreover, a scene can experience structural modifications over time. There are also challenges that are specific to the environment. An approach for dealing with visual distractors is to extract regions of interest from the image, like regions that contain only the elements that are more relevant for the recognition tasks. This solution has a problem: many of the regions may only cover distractors, so they can negatively affect performance, and increasing the number of regions would not improve coverage of the informative areas but also of the irrelevant ones. Attention modules are an approach to select more relevant information from the images that can significantly improve the performance of place retrieval. The image is processed as a whole but the individual features are weighted according to a relevance criterion. An example of an attention module is the convolutional block attention module (CBAM) [10] that infers attention maps along the channel and spatial dimensions.

Finally, the datasets that are openly available and used for visual place recognition focus on different use cases or problems and can have substantial differences among each other. Some of them are non-robotics datasets and can be focused on recognizing famous landmarks or discretely sampled locations collecting images from StreetView like the Pitts30k dataset. Others can be robotics datasets, typically created by recording videos from cameras mounted on a vehicle or smaller robots like the St.Lucia dataset [9].

3. Method Overview

In this section, we describe the state-of-the-art methods for the task of visual geo-localization that we used in our work. Our implementations are based on a ResNet-18 [2] backbone, pre-trained on the ImageNet dataset [8], with the combination of different techniques, that we want to briefly present. A graphical representation of the main modules and their possible combinations is visible in Figure 1.

3.1. Network heads

The problem of visual geo-localization has been traditionally cast as an instance retrieval task, where the query image is compared through appropriate metrics with a large geotagged database, to identify its position. A crucial aspect in this comparison is the ability to generate appropriate image representations, nowadays this is done through the utilization of deep CNNs, in particular by exploiting their convolutional representation. This solution shows better performance and ability in generalization than the previously adopted handcrafted methods. The solutions that we adopt differ on how they aggregate the output of the last convolutional layer of the CNN backbone, an HxWxD map, but they can be categorized into two families:

1. methods derived from hand-crafted representations of local descriptors;
2. pooling methods;

3.1.1 Average Pooling

This is a basic pooling operation where that takes the HxWxD output of the CNN backbone and performs downsampling by averaging the value of each patch of the feature map. In our implementation, for each input, we obtain a D-dimensional vector, where each element is computed by averaging the value of the HxW activations.

3.1.2 NetVLAD

The NetVLAD layer, proposed by Arandjelović *et al.* [1], belongs to the first family of solutions. It's derived from a hand-crafted representation, the one of Vector of Locally Aggregated Descriptors (VLAD) [3]. Intuitively, the NetVLAD solution implements a differentiable, and so trainable, version of VLAD, pluggable into any CNN architecture. The source of discontinuities in VLAD is the hard assignment $a_k(x_i)$ of descriptors x_i to cluster centers c_k, that can retrieve only 1 or 0 depending on if the cluster is the closest cluster to the descriptor. Arandjelović *et al.* solve this problem by replacing the hard assignment with a soft assignment, a function that can retrieve the value in the range between 0 and 1, depending on the distance between x_i and c_k. The final form of the soft assignment is the following:

$$\bar{a}_k(x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_{k'}^T x_i + b_{k'}}} \tag{1}$$

With vector $w_k = 2\alpha c_k$ and scalar $b_k = -\alpha||c_k||^2$. α is a parameter that controls the decay of the response with the magnitude of the distance, it assign the weight of descriptor x_i to cluster c_k, proportional to their proximity. $\{w_k\}, \{b_k\}$ and $\{c_k\}$ are trainable parameters for each cluster k. Given K cluster centers and N D-dimensional local image descriptors as input, the NetVLAD image representation V, represented here as a KxD matrix, but then converted into a vector and normalized, outputs an element like the following for the *j-th*, *i-th* descriptor and *k-th* cluster centre:

$$V(j,k) = \sum_{i=1}^{N} \bar{a}_k(x_i)(x_i(j) - c_k(j)) \tag{2}$$

3.1.3 GeM

The Generalized-Mean pooling (GeM), proposed by Radenović *et al.* in their work [7], belongs to the second family of solutions. The GeM layer takes the HxWxD output of the CNN backbone as input, denotes with X_d the set of HxW activations for feature map $d \in \{1...D\}$ and produces a D-dimensional vector f where the $d \in D$ element is computed as follow:

$$f_d = \left(\frac{1}{|\mathcal{X}_d|} \sum_{x \in \mathcal{X}_d} x^{p_d} \right)^{\frac{1}{p_d}} \tag{3}$$

The parameter p_d can be manually set or learned, since this operation is differentiable, and it can be different per feature map or shared ($p_d = p$), Radenović *et al.* [7] shows that a shared pooling parameter provide better performance. Furthermore, it's important to underline two special case:

1. for $p_d \rightarrow \infty$ we have max pooling
2. for $p_d = 1$ we have average pooling

3.2. Attention module

One of the main problems of the visual geo-localization task is to make the network invariant to distractors. Since we are dealing with images taken at different times and seasons, much information present inside it are not useful: think about peoples and cars that are randomly captured when a photo was taken. These objects are not useful to the final goal of recognizing a place, and we should pay less attention to them, focusing on the other hand on more informative elements, attention modules address this task.

3.2.1 CBAM

The solution to add attention proposed by Woo *et al.* in their work [10], is called Convolutional Block Attention Module (CBAM). Since CBAM is a lightweight and general module, it can be integrated into any CNN architecture with low overhead. The CBAM architecture is composed of two main modules:

- *Channel attention module*: it's the portion of CBAM in charge of producing the channel attention map, that intuitively focuses on "what" is meaningful given an input image. To obtain this information, first, they squeeze the spatial dimension of the input feature map through two parallel pooling methods, average-pooling, and max-pooling, both the output descriptors are then forwarded to a shared Multi-Layer Perceptron with one hidden layer to produce the final channel attention map $M_c \in \mathbb{R}^{Cx1x1}$.
- *Spatial Attention Module*: it's the portion of CBAM in charge of producing the spatial attention map, that

intuitively focuses on "where" is an informative part given an input image. To obtain this information, they apply to the output of the previous stage again two pooling operations, average-pooling, and max-pooling, but this time sequentially. On the concatenated feature descriptor then, they apply a convolutional layer to generate spatial attention map $M_s \in \mathbb{R}^{HxWx1}$.

The two elements are placed sequentially; first, they perform an element-wise multiplication between the intermediate feature map and the channel attention map $M_c \in \mathbb{R}^{1x1xC}$ and then the output of this stage is element-wise multiplied with the 2D spatial attention map $M_s \in \mathbb{R}^{HxWx1}$.

3.3. Loss function and mining procedure

A crucial aspect to let the network understand the best representation that will optimize place recognition performance is the adopted loss function. In our work, we first exploit a classical solution for this task, the triplet loss, and then we try to expand it with the second-order similarity loss. The triplet ranking loss supposes a structure of data of type $(q, \{p_i^q\}, \{n_j^q\})$ where we have respectively the query q, a set of hard positive and a set of hard negatives, so a mining procedure to select these images is needed. The hard positives for each query are computed as all the queries that are within *train positives dist threshold* (default: $10m$) from the query, if a query has no hard positives, it will be discarded because considered useless. Note that not all the hard positives are depicting the same scene, this is because this selection is based only on an approximate GPS position, but different camera orientations and perspectives are possible. The set of hardest negatives instead, is selected as the 10 nearest negatives, i.e. further than 25 meters and nearest in the feature space.

3.3.1 Triplet Loss

Given a training tuple $(q, \{p_i^q\}, \{n_j^q\})$, Arandjelović *et al.* [1] defines the weakly supervised ranking loss as follow:

$$L_\theta = \sum_j l\left(\min_i d_\theta^2(q, p_i^q) + m - d_\theta^2(q, n_j^q)\right) \tag{4}$$

Intuitively, this function measure the distance difference between the query with the best matching positive (the hard positive with most similar descriptors with the query) and the query with each computed hard negative, if this difference is smaller than the margin, then the loss provide a penalization equal to the amount of violation. l is a simple hinge loss $l(x) = max(x, 0)$, that retrieve the amount of violation if it's present, 0 otherwise. The objective of this loss function is to learn a similar representation for geographically close images, and the opposite for geographically far images.

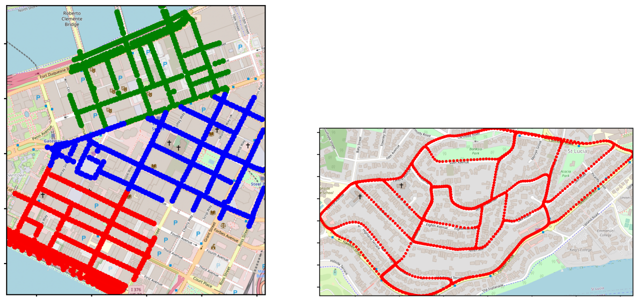
Figure 2. Pitts30k and St.Lucia datasets images distribution.

3.3.2 Second Order Similarity Loss

Following the work proposed by Ng *et al.* [6], we exploit for our work also a second-order similarity loss, that has the following form:

$$L_{SOS} = \frac{1}{|S_t|}\sqrt{\sum_{S_t}\left(\|D_a - D_n\|^2 - \|D_p - D_n\|^2\right)^2} \tag{5}$$

With $S_t = \{(D_a, D_p, D_n)\}$ a set of triplets formed by query, positive and negative images. The final objective function is a combination of the triplet loss together with SOS loss, balanced by a parameter λ.

$$L = L_{triplet} + \lambda L_{SOS} \tag{6}$$

4. Experimental results

4.1. Datasets

We report results on two publicly available datasets as shown in Figure 2.

Pitts30k dataset is a subset of the bigger Pitts250k. It contains $30k$ images downloaded from Google Street View and $21k$ test queries generated from Street View but taken at different times. The datasets are divided into three equal parts for training, validation, and testing, each with $10k$ images and roughly $7k$ queries. Each part is geographically disjoint from others.

UQ St.Lucia [9] dataset is a vision dataset gathered from a car driven in a 9.5 km circuit around the University of Queensland's St.Lucia campus on December 15, 2010. The dataset traverses local roads and encounters several varying scenarios including roadworks, speed bumps, bright scenes, dark scenes, reverse traverses, some loop closure events, multi-lane roads, roundabouts, and speeds of up to 60 km/h.

Head	Pitts30k (train)	Pitts30k	St.Lucia
Average Pooling	81.7	81.5	48.0
NetVLAD	96.0	93.2	71.3
GeM	89.9	89.1	68.3

Table 1. Comparison between different network heads in terms of Recall@5.

4.2. Head selection

As already mentioned, our architecture is based on a ResNet-18 backbone, pre-trained on the ImageNet dataset. In addition to it, we tried the three heads mentioned above in Section 3.1 (Average Pooling, NetVLAD and GeM), arriving to have three different starting models.

Analyzing the behavior of the three models, we decided to implement additional changes using the NetVLAD head, because it is the most performing on the two datasets, as we can see from Table 1.

From this point on, therefore, all proposed implementations have NetVLAD as a starting point.

4.3. Learning rates and optimizers

A crucial decision for the overall performances of a deep model is the choice of the optimizer to work with and its learning rate: on one hand, with the choice of a too small learning rate the optimization procedure can take very long time, even infinite, and it can never reach reasonable values. On the other hand, the choice of a too large learning rate provides a speed up, but for too high value the possibility of overshooting, so overcome the minimum, it's concrete. In our work we focus mainly on two different optimizers, Adam and Stochastic Gradient Descent (SGD) with momentum $= 0.9$, and we perform different experiments with three different learning rates: $lr \in \{10^{-2}, 10^{-5}, 10^{-6}\}$. For our experiments, we put an upper bound of 20 epochs of training, and set a patience mechanism that stop the training if the Recall@5 value does not improve for three epochs.

The results are visible in Figure 3, and they seem to confirm our prediction: for a low value of the learning rate ($lr = 10^{-6}$) brings the optimization procedure to reach the maximum of 20 epochs, without the convergence to a fixed R@5 value, instead of for a high learning rate ($lr = 10^{-2}$), the convergence is much faster.

We find two unexpected behavior: first, for SGD with $lr = 10^{-2}$, we expect a faster convergence than 15 epochs (usually it converges after 8-9 epochs), but we should underline that the maximum recall value is obtained at epoch 06, and then it starts oscillating around this value, without triggering the patience mechanism. Second, for SGD with $lr = 10^{-6}$, we expect a very slow convergence, but it happens after only 7 epochs, we think that this behavior is due

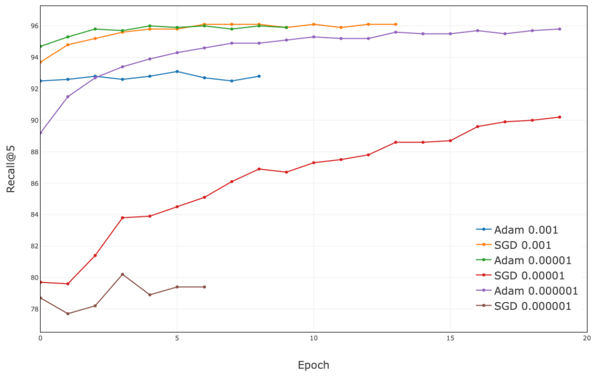

Figure 3. Comparison between the different optimization techniques and learning rates in terms of Recall@5 during different epochs on the training dataset Pitts30k.

Train distance (m)	Pitts30k (train)	Pitts30k	St.Lucia
5	96.4	92.9	70.8
10 (default)	96.0	93.2	71.3
20	96.2	92.8	72.3

Table 2. Comparison between different values in terms of Recall@5 obtained changing distance threshold at train time.

to a stuck in a local minimum ($R@5 = 80.2$) that cause the activation of the patience mechanism. For the next studies, we choose the setting that performs better at test time on the two different datasets (Pitts30k, St.Lucia), that is the Adam optimizer with $lr = 10^{-5}$ ($R@5 = 93.2$ and $R@5 = 71.3$ respectively).

4.4. Positives train distance

As mentioned in section 3.3, at train time we have a parameter that plays an important role: the distance from the query image (in meters) within which we consider as *hard positive* the images of the database. We analyze its impact by performing some experiments with different values.

The results are shown in Table 2; in the mining procedure, we remove all the queries that have an empty hard positive set, for this reason, an immediate result that we can see is the number of queries removed: in the case of distance set to 10 and 20, this number is quite similar (96 and 24 respectively), but if we put a robust constraint (distance set to $5m$), then 2544 query are removed. With distance values of 10 and 20 results are similar, in particular, we obtain better results on Pitts30k for the first, on St.Lucia for the second. Apparently, from the R@5 obtained at train time, the distance set to $5m$ performs better, but this is misleading, it gets high only because we are setting a strict constraint. We can see its real performances in the test values, and it goes down especially for the St.Lucia dataset.

4.5. Positives test distance

Another parameter that plays a fundamental role in the behavior of the network is the distance within which a place is considered to be correctly recognized. In this case, the tests carried out returned results in line with those expected. It is quite predictable that by increasing this parameter the recall values will increase and vice versa.

In our case, we tested the original NetVLAD architecture, trained to consider positive places in a radius of 10 meters, with this parameter first reduced to 10m and then increased to 40m.

As expected, the $10m$ test generated lower results than the original, with a Recall@5 of 88.3 on the Pitts30k, and the one at $40m$ generated higher results, with a Recall@5 of 94.7.

4.6. Data augmentation

Deep learning always benefits from having more amount of data. For this reason, we can artificially increase the dataset size by applying some transformation to the training images preserving their labels. This will help prevent overfitting by adding some variety to the input data, but the final accuracy may be negatively affected if the variations we include transforming the training images are not present in the test set.

We decide to use five different pipelines of transformations:

- *CS-HF*: the image is jittered in contrast and saturation, then it is horizontally flipped.
- *H-RP*: the image is jittered in hue, then the random perspective transformation is applied.
- *B-GS-R*: the image is jittered in brightness, then the relative grayscale is taken and finally random rotation is applied.
- *GS*: the image is grayscaled taking the relative grayscale.
- *BCSH*: the image is jittered in brightness, contrast, saturation, and hue.

From Figure 4 we can see how the Recall@5 changes during the epochs on the different techniques of data augmentation we used. We can immediately notice that the worst behavior is the one of the *B-GS-R* technique, which is the only technique that applied random rotation to the image. This because in our task is not very helpful to rotate an image of 180 degrees or do some strange rotation and it changes drastically the representation of buildings and roads. So, when we ask to recognize a place, the image shows a place that does not exist anymore.

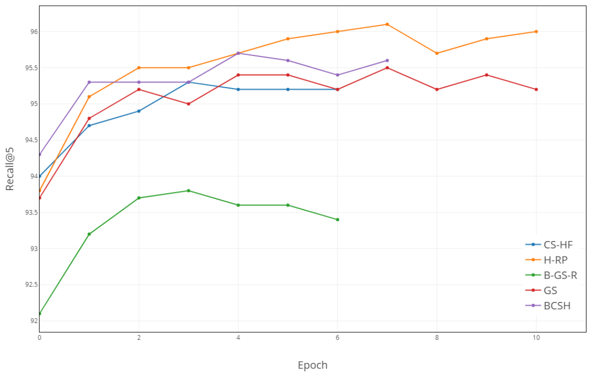

Figure 4. Comparison between the different techniques of data augmentation in terms of Recall@5 during different epochs on the training dataset Pitts30k.

Augmentation	Pitts30k (train)	Pitts30k	St.Lucia
CS-HF	95.3	92.4	77.5
H-RP	96.1	92.8	72.2
B-GS-R	93.8	91.5	64.4
GS	95.5	92.7	71.0
BCSH	95.7	92.5	82.1

Table 3. Comparison between the different techniques of data augmentation in terms of Recall@5 on the two different evaluation datasets.

Instead, the technique that behaves better on the training set is the *H-RP* that reaches a Recall@5 of 96.1. So, we can notice that this technique is helpful.

From the evaluation on the test set, we expect that some modifications in terms of color like jittering the image in some way can improve the results also on St.Lucia because drastically changing the image can help the model to learn and understand better general features instead of specific features correlated to the Pitts30k dataset.

From Table 3 we can see the results on the two validation datasets, Pitts30k and St.Lucia. The best performances on the Pitts30k dataset are given by the *H-RP* augmentation technique that applies some random perspective. Also, we obtain good results with all the other techniques and, as expected from the training results, we have the worst result with the *B-GS-R* augmentation technique.

As we expected, the best results by far on the St.Lucia dataset are given by the *BCSH* augmentation technique that jitter the image in terms of brightness, contrast, saturation, and hue. In this way, the model can understand and learn more general feature and recognize better also the image of St.Lucia that are very different from the one of the Pitts30k dataset.

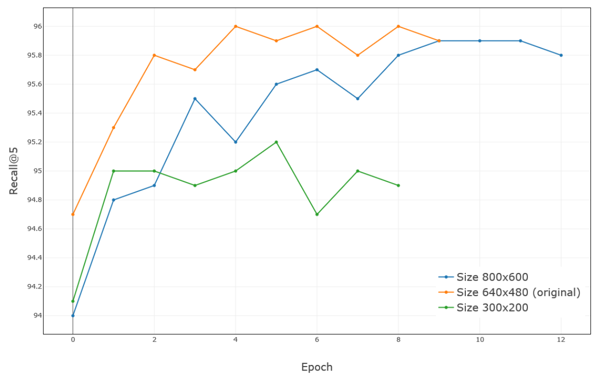

Figure 5. Comparison between models trained with different image sizes in terms of Recall@5 during different epochs on the training dataset Pitts30k.

Image Sizes	Pitts30k (train)	Pitts30k	St.Lucia
640x480	96.0	93.2	71.3
800x600	95.9	92.8	66.5
300x200	95.2	92.6	81.5

Table 4. Comparison between models trained with different image sizes in terms of Recall@5 on the two different evaluation datasets.

4.7. Image Resizing

The size of the images plays a big role in training a neural network. For example on one hand larger images allow to acquire more details and require more time to train, but on the other hand, smaller images focus the attention on the general view of the environment and are therefore generally better if the test dataset has few similarities with the training one.

This behavior was partially confirmed by the results obtained, as we can see from Figure 5. We trained the original NetVLAD network with images taken from the Pitts30k database and resized them one time to $300x200$ and another time to $800x600$.

For the increased resolution, there was no benefit. With Pitts30k the training and test values are practically the same as the original ones. With the tests on St.Lucia, as expected, it behaves very badly due to the different types of images, resulting in 66.5 precision on Recall@5.

The lower resolution instead behaved as expected, with a slight decrease in performance on the Pitts30k, and a big improvement in the test on St. Lucia, which reached 81.5 on Recall@5, as we can see in Table 4 where we show test results.

Attention Layer	Pitts30k (train)	Pitts30k	St.Lucia
NetVLAD (Default)	96.0	93.2	71.3
NetVLAD + CBAM	95.7	93.3	70.8

Table 5. Comparison between models trained with different attention layer in terms of Recall@5 on the two different evaluation datasets.

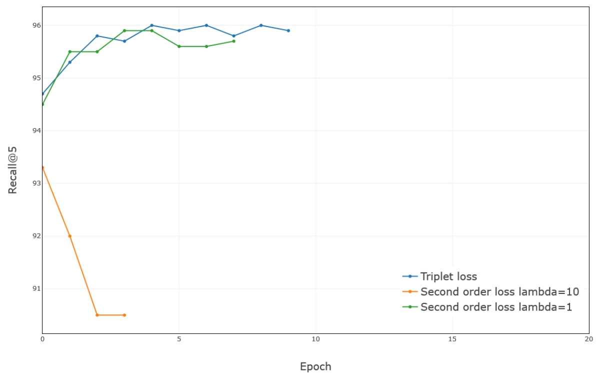

Figure 6. Comparison between models trained with SOS Loss and Triplet Loss in terms of Recall@5 during different epochs on the training dataset Pitts30k.

4.8. Attention layer

Given the importance of eliminating noise in a task such as places recognition, we thought that using an attention layer would bring significant improvements, but this was not the case.

The use of CBAM as an attention layer did not bring any improvement neither during the training nor during the test. The results were exactly in line with the previous ones, even though there was a slight decrease in the Recall@5 in training, but during the test, it did not show this difference, as we can see from Table 5.

A hypothesis to justify this behavior could be that NetVLAD is already a network that packages the incoming information as a series of local feature descriptors, therefore little influenced by an attention layer to carry out a selection of these descriptors, since it is able autonomously to do it through back-propagation, does not change too much the results. This hypothesis should be verified through the use of multiple attention layers.

4.9. Second-order similarity loss

The use of second-order similarity loss does not seem to bring benefits compared to the more used triplet loss. We performed two tests with two different λ values, 1 and 10, but in both cases the results were disappointing.

The test with $\lambda = 1$ obtained similar but lower performances to the previous model. In the test with $\lambda = 10$, on the other hand, the results worsened drastically.

Given the results, we believe the SOS Loss is not suitable for the topic in question as shown in Figure 6.

5. Conclusions

In this work we studied and used some implementations to address the task of visual geo-localization.

First of all, we implement NetVLAD and GeM as baselines and we compare them with the average pooling one to understand that NetVLAD is the most promising baseline in terms of Recall@5. In the whole work, we first train the model on the Pitts30k dataset, and then we test it on both Pitts30k and St.Lucia.

After the first implementation, we decide to do some ablation study and try to improve the model and obtain better performances from it. First of all, we try different learning rats and optimizers like Adam and SGD with momentum $= 0.9$, to understand that the Adam optimizer with a learning rate of $1x10^{-5}$ has the best performances on the training set.

Then we decide to change the parameter that defines the distance at which positives are taken at train time and the parameter which defines the distance at which positive is taken at test time, and we do not find improvement.

Moreover, we try different techniques of data augmentation, to understand that the best one is the *BCSH* that has a Recall@5 on the St.Lucia dataset that is significantly improved with respect to the one we have without it (71.3 vs 82.1).

Also, we decide to change the size of the images, both at train and test time. From that, we understand that if we improve their size, we can obtain improvement on Pitts30k in terms of Recall@5, but the results on the St.Lucia dataset remain the same. Instead, if we decrease the size, we find promising values of Recall@5 also on St.Lucia (81.5).

Finally, we decided to implement the CBAM attention model and the second-order similarity (SOS) loss. Unfortunately, both of them do not provide us improvement on the results on the two datasets.

References

[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition, 2018. 2, 3, 4

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2016. 2

[3] H. Jégou, M. Douze, C. Schmid, , and P. Pérez. Aggregating local descriptors into a compact image representation, 2010. 3

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks, 2012. 1

[5] C. Masone and B. Caputo. A survery on deep visual place recognition, 2021. 1

[6] T. Ng, V. Balntas, Y. Tian, and K. Mikolajczyk. Solar: Second-order loss and attention for image retrieval, 2020. 2, 4

[7] F. Radenovic, G. Tolias, and O. Chum. Fine-tuning cnn image retrieval with no human annotation, 2018. 2, 3

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge, 2015. 1, 2

[9] M. Warren, D. McKinnon, H. He, and B. Upcroft. Unaided stereo vision based pose estimation. In G. Wyeth and B. Upcroft, editors, *Australasian Conference on Robotics and Automation*, Brisbane, 2010. Australian Robotics and Automation Association. 2, 4

[10] S. Wo, J. Park, J.-Y. Lee, and I. S. Kweon. Cbam: Convolutional block attention module, 2018. 2, 3