POLITECNICO DI TORINO

Master degree in Data science and Engineering

Master Thesis

# Deep learning for Sequence-based Visual Geo-localization

Building a sequence to sequence geo-localization system


**Supervisors**
Prof. Barbara Caputo

**Co-supervisors:**
Prof. Nicola Gatti
PhD. Carlo Masone
Dott. Gabriele Berton

Gabriele Trivigno
Student ID: 276807

Academic year 2021 – 2022

**Deep learning for Sequence-based Visual Geo-localization**
Master thesis. Politecnico di Torino, Turin.


October 2021.

# Acknowledgements

Scrivere questa tesi è stato senza dubbio tra i compiti più ardui che mi sia mai trovato a svolgere. Come tale, mi ha fatto realizzare che questa sensazione ha accompagnato tutto il mio percorso universitario: ogni sfida era più ardua della precedente, e arrivato a questo punto posso solo augurarmi di continuare a intraprendere sempre nuove strade, che tra le mille difficoltà possano continuare a farmi crescere.

Al contempo, se c'è una cosa che ho imparato in questi anni, è che nessun successo accademico riempie di gioia quanto condividere un'emozione, un sentimento, un pensiero, una risata, con persone con cui sentirsi capiti. Per questo ringrazio di cuore tutte le persone che mi hanno fatto sentire così; il mio essere arrivato fin qui non dipende che da voi.

Per il resto, le persone da ringraziare sono tantissime.

Ai miei genitori, che non hanno mai avuto dubbi che potessi farcela, semplicemente perchè credono in me più del dovuto; a mia nonna che è sempre la prima ad informarsi sui miei progressi, e sulla mia alimentazione; e a tutta la mia famiglia;

Alla mia ragazza, anche solo per esserlo ancora dopo avermi sopportato nei momenti più stressanti;

A tutti quelli che si sono considerati miei amici, con cui ho scoperto cosa sia la spensieratezza;

Alla prof. Caputo che ha creduto da subito in me, offrendomi mille possibilità, e a Carlo, Gabriele e Riccardo, senza i quali questa tesi non sarebbbe stata possibile;

Ai miei colleghi con i quali era normale, nel mezzo della notte, scambiarsi informazioni su quella scadenza dimenticata, o quel pezzo di codice che proprio non tornava.

# Abstract

Visual geo-localization is the task of recognizing the geographical location where an image was taken by comparing it to a database of geo-tagged images of previously visited places. This is an open topic of research in computer vision, with numerous recent studies that investigate solutions to make these methods more accurate, robust to appearance changes, generalizable and scalable.

This thesis proposes to address two emerging problems in visual geo-localization:

- the first aim is to perform an extensive survey of the existing methods in the literature for such task, in order to 1) establish a fair and clear evaluation protocol of these methods, which is missing for visual geo-localization, and 2) provide guidelines on the most suitable approach for practical application.

- the second contribution, more research oriented, is to assess the extension of the state-of-the-art methods for visual geo-localization from a single image to the case where the input is a sequence of frames. This is a setting that is particularly meaningful for autonomous vehicles or for augmented reality applications.

This second part of the thesis explores the possibility to tackle the sequence-based problem using Transformers-based methods, a solution that has not yet been studied in literature.

# Contents

# Chapter 1

# Introduction

## 1.1 Thesis's objectives

The goal of this thesis is to provide a fair and clear evaluation protocol of the main methods for Visual Geo-Localization (VG), highlighting guidelines for practical use in applications, and finally to asses their extensibility to a multi-image setting, proposing new architectures to deal with such data.

The task at hand, known as Visual Geo-Localization, regards the ability to coarsely estimate the place where a photo was taken based on a set of previously visited locations. It has a wide range of real-world applications. Some examples can be the implementation of 3D reconstruction systems, photo sharing services, augmented reality, and more importantly also autonomous navigation scenarios, where such a system that can pinpoint accurately its spatial coordinates can circumvent the problem of the drop in GPS signal quality that verifies in crowded cities.

In some applications it might be of interest to estimate the 6 degrees-of-freedom position of an object, however the focus of this thesis is on locating a photo within a database spanning a large area (like big cities) with an accuracy of a few meters.

In general, either for single or multi image setting, a reliable VG system should possess the following characteristics:

- **Large capacity**: it should be able to accurately predict the location of queries against a big-sized database containing images for the whole area of big cities;
- **Generalization**: the learnt feature extractors should have the ability to work well even when applied on a city different from the ones seen

Table 1.1: Difference in results obtained by small changes to a VG pipeline. Recall@1 for a ResNet-18 with NetVLAD trained on Pitts30k and tested on Tokyo24/7.

| | Vanilla | Resize (80%) | Data augmentation (brightness = 2) | Predictions refinement (*nearest crop)* | PCA (2048) |
|---|---|---|---|---|---|
| R@1 | 63.4 | 64.3 | 68.6 | 67.0 | 56.6 |

during the training process ;

- **Robustness**: the network should be robust to changes in appearance and lighting.

## 1.2 Related works and main problems

This field has met more and more the interest of researchers in recent years, as it is witnessed by the growing number of publications regarding it in top-tier conferences and workshops, and therefore many different methods are available.

However, among the many approaches in the literature, a few issues can be pointed out and will be addressed in this work:

- **Metrics**: oftentimes research claim state-of-the-art performances based on the single metric of recall on the dataset of choice. Sometimes, especially in practical application, it can be quite useful to analyze the behavior of methods with respect to factors like inference and extraction time, hardware requirements, and ability to scale to bigger datasets.

- **Uniform setting**: A uniform framework to evaluate the performance of VG models can make it easier to pinpoint which elements of a pipeline have more influence on the final result, in which situations. Without such a framework, it becomes harder to directly compare different algorithmic approaches, whose improvements can be hidden by the use of other marginal techniques like weights initialization, data augmentation, use of libraries. In Tab. 1.1 it is shown how some minor modification from the point of view of the method can affect the final value of recall.

- **Sequences**: In many application, mainly in the field of robotics and autonomous driving, it is natural to work with sequences of images rather than single pictures, as they are available and therefore it could be an

additional value to be exploited. However when it comes to extending the task to multi-image input the literature is rather scarce.

## 1.3 My contribution: research & development

Specifically it has been decided to evaluate methods based on image retrieval, that work by comparing the query (the photo whose location is the objective to be found) with a consistent set of images of known location; such a set is called database.

Stemming from the criticality highlighted in Section 1.2, firstly an extensive analysis of the existing methods for VG is proposed. The main point is to contextualize all the approaches in the literature in a uniform setting, allowing for a clear point of view to examine how specific elements of a VG pipeline affect results, taking into account also their complexity (number of parameters, FLOPs, feature dimensionality...) and applicability in a real-world scenario. With the proposed framework a consistent number of experiments have been run, varying the training dataset size as well; this tool will offer to researchers and practitioners a convenient way to experiment with a broad variety of state-of-the-art VG architectures and techniques, with the possibility to vary each step of a VG pipeline. Such a tool can prove to be useful both for researchers in developing novel solutions and for industry practitioners who need to tune their system based on real-world constraints

This work, presented as the first part of the thesis, contains a great extent of valuable results and findings and it is the outcome of a joint effort that was also submitted as a paper that is currently under review for the Benchmark and Datasets track of NeurIPS 2021 [1]. Such benchmark was a team effort. All the team members contributed to the development the framework. Personally, I helped in the whole implementation and in particular I was primarily responsible for the implementation of different losses function, the mining methods, the dataset pre-processing and reformatting module and also of the prediction refinement techniques. As for running experiments, that was also a joint effort and I took care of the ones regarding mining, backbones and prediction refinement.

Subsequently, in the second part of the thesis the lack of methods exploiting multi-image sequences is addressed. This part is more research-oriented, and after presenting the few works existing in the related literature, some new approaches that propose to exploit modern Transformers architectures

to extract valuable information from the sequences are presented. This second part has been carried out together with Riccardo Mereu. My personal contribution consisted, for what concerns the coding part, in developing the dataset module to handle sequences and mining in the dataset; implementation of the baselines, the Transformer-Encoder based aggregator, the Timesformer architecture and a version of the Non-Local layer. As regards the experimental part I took care of running and scheduling the majority of the reported experiments.

# Chapter 2

# Related Works

## 2.1 Visual Geo-Localization

Visual Geo-Localization is a research topic for which the interest in the community has spiked over the last years, due to various reasons. On the one hand there are still nowadays many unsolved challenges, like how to obtain reliable performances when images from different domains are encountered; moreover it has become more and more easy to collect large amounts of geo-tagged images through the use of mobile phones and services like Street View, that allow the creation of comprehensive datasets spanning different geographical location as well as different points in time.

Some relevant surveys in the literature that include a comprehensive study and explanation of the techniques developed over the years are [2, 3, 4]. The field has definitely been influenced by the deep learning wave, as before that it was mainly based on the use of hand-crafted features. A survey on the topic that is more focused on the deep-learning applications can be found in [5].

The result of this rapidly evolving research is that there is not a de-facto standard name for the task and it can be referred to with many alternatives, such as Visual Place Recognition, Image-Based pose estimation, Visual Based Localization, and many variations on the theme.

The majority of the approaches that will be presented in this section are based on the framework of image retrieval, but however some alternatives will be discussed like 3D-based methods and classification.

## 2.2 Related Fields

### 2.2.1 Image Retrieval

A system built for image retrieval is a system that, given a single image called query, and a separate set of images called database (or gallery, dataset) has the purpose of locating from the database an image that is the best match for the query at hand. The definition of best match depends of course on the specific task, and it is usually based on a form of similarity measure. This is a task that has been studied over the years due to the large variety of possible applications, of which identification of object/faces are the most famous ones. It is easy to understand how the task of face recognition is strictly related to the one of VG as in both cases the final answer is obtained from the ground truth provided by the database item which is closest to the query; the nature of the ground truth determines the task as well, it being either the identity of the face or the GPS tag of an image.

Generally speaking, an image retrieval pipeline is mainly composed by the following 4 steps, as it is depicted in Fig. 2.1:

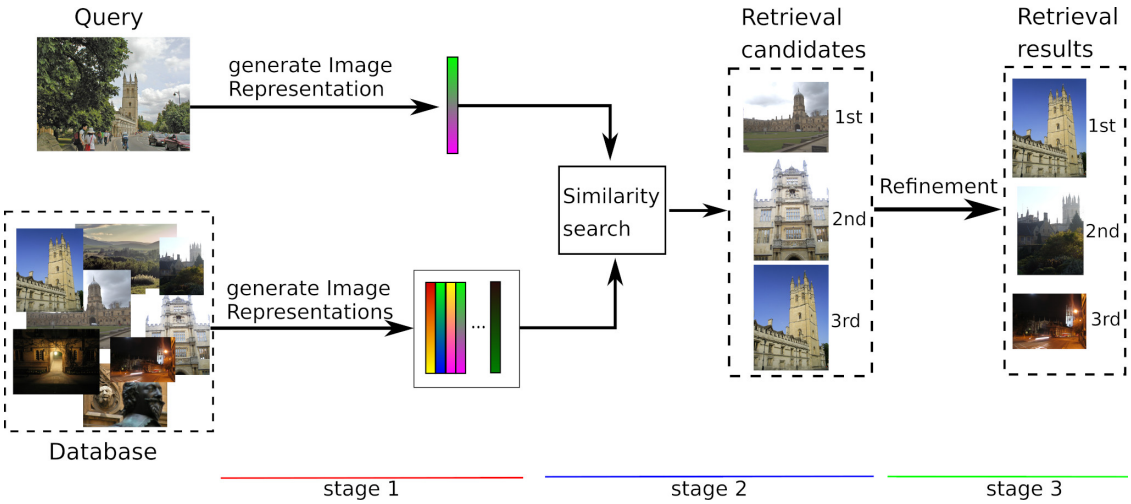



Figure 2.1: Architecture of a standard image retrieval system. Image from [2].

- **Features extraction**, consists in obtaining a representation of the query at hand in the chosen feature space, in a way that should include all the necessary relevant information to describe the image;
- **Similarity Search**, is the part of the pipeline in which an algorithm

evaluates the database items that with higher probability represents the same place depicted from the extracted features;

- **Re-ranking**, a refinement process on the top candidates identified by the similarity search, involving any method that is able to estimate with a better granularity which one is the most likely to be the positive match.

### 2.2.2 Landmark Retrieval

A research field that is somewhat similar to Visual Geo-Localization is the on Landmark Retrieval (LR). In the LR task the focus is shifted from the GPS coordinates of an image to its content depicting or not a particular landmark; where a landmark can be any relevant building, piece of artwork or really any distinguishable object.

The fundamental difference with VG is that in this task the geographical distance is not considered to be relevant; this means that a database item is considered as a valid match even if the landmark of interest present in the query is visible from a long distance. (e.g. a picture of the Tour Eiffel from below matches also with images captured from far away in which the Tour is still visible). Another substantial difference among the 2 task is that in LR typically the set of landmarks of interest is a discrete set, whereas in VG the focus is on estimating the GPS labels in the area of interest in a continuous fashion.

This task has also been extensively studied in the literature and many datasets are available. The most famous one is probably Google Landmark, available in 2 versions ([6], [7]), and it includes landmarks from all over the world. There also exist some smaller, city-level datasets like Paris [8] and Oxford [9]. In some cases it can be interesting to evaluate the performance of models for VG that have been pretrained on the aforementioned Landmark Retrieval datasets. In the scope of the benchmarking work for this thesis, this possibility has been analyzed and will be exposed later on.

### 2.2.3 Visual Localization

Another task related to VG can be identified in the field of Visual Localization (VL), of which an extensive survey can be found in [10]. In VL the objective is, given the query image located in a known environment, to predict accurately the 6 degrees-of-freedom of the camera, that is to infer both spatial position and orientation in the scene. Therefore as regards the location information, VG and VL methods are basically interchangeable, however

the radically different setting of the problem has led the literature to treat separately the 2 tasks. Nevertheless, in [11] the authors evaluate the performances of methods from VG when applied to VL in 3 different modalities:

- **Pose approximation**, where the pose of the query is estimated with a weighted combination of the top-k database items retrieved. Clearly this downstream task allows to build upon a retrieval system from VG;
- **Pose estimation without a global map**, in which the top retrieval from the database are used on-the-fly to compute a 3D map of the environment, and subsequently it is compared to the query to compute the orientation;
- **Pose estimation wit a global map**, in this setting the 3D map of the scene is already known and pre-built, rather than estimated on-the-fly for each query; matches with the 3D-tagged database images are then used to reconstruct the 6 DoF position.

Some relevant datasets for this task, each with different characteristics are **Aachen Day-Night** [12], an outdoor setting with strong viewpoint changes, the famous **RobotCar Seasons** [13] for the autonomous driving setting, therefore presenting little changes in perspective, and **Baidu Mall** [14] with the presence of many cases of occlusion and various disturbances in an urban environment.

## 2.3 Approaches to VG

In this section the main approaches that have been used in the literature for the VG task are presented briefly, and afterwards the most common one, that is the Image Retrieval approach and is also the one that has been deepened in the work of this thesis is described in more details.

### 2.3.1 Image Retrieval Approach

As already mentioned, this is the way in which the task is commonly cast as. Briefly, in this setting the VG problem consists in estimating the GPS location of a query against a geographically-complete database whose images include the ground truth GPS label. The query position is obtained by finding the best match in the database set.

The described approach is of course strictly related to the general setting of retrieval described in Sec. 2.2.1; nevertheless there is a substantial shift in the ultimate goal. In fact, the objective is not to identify among the gallery set the image that is most similar to the query merely based on the similarity of the depicted content, but rather to match its spatial location.

This seemingly uninfluential conceptual difference is actually quite important. It follows that a successful method for VG needs to learn how to extract relevant information about the semantic categories of the environment represented in an image (presence of building, their relative position, form and structure...) because the mere appearance can be subject to heavy changes upon different seasonality or weather conditions; moreover it is not unlikely for geographically distinct points in a city to be quite similar, especially in neighborhoods with a shared architectural style. This issue is well-known in the literature and it is discussed in surveys on the topic like [4, 3] and it referred to as *perceptual aliasing*.

### 2.3.2 Classification Approach

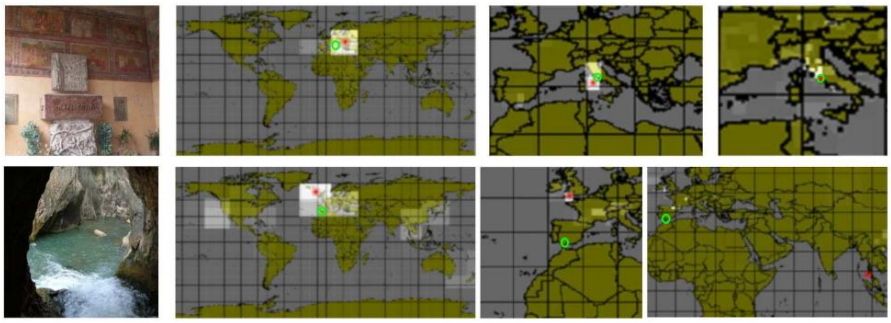


Figure 2.2: Image that shows the working scheme of the VG task cast with a classification approach. Each rows displays an example of query and on the right the prediction (red dot) and truth (green dot) on an increasingly granular grid. Image from [15]

A radically different approach from the retrieval setting is represented by treating the problem as a classification task. This is an idea that has stemmed more recently in the deep-learning related literature for the task, motivated by various factor, one being the outstanding performances that deep classifier networks have been able to achieve when dealing with large-scale datasets (both in terms of data points and in number of classes). Moreover there is also an inspiration from the way in which our human brains work, as typically

we do not need to compare a place singularly with every single one we have ever seen and still we are able to estimate the geographical block in which a picture was taken.

A representation of a geographical grid used for VG is in Fig. 2.2. In this formulation the focus is not anymore to retrieve in a database the specific position of a query, but rather to locate the query into a region, which can be as specific as the task requires it to be. This idea is also a building block for global Geo-Localization, also allowing to use different granularities for the classification to adapt to the specific use-case. Works that treat this theme can be found in [16, 17, 15].

Note that methods that use classification at training time, and revert to retrieval at inference time do not fall into this category.

### 2.3.3 Other Approaches

As already stated, some of the main issues that affect the VG task are related to the difficulty in extracting robust features that are agnostic to changes in viewpoint, seasonality, and domain shift in general. In this sub-section are presented some methods with which researchers have tried to tackle such problems, abandoning the retrieval framework and trying to exploit 3D based methods and Cross-Appearance datasets that explicitly treat the domain shift problem.

#### 2.3.3.1 3D-Based Methods

Methods based on 3D databases are in general less efficient than methods that rely on images, however they find their field of applicability when the application of interest requires an accurate estimate of the pose in terms of location and orientation of the query, similarly to the VL task. A good survey of this kind of methods is included in [13], and in [19] they try to limit the efficiency loss due to the 3D pose estimation by reverting to a traditional bi-dimensional search to retrieve the geographical position and then use 3D data only to compute more precisely the 6 DoF pose.

The methods that tackle this task usually rely on geo-tagged 3D reconstructions of the space, inside which the query is localized, and the match is evaluated also in terms of orientation and not only of position. Therefore there is the need of suitable datasets, that can be built in a much more expensive way than traditional datasets for VG are, using RADAR or special RGB-depth cameras.

Figure 2.3: Image from [18]. In these 2 examples the query is represented in the middle, and the environment around it is the three-dimensional gallery. The method reconstructs the query in the environment taking the pose into account.

#### 2.3.3.2 Cross-Appearance Localization

This sub-section aims at discussing techniques that explicitly take into account domain shifts between the images seen in the database and the queries, showing how researchers have specifically tried to tackle this problem.

**2.3.3.2.1 Cross-domain** In the specific sub-case of Cross-Domain VG, the problem setting is similar to general field of Domain Adaptation in Computer Vision, where datasets and methods acknowledge the presence of different distributions in the data (in the case of VG, between the domains of query and gallery sets). An example of domain shift can be day-night, real-painting or clip-art, as it is shown in Fig. 2.4.

A related work in which authors try to tackle this problem using techniques from Domain Adaptation like Alignment via Discriminative Visual Elements can be found in [20]. In their case queries are represented by paintings, to be matched against normal database items.

**2.3.3.2.2 Cross-view** In an even more challenging setting, the Cross-View VG task tries to exploit databases made from aerial pictures. It is clear what would be the advantage in managing to use such a database, as aerial pictures from satellites are widely available for any place on the globe and a system able to work with them could leverage this easily available datasets. However the challenge lies in the fact that queries, in a real scenario

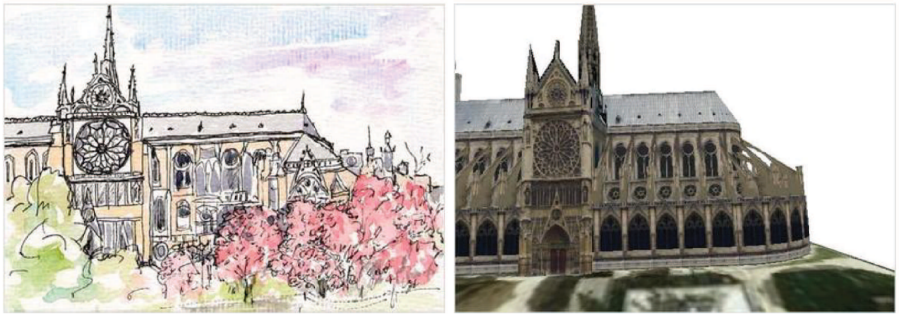


Figure 2.4: Cross-domain: painting vs front-view

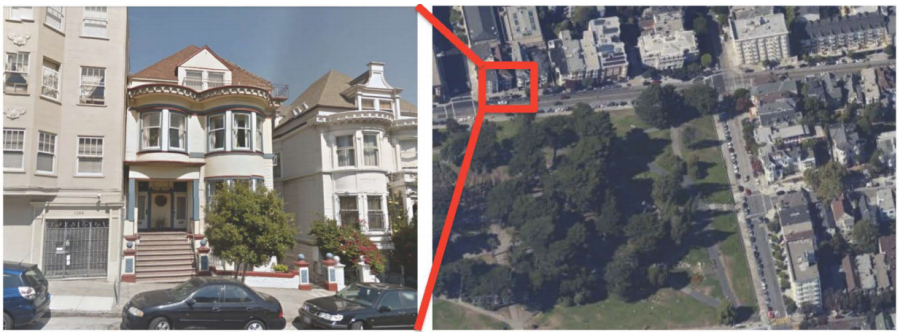


Figure 2.5: Cross-view: aerial vs ground-level

would still be taken by the users and therefore present front or side views, being taken from the ground level. An example of the substantial difference between aerial and ground-level images is visible in Fig. 2.5. Given the convenience that exploiting aerial images would grant, this field has been studied since early works before the advent of Deep-Learning (DL) methods, in works like [21], and more recently by [22]. In the case of [23] the authors propose to use transformations such as image rectification on the queries in order to reduce the domain gap with the aerial-views present in the database.

## 2.4 VG as Image Retrieval Task

From now on, as anticipated, are discussed methods that approach the problem from the retrieval perspective, as it is the one adopted by the most successful works in the literature and it has been the principle point of interest for the experiments run under the scope of this thesis.

The general retrieval pipeline depicted in Fig. 2.1 has been detailed in Sect. 2.2.1. As discussed, the first part of pipeline has the fundamentally

important job of extracting meaningful features from the queries, to be compared with the database later on in the pipeline. Nowadays the most common tool used for this purpose in the DL world is constituted by Convolutional Neural Networks (CNNs), however this research field was popular even before the deep learning wave and therefore in the following subsection are presented first the *Hand-Crafted* extraction approaches and then the Deep ones.

### 2.4.1 Hand-Crafted Representations for VG

The representations obtained from the query can be differentiated mainly among the scope of the descriptors, being it either local or global.

#### 2.4.1.1 Local Descriptors

An extracted representation is defined as local if it only bases its analysis on a subset of the input image. This techniques were most popular before the advent of deep learning. The patches of the images considered could be as small as single pixels. The general mechanism with which these methods work is to first sample the patches with the desired density, and subsequently a kernel is used as a detector to find saliency points by comparing pixels with their neighborhood, and then to extract the feature representation only in the neighborhood of the *keypoints*. Some examples of methods based on this idea that were popular in the beginning of the century are Scale Invariant Feature Transform (SIFT, [24]), which was followed by some variant that tried to improve performances like RootSIFT [25] and SURF [26] which tries to make the method suitable for an on-line usage.

Subsequently, a comparison between images would consist in the pairwise comparison between such local descriptors. However this procedure is not suitable for a large-scale search inside the gallery, so researchers have tried to work around this issue. In [27], the authors propose to use a lightweight classifier to discriminate the most useful features; this approach stems from the observation that the description extractors are task-agnostic and detect any seemingly relevant information from the image, that therefore may not be a discriminative feature for the task of interest, i.e. the one of VG.

Following this chain of reasoning stemmed some of the most popular methods in this research area. In particular, the idea is that since comparing in a pairwise fashion all the descriptors yields a non-computationally-feasible approach, a better strategy would be to instead focus on the statistic of

the descriptors. The first work that proposed this approach was [28], that borrowed from Natural Language Processing (NLP) the idea of clustering descriptors, and then to interpret their cluster assignment as a quantization of the features in terms of a codebook of visual words. This is an approach that had proved itself to be successful in NLP, where the concept of 'word' is more naturally applied and word occurrences are counted. The parallelism that these methods establish is based on seeing the centroids of the obtained Voronoi cells as *visual words.* In this framework, the final image descriptor is obtained with the frequency count of the assignments of features to each visual word (i.e. the obtained centroids), with a *tf-idf* (Term Frequency - Inverse Document Frequency) weighting scheme analogous to the one adopted in Bag of Words approaches for text mining. The described representation is compared with database images using any similarity metric, like Euclidian distance or cosine similarity. One of the advantages of this approach is the fact that it yields fixed-length vectors, of size equal to the chosen number of clusters (or visual words), that are easier to compare among themselves.

One of the main problems that affect this framework is the fact that 'hard' cluster assignments leave no space for nuances and may cause for loss of information whenever some features are close to more than one centroid.

To prevent this kind of problems in a popular work by Jegou et al. [29] (that will also be renewed later on in the DL era) the authors proposed to 'soften' the cluster assignment by taking into account the residual distances between features and their cluster-assigned centroid, allowing to obtain a more clear picture and reduce information loss. This method goes by the name of VLAD: Vector of locally Aggregated Descriptors.

In the paper [30] the authors point out that all the methods in this described category, that are based on building images representations from locally extracted descriptors all need to include 2 different steps in their pipelines:

- **Embedding**: step that aims at extracting features from local patches of the images, possibly into spaces of richer dimensionality while trying to highlight the more distinctive features and avoiding false positives; an example to this end is VLAD in the way that their soft-assignment suppresses importance of features that are close to more than one cluster, or also the Triangular embedding proposed in [30].

- **Aggregation**: to combine the various local embeddings into a single, compact representation.

#### 2.4.1.2 Global Descriptors

If one should posses the ability to process images as a whole, the need for the 2-steps (Embedding - Aggregation) discussed in the previous paragraph is eliminated. This is the principle on which Global-descriptor-based methods build upon.

Therefore this approaches require the ability to extract directly from the image an holistic set of features able to capture all the essential information. This eliminates also the need of the detection step that is performed in the embedding phase for Locally-describing approaches, and therefore can provide more efficient methods, for which the drawback is the added obstacle of being able to extract meaningful features directly on the whole image. An example of methods that treat images globally is GIST [31], a popular work from which many optimized version stemmed out like [32] that reduces its memory footprint. In general with respect to local descriptors these methods can turn out to be less robust, especially when dealing with occlusions and changes in viewpoint; whereas in other cases of domain shift such as weather or illumination changes global descriptors can actually yield stable descriptors as showed in the work of [33].

### 2.4.2 Deep Learning Representations for VG

In the last decade CNNs have become the Off-The-Shelf architecture for processing images (and in some cases videos as well), surpassing hand-crafted methods in all visual task since the famous introduction of Alexnet [34] at the ImageNet [35] challenge in 2012. Beyond the initial success in image classification, studies in all fields of Computer Vision have proven the excellent transferability of CNNs on different domains when trained on large and general purpose datasets like the famous ImageNet [35].

Image Retrieval is not an exception in this sense and the advent of CNNs brought many novelty and better results for VG as well, and in this section they are going to be object of discussion as they are nowadays the most broadly used.

In very recent years the community has shown an interest in the application of Transformer architectures, originally meant for NLP, to Computer Vision tasks. In this first section about Related works the focus is kept only on CNN-based methods, as in the VG literature there are little or no works exploring these new architectures, and they are discussed later on in the section about the innovative contribution of this thesis.

#### 2.4.2.1 Fully connected representations

The works that pionereed the use of deep learning for VG stemmed after works like [36] that showed how the final classification layer of a CNN can contain a surprisingly effective representation for Image Retrieval, specifically by considering the vector of activations of the said FC (Fully-Connected) layer of network pre-trained on ImageNet.

Further works ([37]) proved that it was possible to improve upon this result by adopting metric learning approaches using a triplet loss to fine-tune the network specifically for the retrieval task.

From these early results researchers could conclude that representations extracted in this fashion present similar characteristics as Global descriptors, sharing their weaknesses as well like the lack of robustness to occlusion and viewpoint changes. To work around these challenges some local version of these approaches were developed, and even though performances were comparable with the ones of traditional methods, important issues remained such as the increased computational cost, high number of parameters that is characteristic of FC representations, and moreover the fact that this constrained input size to be fixed, yielding overall less flexible methods.

#### 2.4.2.2 Deep Feature-maps based representations

More recent works have moved on from the use of FC-based representations, upon the intuitions that the feature maps of generic size $CxHxW$ at the exit of the n-th convolutional layer of a network contain a spatial-related representation of the features extracted from the image. In the first attempt in [38], Babenko et al. tried to use such feature maps by simply flattening them into a normalized vector and using it as a descriptor. This proposition paved the way for the current state-of-the-art techniques, even though it proved to be too simple and with sub-optimal results compared to FC-based techniques, mainly due to the fact that taking the feature maps directly as a representation causes the loss of the spatial information that was encoded in those tensors.

Building upon these concepts are now presented the main techniques that are considered state-of-the-art in terms of either performances or efficiency, dividing them in 1) aggregation-based methods and 2) pooling-based methods

**2.4.2.2.1 Aggregation of convolutional feature maps** The methods in this paragraph have in common the perspective of seeing the $CXHXW$

feature maps as a lattice made up of C-sized descriptors, mapped to spatial locations. In other words, the interpretation given to the ouput of convolutional layer is to see these tensors as an ensemble of local descriptors, therefore allowing to borrow the wisdom developed for hand-crafted methods that used to rely on local descriptors as well.

Therefore the only missing step to build an end-to-end trainable method is the aggregation step, of which the most successful example is given by NetVLAD [39], that takes on the concept of soft-cluster assignment of VLAD [29] and implements it with a differentiable layer that is therefore pluggable on top of any convolutional architecture. The working concept remains pretty much the same, except that to achieve differentiability the assignment is not anymore a binary variable but it becomes continous with a softmax normalization, so that the sum of the cluster assignments is 1. The methods than still relies on residual distances between the computed features and the cluster centroids to use as a descriptor. This implementation contains more trainable parameters than the original VLAD and it achieves state-of-the-art results while providing great flexibility as well.

The downside of this approach is that the final descriptor is made up by the residual of the cluster assignments from each of the centroids ( which are traditionally 64) for each $HxW$ input grid. This last number varies depending on the backbone, and it is typically one of $\{256, 512, 1024\}$. and therefore the final size varies from 16384 to 65536 which leads ultimately to a considerable memory footprint.

For this reason in many works it is considered the possibility of using a dimensionality reduction technique such as PCA to reduce the dimensionality of the final image descriptor.

**2.4.2.2.2 Pooling of convolutional feature maps** As an alternative to performing an aggregation step it can be considered the possibility to use less elaborate schemes to rely more on the raw feature maps. This methods stem from the belief that maps from late layers of convolutional networks already have a significant discriminative capability, that can be exploited by computing their statistics. The most basic idea building on this concept was MAC proposed by [40] (Maximum Activations of Convolutions) which performs max-pooling on each of the feature maps, drastically reducing dimensionality and obtaining directly a representational vector.

An improvement on this concept was brought by SPoC ([41], Sum-Pooled Convolutions) that instead of max-pooling uses sum-pooling, showing to achieve better results especially when dealing with scale changes or occlusion,

as showed by [42] where the authors argue that using sum rather than the max operation yields features more robust to local distractors.

A further approach that tries to bring together the advantages of these previous 2 is represented by R-MAC (Regional-Mac) [43], where the max-pooling is computed separately on patches of the feature maps which are finally sum-pooled and normalized.

The other pooling-based methods that is considered as state-of-the-art is the popular GeM layer (Generalized Mean Aggregation, [44], which performs the average-pooling on each of the feature maps returning a compact descriptor of the same size as the number of channels of the CNN extractor. Specifically, rather than simple average it is the generalized version, which therefore requires only one extra parameter that is shared among all the feature maps.

### 2.4.3 Similarity Search

In the context of VG cast as Image Retrieval problem that is under discussion, after having extracted a proper descriptor for the queries, the next step in the pipeline regards the similarity search inside the database.

The main criteria used to establish similarity in the literature is traditionally the $L_2$ norm (Euclidian distance), and less commonly the cosine similarity. Therefore the database match for the query at hand is found by comparing their distances in the chosen feature space; it follows that the Pooling-based methods presented in the previous section present an advantages with respect to Aggregation-based ones as the dimensionality of the extracted features is much lower, as well as memory footprint and computational time required. For this reason as anticipated before, methods with highly-dimensional features like NetVLAD if the retrieval time is critical to the application can make use of the PCA to bring down the dimensionality as mentioned in [39].

The main technique used to efficiently retrieve the top database matches is represented by K-Nearest Neighbors methods (kNN).

#### 2.4.3.1 K-nearest neighbors

The kNN is a simple yet popular technique to perform searches in arbitrarily high dimensional spaces. Throughout the course of the thesis to perform the experiments has been used the Faiss library [45] provided by Facebook, which implements this method while conveniently providing a number of possible

optimizations.

In fact, in cases in which the chosen database for retrieval scales to the million of samples, it may become unfeasible to perform an exact kNN and therefore some approximate kNN that rely on compress indexes can be used, trading accuracy for execution time. Some examples are: (1) Inverted Index (IVF) [28]; (2) Product Quantization [46]; (3) Inverted Multi-Index [47]; (4) Hierarchical Navigable Small World graph (HNSW) [48].

### 2.4.4 Candidates re-ranking

As show in Fig. 2.1 and discussed in Sect. 2.2.1, it is possible to perform a refinement step on the top-n retrieval from the similarity search, in an attempt to improve the ranking with this post-processing. This can be a clever workaround to the loss in accuracy that can derive from the use of approximate indexes to speed-up the kNN search.

There are a few distinct categories for these techniques:

- **Spatial Verification**: use of geometric verification to assess the correspondence of features between images and based on this estimate the correctness of the match. A popular example is given by RANSAC [49];
- **Non-Geometric**: to indicate methods that are not based on geometric assumptions and are specifically hand-crafted for the adopted pipeline like it is done in [50] for R-MAC.
- **Query expansion**: QE, a successful approach first presented in [25] and then renewed by many other works, that mainly consists in using the first-time retrieval candidates to combine them with the query with the aim of producing a more representative feature embedding to be used again as the key for a new search in the database.
- **Diffusion**: this category proposes to exploit the information in the geodesic structures in the data manifold. The idea is to build a graph with random walks in the database where edges represent similarity among database items, and then to use such a graph to guide the retrieval refinement.

## 2.5 Benchmarks in VG

Since the first part of the work for this thesis was focused on an extensive benchmarking of the existing literature in VG to asses in a standardized

framework which methods perform best in which situations, this section is dedicated to other similar works in literature, even though their number is quite scarce. Indeed, the very reason that brought the idea for this work was this lack of a clear framework to understand, especially for new practitioners who are new to the field, what are the main techniques and what are pros and cons of each one, when compared fairly against each other, eliminating possible disturbances from other minor differences.

To this end, a similar work relative to the VG task was published earlier this year (2021) by [51], whereas the already mentioned [11] does the same for the field of VL.

However, in [51], the authors, while performing a substantial number of valuable experiments, their aim is different as they directly compare methods that were proposed by their creators, and therefore their results are obtained without a common ground and each rely on different training datasets, mining procedures and so on, and our experiments will show how relevant each of these factors are. So the argument is that this approach leads to unstructured experiments because each method is evaluated in its own arbitrary setting, hence in this scenario, it is hard to distinguish which are the factors that determine a particular performance.

In this work the focus was building a modular software to enable an easy and automatized comparison of a substantial number of VG system, allowing as well to compose them to test out custom alternatives.

Another difference with respect to [51] and [11] is that whereas they do provides an analysis on the dimensionality of the features and the relative inference times, in this work were taken into account more objective statistics in order to eliminate the hardware variability factor. For example some parameters according to which methods have been evaluated are training cost, memory footprint and FLOPs of models, showing in which situations these parameters could become critical to the efficiency of the application.

## 2.6 Sequence-Based VG

Even though for a number of interesting applications of VG, especially in the fields of robotics and autonomous systems, multiple images are inherently available, such setting has not received a lot of attention in the literature, and mainly the proposed works focus on the use of single images. One clear example of a field that could benefit from the exploitation of sequence data is Visual Simultaneous Localization and Mapping (known as Visual SLAM).

In the SLAM setting, the objective for the robotic system is to map to surrounding environment, that is unknown, and subsequently to localize itself exploiting only visual elements. It follows that in this setting an important point is the ability to detect loops, in order to understand whether a certain point had already been explored or not. Upon detection of a loop with high confidence, the system is able to connect the explored points in a reliable map. Similar procedures are implemented as well in fault recovery systems; specifically some examples are the need for a mobile robot to re-localize itself after an external events of any kind has caused to lose its tracking state.
The bottom line is that all of these situations by the nature of the task need to generate a stream of visual data that is processed in order to extract the described pieces of information. Therefore it is clear how the development of a system that is able to treat sequences rather than single frames can be of interest for this applications, exploiting the additional contextual information that can be extracted from sequences to provide more accurate predictions and reduce false positives.
Nevertheless, the specific field of Sequence-based Visual Geo-localization has been studied in a limited number of works, trying to exploit temporal hints as well as the multi-viewpoints that can be present in sequences. Contrarily, regarding the processing of videos for Action Recognition, there is a rich set of proposed approaches, mainly oriented to classification [52, 53]. Moreover, considering the broad topic of processing multi-frame data, numerous research topic have diffused focusing on different aspects. Some examples are video captioning [54], scene recognition [55], 3D shape reconstruction and semantic segmentation [56, 57], and video Personal-ID [58]. A traditional approach in the literature to extract spatiotemporal features has been to expand the inductive bias of convolutional networks to the temporal axis, and use 3D kernels [59, 60]. However, 3D convolutional networks come with high computational requirements and tend to yield heavyweight networks. An additional consideration is that the majority of the listed fields, especially those oriented to action recognition, are cast as classification problems on a limited set of categories. In the field of VG the setting is different as the feature extractor needs to learn to represent the semantic elements in a scene of common places, everywhere in a city, that can allow to recognize that very specific point against a vast gallery of examples.

### 2.6.1 Current Approaches

The majority of works in the literature regarding sequence-based localization come from the field of robotics, and traditionally rely on hand-crafted methods. Similarly to what was done in single-image VG in the pre-Deep Learning era, mainly these methods rely on the use of local descriptors extracted from single frames, which are then aggregated sequentially over the frames evaluating differences in the hand-crafted representations, considering local time windows. Some examples are [61, 62, 63], or FAB-MAP [64]. All of these approaches have showed to provide improvements with respect to single-image methods when the input data contains notable shifts in appearances, validating the usefulness of exploiting multi-frame information.
A popular method, among the ones mentioned, is SeqSLAM [61] that is based on a sequential search exploring different possible paths in a matrix. The matrix, for each candidate match, is made up of the differences between the vectors representing each frame, and subsequently the sequential path encountering the lowest penalty among the possible sequences to be matched is deemed the best match. Other works tried to complement this approach using more refined sequence scoring methods, exploiting visual odometry techniques or camera-speed information [62, 65]; accounting for different possible trajectories [66], or even exploit attention applied to different possible trajectories [67].
Finally, among the last cited hand-crafted methods there are works based on the Bag of Words representation like DBoW [68], similarly to what is done in VLAD [29]. In [68] the authors combine the methods of [69] (BRIEF) and [70] (FAST) to obtain a system able to track the evolution of features with the same structure over the frames. It is equivalent to impose a constraint on the temporal evolution of features to be consistent.
The described methods, all based on hand-crafted features, have in common a series of weaknesses:

- they all rely on the notion of ordering of frames; therefore the extracted features do not depend exclusively on the semantic elements in the frames;

- they are structured to work with elevated number of frames, which are required to be consecutive; rely on long-term sequence matching, i.e. both query and database sequences must have many consecutive matching frames;

- they are based on the assumption that frames query and database sequences have a 1:1 matching, or at least a linear relationship.

Additionally, the described methods are not able to process the sequence in its entirety and are based on the matching of local features on a per-frame basis. This matching of single image descriptors is a well-studied topic in the VG, whereas the extraction of temporal relationships, encoded in a single descriptor valide for the entire sequence is a yet to be explored topic.

The main works in the VG literature that do address this topic, are quite recent, and are the following: [71, 72, 73] Starting from [71], this works introduce the use of deep architectures. The authors propose to use a convolutional backbone as a single-image feature extractor (specifically a ResNet-50 [74]), combined with three different aggregation techniques for the purpose of collecting the multi-frame features into a single descriptor. The three approaches are, ordered by their complexity, simple concatenation of descriptors, use of a Fully Connected layer, and use of a LSTM recurrent network [75]. The first two do not account for the temporal structure, whereas the third does and it consists in learning how to update the sequence descriptor step-by-step with features form each frame. Some more details about these baselines method can be found in Sec.5.4.1.
As regards SeqNet, the name of the method proposed in [72], it proposed to fuse the popular hand-crafted method of SeqSLAM [61], with modern deep-networks from the single-image VG task. Their method is made up of two main components, hierarchically combined. First, a standard convolutional network is paired with NetVLAD to obtain single image descriptors. These descriptors get fed to a separate downstream architecture that uses 1-dimensional convolutions as a weighted pooling over the temporal axis of the frames (approach from [76]); subsequently there is an average pooling over the sequence (SAP) and a normalization step. The resulting descriptors are used to select the most promising candidates with a similarity search through the available gallery, as a first filter. To select the final prediction among the top-k already chosen, this method resorts to the search for sequential path in the image-matching matrix as done in SeqSLAM. This final step selects the match based on the matching score of the k pre-selected sequences; however relying on this method brings back the same weaknesses highlighted, especially the assumption that there exist a direct correspondence between frames in the query and the database, which in general can be true only on specific datasets, and in practice limits the applicability of this method to

real-world applications.

The last of the considered works is [73], which proposes the 'Delta Descriptors' method. This method uses an unsupervised module that computes the differences over the temporal axis, of the features extracted from the single frames of the sequence. This approach is specifically focused on the setting of autonomous mobile systems, where the availability of sensors creates a stream of images. One issue in this setting, is that two sequences representing the same places, but collected with an initial shift between them, can present rather different descriptors due the initial offset that results in different appearances across the frames. Therefore the authors propose to overcome this issue by using a descriptor based on the moving differences of features across the frame, claiming that this can yield representations robust to shifts in domain and appearance. Even though they show to achieve satisfactory results with this approach, the main drawback is that to obtain stable sequence descriptors, avoiding false positives, the methods needs long sequences, around 60 frames and more. Otherwise, on short sequences this approach is more likely to yield uninformative descriptors, if the level of overlap between frames is high.

## 2.7 Self-Attention and Vision Transformers

### 2.7.1 The Attention Mechanism

The concept of *attention* was originally introduced in the field of Natural Language Processing (NLP), specifically for neural translation [77], as a mechanism to allow networks in keeping information across long sentences. Traditionally, tasks involving sequences have been addressed with Recurrent Networks (RNNs), and more recently LSTM or GRU ([75, 78] structured in an encoder-decoder scheme [79, 80, 81]. The working concept of these architecture is to use the encoder to obtain an embedded representation of the input sequence (split into tokens) to be fed to the decoder that generates the final output. Due to the recurrent architecture of the networks used, when the length of the treated sequences starts to grow, the limits of such approaches emerge. In general, recurrent network have issues in propagating relevant information though numerous time steps.
Another issue is represented by the fact that the decoder only communicates with the encoder through the hidden vector that is its ouput; this can lead to a misalignment between the two, and difficulties in focusing on relevant parts of the sequence. In [77] the authors propose to let the decoder access

the hidden state of the encoder, and to build a context vector $\mathbf{c_j}$ updated dynamically at each time step $j$. The context vector is computed on each input token based on the hidden states of the encoder $\mathbf{h_i^{in}}$ to obtain an attention score $\alpha_{ji}$. evaluate an attention score. Mathematically:

$$e_{ji} = a(\mathbf{h_i^{in}}, \mathbf{h_j^{out}}) \tag{2.1}$$

$$\alpha_{ji} = \frac{e_{ji}}{\sum_i e_{ji}} \tag{2.2}$$

$$\mathbf{c_j} = \sum_i \alpha_{ji} \mathbf{h_i^{in}} \tag{2.3}$$

The $a(\cdot, \cdot)$ represents a function to evaluate the pairwise similarity of tokens, and it is called alignment function. The attention score is obtained normalizing these similarities. In this first work [77] the scoring function was a simple shallow classifier, learnt together with the rest of the model. Subsequently, this methods was categorized as *additive attention*. In the following years the categorization of attention mechanisms based on the scoring function has been enlarged, and more modern ones are location-based [82], general and content attention [83], and finally dot-product attention [83], later refined by [84] in its scaled version. Summarizing, the concept of attention consists in a mechanism that permits to a network to selectively attend on a chosen segment of the input data on which base the final predictions.

### 2.7.2 Self-Attention

A particular attention technique is the one of *Self-attention*, and it is characterized by the fact that the input sequence is as well the target output, and the model has to learn how to deconstruct it into its meaningful elements and them to reconstruct it. This techniques have been around for some time in the field of NLP, applied to machine reading and automated captioning [85, 86, 87]. In 2017 the Transformer architecture was proposed by [84], making self-attention the cornerstone element of the layers of the proposed architecture, and since then it has been vastly adopted in virtually any NLP-related task, and more recently in Computer Vision topics as well. Even before the introduction of Tranformers, researchers working on Vision topics had been experimenting with the concept of attention, and the first to define it as *Visual Attention* were the authors of [88], describing as the ability of a network to learn to detect the most relevant area of an image, and to capture structural relationships between distant regions. Some extensive

and accurate surveys on the topic of Visual Attention can be found in [89, 90, 91].

The rest of this section introduces the Non-Local block which is an implementation of attention applied to CNNs, before moving on to the use of Transformer-Encoder layers directly on images.

Non-Local networks were proposed by [92], and follow a concept similar to the non-local averaging [93] used for image denoising. The Non-Local block is a flexible differentiable layer that can be plugged in at any point of a CNN, and can take into account also the temporal dimension if more images are present. The main idea is to build a contextual vector to highlight the salient region of an image by looking at the extracted features and their relationships rather than directly at the input. Since it is computed using the entire feature maps, it allows to go beyond the inductive bias of convolutional networks, analyzing structural relationships from different regions all across the image rather than looking only in single neighborhoods. A more detailed explanation of this module can be found in Sec. 5.4.6.
Some alternatives were proposed to reduce the otherwise quadratical cost with respect to quantity of feature maps. [94] reduces such complexity by drawing a criss-cross path and accounting only for features that are encountered on this path. A slight variation can be found in Local Relation Net [95], that proposes a block that applies attention on local windows to compute dynamically the weights of the network based on the similarity score between the features in the considered region.
Concluding this section, a notable attempt to put aside convolutional-based blocks to rely only on self-attention was proposed by [96]; to counteract the loss of spatial information the authors added a spatial position encoding token [97], somewhat anticipating the method that will be used in Vision Transformers. However, the obtained architecture had sub-optimal results with respect to attention augmented-CNNs.

### 2.7.3 Transformers

The nowadays popular Transformer architecture was first proposed in 2017 in [84]. As anticipated, it makes Self-Attention the key operation on which its building blocks are based, that allows the network to attend to all the input data with a complexity in terms of number of computing steps required that is constant, $O(1)$. In this way nor recurrent nor convolutional blocks are employed, and lacking any inductive bias they are suited to treat any data domain. Regarding sequences, they have shown outstanding advantages with

respect to traditional RNNs both in terms of 'memory', that it the ability to capture long range dependencies in the input, and in terms of parallelization as well, since there are no sequential-based operation.

The specific scoring function used to compute the attention is *scaled dot-product*. To build the attention vector, each token is represented in three different spaces, given by the following vectors: query $Q$, key $K$ and value $V$. To compute the score for a single token, the Query representation is compared to the Key of all the other tokens, to evaluate their similarity. Finally the softmax is applied to normalize the output in order to use it as a contextual weighting mask on the Value representation, which will then be residually summed to the input token. Precisely, the formula in matrix form is:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

where the introduction of the factor $1/\sqrt{d_k}$ is to prevent the softmax operation from yielding too small gradients. Another element of novelty introduced in this work, is the idea of splitting the tokens, which can have a dimensionality of the order of hundreds. In order to do so, a set of attention heads, rather than a single one, is used to split the token in segments and have each head attend a single segment. This enhances the degree of parallelization obtainable and also allows the different heads to capture different levels of structural relationships between the different elements in the set of tokens. Therefore the set of weights to obtain the Query, Key, Vector projections of the token are differentiated for each of the $h$ attention heads $\{W_i^q, W_i^k, W_i^v\}$, with $i \in \{0, \ldots, h-1\}$. In the original paper [84], the authors propose $h = 8$. For a more detailed discussion on the way that Transformer architectures operate, refer to Sec. 5.4.2.

### 2.7.4 Vision Transformers

As already stated, following the introduction of Transformers [84], a remarkable wave of applications with outstanding results has followed in the NLP community. Some of the most popular among the developed architectures are Bidirectional Encoder Representations from Transformers (BERT) [98]; from Google Brains labs three different versions of GPT were proposed in succession [99, 100, 101], the Generative Pre-trained Transformer; another examples is Text-to-Text Transfer Transformer) [102]. More recently, optimized versions of BERT have been widely adopted, like RoBERTa (Robustly

Optimized BERT) [102].
The interest for these kind of architectures has not been limited to NLP application, and with a short delay many researchers in Computer Vision have started pursuing the target of building a Transformer-based architecture able to operate on visual data. For an extensive summary of this phenomenon, two complete surveys are [103, 91].

Some of the first proposed approaches to remove convolutional layers were [104, 105]; the authors replace convolution with single-headed only self-attention. Nonetheless, the real breakthrough towards successful implementation of a full Transformer-based architecture able to obtain state-of-the-art result on Image Recognition was ViT, appropriately called Vision Transformer [106]. The main idea is to cut the imagtes in 16x16 patches which are then flattened with a learnt linear transformation in embeddings that are used as tokens. The authors turned the disadvantage of transformers, that due to the lack of any inductive bias require large datasets to properly learn the data distribution, into the strength of this architecture that is able to outperform state-of-the-art CNNs on large-scaled datasets. Therefore pre-training becomes an important factor, and the authors use enormous datasets like JFT [107] before moving on to ImageNet to *fine-tune* the model, which however is itself a rather big dataset. These characteristics are certainly useful, but high computational requirement e large scale datasets are of course also a weakness; therefore many recent works focused on making more accessible the use of vision transformers.
A famous example is DeiT [108], that eliminates the need of a heavy pre-training by using a helper CNN network as a teacher. Other interesting approaches focus on re-injecting the bias of convolutional layers in the early stages of the network, that is certainly useful especially to detect low level features. Moreover, this yields substantially lighter models, that require as well less data to train. To fall in this category are LocalViT [109], LeViT [110], and the so-called Convolutional Vision Transformer, CvT [111]. An additional example of this network, that is especially focused on lowering transformer requirements both in terms of computation, memory and data needed, is the Compact Convolutional Transformer, CCT [112]. The latter has been thoroughlly explored throughout the work for this thesis and a more detailed description can be found in Sec. 5.5.6.

Finally, the last architecture mentioned is not properly a vision transformer, but has been developed in the field of Action Recognition. It has been called Timesformer [113], and its building concept is to extend the Self-Attention mechanism to not only extract contextual information tokens

coming from an image, but to extend its scope to include the temporal axis and therefore attend to token from different frames as well. This allows the network to encode spatiotemporal structural relationship as well. This model achieves state-of-the art results on video understanding tasks. The authors propose different version of this spatio-temporal attention, differentiating themselves in computational cost, and specifically for this work the *Dividede Space-Time attention* has been implemented to explore its adaptability to the VG task; a more thorough explanation of its functioning is provided in Sec. 5.4.4.

# Chapter 3

# Datasets

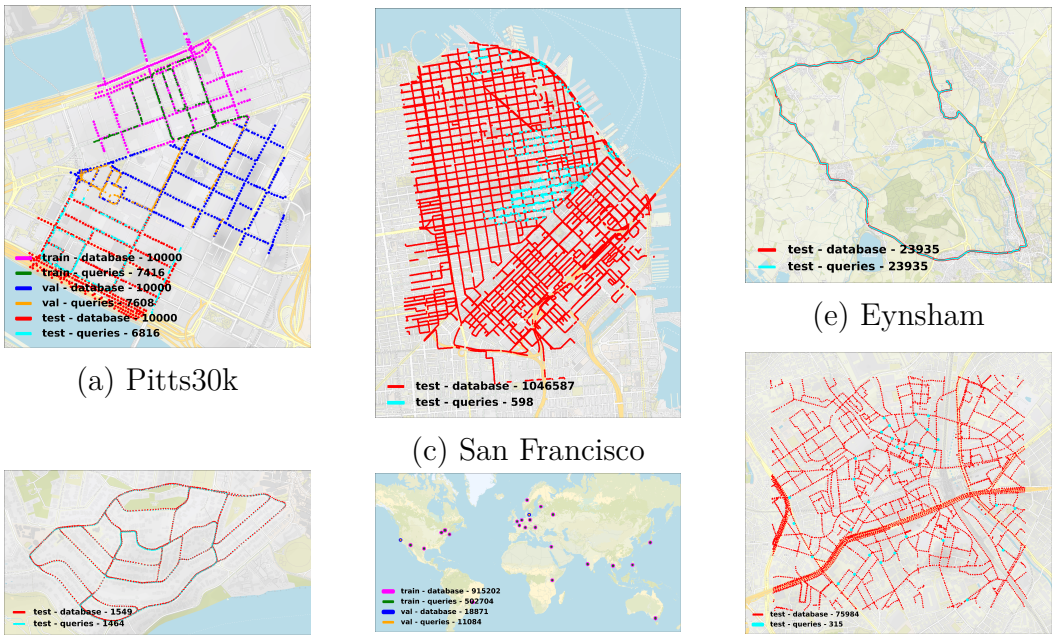
## 3.1 Datasets for benchmarking

In the scope of the benchmark, it is important to use a rich and variegated collection of datasets, to evaluate the performances of the models under different scenarios of both scale, image-type (panorama, front-view, phone...) in order to have representative results for a number of possible real-life use-cases. Tab. 3.1 reports some statistics about the 6 datasets that were chosen, whereas Fig. 3.1 shows their geographical coverage. In this section a general explanation about the overall datasets characteristics is given, and later on a brief paragraph is dedicated to each one of them.

The high grade of variability contained in this datasets allows to perform a thorough evaluation of model performances with respect also to the potential applications of interest; for example datasets containing panorama views may be better indicators of performances for pose estimation tasks, whereas front

Table 3.1: **Datasets characteristics:** Regarding images types: cropped from a 360° panorama undistorted; "front-view" refers to only one forward facing view available; "phone" is for smartphone-taken pictures. Both "panorama" and "front-view" viewpoint is a camera on the rooftop of a moving car. Table from [1].

| | **# train database/queries** | **# val database/queries** | **# test database/queries** | **Dataset size** | **Database type** | **Database img. size** | **Queries type** | **Queries size** |
|---|---|---|---|---|---|---|---|---|
| Pitts30k | 10K / 7.4K | 10K / 7.6K | 10K / 6.8K | 2.0 GB | panorama | 480×640 | panorama | 480×640 |
| MSLS | 915K / 503K | 19K / 11K | 39K / 27K | 56 GB | front-view* | 480×640** | front-view* | 480×640** |
| Tokyo 24/7 | 0 / 0 | 0 / 0 | 75K / 315 | 4.0 GB | panorama | 480×640 | phone | variable |
| R-SF | 0 / 0 | 0 / 0 | 1.05M / 598 | 36 GB | panorama | 480×640 | phone | variable |
| Eynsham | 0 / 0 | 0 / 0 | 24K / 24K | 1.2 GB | panorama | 512×384 | panorama | 512×384 |
| St Lucia | 0 / 0 | 0 / 0 | 1.5K / 1.5K | 124 MB | front-view | 480×640 | front-view | 480×640 |




(a) Pitts30k

(c) San Francisco

(e) Eynsham

(b) St Lucia

(d) MSLS

(f) Tokyo 24/7

Figure 3.1: **Geographical coverage of various datasets,** generated using the software developed for [1]. Figure from [1].

and side view images may be more relevant for autonomous driving scenarios.

In order to be successfully used to train a VG system, a dataset should hold the following characteristics:

- **Density** of the database so that queries can have with good probability some positive matches;
- **Large-scale** to avoid overfitting on the appearance of a small area and not be able to generalize;
- **Possess GPS ground truth** for obvious reasons;
- **Include a Time Machine**, that is including images collected over the years; as showed by [39] it is fundamental to obtain robust models that are able to focus only on the semantic aspect of the images and not the ones that are subject to change overtime and with seasonality/lighting.

It is important to point out that for many other fields like Landmark Retrieval this checklist is not verified and for example datasets like Google Landmarks do not include GPS labels or a Time machine. In some other

cases, like for Visual Localization, datasets can tend to be very dense, covering a smaller geographical area as their purpose is to precisely estimate a 6 DoF pose in a small environment.

It has been decided to use only two datasets to train the considered methods, specifically Pitts30k [39] and Mapillary Street-Level-Sequences (MSLS) [114] because of their complementary characteristics, other than the fundamental fact that they both are generated over a timespan of several years, which is of paramount importance for training as explained. In particular, Pitts30k presents an homogeneous distribution of urban views all taken from panorama images, in a relatively small size ($7.4K$ queries only for training), and even if the validation and test sets are disjointed between them and the train set as well, they are still all panorama images in the urban context. As regards MSLS instead, it is a dataset that proposes itself as the ultimate training dataset for many different applications in VG, and contains a wide range of seasonal, lighting and weather conditions from dozens of cities all around the world, with over $1.6M$ images overall. Moreover the validation set is made up of a set of cities disjointed from the one of training, so that the images on which methods are tested are in totally different countries or even continents than the training ones thus providing more indications on the generalization capability of methods. It is worth mentioning that for MSLS the authors never released the GPS labels for test split, therefore like previously done in the literature by [115] results are computed on the validation split; even though this is not ideal, given the enormous variability between training and validation in the dataset, in the end it is not a problem and results can still be seen as a comprehensive score.

Finally, to obtain a complete evaluation of the property that training on a dataset rather than another can confer to models, the other 4 datasets are used as a strong test-suite against which their generalization capabilities can be thoroughly verified. Specifically, the other datasets are the following:
Tokyo 24/7 [116], Revisited San Francisco (R-SF) [117, 118], Eynsham [119] and St Lucia [120]. Revisited San Francisco refers to the dataset presented in [117] but using the refined query poses computed by [118].
Throughout the thesis it has also been analyzed the impact that pre-training on datasets other than ImageNet can have, specifically by using scale landmark retrieval or classification datasets, such as Google Landmark [6, 7] and Places 365 [7], to understand if the learning performed on such datasets is transferable with good results to the VG task.

## 3.2 Mapillary SLS

**Mapillary SLS** [114] contains data separated by cities from all continents, with different domains regarding seasonality, lighting, weather and also data spans over decades. As mentioned above it does include a test split for which however labels were never released.

## 3.3 Pittsburgh

Namely **Pitts30k** in the version used, it comes from the work of Arandjelovic [39] which originally proposed it as a restriction of Pitts250k [121]. It is considerably smaller than MSLS as showed in Tab. 3.1, and it is collected using the APIs from Street View, completely in Pittsburgh; to enhance the robustness of models there is a 2-year time difference between query and gallery images. Other kinds of domain shifts are however limited.

## 3.4 San Francisco

For **San Francisco** [117] instead of Street View, it has been collected with a camera on car rooftop going around the city to build a substantial gallery ( $1M$ images) whereas queries only amount to a few hundreds and were collected manually using smartphones. Therefore many works have tried to label the queries using Visual Localization approaches, and accuracy was the factor driving the choice towards the more accurate query coordinates contained in the R-SF version from [118].

## 3.5 Tokyo 24-7

Proposed by [116] it has a considerably large database, in contrast to a few hundreds query only. Whereas the database is build using the Street View APIs, queries come from phone-taken pictures, and there is a specific focus on having the queries depict the same places under different lighting condition. In fact queries are collected manually for this purpose.

Due to the limited number of queries that would limit the training capability, in some works [39, 122] it has been used the Tokyo Time Machine to that purpose.

## 3.6 St. Lucia

**St Lucia** [120] is a more old-fashioned datasets that was popular in the early days of VG, collected in a the St Lucia neighborhood of Brisbane. Before the Street View APIs became convenient as they are now, this dataset is build by sending a car around a loop with a camera; in this particular case queries and database are obtained choosing different laps. Using the purposely developed software it has been pre-processed to reduce the density to 5 meters separated frames.

## 3.7 Eynsham

**Eynsham** [119] is also an old-fashioned datasets that was popular in the early days of VG, and it is made up of single-channel images (i.e. grayscale) obtained from a car-rooftop camera on a car around the city of Oxford. Therefore it contains countryside setting images.

Specifically, queries and databases are collected on the same path, during separate loops performed around the city.

In Fig. 3.2 are shown exemplifying pairs of queries and their positive matches in the database. This should let the reader get a sense of the strong generalization capabilities that models require to perform inference on difference datasets and as well the broad spectrum of changes in illumination and environment in general that datasets present.

(a) Pitts30k

(c) San Francisco

(e) Eynsham

(b) Tokyo 24/7

(d) MSLS

(f) St Lucia

Figure 3.2: **Examples of queries and one of their positive matches** for all the considered datasets.

# Part I

# Chapter 4

# Benchmarking

This chapter is devoted to discussing the methodologies, architectures and results regarding the first part of this thesis, whose aim was to establish an evaluation framework for Visual Geo-localization, which is missing, and provide guidelines on the most suitable approach for practical application. Therefore the contribution in this first part has been a thorough exploration of the VG field, by providing an extensive benchmark of the most popular and effective methods proposed in the literature over the years, for each of the specific aspects of a general VG pipeline. For each specific step, a number of alternatives have been evaluated through an exhaustive set of experiments, and discussed taking into account real-world constraints. This considerable amount of work is the outcome of a group effort, including a submission to NeurIPS Benchmark and Datasets 2021 track. For this reason all the figures and tables reported in this chapter will cite this submitted paper [1].

## 4.1 Methodology

This section begins with a clarification of the general setting in which all the discussed experiments have been run. Fig. 4.1 reports the adopted model of a VG pipeline, whose elements are modularly switched to implement each tested method. Results are then reported in Sect. 4.2

Reminding that the adopted problem-framing is the one of Image Retrieval, the working scheme of the pipeline at inference time is the following: upon receiving a new **query** whose location is unknown, the system runs the image though a feature extractor and afterwards performs a search into the available database to retrieve the best match for the extracted features.

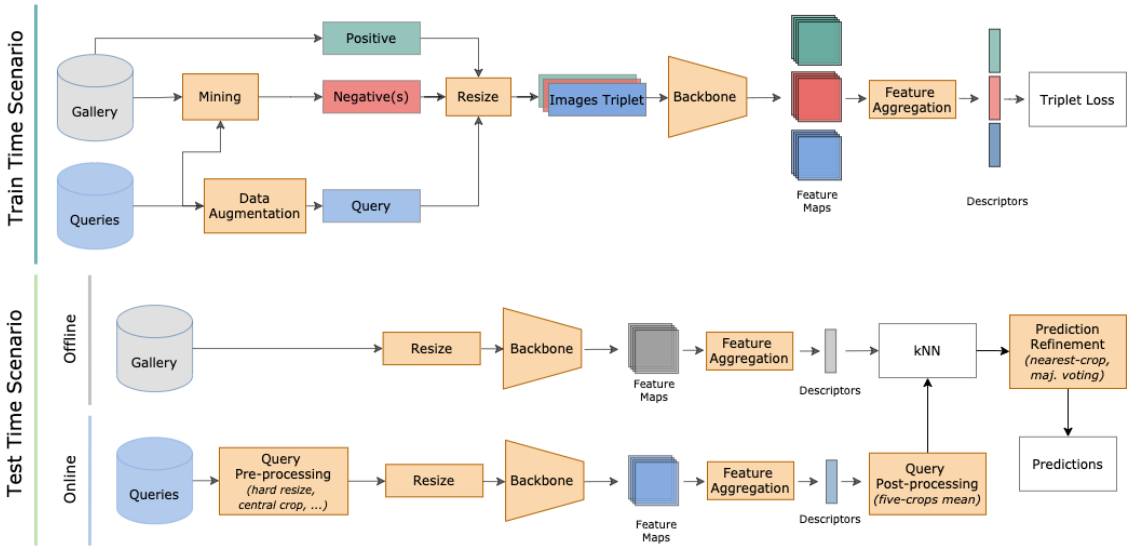



Figure 4.1: **Schematics of a VG pipeline.** The orange block in the figure represent the switchable modules in a general VG system, for which a broad variety of methods was analyzed in the set of experiments performed. Diagram from [1].

Therefore in a deployed VG application the system can compute the feature representation of the database before-hand in an offline fashion, to reduce to the minimum the operations required to compute the result for queries that are submitted by users. Throughout this work this will be taken into account to evaluate the efficiency of methods, as it means that the scale of the database mainly determines the memory requirement, whereas the size of the descriptors and the backbone used for feature extraction determine together the inference time. Note that the retrieval stage is performed using the already cited Faiss [45] library; using in most of the cases an exact kNN and in 4.2.5 some approximate kNN with more efficient indexing are evaluated.
Fig. 4.1 is exemplifying of the many design choices that need to be defined by an engineering process for the desired application; and all of these factors such as the choice of the backbone, the search engine, mining techniques all have a significant impact on both performances and efficiency of the system. For this reason when analyzing different methods in the literature there is not a clear-cut point to look at to understand the performance difference, therefore this benchmark should help in this sense providing a systematical and standard framework to analyze the impact of design choices (both at train and inference time), including a substantial number of state-of-the-art

methods.

### 4.1.1 Experiments Summary

Specifically, from the general pipeline reported above, in the following sections are investigated in details:

- **Backbone**: the first major step in a VG pipeline is the feature extraction from the images. Usually pretrained convolutional networks are used as a backbone, thus this choice heavily influences the results and the weight of the final system (Sec. 4.2.1);
- **Aggregation method**: the feature maps extracted by a convolutional backbone are commonly aggregated via pooling or other methods to produce a single global descriptor to be used for the retrieval (Sec. 4.2.2);
- **Mining**: an efficient way to mine negatives to be fed to the triplet loss is an important factor in a VG system (Sec. 4.2.3 );
- **Pre-Training**: it has also been studied if the system can benefit from backbones trained on datasets other than ImageNet. (Sec. 4.2.4);
- **Inference Time**: real-world applications of VG often are required to work in real-time, providing fast localization prediction. For this reason the inference time required to output the best database match for a new query submitted to the system is a fundamental element to evaluate for a given method. A comprehensive analysis has been carried out to understand which factors affect inference time, and since the time to perform the similarity search through the database is certainly an impacting element, different optimization techniques regarding the indexing and retrieval procedures have been explored. (Sec. 4.2.5);
- **Data augmentation, Pre/Post processing** : Data augmentation provides a simple yet efficient way to improve robustness in the model. It has been applied to the queries, as in a real world scenario they might come from different sources. Among preprocessing techniques resizing input images is an option, both at train and test time. While this factor is often overlooked in research papers, it is shown that it can have a strong impact on results and computational complexity. Other pre and post-processing techniques applied to the queries are investingated to see if they can lead to better results and allow parallelization of multiple images in a batch. (Sec. 4.2.6)

### 4.1.2 Technical protocol

Throughout all the performed experiments, the different methods were trained, as mentioned on the datasets of Pitts30k [39] and Mapillary SLS [114]. As for the loss function adopted, the chosen one is from the metric learning approach that is common to all state-of-the-art methods and consists of a **triplet loss**, where a triplet is constituted by the query image (also called anchor), one of its positive matches in the database and one of its negative matches.

The procedure to select the positive/negative samples is called **Mining** and is a critical factor to both performances and efficiency of the system. The metric used for validation is the recall, which evaluates the fraction of images whose correct match was within the first N candidates retrieved by the system. The distance that defines a positive match is 10 m during training(de-facto standard set after the work of [39]), in the scope of the mining procedure, and it is usually 25 m at inference time. This metric for top-n candidates is usually indicated with R@N. To guarantee the stability of the reported results, all experiments are repeated three times and are reported with standard deviation.

As for the training procedure, it was found to be effective the use an early stopping criterion, allowing for no more than 3 epochs without improvements. To overcome the enormous variability in scale present in the chosen training datasets, it has been decided to uniform the duration of an epoch to 5000 images. The optimizer of choice is the renowned Adam stochastic optimizer [123], working with forward passes made up of batches of 4 triplets; specifically a triplet is considered as the association of a query and its positive with 10 distinct negative images.

To publish the output of this extensive analysis, also a website has been developed and will be mantained, publicly available at `https://deep-vg-bench.herokuapp.com/`. [1]. References to the developed codebase can be found on the website, with instruction on how to use it to experiment with every aspect of the pipeline in Fig. 4.1. Furthermore, two additional repositories have been released to download and process all the datasets used, and to run the pre-training on the Landmark Retrieval datasets.

[1] `https://deep-vg-bench.herokuapp.com/`

## 4.2 Experiments

Experiments were conducted to individually evaluate all the components highlighted in Fig. 4.1. Firstly, the analysis starts by comparing results with various backbones, rigorously showing how this affects not only the results, but also inference complexity (in FLOPs), and size of the model (Section 4.2.1). Similarly, different state-of-the-art aggregation methods are analyzed, providing a number of useful statistics (Section 4.2.2). Another often neglected factor is the problem of computing the predictions at test-time. While at train time usually images come from a single domain and have the same resolution, test-time photos can come from a variety of different sources, such as mobile phones, and could be horizontally or vertically oriented. To solve this problem, a common workaround is to pass test queries one by one to the network [39] or crops of the image [124], without investigating how this affects the results. Sec 4.2.6 investigates on this point.

### 4.2.1 Backbones

As discussed in Sect. 2.4.2.2, the feature extraction step of the pipeline is of paramount importance to the successfulness on the task. For this purpose various convolutional backbones are evaluated, as the majority of state-of-the-art works adopt these kind of networks; in Chap. 5 some innovative alternatives will be explored.
The different architectures that were tested are reported in Tab. 4.1, to measure their effect on the retrieval performance. In order to complete the pipeline, they were combined with the 2 most popular aggregation methods in the literature: GeM [44], which is state-of-the-art for the lightweightness of the resulting descriptors, and NetVLAD [39], with higher output dimensionality but usually higher performances. The backbones were the following four, all very popular in the Computer Vision community: (1) VGG-16 [125], ResNet-18 [74], and its heavier versions ResNet-50, and ResNet-101.

Even though this may not seem an elevated number of backbones, this are in practice the most common in the literature (see the works of NetVLAD and gem themselves [39, 44], earlier approaches like [124, 43], and newer contributions as well like [126]), due to the fact that the transferability of their learning from the image recognition task has been proved over and over in a variety of Vision-related task, and therefore they represent basically an Off-The-Shelf commodity for researchers. Nonetheless among them there still

are some tradeof that are hereby discussed.
It is worthwhile specifying that for all the ResNets versions, they have been truncated at the *conv4_x* layer, as it represents the optimal point in the tradeof of network extractive capacity and computational cost; in the following Tab. 4.2 this choice is further clarified. As for the VGG16 instead, it has been used the convolutional backbone in its entirety removing only the pooling before the fully connected classifier. Tab. 4.1 shows the results of this experiments.

Table 4.1: **Backbones:** this table shows how changing the backbone influences computational requirements and recalls. Table from [1].

| Backbone | Aggregation Method | Features Dim | FLOPs | Model Size | Training on Pitts30k | | | | | | Training on MSLS | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | R@1 Pitts30k | R@1 MSLS | R@1 Tokyo 24/7 | R@1 R-SF | R@1 Eynsham | R@1 St Lucia | R@1 Pitts30k | R@1 MSLS | R@1 Tokyo 24/7 | R@1 R-SF | R@1 Eynsham | R@1 St Lucia |
| VGG-16 | GeM | 512 | 188.01 GF | 56.13 Mb | 78.5 | 43.4 | 39.9 | 40.4 | 70.2 | 46.4 | 70.2 | 66.7 | 43.6 | 32.1 | 80.4 | 79.9 |
| ResNet-18 | GeM | 256 | 17.29 GF | 10.63 Mb | 77.8 | 35.3 | 35.3 | 34.2 | 64.3 | 46.2 | 71.6 | 65.3 | 42.8 | 30.5 | 80.3 | 83.2 |
| ResNet-50 | GeM | 1024 | 40.61 GF | 32.71 Mb | 82.0 | 38.0 | 41.5 | 45.4 | 66.3 | **59.0** | **77.4** | 72.0 | **55.4** | 45.7 | **83.9** | 91.2 |
| ResNet-101 | GeM | 1024 | 86.29 GF | 105.36 Mb | **82.4** | **39.6** | **44.0** | **52.5** | **69.0** | 57.6 | 77.2 | **72.5** | 51.0 | **46.9** | 83.6 | **91.6** |
| VGG-16 | NetVLAD | 32768 | 188.09 GF | 56.38 Mb | 83.2 | 50.9 | 61.4 | 64.6 | 74.4 | 50.1 | 79.0 | 74.6 | 61.9 | 57.1 | 84.2 | 86.7 |
| ResNet-18 | NetVLAD | 16384 | 17.27 GF | 10.76 Mb | 86.4 | 47.4 | 63.4 | 61.4 | 76.8 | 57.6 | **81.6** | 75.8 | 62.3 | 55.1 | 87.1 | 92.1 |
| ResNet-50 | NetVLAD | 65536 | 40.51 GF | 33.21 Mb | 86.0 | 50.7 | 69.8 | 67.1 | **77.7** | 60.2 | 80.9 | 76.9 | **62.8** | 51.5 | **87.2** | 93.8 |
| ResNet-101 | NetVLAD | 65536 | 86.06 GF | 105.86 Mb | **86.5** | **51.8** | **72.2** | **67.5** | 74.0 | **63.6** | 80.8 | **77.7** | 59.0 | **56.1** | 86.7 | **95.1** |

Looking at the sheer numbers of Top-1 Recall (R@1) the backbone providing the best performances appears to be the ResNet-101; nevertheless it is important to consider the trade-of with its computational requirements in terms of the yielded dimensionality of descriptors and model FLOPs and compare them with the gap in recall. Taking into account these factors, it is evident how the less efficient choice turns out to be the VGG-16 (which however has been used in many proposed works in the literature), and how Resnets-18 and -50 represent in basically all the cases the best choice overall. Specifically, the extremely lightweight Resnet-18 (r18) seems to be the go-to alternative for the majority of applications; and trading it for a Resnet-50 may be evaluated in some specific applications in which computational and time requirements are not critical.
Another important consideration that surfaces from the analysis of this results is the substantial impact that the characteristics of the training datasets have on the generalization capability of the models. Infact models trained on Pitts30k, especially with GeM, show overall lower results than the one obtainable by using a more comprehensive dataset like MSLS.

This is particularly evident in the cases of St.Lucia and Eynsham which include images in the countryside, and thus in a different domain than the urban-only that the model had seen in Pitts30k during training. In these

cases the broad variability provided by MSLS proves to yield superior generalization capability with improvements in recall from 15% up to 30%. These considerations are an additional support to the motivations of this work, showing that comparing directly between them methods that have been trained in different settings, on different training datasets as it happens in mentioned works like [51] may paint an inaccurate picture of the property that design choices confer to the methods.

Table 4.2: Focus on **ResNets:** Possible advantages obtainable by truncating this backbone *conv4_x* or *conv5_x* for VG. Table from [1].

| Backbone | Aggregation Method | Features Dim | FLOPs | Model Size | Training Dataset | R@1 Pitts30k | R@1 MSLS | R@1 Tokyo 24/7 | R@1 R-SF | R@1 Eynsham | R@1 St Lucia |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ResNet-18 *conv4_x* | GeM | 256 | 17.29 GF | 10.63 MB | Pitts30k | 77.8 ± 0.2 | 35.3 ± 0.5 | 35.3 ± 1.1 | 34.2 ± 1.7 | 64.3 ± 1.2 | 46.2 ± 0.4 |
| ResNet-18 *conv4_x* | NetVLAD | 16384 | 17.27 GF | 10.76 MB | Pitts30k | **86.4 ± 0.3** | **47.4 ± 1.2** | **63.4 ± 1.2** | **61.4 ± 1.5** | **76.8 ± 1.2** | **57.6 ± 3.3** |
| ResNet-18 *conv5_x* | GeM | 512 | 22.33 GF | 42.67 MB | Pitts30k | 77.9 ± 0.3 | 34.4 ± 0.4 | 34.4 ± 0.6 | 36.9 ± 0.3 | 59.1 ± 1.3 | 51.2 ± 1.3 |
| ResNet-18 *conv5_x* | NetVLAD | 32768 | 22.28 GF | 42.92 MB | Pitts30k | 79.6 ± 0.5 | 47.1 ± 1.8 | 48.9 ± 2.5 | 49.1 ± 3.6 | 70.5 ± 1.0 | 54.4 ± 2.7 |
| ResNet-50 *conv4_x* | GeM | 1024 | 40.61 GF | 32.71 MB | Pitts30k | 82.0 ± 0.3 | 38.0 ± 0.1 | 41.5 ± 1.8 | 45.4 ± 2.0 | 66.3 ± 2.5 | 59.0 ± 1.4 |
| ResNet-50 *conv4_x* | NetVLAD | 65536 | 40.51 GF | 33.21 MB | Pitts30k | **86.0 ± 0.1** | **50.7 ± 2.0** | **69.8 ± 0.8** | **67.1 ± 2.3** | **77.7 ± 0.4** | **60.2 ± 1.6** |
| ResNet-50 *conv5_x* | GeM | 2048 | 50.54 GF | 89.88 MB | Pitts30k | 79.8 ± 0.5 | 41.5 ± 0.7 | 48.0 ± 2.5 | 44.3 ± 1.0 | 65.2 ± 1.4 | 57.5 ± 1.5 |
| ResNet-50 *conv5_x* | NetVLAD | 131072 | 50.35 GF | 90.88 MB | Pitts30k | 79.6 ± 0.2 | 46.2 ± 0.5 | 54.7 ± 2.6 | 51.2 ± 2.5 | 69.8 ± 1.0 | 53.0 ± 4.1 |
| ResNet-18 *conv4_x* | GeM | 256 | 17.29 GF | 10.63 MB | MSLS | 71.6 ± 0.1 | 65.3 ± 0.2 | 42.8 ± 1.1 | 30.5 ± 0.8 | 80.3 ± 0.1 | 83.2 ± 0.9 |
| ResNet-18 *conv4_x* | NetVLAD | 16384 | 17.27 GF | 10.76 MB | MSLS | **81.6 ± 0.5** | **75.8 ± 0.1** | **62.3 ± 1.6** | **55.1 ± 0.9** | **87.1 ± 0.2** | **92.1 ± 0.7** |
| ResNet-18 *conv5_x* | GeM | 512 | 22.33 GF | 42.67 MB | MSLS | 73.5 ± 0.5 | 68.4 ± 0.8 | 41.0 ± 0.8 | 38.6 ± 1.8 | 79.4 ± 0.5 | 84.7 ± 0.7 |
| ResNet-18 *conv5_x* | NetVLAD | 32768 | 22.28 GF | 42.92 MB | MSLS | 75.7 ± 0.7 | 75.7 ± 0.6 | 49.9 ± 1.6 | 41.3 ± 0.2 | 84.1 ± 0.4 | 91.3 ± 0.4 |
| ResNet-50 *conv4_x* | GeM | 1024 | 40.61 GF | 32.71 MB | MSLS | 77.4 ± 0.6 | 72.0 ± 0.5 | 55.4 ± 2.5 | 45.7 ± 1.0 | 83.9 ± 0.6 | 91.2 ± 0.7 |
| ResNet-50 *conv4_x* | NetVLAD | 65536 | 40.51 GF | 33.21 MB | MSLS | **80.9 ± 0.0** | **76.9 ± 0.2** | **62.8 ± 0.9** | **51.5 ± 1.2** | **87.2 ± 0.3** | **93.8 ± 0.2** |
| ResNet-50 *conv5_x* | GeM | 2048 | 50.54 GF | 89.88 MB | MSLS | 74.7 ± 0.4 | 70.6 ± 0.6 | 46.3 ± 1.3 | 42.1 ± 0.5 | 82.5 ± 0.5 | 89.8 ± 0.4 |
| ResNet-50 *conv5_x* | NetVLAD | 131072 | 50.35 GF | 90.88 MB | MSLS | 74.7 ± 0.2 | 75.2 ± 0.5 | 52.4 ± 0.8 | 44.0 ± 1.1 | 85.5 ± 0.4 | 91.3 ± 0.7 |

As mentioned in the previous paragraph, Tab. 4.2 shows the motivation behind the choice of using as a default strategy the truncation of the ResNets backbone to the *conv4_x* layer, with respect to the alternative of using all the convolutional layers up to *conv5_x* (refer to the original ResNet paper [74] for a detailed explanation on the layers structure). The table show clearly that truncating at *conv4_x* achieves the best results across the board with NetVLAD, and it is at least comparable to the results with GeM; however there is a clear advantage with respect to the *conv5_x* version, which is not the slightly lower FLOPs required, as in the number of channel which is drastically lower (exactly half), providing a consistent speed-up in retrieval time as will be shown in 4.2.5.

### 4.2.2 Aggregation

This section studies all the most used aggregation methods in the literature that were discussed in the Related Works in Sec. 2.4.2.2.1 and Sec. 2.4.2.2.2. They all work by plugging themselves on top of a feature extractor and provide a single (more or less compact) descriptor to be used for retrieval.

These experiments study the obtained performances using different training datasets, and also relating them to feature dimensionality that each method leverages, to understand the impact of it. To vary the size of the final descriptor a few techniques have been used; for the purpose of reducing the size a common choice in the literature is to perform a PCA reduction, to be learnt on the training set only [39]. Contrarily if one whishes to obtain bigger descriptors a fully connected layer can be added to obtain the output, as it was done in this work to explore how and if this can improve the retrieval performances,.

Among the experimented aggregators there are the following methods: CRN [124] is a rudimental form of attention(published years before the latest self-attention techniques) that performs a contextual re-weighting of the features by building a weighted mask that remodulates the feature maps at the output of the backbone; moreover are present all the principal pooling techniques that were exposed in Sec. 2.4.2.2.2 and that in general are less robust than GeM [44], but were tested anyway since this represents a strong test-suite to validate this affirmation. These mentioned pooling methods are MAC [40], its region-wise applied version R-MAC [43], which are both based on max-pooling; SPoC [41] which instead relies on sum-pooling, and the most recent one which is RRM [127]. All this results are in Tab. 4.3

Moving on to discuss the conclusions that can be drawn from Tab. 4.3, it is worth specifying that in this case the FLOPs or size of the aggregation modules was not specified as in every case it is considerably lower with respect to the feature extractor, and it is therefore negligible. The table is organized in two main sections, with the first half reporting results of methods trained on Pitts30k, and the second half dedicated to MSLS; the dataset as it is clear from the results heavily affects the outcomes.
If from the previous table on backbones, Tab. 4.1 it seemed that to use NetVLAD was absolutely the better choice in any case, to this more accurate analysis it surfaces how each of the 2 methods has its strengths. In particular, it surfaces that when training on a more restricted dataset like Pitts30k, NetVLAD achieves both better performances and generalization capabilities; the latter is also true in general, probably thanks to the higher dimensionality that the method provides. Also, referring to the case of training on Pitts30k, NetVLAD shows to outweigh the pooling-based methods and GeM in particular even when the dimensionality is made equal with the use of PCA for NetVLAD and of FC layers for GeM; a motivation for this

Table 4.3: Comprehensive report of **Aggregation methods**, complemented with yielded descriptor dimensionality and collected by backbone. Table from [1].

| Backbone | Aggregation Method | Features Dim | Training Dataset | R@1 Pitts30k | R@1 MSLS | R@1 Tokyo 24/7 | R@1 R-SF | R@1 Eynsham | R@1 St Lucia |
|---|---|---|---|---|---|---|---|---|---|
| ResNet-18 | SPOC [41] | 256 | Pitts30k | 60.6 ± 0.9 | 16.5 ± 0.5 | 15.2 ± 1.1 | 10.4 ± 0.3 | 41.0 ± 2.0 | 29.0 ± 1.5 |
| ResNet-18 | MAC [40] | 256 | Pitts30k | 57.3 ± 0.5 | 25.6 ± 0.4 | 15.2 ± 1.3 | 15.5 ± 0.3 | 49.6 ± 0.7 | 26.6 ± 1.0 |
| ResNet-18 | RMAC [43] | 256 | Pitts30k | 63.2 ± 0.4 | 28.7 ± 0.6 | 22.7 ± 2.3 | 30.5 ± 1.4 | 64.0 ± 0.7 | 42.8 ± 1.3 |
| ResNet-18 | RRM [127] | 256 | Pitts30k | 68.2 ± 0.5 | 21.4 ± 0.8 | 25.4 ± 1.4 | 21.7 ± 1.8 | 51.9 ± 0.8 | 33.7 ± 0.3 |
| ResNet-18 | GeM [44] | 256 | Pitts30k | 77.8 ± 0.2 | 35.3 ± 0.5 | 35.3 ± 1.1 | 34.2 ± 1.7 | 64.3 ± 1.2 | 46.2 ± 0.4 |
| ResNet-18 | GeM + FC 256 | 256 | Pitts30k | 72.4 ± 0.7 | 26.4 ± 0.5 | 27.5 ± 1.2 | 29.0 ± 1.2 | 59.3 ± 1.0 | 39.1 ± 0.8 |
| ResNet-18 | NetVLAD + PCA 256 | 256 | Pitts30k | 80.7 ± 0.7 | 38.3 ± 1.2 | 41.7 ± 0.8 | 35.9 ± 1.8 | 68.9 ± 1.1 | 45.4 ± 2.2 |
| ResNet-18 | CRN + PCA 256 | 256 | Pitts30k | **82.0 ± 0.7** | **43.6 ± 0.7** | **47.7 ± 0.9** | **45.1 ± 0.3** | **71.3 ± 0.8** | **51.3 ± 3.4** |
| ResNet-18 | GeM + FC 2048 | 2048 | Pitts30k | 75.0 ± 0.4 | 29.9 ± 0.6 | 34.5 ± 0.4 | 36.1 ± 0.2 | 63.7 ± 0.3 | 45.1 ± 2.1 |
| ResNet-18 | NetVLAD + PCA 2048 | 2048 | Pitts30k | 85.0 ± 0.4 | 45.0 ± 1.5 | 56.6 ± 0.7 | 53.2 ± 2.4 | 75.4 ± 1.1 | 54.6 ± 3.0 |
| ResNet-18 | CRN + PCA 2048 | 2048 | Pitts30k | **85.7 ± 0.3** | **50.6 ± 0.6** | **61.0 ± 1.6** | **62.8 ± 1.2** | **77.4 ± 0.5** | **61.1 ± 2.7** |
| ResNet-18 | NetVLAD [39] | 16384 | Pitts30k | 86.4 ± 0.3 | 47.4 ± 1.2 | 63.4 ± 1.2 | 61.4 ± 1.5 | 76.8 ± 1.2 | 57.6 ± 3.3 |
| ResNet-18 | CRN [124] | 16384 | Pitts30k | **86.8 ± 0.1** | **53.2 ± 0.7** | **68.8 ± 1.0** | **69.0 ± 0.6** | **79.1 ± 0.3** | **64.8 ± 3.2** |
| ResNet-50 | SPOC [41] | 1024 | Pitts30k | 60.9 ± 0.5 | 19.2 ± 0.4 | 14.0 ± 0.5 | 9.0 ± 0.7 | 40.5 ± 2.3 | 27.1 ± 1.5 |
| ResNet-50 | MAC [40] | 1024 | Pitts30k | 77.6 ± 0.2 | 36.2 ± 0.7 | 36.2 ± 1.4 | 34.8 ± 0.7 | 72.9 ± 0.3 | 51.3 ± 2.4 |
| ResNet-50 | RMAC [43] | 1024 | Pitts30k | 74.9 ± 1.0 | 34.8 ± 0.8 | 41.8 ± 0.6 | 46.4 ± 1.0 | 73.1 ± 0.7 | **68.7 ± 0.5** |
| ResNet-50 | RRM [127] | 1024 | Pitts30k | 72.8 ± 0.2 | 27.9 ± 0.6 | 28.3 ± 0.8 | 28.6 ± 1.0 | 65.9 ± 0.9 | 45.1 ± 1.7 |
| ResNet-50 | GeM [44] | 1024 | Pitts30k | 82.0 ± 0.3 | 38.0 ± 0.1 | 41.5 ± 1.8 | 45.4 ± 2.0 | 66.3 ± 2.5 | 59.0 ± 1.4 |
| ResNet-50 | NetVLAD + PCA 1024 | 1024 | Pitts30k | 83.9 ± 0.7 | 46.5 ± 2.0 | 59.4 ± 1.2 | 53.2 ± 3.8 | 72.5 ± 0.3 | 57.7 ± 2.0 |
| ResNet-50 | CRN + PCA 1024 | 1024 | Pitts30k | **84.1 ± 0.4** | **49.9 ± 0.8** | **64.6 ± 1.2** | **58.8 ± 0.1** | **74.3 ± 0.2** | 63.4 ± 0.4 |
| ResNet-50 | GeM + FC 2048 | 2048 | Pitts30k | 80.1 ± 0.2 | 33.7 ± 0.3 | 43.6 ± 1.6 | 48.2 ± 1.2 | 70.0 ± 0.3 | 56.0 ± 1.7 |
| ResNet-50 | NetVLAD + PCA 2048 | 2048 | Pitts30k | 84.4 ± 0.4 | 47.9 ± 2.0 | 62.6 ± 1.7 | 56.0 ± 2.9 | 74.1 ± 0.4 | 58.9 ± 1.6 |
| ResNet-50 | CRN + PCA 2048 | 2048 | Pitts30k | **84.7 ± 0.3** | **51.2 ± 0.8** | **67.1 ± 0.7** | **62.3 ± 0.3** | **75.8 ± 0.2** | **65.0 ± 0.1** |
| ResNet-50 | NetVLAD [39] | 65536 | Pitts30k | **86.0 ± 0.1** | 50.7 ± 2.0 | 69.8 ± 0.8 | 67.1 ± 2.3 | 77.7 ± 0.4 | 60.2 ± 1.6 |
| ResNet-50 | CRN [124] | 65536 | Pitts30k | 85.8 ± 0.2 | **54.0 ± 0.8** | **73.1 ± 0.3** | **70.9 ± 0.2** | **79.7 ± 0.1** | **65.9 ± 0.4** |
| ResNet-18 | SPOC [41] | 256 | MSLS | 44.2 ± 1.0 | 39.5 ± 0.5 | 20.3 ± 1.3 | 9.5 ± 0.9 | 62.3 ± 0.6 | 58.8 ± 0.8 |
| ResNet-18 | MAC [40] | 256 | MSLS | 60.4 ± 1.1 | 54.7 ± 1.8 | 20.4 ± 2.6 | 18.9 ± 2.0 | 76.3 ± 1.2 | 69.2 ± 1.2 |
| ResNet-18 | RMAC [43] | 256 | MSLS | 58.1 ± 1.2 | 48.9 ± 2.0 | 29.1 ± 2.0 | 34.3 ± 1.4 | 73.3 ± 1.1 | 63.7 ± 2.7 |
| ResNet-18 | RRM [127] | 256 | MSLS | 60.8 ± 1.5 | 54.9 ± 2.6 | **44.4 ± 2.1** | 30.9 ± 2.8 | 75.7 ± 1.5 | 68.7 ± 1.4 |
| ResNet-18 | GeM [44] | 256 | MSLS | 71.6 ± 0.1 | 65.3 ± 0.2 | 42.8 ± 1.1 | 30.5 ± 0.8 | 80.3 ± 0.1 | 83.2 ± 0.9 |
| ResNet-18 | GeM + FC 256 | 256 | MSLS | 68.6 ± 1.1 | 59.6 ± 2.6 | 41.9 ± 2.7 | 31.3 ± 0.5 | 78.5 ± 2.0 | 76.1 ± 3.4 |
| ResNet-18 | NetVLAD + PCA 256 | 256 | MSLS | 74.2 ± 0.2 | 70.6 ± 0.3 | 43.6 ± 0.5 | 34.7 ± 1.7 | 84.4 ± 0.4 | 89.8 ± 0.5 |
| ResNet-18 | CRN + PCA 256 | 256 | MSLS | **74.5 ± 0.8** | **72.1 ± 0.1** | 44.1 ± 1.4 | **35.1 ± 2.4** | **84.8 ± 0.3** | **91.6 ± 0.4** |
| ResNet-18 | GeM + FC 2048 | 2048 | MSLS | 71.9 ± 1.0 | 64.0 ± 1.2 | 51.8 ± 0.9 | 37.6 ± 1.3 | 81.1 ± 0.9 | 79.2 ± 0.9 |
| ResNet-18 | NetVLAD + PCA 2048 | 2048 | MSLS | **80.4 ± 0.4** | 74.6 ± 0.2 | 55.6 ± 1.2 | 47.4 ± 1.1 | 86.4 ± 0.3 | 92.2 ± 0.3 |
| ResNet-18 | CRN + PCA 2048 | 2048 | MSLS | 80.1 ± 0.8 | **75.8 ± 0.1** | **57.2 ± 2.3** | **47.8 ± 2.7** | **86.8 ± 0.3** | **93.2 ± 0.4** |
| ResNet-18 | NetVLAD [39] | 16384 | MSLS | **81.6 ± 0.5** | 75.8 ± 0.1 | 62.3 ± 1.6 | **55.1 ± 0.9** | 87.1 ± 0.2 | 92.1 ± 0.7 |
| ResNet-18 | CRN [124] | 16384 | MSLS | 81.3 ± 0.7 | **76.8 ± 0.0** | **63.8 ± 1.4** | 53.9 ± 2.0 | **87.5 ± 0.2** | **93.7 ± 0.1** |
| ResNet-50 | SPOC [41] | 1024 | MSLS | 47.5 ± 1.3 | 47.9 ± 1.5 | 20.6 ± 1.6 | 8.9 ± 1.0 | 68.3 ± 0.5 | 68.6 ± 1.4 |
| ResNet-50 | MAC [40] | 1024 | MSLS | 76.0 ± 0.2 | 67.4 ± 1.6 | 45.3 ± 1.0 | 44.4 ± 2.6 | 84.6 ± 0.4 | 86.0 ± 0.7 |
| ResNet-50 | RMAC [43] | 1024 | MSLS | 70.1 ± 0.8 | 62.0 ± 0.5 | 52.1 ± 2.3 | **54.3 ± 1.8** | 80.6 ± 0.5 | 85.9 ± 1.0 |
| ResNet-50 | RRM [127] | 1024 | MSLS | 69.3 ± 1.0 | 67.4 ± 0.4 | 53.7 ± 0.8 | 43.7 ± 1.0 | 84.3 ± 0.5 | 84.8 ± 1.1 |
| ResNet-50 | GeM [44] | 1024 | MSLS | **77.4 ± 0.6** | 72.0 ± 0.5 | **55.4 ± 2.5** | 45.7 ± 1.0 | 83.9 ± 0.6 | 91.2 ± 0.7 |
| ResNet-50 | NetVLAD + PCA 1024 | 1024 | MSLS | **77.4 ± 0.2** | 74.8 ± 0.3 | 51.3 ± 1.3 | 39.0 ± 1.3 | 85.2 ± 0.3 | 92.9 ± 0.3 |
| ResNet-50 | CRN + PCA 1024 | 1024 | MSLS | 77.3 ± 0.3 | **75.6 ± 0.0** | 51.8 ± 1.1 | 38.8 ± 1.0 | **85.7 ± 0.3** | **94.1 ± 0.2** |
| ResNet-50 | GeM + FC 2048 | 2048 | MSLS | **79.2 ± 0.6** | 73.5 ± 0.8 | 64.0 ± 3.9 | **55.1 ± 2.4** | 86.1 ± 0.7 | 90.3 ± 1.0 |
| ResNet-50 | NetVLAD + PCA 2048 | 2048 | MSLS | 78.5 ± 0.2 | 75.4 ± 0.2 | 52.8 ± 0.4 | 42.6 ± 1.3 | 85.8 ± 0.3 | 93.4 ± 0.4 |
| ResNet-50 | CRN + PCA 2048 | 2048 | MSLS | 78.3 ± 0.3 | **76.3 ± 0.1** | **54.3 ± 0.7** | 42.8 ± 1.6 | **86.2 ± 0.4** | **94.4 ± 0.2** |
| ResNet-50 | NetVLAD [39] | 65536 | MSLS | **80.9 ± 0.0** | 76.9 ± 0.2 | 62.8 ± 0.9 | 51.5 ± 1.2 | 87.2 ± 0.3 | 93.8 ± 0.2 |
| ResNet-50 | CRN [124] | 65536 | MSLS | 80.8 ± 0.2 | **77.8 ± 0.1** | **63.6 ± 0.5** | **53.4 ± 1.4** | **87.5 ± 0.4** | **94.8 ± 0.3** |

fact can be found by reckoning that FC layers bring with themselves a considerably large number of parameters that is likely to overfit on the small

dataset provided by Pitts30k.
However, it is not to be underestimated the potential of the GeM pooling, infact in the situation in which its own authors suggest to use it, that is on comprehensively large scale datasets, it does indeed take over with respect to NetVLAD. This is an important consideration since it needs not to be forgotten that even with the addition of a FC layer, GeM is still able to offer a lower output dimensionality than its competitor, which is an important factor in retrieval time; furthermore the idea of NetVLAD of using residual distances from clusters of visual words, however powerful and able to yield quite robust descriptors, has the defect of being highly sensitive to the initialization of said clustering, making it overall harder to train successfully. GeM instead adds a single parameter to the backbone and its pooling is really lightweight and makes the training procedure shorter and smoother as well.

Finally, it is worth mentioning the remarkable results that the attention mechanism proposed in CRN offers, demonstrating its usefulness in making the model more robust across the board, independently from the backbone of choice and even the size of the feature space. It is especially interesting how it often performs slightly worse than NetVLAD on the training dataset, offering instead better generalization on different testing datasets, proving to yield superior generalization capabilities when subject to various domain shifts. The price at which this property come is an additional training stage. This discussion renders understandable why NetVLAD and GeM represent the most commonly adopted state-of-the-art methods, as each one of them represents a good trade-of of retrieval performance vs efficiency, each one slightly more inclined towards one of the 2; the considerations highlighted should also make it clearer in which situations it is better to pivot towards one or the other.

#### 4.2.2.1 Memory footprint of the aggregation methods

In a real-world system for VG, an important factor to keep an eye on is represented by the dimensionality of the features on which the system relies to obtain images representations. As it was already underlined, when such a system is deployed, all the features of the database should be precomputed offline and be stored in RAM and efficiently indexed in order to be ready to perform similarity searches when new queries arrive. It follows that for feasibility, a method that employs very large descriptor can only be applied

with databases of limited size.

Table 4.4: Analysis of **Memory footprint** related to feature dimensionality (determined by backbone + aggregation method) and database size. The reported number represents an estimate of the minimum RAM required to run an exact kNN similarity search. Table from [1].

| Features Dim. | Pitts30k | MSLS | Tokyo 24/7 | R-SF | Eynsham | St. Lucia |
|---|---|---|---|---|---|---|
| 256 | 0.01 GB | 0.04 GB | 0.07 GB | 1.00 GB | 0.02 GB | 0.001 GB |
| 1024 | 0.04 GB | 0.15 GB | 0.29 GB | 4.01 GB | 0.09 GB | 0.006 GB |
| 2048 | 0.08 GB | 0.30 GB | 0.57 GB | 8.01 GB | 0.18 GB | 0.011 GB |
| 16384 | 0.61 GB | 2.38 GB | 4.58 GB | 64.09 GB | 1.46 GB | 0.092 GB |
| 65536 | 2.44 GB | 9.52 GB | 18.31 GB | 256.35 GB | 5.86 GB | 0.366 GB |

To give some concrete numbers on the requirements that can follow the design of a VG system, Tab. 4.4 shows the minimum amount of memory required to store the feature embedding of the whole database. It is clear from the table the difficulty in scalability posed by big datasets such as the ones present in R-SF, and how the required RAM can quickly scale to hundreds of GigaBytes if for example the VG system employs a ResNet101 as backbone and NetVLAD as aggregator, resulting in an embedding dimensionality of 65536; contrarily it emerges the advantage in using some more lightweight pooling techniques like GeM that basically pose no bottleneck (with realistic database size) to scalability.

### 4.2.3 Mining

As it is clear from the analysis carried out up until now, the state-of-the-art methods all make use of metric learning approaches, specifically of the triplet loss. As highlighted in Sec. 4.1.2, this requires a dedicated procedure called *Mining* in order to find in the database a set of suitable positive and negative images to use to form the triplets.

The process is particularly delicate as the quality of the triplet can significantly affect the ability of the network to learn, rendering the entire process useless if images were to not be selected correctly. In particular, as regards the positives, it is important to find images that are indeed very similar to the query, otherwise the process would be asking to the network to bring closer in the feature space images that are not actually a correct match. Therefore the criteria is to take, among the images whose labels tells that they are within a given threshold (traditionally $10m$ during training [39]), the closest one in the embedding space to the query at hand. The reason behind this is

that two images, even if their GPS coordinates tell that they are very close (i.e. less than $10m$), this says nothing about the viewpoint which can be rather different and thus lead to different appearance, which would confuse the network; instead by choosing the closest one in the feature space the process make sure to only pick positives that depict the same scene.

As for the negatives, the challenges are numerous. First of all, the notion of positives being within a threshold already limits the search space, moreover the set of positives for each query can be computed beforehand only once and retrieving them becomes very efficient during training. Instead, any image outside of this threshold in the whole database is a negative candidate, and therefore negative mining is an important topic in the literature. Ideally, one would want to select the hardest possible negatives, i.e. images that, while outside the threshold for positive matchings, should however be as similar as possible to the query in order to give to the network examples against which it is 'hard' to learn how to discern among them. To give an example, if for a query in an urban environment, some images depicting countryside scenes were to be picked as negatives, their feature representation would already be so different that the network could not possibly learn anything useful from triplets of such kind.

The first method that became popular was proposed by the authors of NetVLAD in [39], which if will be referred to from now on as *full database mining*. The idea is to compute the features of the entire gallery available (this set of pre-computed features is called *cache*), and subsequently to retrieve among them the closest positive and hardest negative according to the criteria explained above. The point is that as training goes on, the network learns how to improve the embeddings and its weights are updated accordingly, therefore the cache needs to updated every so on in order for the mining procedure to be effective.

Originally the authors propose to refresh such cache every 1000 forward passes of queries, which is reasonable and in practice gives good result, and it is the best method on the dataset that the same authors in [39] propose, that is Pitts30k. However, as it surfaces in the experiments presented in this section, as soon as one tries to scale the training of a VG system to a substantially larger-scale dataset (in particular what matters in this sense is the size of the gallery) it soon becomes unfeasible to compute a feature representation for the whole dataset every so on. In fact such a procedure scales in a directly proportional fashion to the size of the dataset, both in term of computational time and of memory footprint to store the cache; also because as the dataset grows bigger, the percentage of images after which the

cache needs to be recomputed ($\frac{1000}{db\_size}$) gets smaller, rendering the training unfeasible overall.

The first one to address this problem were the team of Mapillary [114], as when they proposed their dataset this issue became evident and a more suitable criteria to mine the triplets was needed. This mining method will be called in the following tables *partial database mining*, and it is as simple as it is effective. Instead of computing the embedded representation of the whole database, the choice is to randomly sample a thousand images from the available gallery, and to only compute their representations among which to look for the hardest negatives.

However surprising it may seem, this approach in practice shows results comparable to the full version, while in contrast drastically reducing the spatial and time requirements. (to get a sense of the memory requirements of storing the features for the whole database refer to Sec. 4.2.2.1) Tab. 4.5 contains all detailed results using various backbones and the aggregation methods that were highlighted as more representative choices in Sec. 4.2.2, NetVLAD and GeM.

Table 4.5: Results showcase of different **Mining strategies.**. Table from [1].

| Backbone | Aggregation Method | Mining Method | Training Dataset | R@1 Pitts30k | R@1 MSLS | R@1 Tokyo 24/7 | R@1 R-SF | R@1 Eynsham | R@1 St Lucia |
|---|---|---|---|---|---|---|---|---|---|
| ResNet-18 | GeM | Random | Pitts30k | 73.7 ± 0.7 | 30.5 ± 0.5 | 31.3 ± 0.8 | 24.0 ± 1.2 | 58.2 ± 1.4 | 41.0 ± 1.2 |
| ResNet-18 | GeM | Full database mining | Pitts30k | **77.8 ± 0.2** | **35.3 ± 0.5** | **35.3 ± 1.1** | **34.2 ± 1.7** | **64.3 ± 1.2** | **46.2 ± 0.4** |
| ResNet-18 | GeM | Partial database mining | Pitts30k | 76.5 ± 0.3 | 34.2 ± 1.3 | 33.9 ± 1.4 | 32.9 ± 0.7 | 64.0 ± 2.4 | 45.6 ± 0.9 |
| ResNet-18 | NetVLAD | Random | Pitts30k | 83.9 ± 0.5 | 43.6 ± 0.5 | 55.1 ± 1.3 | 53.8 ± 1.1 | 76.3 ± 0.6 | 53.5 ± 1.4 |
| ResNet-18 | NetVLAD | Full database mining | Pitts30k | **86.4 ± 0.3** | **47.4 ± 1.2** | **63.4 ± 1.2** | 61.4 ± 1.5 | **76.8 ± 1.2** | **57.6 ± 3.3** |
| ResNet-18 | NetVLAD | Partial database mining | Pitts30k | 86.2 ± 0.3 | 47.3 ± 0.4 | 61.2 ± 0.5 | **62.9 ± 0.3** | 76.6 ± 0.5 | 57.1 ± 1.6 |
| ResNet-50 | GeM | Random | Pitts30k | 77.9 ± 1.0 | 34.3 ± 1.3 | 40.1 ± 1.0 | 35.5 ± 3.0 | 63.8 ± 0.9 | 52.3 ± 1.4 |
| ResNet-50 | GeM | Full database mining | Pitts30k | 82.0 ± 0.3 | 38.0 ± 0.1 | 41.5 ± 1.8 | 45.4 ± 2.0 | 66.3 ± 2.5 | 59.0 ± 1.4 |
| ResNet-50 | GeM | Partial database mining | Pitts30k | **82.3 ± 0.0** | **39.0 ± 0.4** | **43.5 ± 0.2** | **45.5 ± 1.7** | **67.7 ± 1.4** | **61.0 ± 2.0** |
| ResNet-50 | NetVLAD | Random | Pitts30k | 83.4 ± 0.6 | 45.0 ± 0.3 | 61.9 ± 2.1 | 55.8 ± 1.5 | 75.0 ± 1.8 | 52.6 ± 1.2 |
| ResNet-50 | NetVLAD | Full database mining | Pitts30k | **86.0 ± 0.1** | **50.7 ± 2.0** | **69.8 ± 0.8** | **67.1 ± 2.3** | **77.7 ± 0.4** | **60.2 ± 1.6** |
| ResNet-50 | NetVLAD | Partial database mining | Pitts30k | 85.5 ± 0.3 | 48.6 ± 3.1 | 66.7 ± 4.1 | 65.0 ± 4.3 | 77.6 ± 1.3 | 59.0 ± 4.1 |
| ResNet-18 | GeM | Random | MSLS | 62.2 ± 0.3 | 50.6 ± 0.6 | 28.8 ± 0.8 | 17.1 ± 1.0 | 70.2 ± 0.6 | 71.4 ± 1.0 |
| ResNet-18 | GeM | Full database mining | MSLS | 70.1 ± 1.1 | 61.8 ± 0.5 | **42.8 ± 1.4** | **31.3 ± 1.2** | 79.3 ± 0.2 | 81.0 ± 0.9 |
| ResNet-18 | GeM | Partial database mining | MSLS | **71.6 ± 0.1** | **65.3 ± 0.2** | **42.8 ± 1.1** | 30.5 ± 0.8 | **80.3 ± 0.1** | **83.2 ± 0.9** |
| ResNet-18 | NetVLAD | Random | MSLS | 73.3 ± 0.7 | 61.5 ± 1.4 | 45.0 ± 1.5 | 34.8 ± 0.2 | 84.9 ± 0.3 | 79.7 ± 1.7 |
| ResNet-18 | NetVLAD | Full database mining | MSLS | - | - | - | - | - | - |
| ResNet-18 | NetVLAD | Partial database mining | MSLS | **81.6 ± 0.5** | **75.8 ± 0.1** | **62.3 ± 1.6** | **55.1 ± 0.9** | **87.1 ± 0.2** | **92.1 ± 0.7** |
| ResNet-50 | GeM | Random | MSLS | 69.5 ± 1.2 | 57.4 ± 1.1 | 43.5 ± 3.3 | 31.1 ± 0.9 | 78.8 ± 0.5 | 78.3 ± 1.2 |
| ResNet-50 | GeM | Full database mining | MSLS | 77.3 ± 0.3 | 69.7 ± 0.2 | 52.4 ± 1.7 | 45.3 ± 0.2 | **84.2 ± 0.0** | 91.0 ± 0.2 |
| ResNet-50 | GeM | Partial database mining | MSLS | **77.4 ± 0.6** | **72.0 ± 0.5** | **55.4 ± 2.5** | **45.7 ± 1.0** | 83.9 ± 0.6 | **91.2 ± 0.7** |
| ResNet-50 | NetVLAD | Random | MSLS | 74.9 ± 0.4 | 63.6 ± 1.3 | 41.9 ± 1.6 | 34.6 ± 2.3 | 85.5 ± 0.2 | 80.9 ± 0.4 |
| ResNet-50 | NetVLAD | Full database mining | MSLS | - | - | - | - | - | - |
| ResNet-50 | NetVLAD | Partial database mining | MSLS | **80.9 ± 0.0** | **76.9 ± 0.2** | **62.8 ± 0.9** | **51.5 ± 1.2** | **87.2 ± 0.3** | **93.8 ± 0.2** |

Tab. 4.5 reports the outcomes separated by training dataset, including also among the mining stategies, the 'Random' one. Random mining consists

in randomly sampling negatives without performing any mining to choose among them, which is not very clever and it is not used in practice, however it is interesting as a baseline result against which to compare the other 2.
It is indeed an unexpected outcome seeing how on the smaller Pitts30k the *Random* strategy only shows minimal drops in recall, ranging from 3% to 5%. This figure needs to be related to the context of the datasets, recalling that Pitts30k contains densely sampled images from the streets of the homonym city, where therefore the percentage of negative images that would look very similar to the query is substantial. This intuition is confirmed from the second half of the table in which it is visible how the drop increases up to 12% when the dataset is MSLS which presents a much richer variability in terms of domains.
Another notable fact to report is that, as it is noticeable in the table, the row corresponding to full mining with NetVLAD as aggregation method is left empty, and the reason is that training was so expensive that it was not even close to convergence even after five days of execution of multi-gpu training, witnessing the unbearable cost that full mining brings when applied on such large-scale datasets. As a counterexample, the rest of the experiments were able to terminate in under a day of execution time.

This section mainly confirms that hard negative mining is important and that the approximate strategy proposed by the authors of MSLS has much lower cost while performing similarly well and even sometimes better.

### 4.2.4 Pre-training

As it was stated since Sec. 2.4.2.2, one of the characteristics that made popular the use of Deep Learning and CNNs in particular, for the VG task (as for many others in the literature) was, beyond their great representational capacity, their good transferability properties. Transferability means that the network can be pre-trained on a general purpose dataset, and the network will retain the learnt ability to extract meaningful features and can therefore be used on a different task. In practice, to obtain the best possible result, the pre-trained network is trained again for the specific task at hand, usually with lower learning rates and such technique is called *fine-tuning*.
For these reasons in this section the analysis is dedicated to the role of the chosen dataset for the pre-training of the network. As explained, this is traditionally ImageNet, due to its great variability of images that confer to the networks trained on the ability to extract robust features. Specifically, using

Table 4.6: Impact of the **training dataset**, including Landmark Retrieval datasets. Table from [1].

| Source | Loss | Training Dataset | Backbone | Aggregation Method | R1 Pitts30k | R1 MSLS | R1 Tokyo 24/7 | R1 R-SF | R1 Eynsham | R1 St Lucis |
|---|---|---|---|---|---|---|---|---|---|---|
| [44] | Triplet | GLDv1 | ResNet-50 | GeM + FC 2048 | **84.1** | 69.5 | **77.8** | **76.4** | 61.8 | 77.3 |
| [44] | Triplet | Sfm120k | ResNet-50 | GeM + FC 2048 | 83.4 | 64.5 | 75.2 | 75.6 | 68.8 | 73.9 |
| - | Triplet | Pitts30k | ResNet-50 | GeM + FC 2048 | 80.1 | 33.7 | 43.6 | 48.2 | 70.0 | 56.0 |
| - | Triplet | MSLS | ResNet-50 | GeM + FC 2048 | 79.2 | **73.5** | 64.0 | 55.1 | **86.1** | **90.3** |
| [44] | Triplet | GLDv1 | ResNet-101 | GeM + FC 2048 | **85.1** | 72.4 | **77.8** | **79.8** | 61.6 | 83.4 |
| [44] | Triplet | Sfm120k | ResNet-101 | GeM + FC 2048 | 83.9 | 64.7 | 77.5 | 78.3 | 62.8 | 76.3 |
| - | Triplet | Pitts30k | ResNet-101 | GeM + FC 2048 | 82.4 | 40.0 | 47.2 | 57.5 | 75.9 | 61.7 |
| - | Triplet | MSLS | ResNet-101 | GeM + FC 2048 | 79.1 | **75.3** | 61.9 | 54.9 | **86.0** | **92.5** |

the open source codebase that has been released for this work, it is possible to automatically download state-of-the-art pretrained models that have been published by the authors of [44] at [2]. The ratio of this comparison is to study whether or not dataset tailored for Landmark Retrieval, which as explained in Sec. 2.2.2 is related to VG yet with substantial differences, can be used with satisfactory results to train models for the purpose of VG. The comparison is done with the traditional training datasets used throughout this work, which are Pitts30k and MSLS. Note that this experiments were performed using a lightweight GeM aggregator expanded with an FC layer, for reasons of fairness of comparison with respect to the models pre trained on the other datasets released by [44] that made use of such aggregation method.

The released software also enables experimenting both with these pre-trained models, and also to carry out further experiments for example by using these models as a starting point for a further fine-tuning stage.

Tab. 4.6 contains the outcomes of the run experiments. It certainly stands out to a first glance the fact that using the GLDv1 [6] dataset can represent a nice alternative to strictly VG-oriented datasets as it it able to provide interesting results. As regards Sfm120k, even though it tends to stay consistently below its counterpart GLDv1, the gap is only of a few percentual points, it still offers decent performances. The characteristic that these 2 Landmark Retrieval dataset share is their large scale, with a broad variety of viewpoints and orientation, which therefore bestows on models trained on them some nice robustness and generalization capabilities.
Another consideration that can be pointed out and that confirms the findings

[2] `https://github.com/filipradenovic/cnnimageretrieval-pytorch`

highlighted in the analysis of Tab. 4.3 is how the dataset that is less likely to yield robust models is the small Pitts30k, as the table shows poor generalization performances when tested on the others datasets. As discussed also in Sec. 4.2.2, this is due both to the limited variability of the images present in Pitts30k, that does not allow models to build a comprehensive embedding representation for scenes other than the urban ones of Pittsburgh, and is also partly due to the chosen aggregator which includes a FC layer which introduces an elevated number of parameters that is more likely to overfit on small datasets.
Another fact that surfaces from these experiments is that for Tokyo 24/7, and as well for San Francisco it seems to be a better choice relying on the Landmark Retrieval datasets that outperform both Pitts30k and MSLS. These outcomes can be understood in relationship with the characteristics of these two (R-SF and Tokyo) with respect to MSLS; in fact they are made up of images with views at 360° and queries are smartphone-captured, therefore there is a larger domain gap compared with the front-view pictures that MSLS contains and that were taken from a car roof-top camera. Other than this specific case, overall the wide variability of seasonality, lighting and scenario of MSLS shows its value on the other datasets proving good generalization overall.

The bottom line that can be highlighted is that the training dataset should be a choice to be carefully evaluated, not limiting the options to strictly VG-related datasets, and that the domain characteristics of the images that the application is going to encounter should be taken into account and the chosen dataset should match these characteristics as much as possible (see the example of Tokyo and R-SF above). Another suggestion can be to tune the number of parameters in relationship to the dataset size as well; more is not always better.

### 4.2.5 Inference Time

This section is devoted to the analysis of the runtime required by a deployed VG-based application, when fed with a new query to localize. Such a computational time is known as *Inference time*, $t_i$. Due to the structure of the pipeline, such variable can be analyzed in light of its two main components:

- **Embedding**: $t_e$, time required by the chosen feature extractor of the system to compute a descriptor for the image at hand. It is influenced

Table 4.7: **Time to extract feature representation** of a single image sized 480×640, reported in milliseconds. To fairly estimate this figure the results is computed averaging the time required to compute the feature representation for 10 thousand images. Table from [1].

| Aggregation | VGG16 | ResNet-18 *conv4_x* | ResNet-18 *conv5_x* | ResNet-50 *conv4_x* | ResNet-50 *conv5_x* | ResNet-101 *conv4_x* | ResNet-101 *conv5_x* |
|---|---|---|---|---|---|---|---|
| GeM | 12.3 | 4.1 | 3.9 | 6.7 | 7.3 | 9.6 | 10.2 |
| NetVLAD | 13.0 | 4.4 | 4.4 | 8.5 | 8.3 | 11.5 | 11.3 |
| NetVLAD - PCA 256 | 16.6 | 6.0 | 7.7 | 15.3 | 22.4 | 18.3 | 24.9 |
| NetVLAD - PCA 2048 | 40.6 | 17.5 | 30.8 | 61.8 | 117.2 | 63.6 | 115.1 |

mainly by the choice of the backbone (in case of a CNN, by its number of layers and of parameters, FLOPs) and the resolution of the query;

- **Retrieval**: $t_r$, time required by the chosen retrieval algorithm to find a suitable set of candidates for the best match of the query. The retrieval algorithm is traditionally in all the state-of-the-art literature, a kNN search. This computational time is heavily affected by the dimensionality of the embeddings extracted during the previous step, and as well by the size of the gallery in which the kNN has to search for matches.

$$t_i = t_e + t_r$$

So Inference time can be studied based on these 2 components. The rest of section is devoted to this analysis, starting from Embedding (or extraction) time.

Tab. 4.7 reports the results obtained for each aggregation method varying the backbone, in terms of time needed to build the final image descriptor. As already discussed above, backbone and aggregation are the only factors that play a role, together with image size. Specific hardware of course is of influence as well, but the proportionality among the results should hold true. Looking at the result it can be seen how using PCA, which can drastically reduce the retrieval time (as the rest of the section will show), can actually cause the biggest difference when it comes to embedding time. This is mainly caused by the fact that the standard PCA implementation used by researchers (provided by the Scikit-Learn library) does not make use of the GPU and relies on processor, causing the gap that shows up in the table. It also emerges that the implementation of NetVLAD alone is efficient enought to not cause any noticeable efficiency drops compared to GeM.

Overall, considering the scale of the table which is in milliseconds, it can be concluded that for a system that needs to work in reasonably short timespan, none of this methods should create noticeable delays, which is not true

for the retrieval time that will be analyzed next. It is however worth noting how using a ResNet-101 with respect to a ResNet-18 does cause a three-fold increase in the time required for extraction, and as it was extensively analyzed in Sec. 4.2.1, it only provides a minimal improvements in the results; therefore once again it is advisable to stick with smaller versions of the ResNet.

Moving on to discuss retrieval time, as already specified the analysis focuses on a kNN search. K-Nearest Neighbor (kNN) is the predominant method in the Deep Visual Geo-localization and Image Retrieval literature for matching query and database descriptors [39, 41, 128, 129, 72, 130, 43, 127, 126, 3, 2, 6, 44, 36, 40, 131, 116, 114, 51].
Since as stated the main factors affecting the retrieval time are numerosity of database images, and dimension of the embedding, Fig. 4.2 reports the elapsed retrieval time related to these variables. The figures in the plot have been obtained averaging the time required to retrieve the top 10 candidates for a thousand queries. The reason for this is that, due to optimizations that are obtained by parallelizing the search, if one were to compute the matches for only a 100 query it would get a total computational time that is lower than the one required for a thousand. Therefore to fully exploit the features of the Faiss [45] library it is better to batch together queries in order to speed up the process.

The plot in Fig. 4.2 shows how the matching time depends linearly on the number of images in the database and the dimensionality of the features. As the dot line represents the factor of comparison with the time required for feature extraction, it follows that as embedding dimensionality and database size scale up and the curves in the plot go above the dotted line, that is when retrieval becomes the critical factor for efficiency. Roughly it can be pointed out that to avoid bottlenecking the whole process, one should try to keep the product of descriptor dimensionality and size of the available gallery below $10^9$, as an order of magnitude.
Based on this reasoning, it is easy to understand why retrieval time is often considered as the most critical factor to the efficieny of a VG application, especially if it is meant to work in real-time. The chart also shows that if a system was to have a descriptor dimensionality of 2048, which is a reasonably small figure, when faced with a database containing images in the order of millions (which is not uncommon, like the San-Francisco dataset), even such a limited feature dimensionality would cause a ten-fold increase in the overall Inference time. Using a standard NetVLAD implementation, the

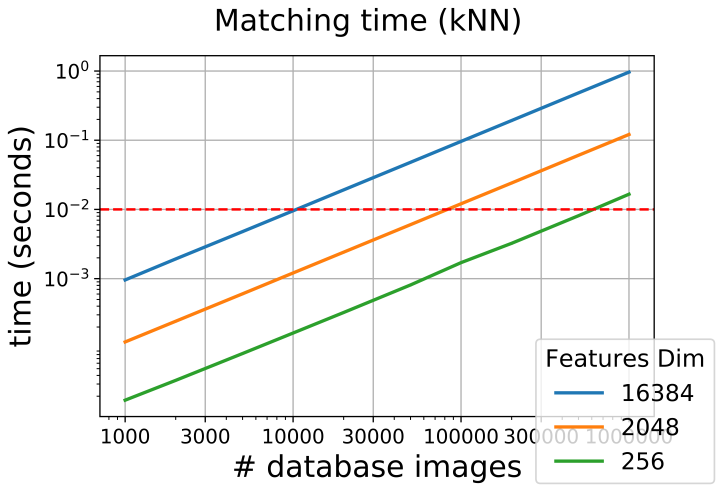



Figure 4.2: **Average Retrieval time for a single query.** Using an efficient parallelization of the the process, this curves show a linear dependence on the database size, plus a gap factor that depends on the feature dimensionality. The dotted line represents the time required for a VG system made up of a ResNet-101 feature extractor and GeM pooling to output an image descriptor. Chart from [1].

elapsed time would get multiplied by again a factor of 10. It is also important to remember that choosing a ResNet-18 is generally the better choice as it guarantees state-of-the-art results while keeping a lower number of parameters to train, and only requires one third of extraction time. Therefore in a potential system made up of a Resnet-18 + NetVLAD aggregator, the required time for retrieval would outweigh the embedding time by a factor of 300, constituting a serious bottleneck for the efficiency of the system.
For this reason, the following sub-section explores the possibility of using efficient indexing and approximate kNN, to reduce the cost of retrieval.

#### 4.2.5.1 Efficient indexing

This subsection studies the trade-of of computational time vs accuracy that can be adjusted using approximate kNNs, and as well the reduction in memory footprint that can be obtained using efficient indexing techniques. The figure below (Fig. 4.4) reports the figures to study such trade-of for each of the datasets taken into consideration throughout this work. Other than the traditional exact kNN, the following methods have been taken into account:

- **Inverted Indexes** known as IVF [28]. The concept of inverted indexes

is the one of *mapping from content*. The equivalent concept, applied to a book for example, would be given a word to return its location in the book. The way in which this idea is applied to vectors in the feature space is by computing a clustering, and then to associate all the vectors in a cluster with their Voronoi cell. This renders the search much faster because it eliminates the need to search among the whole set and the search is restricted to a cluster and its neighbouring ones. While this does speed up the process, it also introduces 2 additional hyperparameters to set. The first one is the number of Voronoi cells in which to split the search space, which has been kept fixed to 1000 throughout the performed experiments in Fig. 4.4, and the second one is the search radius, in terms of number of neighbouring Voronoi cells to extend the search to. For this second hyperparameter the values of 1 and 10 have been tested, as reported in the legend. So this provides a speed-up at retrieval time without causing any accuracy losses;

- **Product Quantization** known as PQ [46]. The main focus of this technique is reducing the memory footprint required by the search, trading for it some accuracy in the retrieval. The working concept is the following: given the collection of the database feature embeddings, they get split into equal-sized subvectors. The number of sub-vectors in which embeddings are split is an hyperparameter of the process which is called $m$. It is usually set to 8. Then, on the sub-space created by the set of sub-vectors, a clustering procedure is performed, so that the representation of each (full) vector is drastically reduced to only a small number of IDs (precisely, $m$) representing the centroids to which each of its subvector has been assigned. Fig. 4.3 shows the advantage brought by this idea with respect to the memory requirements of a full clustering like the one performed by IVF. The main drawback in this approach is the sure drop in recall due to the approximation introduced by the quantization of the subvectors. The number of bits used to hold the centroids can mitigate this phenomenon, but it is unavoidable due to the nature of the method.

- **Composed index: IVF + PQ** known as IVFPQ. PQ is a technique that can yield major improvements in the search time. A way to further reduce the search time, is to combine the 2 previously mentioned techniques. In paricular, the quantized vectors are associated to their Voronoi cells, and only a subset of the neighbouring cells is used to perform the search for a new vector. Losses in recall remain the issue,

however the speed-up that can be obtained is remarkable.

- **Inverted Multi-Index** [47]. It follows a similar approach to the IVFPQ index, and the results are the same, meaning that it provides substantial speed-ups in terms of retrieval time, paying the price of loosing in recall. The idea borrowed from the PQ technique is to split each vector in a subset of sub-vectors, therefore splitting its representation in a multi-dimensional fashion. Inverted Multi-Indexes do the same, and replace each vector representation with a multi-dimensional table, in which for each entry contains a compressed representation of the subvector. This compressed representation are then kept in an efficient data-structure similar to the one of inverted indices that associates each centroid-like structures to the sub-vectors that are close to it, ordered by increasing distance.;
- **Hierarchical Navigable Small-World graphs** known as HNSW [48]. The authors propose to represent the vector space with a graph-like structure, in which each vector is associated to a node, and the set of edges is built to reflect the distances among vectors in their original space. It provides a speed up in the search without loosing in precision.

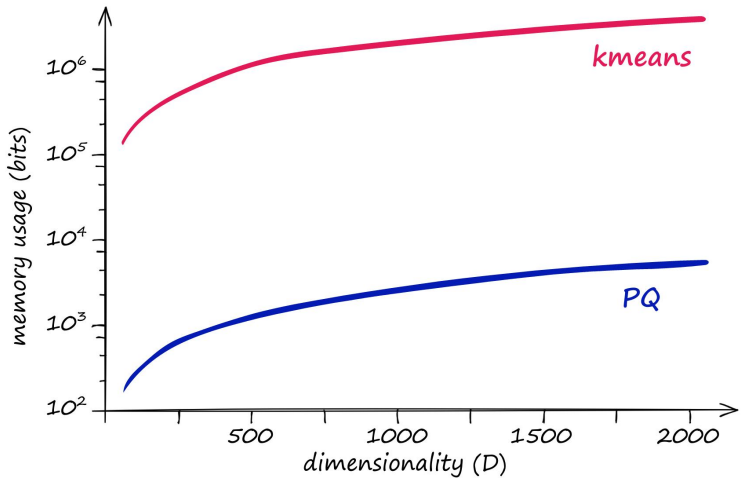



Figure 4.3: **Reduction in memory footprint** obtained using Product Quantization technique with respect to a IVF index alone that performs clustering without the quantization operation. Credits to `https://www.pinecone.io/learn/product-quantization`.

The results showed in Fig. 4.4 follow the expectation. Of course the dataset in which improvements are more substantial in terms of time saving

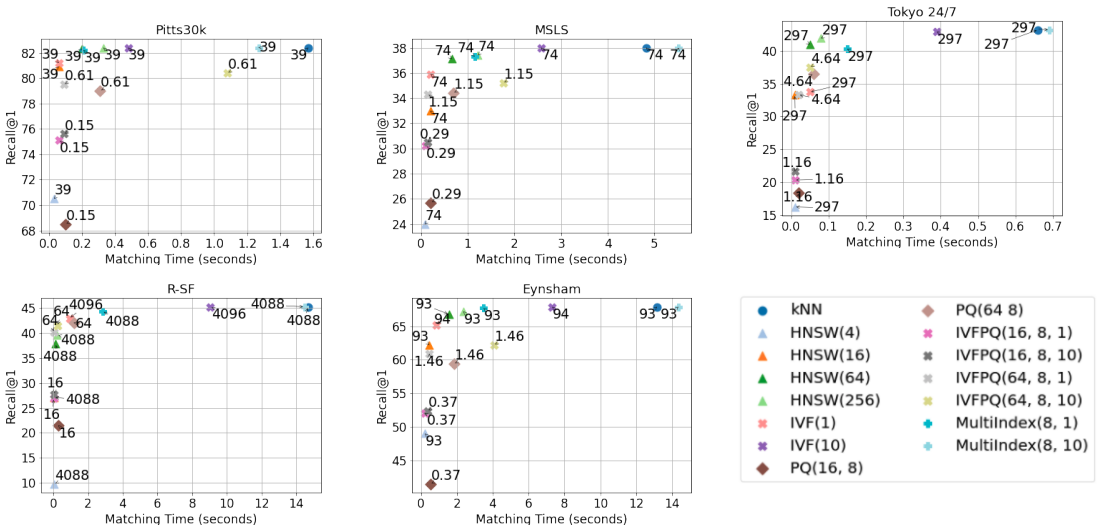



Figure 4.4: Analysis of **Efficient indexing techniques for the kNN**. Each plot refers to a different dataset, and the pipeline used for testing contains as a backbone the ResNet-50 whose feature maps are aggregated using GeM, resulting in a final descriptor dimensionality of 1024, which had been trained on the Pitts30k dataset. The charts compare the elapsed time measured in seconds, for each of the indexing technique to retrieve the top-10 candidates for the complete set of queries present in the various datasets. The time elapsed is related to the accuracy achieved in terms of R@1 reported on the y axis. Each data point in the graph is equipped with a number on the side, that indicates the minimum memory that that particular retrieval technique required, in MegaBytes. Plots from [1].

is San Francisco, which is the one containing a bigger database. Inverted File Indexes and Hierarchical Navigable Small-World graphs prove to be effective reducing the matching time in average by a 50%, without loosing any performances. However, they do not reduce the memory footprint.

In terms of time requirements, the methods that stand out the most are IVFPQ and Inverted Multi-Indexes, which can reduce the retrieval time by a factor of 20, with great memory efficiency as well. In the case of San-Francisco, which is the most demanding dataset, the memory footprint using IVFPQ goes from over 4 Gb to only 64 Mb, which is remarkable. In terms of performance lost due to this trade-of, there is a drop to be expected of anywhere from 4 to 8 %.

Concluding this analysis, the optimal choice depends as always on the

application, however some general considerations can be drawn out. Inverted Indexes should be a safe bet in all cases, as they provide a moderate but interesting speedup, without trading any performances for it; whereas application in which time and memory constraints are critical, for example embedded systems or real-time applications, Inverted Multi-Indexes and IVFPQ are the clear choice. An important additional consideration is that, as a countermeasure to the precision drop caused by approximate searches, it is possible to use Candidates Re-Ranking techniques on the top retrieval candidates like the ones presented in 2.4.4 to improve results.

### 4.2.6 Data augmentation and pre/post processing

This section explores the application of techniques like data augmentation and pre or post processing which are often disregarded as less important but can actually make a substantial difference. All the techniques presented in this section are thought as to be applied on the queries only, as depicted in Fig. 4.1, as it is on the queries that the network learns how to extract robust features to differentiate them from the contrast offered by the negatives in the triplets. An exception to this rule is, for data augmentation, the application of random flipping on the horizontal axis, in which case it gets applied to all the items in the triplets.

**Data augmentation**

The analysis starts from Data augmentation, which in modern Deep Learning is a omnipresent component. The focus is mainly in understanding if there are some techniques that work better or not on a specific dataset, relating results to the characteristics of each dataset and therefore pointing to what a specific application might require. The experiments are run on a pipeline made up by a Resnet-18 as backbone, and a NetVLAD aggregator while training on Pitts30k. Results are reported in Fig. 4.5 testing on multiple datasets.

As anticipated, results confirm the considerable impact of augmentation on the outcomes, with variations depending on the specific characteristics of each dataset. Starting as usually from Pitts30k, once again the modest size and more importantly, limited variability that it offers is once again the determining factor of the fact in general any form of augmentation is no useful, except in some specific and rare cases. The homogeneity of the scenes that it contains make so that any technique that forces the network to collect more general features causes performances to drop.

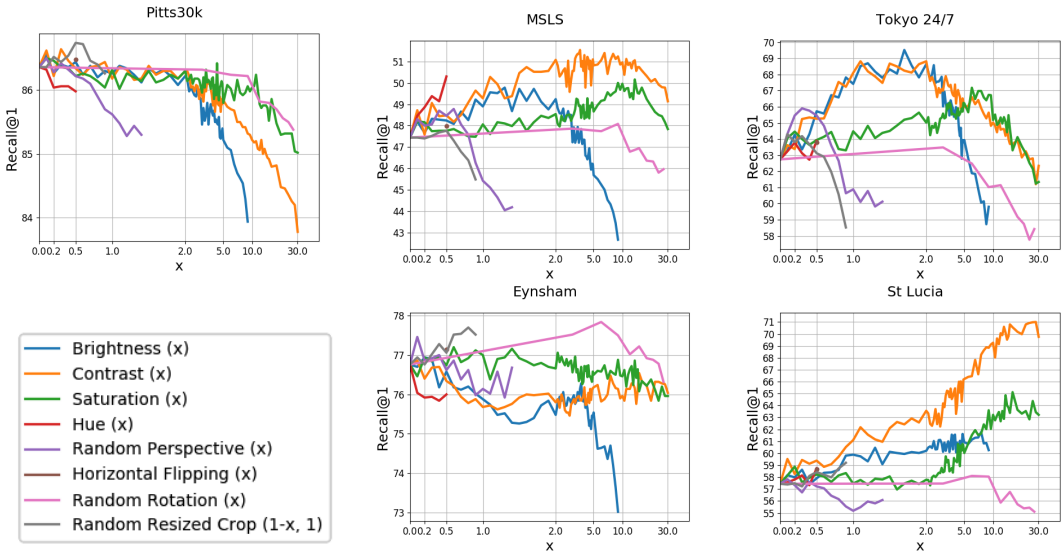



Figure 4.5: Various techniques of **Data Augmentation** applied for each dataset. Their implementation is used from the torchvision library for image transformations. The transformations regard changing the visual attributes of the images, like brightness, hue, saturation and contrast. Also random changes of perspective are tested, as well as mirroring modifications like flipping. Also stretching the scale with resizing has been tested. The higher the parameter of the transformation, the more drastic changes are applied. Plots from [1].

Contrarily, on the other datasets, which is important to highlight were not encountered during training and represent a completely new test suite, data augmentation shows its powerful effects. It is remarkable how this simple technique can manage to yield networks that learn how to extract features that are overall more informative and more robust to domain shifts, simply by modifying the appearances of training queries.
Among the more effective augmentation can be for sure pointed out how injecting random contrast and brightness, except that for Eynsham (it is important to note here that it is a grayscale dataset) bring quite impressive improvements, up to 5 % on Tokyo and MSLS, and even more on St.Lucia, while not causing any noticeable drops on the original training dataset (Pitts30K).

Concluding this analysis, it can be highlighted that there is not a single technique that is a clear win-win for all situations, but there are some useful considerations to point out. For example, in the case of horizontal flipping,

which in the graphs is a dot in correspondence of the value 0.5 of the x axis (in this case is the probability as the flip), and resizing as well with a cut of 50% of the resolution, they both show to provide an improvement, however small, but consistent across the board without ever causing any drops. This is probably due to the fact that such transformation do not alter the appearance of the image, and therefore can turn out to be useful for learning practically with every dataset. An additional consideration can be that jittering modifications (changing contrast and brightness for example) are useful especially if the expected domain of testing of the application contains several domain variations in the appearance (e.g. lighting, weather). This is the case for example of Tokyo and San Francisco, and in fact it is noticeable how for these two datasets such transformation provide substantial improvements. In the case of Eynsham, instead, being it a grayscale dataset, the transformations that turn out to be more useful regard changes in perspective, and flipping.

**Pre/Post -Processing**

As detailed in previous explanation of the dynamics at a play within a real application of a VG system, once the system is operating and receives new queries from the users to be localized, it can likely happen that some of those queries will have different resolution with respect to the ones contained in the database. For these reason, many datasets preemptively include queries with this characteristics in their test sets; it is the case of Tokyo [116], San-Francisco [13], and it is even more common in datasets devoted to the LR field [9].

Commonly in the literature ([39, 131, 44], a practical workaround to this issue is to take the simplest approach and forward the queries through the network in a 1-by-1 fashion. The ratio of this choice is that it is not possible to batch together images with different shapes, and therefore to avoid further processing the simplest alternative is to not batch at all images and perform single forwards. The problem of this choice is that it definitely causes delays in the process, as it does not fully exploit the parallelization capabilities of GPUs; such delay gets more and more relevant as the number of queries that the application receives increases. The goal of this section is to investigate if it is possible to use different approaches in order to speed-up the computation while also possibly gaining in recall performances. The classification of the methods adopted is reported in the general scheme of the proposed pipeline (Fig. 4.1, dividing them in Pre/Post processing method, and eventually refinement of the prediction. The adopted methods are the

following:

- **Hard Resize**: it is a **Preprocessing** technique that consists in a smooth rescaling of the query images to match the standard size that the images in the database present. Therefore if in a dataset (like it is the case for Pitts30k) queries and galleries already present the same size, the method represents the identity transformation. Allows batching of images originally of different sizes;

- **Single Query**: it is among the **Preprocessing** techniques and it still performs a rescaling of the queries, but with a milder approach as it respects the original width to height ratio and only forces the short side to match the dimension found in database images. Does not allow for batching if queries have different aspect ratios;

- **Central Crop**: it also a form of **Preprocessing**, and the procedure to obtain an image of the same size as the database images is to crop a central region from the query of the right dimensions; if the query is too small it gets first enlarged to allow for such a central crop to be taken;

- **Five Crops**: as the last option implemented for **Preprocessing**, consists in taking more than one sample from each query and the choice of the best match is left after the feature evaluation to be performed with different strategies explained below. Specifically, the five samples are taken as square patches of size equal to the shortest dimension of the query.

- **Mean** is a **Postprocessing** technique that can be applied together with the Five Crops approach; specifically it is applied after the descriptor evaluation has been completed, by taking the mean of the distances of database images from each of the query crops to pick the best match.

- **Nearest Crop** is a **Prediction Refinement** step applicable after a Five Crops preprocessing, and it is based on the idea that instead of taking the average distance, the choice should be based on the crop that yields the minimum distance with an image in the database. This is because the average crop distance can be misleading if a particular crop represents a non-informative scene and therefore pollutes the evaluation;

- **Majority Voting** falls within the **Prediction Refinement** category, also to be applied together with Five Crops preprocessing. The idea is

that besides the raw distance value of each crop from a database item, which can vary on magnitude depending on the specific appearance, it should matter if more than crops agree with each other in selecting a certain image as their match. Therefore the final similarity for a given database image is weighted by how many crops agreed on that item.

Table 4.8: **Analysis of Pre and Post processing techniques** applied to queries at test time for inference. Table from [1].

| Backbone | Aggregation Method | Pre/Post-Processing Method | Pre-Proc. | Post-Proc. | Batch Parall. | Training Dataset. | R@1 Pitts30k | R@1 MSLS | R@1 Tokyo 24/7 | R@1 R-SF | R@1 Eynsham | R@1 St Lucia |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ResNet-18 | GeM | Hard Resize | Y | N | Y | Pitts30k | **77.8 ± 0.2** | 35.3 ± 0.5 | 31.8 ± 0.9 | 33.2 ± 2.1 | **64.3 ± 1.2** | **46.2 ± 0.4** |
| ResNet-18 | GeM | Single Query | Y | N | N | Pitts30k | **77.8 ± 0.2** | **35.6 ± 0.6** | 35.3 ± 1.1 | 34.2 ± 1.7 | **64.3 ± 1.2** | **46.2 ± 0.4** |
| ResNet-18 | GeM | Central Crop | Y | N | Y | Pitts30k | **77.8 ± 0.2** | 34.8 ± 0.5 | **36.4 ± 1.1** | 32.6 ± 1.4 | **64.3 ± 1.2** | **46.2 ± 0.4** |
| ResNet-18 | GeM | Five Crops Mean | Y | Y | Y | Pitts30k | 75.4 ± 0.3 | 30.2 ± 0.2 | 35.9 ± 0.5 | 34.4 ± 2.0 | 59.1 ± 0.7 | 43.3 ± 0.8 |
| ResNet-18 | GeM | Nearest Crop | Y | Y | Y | Pitts30k | 74.8 ± 0.1 | 28.3 ± 0.3 | 33.8 ± 1.3 | **35.7 ± 1.6** | 55.5 ± 0.8 | 39.4 ± 0.5 |
| ResNet-18 | GeM | Majority Voting | Y | Y | Y | Pitts30k | 75.1 ± 0.0 | 29.1 ± 0.4 | 34.8 ± 1.5 | 35.3 ± 1.3 | 51.8 ± 0.2 | 41.3 ± 0.5 |
| ResNet-18 | NetVLAD | Hard Resize | Y | N | Y | Pitts30k | **86.4 ± 0.3** | 47.4 ± 1.2 | 58.3 ± 1.4 | 58.9 ± 1.1 | 76.8 ± 1.2 | **57.6 ± 3.3** |
| ResNet-18 | NetVLAD | Single Query | Y | N | N | Pitts30k | **86.4 ± 0.3** | 47.5 ± 1.3 | 63.4 ± 1.2 | 61.4 ± 1.5 | 76.8 ± 1.2 | **57.6 ± 3.3** |
| ResNet-18 | NetVLAD | Central Crop | Y | N | Y | Pitts30k | **86.4 ± 0.3** | **48.0 ± 1.3** | 63.2 ± 0.2 | 57.8 ± 0.4 | 76.8 ± 1.2 | **57.6 ± 3.3** |
| ResNet-18 | NetVLAD | Five Crops Mean | Y | Y | Y | Pitts30k | 85.1 ± 0.2 | 45.3 ± 1.3 | 63.0 ± 0.7 | 60.9 ± 1.7 | **78.9 ± 0.9** | 54.6 ± 2.8 |
| ResNet-18 | NetVLAD | Nearest Crop | Y | Y | Y | Pitts30k | 84.8 ± 0.2 | 46.0 ± 1.5 | **67.0 ± 1.4** | **64.8 ± 0.7** | 75.7 ± 1.4 | 53.0 ± 2.5 |
| ResNet-18 | NetVLAD | Majority Voting | Y | Y | Y | Pitts30k | 84.8 ± 0.3 | 45.2 ± 1.4 | 66.9 ± 1.1 | 64.7 ± 0.7 | 77.1 ± 1.1 | 53.4 ± 2.3 |
| ResNet-50 | GeM | Hard Resize | Y | N | Y | Pitts30k | **82.0 ± 0.3** | 38.0 ± 0.1 | 34.6 ± 1.4 | 40.7 ± 1.8 | **66.3 ± 2.5** | **59.0 ± 1.4** |
| ResNet-50 | GeM | Single Query | Y | N | N | Pitts30k | **82.0 ± 0.3** | **38.2 ± 0.3** | 41.5 ± 1.8 | 45.4 ± 2.0 | **66.3 ± 2.5** | **59.0 ± 1.4** |
| ResNet-50 | GeM | Central Crop | Y | N | Y | Pitts30k | **82.0 ± 0.3** | 37.5 ± 0.3 | 40.4 ± 0.9 | 41.0 ± 2.6 | **66.3 ± 2.5** | **59.0 ± 1.4** |
| ResNet-50 | GeM | Five Crops Mean | Y | Y | Y | Pitts30k | 80.4 ± 0.1 | 33.2 ± 0.1 | 39.8 ± 2.0 | 43.8 ± 0.9 | 65.0 ± 2.4 | 54.4 ± 1.3 |
| ResNet-50 | GeM | Nearest Crop | Y | Y | Y | Pitts30k | 79.2 ± 0.2 | 30.8 ± 0.2 | **43.5 ± 1.4** | **46.9 ± 1.4** | 63.5 ± 2.2 | 52.6 ± 1.4 |
| ResNet-50 | GeM | Majority Voting | Y | Y | Y | Pitts30k | 79.7 ± 0.0 | 31.5 ± 0.1 | 43.0 ± 2.0 | 44.8 ± 1.2 | 62.9 ± 2.3 | 52.8 ± 0.9 |
| ResNet-50 | NetVLAD | Hard Resize | Y | N | Y | Pitts30k | **86.0 ± 0.1** | 50.7 ± 2.0 | 64.3 ± 1.9 | 64.3 ± 1.2 | 77.7 ± 0.4 | **60.2 ± 1.6** |
| ResNet-50 | NetVLAD | Single Query | Y | N | N | Pitts30k | **86.0 ± 0.1** | 50.6 ± 1.9 | 69.8 ± 0.8 | 67.1 ± 2.3 | 77.7 ± 0.4 | **60.2 ± 1.6** |
| ResNet-50 | NetVLAD | Central Crop | Y | N | Y | Pitts30k | **86.0 ± 0.1** | **50.9 ± 1.9** | 68.3 ± 1.4 | 64.6 ± 2.2 | 77.7 ± 0.4 | **60.2 ± 1.6** |
| ResNet-50 | NetVLAD | Five Crops Mean | Y | Y | Y | Pitts30k | 84.7 ± 0.1 | 47.4 ± 1.9 | 68.0 ± 2.2 | 66.5 ± 1.5 | **78.6 ± 0.3** | 54.3 ± 2.8 |
| ResNet-50 | NetVLAD | Nearest Crop | Y | Y | Y | Pitts30k | 84.2 ± 0.2 | 47.0 ± 1.7 | 72.3 ± 1.3 | **68.4 ± 0.8** | 76.8 ± 0.5 | 52.3 ± 2.3 |
| ResNet-50 | NetVLAD | Majority Voting | Y | Y | Y | Pitts30k | 84.3 ± 0.2 | 47.1 ± 1.7 | **72.8 ± 0.8** | 68.1 ± 1.3 | 77.5 ± 0.4 | 53.4 ± 2.2 |

Analyzing the findings in Tab. 4.8, it should be specified right away how the fact that for datasets like Pitts30k and Eynsham (and St Lucia as well) results stay identical varying the pre-processing step is simply explained by the fact that such datasets do not include a variability in terms of query resolution; therefore any of those strategies is equivalent. It follows that for such datasets, there is no need for more complex pre/post-processing methods, as the aforementioned techniques already give the best results.
Perhaps the more interesting results can be found regarding the datasets that are more challenging in these sense; Tokyo and San Francisco, in fact, present a broad variability in the queries and many of them, being taken using smartphones, present the unusual condition of being taller than they are large (portrait mode). The table shows clearly that a more 'careful' treatment rather than a brute Hard Resize can yield noticeable benefits ( up to 9% in recall) using Refinement strategies. Nearest Crop usually outperforms the other, as it allows for the more representative crop to find its match and not

take into account other parts of the images that may be rather uninformative and unhelpful to the localization; however also Majority Voting works nicely providing a consistent bump up in performances on the mentioned datasets. as well.

Concluding, the suggestions that can be drawn out from this analysis depend, as it is always the case, on the kind of data that the designed application is going to encounter. If resolution changes are not part of the picture, for example if the system receives only standardized images, then this steps are not a big deal and an Hard Resize will do the job. However if that is not the case, it is very much advisable to spend the extra time for implementing this concepts as the results will improve noticeably. The only technique that is perhaps never a good choice is Single Query; which shows poorer performances than the other exposed techniques, and does not allow parallelization either. Contrarily, all the other techniques have the advantage of guaranteeing consistent speed-up in the overall inference time, allowing to fully leverage the hardware capability; which is a fundamental pre-requisite for the scalability of a VG system, especially if it can find itself serving multiple queries from multiple users at the same time.

# Part II

# Chapter 5

# Sequence-based Visual Geo-Localization

This chapter addresses the second part of the contributions of this thesis, which is devoted to analyzing the novel field of Sequence-based Visual Geo-localization, which is of potential interest for many applications and yet it has been scarcely studied by the community. As mentioned in Sec. 2.6, the ability to leverage sequences of images is a natural extension to the VG task especially for applications like autonomous driving or SLAM settings in robotics. In all these cases sequences are easily accessible by the system, and therefore it is important to develop adequate models to handle such data. In particular, one of the motivations of considering the multi-frame problem was given by the rising popularity of Transformers in Image Recognition tasks, as their architecture is naturally oriented towards handling sequence-like data, and therefore exploring their applicability to the VG task is one of the goals of this chapter.

## 5.1 The task

As the Sequence-based Visual Geo-Localization (S-VG) is such a young research field, the literature is still lacking a clear definition of the task. One of the earliest works using Deep Learning to address the task, giving the first definition in order to define their task was [71]. Their proposition was renewed in the MSLS paper [114], where the authors together with the dataset address as well the task definition, in three different versions, expanding the proposition of [71] to include more general situations. The definitions are the

following:

- **im2seq**: define matches from single queries, as they traditionally are treated in the VG task, to sequences;
- **seq2im**: use a stack of multiple queries as input sequences to match against single database images;
- **seq2seq**: use a stack of frames as both queries and database items, matching therefore between sequences.

They also defined a multi-frame match between two sequences, or a single frame and a sequence, with the necessary condition of having at least one frame within the given threshold. Thresholds are kept the same as they traditionally are in the im2im VG task (25 m for positives during inference, 10 during training [39]). The work for this thesis is meant to be an explorative delve into this field for which there are little to no references in the literature to provide supporting hints and direction, and it has been chosen to stick with the **seq2seq** approach. The motivation behind the choice was to study the ability of different pipelines to extract robust and representative features from sequences of images, in order to exploit the potential hidden in this additional source of data. Therefore it was preferred to have the same multi-frame setting in both the queries and the database.
It needs to be acknowledged that, however, the other formulations of the task are just as interesting and in some practical applications where the objective is the localization of an object in space at the end of a path, the task would be formulated differently in a similar fashion to the seq2im described above, but where the meaning of a match is that that frame represents the point of arrival at the end of the sequence.

Summing up, the task addressed throughout this chapter follows the specifications provided by [114] for the so-called **seq2seq** setting, and the focus of the analysis will be the ability of models to extract additional information from sequences rather than single frames, hoping to improve performances while also keeping an eye on the computational cost, which is a natural concern when extending the number of data to treat at once. A particular point of interest of such analysis is to experiment with the use of Transformer architectures, introduced in Sec. 2.7 originally proposed in the literature for NLP, that lately have shaken the interest of the community gaining a lot of popularity in Computer Vision tasks as well.

## 5.2 Objectives

The aim of this brief section is to define what are the main motivations guiding the experiments throughout this work, and which are the questions addressed that may be of interest to the community. As already mentioned, the interest stemmed mainly from the observation that in many applications (autonomous driving, SLAM) multi-frame images are naturally available and therefore it would be rather useful if such availability of data could be properly exploited by the adopted VG pipeline.
Secondly, as also mentioned above, a particular point of interest was to verify the applicability of Transformer architectures, lately introduced with success in the field of Video Analysis, specifically for Action Recognition. Indeed the Transformer architecture is based on the concept of Self-Attention applied to sequence of tokens; which makes it naturally suited to treat tasks in which sequences are involved, making it an appetible candidate for the task of interest.
Another point that sustains the choices made during the experiments, is the observation that, unlike in the field of Action Recognition or Video Stream Analysis where in general it is desirable for the model to be able to deal with large numbers of frames, for the task of VG this is not the case due to many reasons. First of all, the training procedure that has proven to be the most successful for the VG task is based on metric learning and therefore on the use of triplets comparing a query, its positive and a number of negatives which is usually 10 [39]. Such a number of negatives is critical to successful learning and therefore cannot be drastically lowered (a small set of experiments will be showed to support this affirmation), therefore the number of frames to be used gets multiplied by a factor of 12 (images for each triplet) rendering the procedure unfeasible soon.
However, there is also a more conceptual reason to support the choice of disregarding long sequences. In fact, for a system whose aim is to localize a sequence in space, it is not meaningful in practical application to consider sequences that span over long distances, losing the concept of *localization*. To give some numbers in order to give a sense of the distances that are the object of discussion, consider the following. First of all, the idea of exploiting more frames is based on the ability to capture the semantic categories inside each frame, compare them across the sequence and finally obtain a single sequence descriptor which is in the end more informative than a single-frame descriptor would be. Therefore it would not make sense to have frames with identical appearance, and the bottom line of this reasoning is that frames

should be separated in space by a distance of at least $3m$ or more. Hence, sequences of more than 15 frames would span over a distance of more than $50m$, which as said earlier starts to lose the concept of localization of a specific place. Moreover, due to the definition of the task as it was proposed by [114], two sequences would match even if they had a single frame in common, which for long sequences may result in paths that are diverging but still share the starting or middle part; the result would be asking to the network a task as impossible as it is nonsensical.
It follows that, given the previous considerations and the general setting with which the **seq2seq** has been defined in the literature, it would not be of interest to spend time in developing models tailored for a sequence with an elevated number of frames, beyond 30. Instead, the focus of the methods developed for this thesis is to use short sequences in order to keep the task about precise geo-localization and to increase the performance obtainable with single images by exploiting the extra information available from the multiple frames. Such an approach is also able to preserve the efficiency of the system keeping low latency which can be a critical aspect in systems that need to operate in real-time, with high precision and low delays (one above all being autonomous driving applications).

## 5.3 Dataset and setting

The availability of datasets is another point where the lack of intensive studies in the literature emerges, as their number is quite scarce. Most of the works based on sequences localization were done in the field of robotics (mainly SLAM) and relied on rather small sets of images which were not suitable for the goals of this work, that is testing deep architectures whose large representation capability requires large datasets in order to obtain successful learning without overfitting. In some works like [72] the authors perform a custom processing of the Oxford Robotcar [132] and Brisbane to obtain sequences, and another popular S-VG dedicated dataset, Nordland [133]. The latter is a rather suggestive dataset containing sequences of frames about the trip of a train around Norwegian fjords for over 3000 km, repeated across the years to include seasonal changes. However spectacular such images might be, the critic that can be raised to such a dataset is the images contained are all from the same country-side setting, full of rather spectacular landscape views which however are not very useful for a model that has to learn features as robust as possible in order to be able to generalize to different settings, as

the models presented in Chapter 4. Training on this dataset would yield a model almost unable to detect any of the semantic categories that are likely to be found in an urban setting for example.

Comes to the rescue the Mapillary [114] dataset, already widely used in the experiments presented in Chapter 4, in its single frame version. However, the dataset has been proposed originally with the additional purpose of promoting the field of S-VG, by including in this dataset the capability to generate sequences of any desired length between 3 and 300 frames per sequence. Therefore the dataset of choice was this last one, coming with the great advantages of:

- **Domain variability**: The same characteristics that made MSLS a good training dataset for the single-frame VG models still apply in this case. The dataset includes a broad variability of lighting, scenario (urban, countryside, suburbs) over a large timespan and cities from all over the world, allowing for training procedures that yield robust models that are more likely to learn a large variety of semantic categories;
- **Flexibility**: The capability to generate sequences of any desired length is a great feature for an experimental process such as the one proposed, as it easily allows to test the extractive ability of models under different conditions;
- **Modularity**: The dataset is structured on a per-city categorization, therefore it is possible to choose only some cities (or to exclude some as well), providing a wide range of experimental possibilities to understand the strength and weaknesses of models;
- **Open source code**: The authors released an open-source codebase that allows researchers to just plug in their code, focusing only on the more relevant implementation issues regarding their pipelines.

Nevertheless, there are also some criticalities regarding the Mapillary dataset that emerged during the work for this thesis, and for the sake of completeness they are hereby reported as these are reflections that can be useful for the community as they have been maturated across many months of use and experimentation:

- **Density of the database**: about the distribution of the database images with respect to the query, an observation raised after a deeper

analysis is that the database images are only present in geographical locations where some queries as present as well; this does not limit the applicability of the dataset but it is, however, a not-so-likely scenario for a real-life application in which one would like to have uniform geographical coverage in the database, and then for the queries there are really no constraints as they can be distributed in any way inside the area covered by the database. Ideally, queries should be uniformly distributed as well to test the learning capability of the model in all locations, but however, the point is that having a database distribution that is along the lines of the one of the queries is not very realistic;

- **Absence of test labels**: the authors, following the idea of proposing a challenge on their newly released dataset, did not release initially the labels. However, even afterwards the labels have not been published and the authors upon request replied that they are not planning to. This is not a big issue and the easy workaround is to just use the validation split, as the latter is made up of completely different cities than the one in the training split this is not an unovercomable issue but it does lead to the waste of all the images in the test set;

- **Open source code**: while this is in principle a positive aspect, as it was listed as well above, the code provided by the authors did not stand the test of time. In fact, many issues were found in the released implementation, causing several delays in the development process. It is only advisable that the authors acknowledge such problems and provide patches to what could otherwise be a great contribution to the community.

## 5.4 Architectures

In this section are described the main architectures used to evaluate different aspects of the **seq2seq** task for VG. Remembering once again the experimental nature of the setting tackled, one of the main challenges faced was the inability to draw a clear strategy from the beginning as for the methods and architectures to test, having to go back and forth in the process of hypothesis and experimental verification, in many cases seeing the hopes for a particular implementation get shattered against the test of reality. Therefore the structure of the following section reflects the twist and turns encountered during the highly challenging yet instructive path.

### 5.4.1 Baselines

Before engaging in more elaborate implementation, it is important to set the baseline results against which to draw comparisons. In situations like this, the massive amount of experiments run for the purposes of the work exposed in Chapter 4 proves its usefulness, even if the field of application is different. As the first option for baselines, it has been chosen to test methods from single image VG, with small changes to adapt them to the multi-frame setting of the task. Specifically, among the many possible backbones that the literature on VG has adopted, the thorough discussion presented in Sec. 4.2.1 tells us that using a ResNet-18 is likely to be the best choice being the most efficient network guaranteeing state-of-the-art results as well. As for aggregation methods, following the analysis in Sec. 4.2.2 it becomes more clear that the most interesting techniques to try out are NetVLAD and GeM, for the generally higher accuracy achieved by the former, and the lightweight nature of the latter.
Such methods are the most interesting ones to be used as baselines results like the fact that they achieved state-of-the-art performances on the single image VG task proves that they are the most suited to extract a meaningful representation of the semantic elements present in single images; hence the main challenge lies in being able to exploit additional information thanks to the extra frames that was not possible to obtain without specifically tailored methods. Therefore the challenge undertaken will be deemed as successful if some of the proposed methods can outperform the mentioned baselines.

Another possible source for baseline methods could be the small portion of works present in the literature. The more interest-worth approaches were exposed in Sec. 2.6, specifically from the literature [73, 71, 72]. As for the approaches proposed by [71], they are briefly summarized here:

- **Descriptor Grouping** Plain concatenation of features extracted by a convolutional backbone;
- **Descriptor Fusion** Aggregation of the concatenated features using in the end a Fully Connected layer;
- **Recurrent Descriptors** Use of a ConvLSTM module to be sequentially fed with the computed features from the backbone, so that the descriptor for the multi-frame input is sequentially updated.

The first two approaches are the more natural way of extending pipelines that have proven to be successful on the *im2im* task, by modifying only the

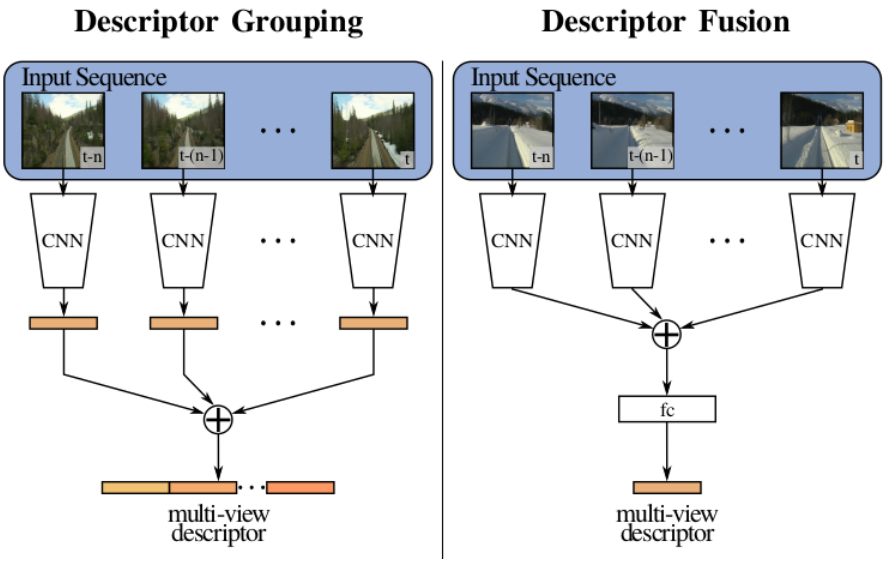



Figure 5.1: Representation of the main 2 techniques used to convert single image methods into baselines for the **seq2seq** task.Credits to [71].

way outputs are collected in order to generate a single sequence descriptor in the end. These propositions have been incorporated into the baselines methods, paired with NetVLAD and GeM as their single image descriptor extractor. Note that, especially with NetVLAD, the idea of concatenating single frames feature has the obvious issue of creating enormous descriptors causing all the problems that were largely analyzed in Chapter 4.
As for the second proposition, it has been incorporated as well in the set of considered baselines, again pairing it with the popular single-image feature extractor pipelines of NetVLAD and GeM.
Regarding the third option about using recurrent networks, it has been decided to set aside this approach without including it in the considered baselines. Reasons for this choice are multiple; starting from the fact that as the same authors in [71] show how in terms of pure recall performances it performs worse than the other 2, probably due to the inner difficulty when training recurrent LSTM networks in obtaining proper weights for the forget gate to track dependencies from the previous frames; even with the short sequences that are treated it can happen that the final descriptor is mainly based on the last frame. The main selling point for these methods in the mentioned paper is that it is able to strongly outperform the others in a situation of reverse-order frames and randomly sampled ones; in such cases, it becomes important that the descriptors are not just sequential stacking

of frames information but are a meaningful representation of the semantic elements present in the sequence overall, and of how they are intertwined between frames. However, according to the objectives of this work highlighted above, the main focus is high-performance precise Geo-Localization, and therefore such a model that has sub-optimal performances in standard situations does not reflect this purpose, whereas the methods that will be proposed will have a focus on all-round accurate recall. Furthermore, one of the initial motivations was to experiment with Transformer architectures which, with respect to Recurrent structure, avoid the problem of difficulties in maintaining context, and are more suitable for efficient parallel computation.

As for the work of [73], however interesting per se, as pointed out by [72], it is based on an unsupervised adaptation of the single-frame descriptors into a sequence representation, in a fashion that highly relies on the order of the encountered frames, requiring also elevated number of frames to achieve the proposed results. Hence it is clearly out-of-line with the objectives and motivations of this work and therefore has not been tested as a baseline.

Regarding the work presented in [72], it is the one that aligns the most with the goals of this thesis. In spite of this, their methods have not been taken into account, since it relies on a two-stream network, that processes frames both singularly with traditional VG methods, and both as a sequence using a Temporal Convolution to perform a weighted pooling over the temporal axis. Finally, they use a method that comes from traditional SLAM methods in which a sequential search in a matrix is used to match the top retrieval candidates obtained with the temporal-pooled descriptors and their single-frame processed counterparts. Such an approach is rather different than the methods developed for this thesis that rely only on the goodness of the representation that the model is able to extract, and comparing them with such method highly based on Prediction Refinement techniques would not constitute a fair evaluation setting. Moreover, the authors only report results using a subset of the cities contained in MSLS, without evaluating on the standard validation split, and also their results, in that case, are comparable with the single-descriptor only performances; therefore the baselines mentioned above were considered as representative enough.

### 5.4.2 The Transformer Encoder

The original Transformer architecture was proposed in [84], and it was originally meant for Natural Language Processing. The key idea from which the author start to build this architecture is that state-of-the-art methods ([134, 135]) for sequential computation had either linear or logarithmic growth in the number of computations needed for information to flow from a given point in the past of the sequence to the present. What this means is that it becomes hard to keep a context tracking long-range dependencies, and therefore the idea of Transformers is to make this number of computation a constant ($O(1)$ in the Big-O notation) through the employment of a mechanism called *Self-Attention*. The network is structured with an Encoder-Decoder structure which is depicted in Fig. 5.2.

The encoder generates a hidden embedding that is then fed to the decoder; the decoder works in an auto-regressive fashion, meaning that the output sequence is generated one token at a time and each output is fed back to the decoder input to generate the next result and so on. While based on a similar structure as Recurrent encoder-decoder architectures, the innovation that Transformers represent lies in the ability to process variable-length sequences without any kind of recurrent computations, using residual blocks augmented with attention. Mathematically, the task of a decoder is to obtain the target sequence of vectors $\mathbf{Y}_{1:n}$, when fed with the sequence $\mathbf{X}_{1:n}$, therefore to estimate the following conditional distribution:

$$p_{\gamma_{enc},\gamma_{dec}}(\mathbf{Y}_{1:n}|\mathbf{X}_{1:n}) \tag{5.1}$$

Where $\gamma$ represents respectively the set of parameters of the encoder and decoder. The encoder is used to build a mapping between the input sequence and a sequence of hidden embeddings :

$$f_{\gamma_{enc}} : \mathbf{X}_{1:n} \rightarrow \overline{\mathbf{X}}_{1:n} \tag{5.2}$$

Since the decoder estimates the final posterior distribution based on the hidden $\overline{\mathbf{X}}$, using Bayes Theorem is it possible to write the expression explicitly with respect to $\overline{\mathbf{X}}$ and all the $\mathbf{Y}_{1:j}$ in the past (remember the auto-regressive behavior):

$$p_{\gamma_{dec}}(\mathbf{Y}_{1:n}|\overline{\mathbf{X}}_{1:n}) = \prod_{i=1}^{n} p_{\gamma_{enc}}(y_i|\mathbf{Y}_{1:i-1}, \overline{\mathbf{X}}_{1:n}) \tag{5.3}$$

This is the setting generally applied in tasks that need to re-build an output sequence. As regards the work for this thesis and the VG task as well,

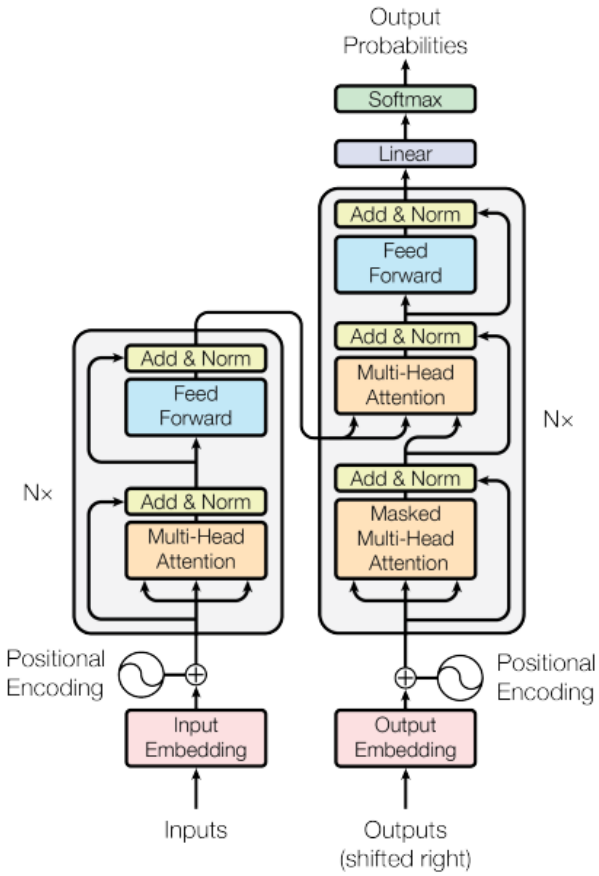



Figure 5.2: Working scheme of the Encoder-Decoder architecture as originally proposed by the authors. The picture represents a single encoder and a single decoder and they can also be modularly stacked on top of each other. Credits to [84].

the interest lies in obtaining an embedding that represents the image or, in this specific case, the sequence. Therefore in the employed transformer architectures, the decoder part has been dropped, and they have been used in an encoder-only fashion as it is common for retrieval and classification tasks. Note that the dimensionality of the embedding is the same as the one of input tokens, and the reason is that transformer blocks rely on residual sums inside each block and therefore dimensionality is kept fixed.
Moving on, let's describe the other fundamental concept introduced by transformers, which is the one of Self-Attention that is present in each block. As [136] precisely points out, the *bi-directional* nature of the implemented attention makes so that the context vector that is used to weight the representation

of a token at each step contains information from each step of the sequence. Therefore what happens as tokens get processed by each encoder block is that each block outputs an enriched representation of the tokens, obtained by weighting contextual information from every single other token in the sequence; each layer can learn to embed in the context different and more high-level features as tokens go up in the architecture ( [84] ). As it is defined by [136], in the first block vectors get transformed from "*context-independent representations*" to "*context dependent representations*". Technically, attention works in the following way:
For each token $\mathbf{x}_i$ in a sequence $\mathbf{X}_{1:n}$, there is an associated **Key**, $\mathbf{k}_i$, a **Value** $\mathbf{v}_i$ and a **Query** $\mathbf{q}_i$ that are nothing other representations obtained through the product of the input token with trainable matrices, one per each of the three representation; note however that the trainable weight matrices are shared for all tokens. Once the tokens have been projected into the three spaces of query, value and key they are used to compute the contextual representation in the following way.
For each of the tokens, its query $\mathbf{q}_i$ is multiplied with dot product by all the key vectors from the other tokens. Query and Key vector share their dimensionality $d_k$. The ratio of using the dot product is to evaluate which of the key vectors of other tokens is more similar in this mapping space to the query of the token at hand. Finally, the result of the dot products gets Softmaxed in order to make it a normalized weighting vector, which is then used to obtain the representation for the token at hand by multiplying such Softmaxed weighting vector with the values $\mathbf{v}_i$ vector of the whole sequences. Such representation is then summed with the original token $\mathbf{x}_i$. Mathematically:

$$\forall i \in \{1..n\} \tag{5.4}$$
$$\mathbf{q}_i = W_q \mathbf{x}_i \tag{5.5}$$
$$\mathbf{v}_i = W_v \mathbf{x}_i \tag{5.6}$$
$$\mathbf{k}_i = W_k \mathbf{x}_i \tag{5.7}$$
$$\mathbf{Att}_i = Softmax\left(\frac{\mathbf{q}_i^T \mathbf{K}_{1:n}}{\sqrt{d_k}}\right)\mathbf{V}_{i:n} \tag{5.8}$$
$$\mathbf{X}_i^{'} = \mathbf{X}_i + \mathbf{Att}_i \tag{5.9}$$

Moreover, such computation is particularly efficient as it is suitable to be batched in a single tensor set of multiplication obtaining a matrix with the embeddings for all the token $\mathbf{X}_{1:n}^{'}$. Fig.5.3 shows a rather explicative picture of the formulas reported above.

Moreover in their work, the authors of [84] propose to replicate the exposed attention process across many *Attention-Heads*, building the so-called *Multi-Head Attention* that consists basically in repeating the process with many (around 10, usually) different learnable matrices $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v$. A representation of this is showed in Fig. 5.4.

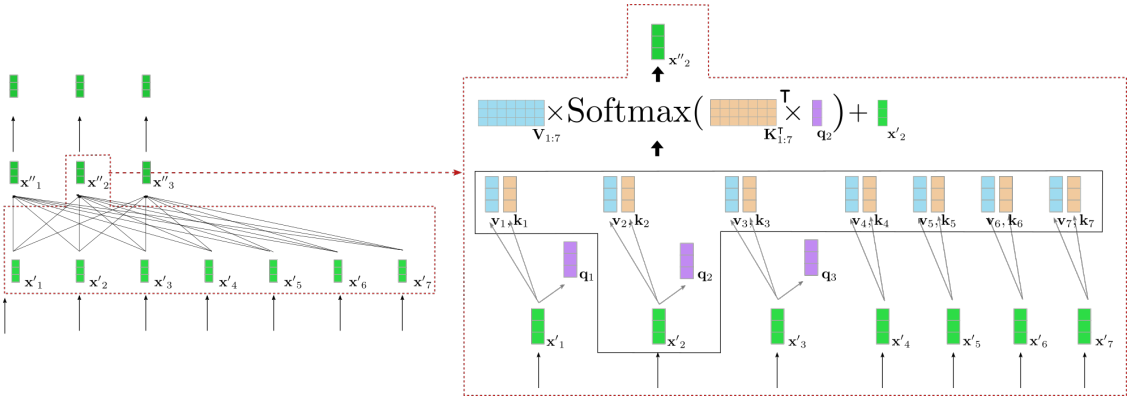



Figure 5.3: Image particularly explicative of the computations used to build the bi-directional contextual vector used to weight the representation of each token. Credits to [136].

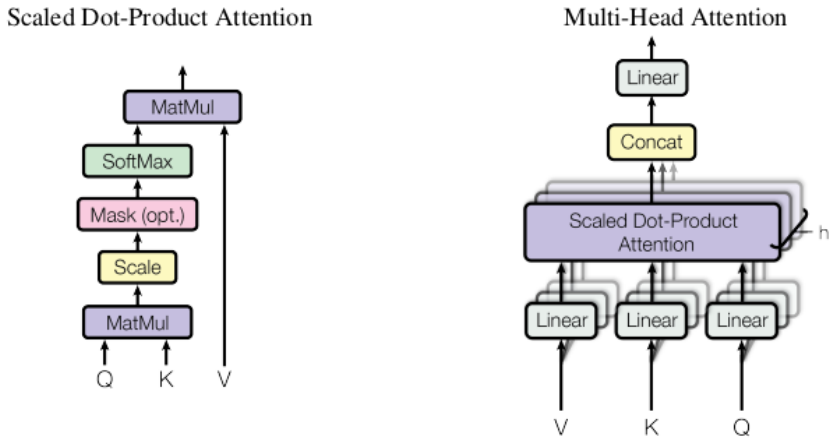



Figure 5.4: Representation of the implementation of the concept of self-attention used to build a context among tokens. Credits to [84].

Such a network needs to receive embedded tokens and cannot, therefore, be fed directly images. Therefore the methodology adopt to implement an S-VG

pipeline exploiting this vanilla transformer encoder, has been to use a ResNet-18 + GeM standard extractor applied to the single-images in the sequence, and to take the single representation as input token for the encoder, which therefore received during training tensors of size $(2+n\_neg, n\_frames, 256)$ for each triplet. Using NetVLAD instead turns out to be unfeasible, due to the fact that transformers encoders need to keep the dimensionality of the tokens fixed in order to perform residuals sum, hence using the 16384 descriptor size of NetVLAD yields unsustainable memory requirements. In particular, the tested architecture used the described vanilla Transformer Encoder with 6 hidden layers, 16 attention heads and obviously a final descriptor size of 256, the same as GeM, which is of course an ambitious experiment with such compact descriptors.

### 5.4.3 ViT - The Vision Transformer

After the original Transformer paper [84] was published in 2017, such models gained quite a lot of popularity in text mining and really any kind of NLP task where they achieved remarkable results (the most famous example being BERT [98]) outperforming all pre-existing methods in the literature. Such outstanding results suscitated the interest of the Computer Vision community, and after some trials and errors in 2020, the popular ViT architecture was published in [106], in which the authors adapted the Transformer architecture and the concept of attention to work with imaging data. One of the main problems that researchers had to solve in order to successfully apply Transformers to images, is the fact that such architecture comes at a cost, which to be specific is $O(n^2)$ in the number of tokens, because they need to be processed pair-wise; therefore a naive implementation having each pixel of an image perform contextual attention on every other pixel would yield an understandably unfeasible approach. The authors tackled such problem cleverly by using a Fully Connected layer to obtain a learn embedding for each patch of the input image; specifically, each embedding is obtained from a 16x16 patch of the original image and mapped into a space of size either 768 or 1024, or more (the authors release different-sized versions of ViT). Such patches are accompanied by a positional embedding purposely inserted to distinguish patches based on their initial location in the image.
Furthermore, one other issue standing in the way of successful implementations of Transformers for Image Recognition was the fact that the inductive bias introduced by Convolutional Network seemed to provide a decisive advantage with respect to transformers. By inductive bias, it is intended the

fact that convolutional kernels process images based on the fundamental hypothesis of equivariance to translation and locality, which in other words mean that the kernel weights can learn to detect visual features which are not dependent on a specific relative position inside the images, and the same kernels can therefore be applied to recognize that feature in whatever part of the image. Locality instead means that each feature in an image is recognizable just by looking at its neighborhood without the need of analyzing dependencies with pixels in the rest of the image, i.e. non-locally.

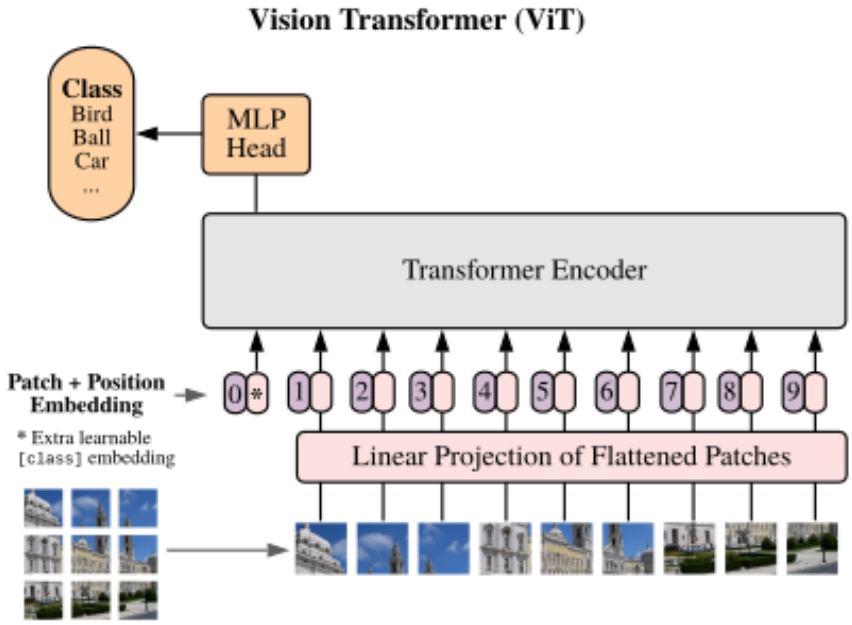



Figure 5.5: Architectural representation of the Vision Encoder-only Transformer. Shows the principle on which it is based, that is by flattening patches of the images, and using in the end the artificially added classification token as the final representation. Credits to [106].

The fundamental discovery of the authors of ViT is that such inductive bias is determinant when dealing with relatively small-sized datasets, requiring much fewer data to learn meaningful representation. Contrarily, when the dataset size scales to hundreds of millions of images, the authors found how irrelevant the initial inductive bias becomes, and how the superior representational capability of transformers emerges at such scale, even outperforming state-of-the-art convolutional architectures on such large datasets.
As for the network structure, the authors propose its application for Image

Recognition tasks and therefore apply the Encoder-only version of the transformers. To obtain the final embedding, the authors follow the example of BERT [98], and append an initially meaningless token, known as classification token (or CLS), which is learnable and is updated by the encoder layers at each step, to be collected in the end to be used as the global representation of the image. An architectural representation of the described architecture is reported in Fig. 5.5.

Coming back to the **seq2seq** task for VG, the way in which such architecture has been tested is in its version with a 768-dimensional feature embedding, built from 16x16 patches of the original images. It has been used as an alternative to the traditional approaches consisting of a convolutional backbone followed by a specific aggregator such as GeM or NetVLAD, in order to test the representational capacity of transformers to directly encode in the final descriptor a meaningful descriptor without the need for an additional aggregation step.
In order to adapt the network to the multi-frame context, the same techniques used for the baselines of ResNet-18 with GeM or NetVLAD has been used, which are the Descriptor Grouping and Fusion from [71]. Practically speaking, the tested configuration had ViT as a feature extractor, followed by two different multi-frame aggregation techniques: concatenation of descriptors, and usage of a Fully Connected layer, with various sequence lengths.

### 5.4.4 The Timesformer

The two approaches presented up until now are nice applications of the Transformer concept in general. However, coming to the specific task being tackled, they are not ideal as the way in which the sequences are treated is either by extracting single-frame level features (ViT - CAT, ViT - FC) using a transformer, or using a convolutional backbone (ResNet-18 - GeM - Transformer) and then aggregated using a transformer encoder.
It would be interesting to have an approach able to exploit the natural ability of transformers in dealing with sequences directly applied to the sequences present in the task at hand. This same question was addressed by researchers in the field of Action Recognition, which had the same problem of wanting to learn spatio-temporal relationships, after the observation that semantic elements in a sequence of frames can be analyzed in relationship with each other just like words in a sentence, as Transformers for NLP do. From this observation stems the concept for Timesformer, an architecture proposed by

[113] which extends the concept of spatial attention that was introduced by ViT [106], to space-time attention over the three-dimensional volume represented by a sequence of images.
This rather clever idea, which works with a similar approach as the one of ViT, interprets a sequence of frames as a chain of patch embeddings extracted from each image and accompanied with a positional embedding that transforms the original sequence from a series of frames to a chain of tokens each one deriving from a patch of a frame. This idea brings radical modification to the standard transform encoder (again, as the task is related to obtaining a sequence-level descriptor, the transformer is used in its Encoder-only version), which however proves to achieve remarkable results, in many cases outperforming state-of-the-art, convolutional based methods.
One of the intuitions on which the authors base their work is that, much like in the case of Image Recognition, the inductive bias of convolutional layers stops to be useful after dataset sizes scale beyond a certain point, where instead the representational capacity and the non-local contextual weighting of transformers prove their strength. Moreover, dealing with sequences of frames, the locality of convolutions makes it hard to track long-range spatio-temporal dependencies in the 3D volume represented by the video.
However powerful the described idea might be, it includes a similar computational problem to the one that the authors of ViT had to tackle: the original implementation of the self-attention scales quadratically with respect to the number of tokens; if for single images this was solved using patch embeddings, due to the additional dimension introduced by the additional frames, the overall cost becomes quickly overwhelming. To address this issue the authors propose different versions of the concept of spatio-temporal attention, with different strategies to cover the 3D volume. Such versions are discussed below with formulas as reported by the authors in [113].
Therefore in this formalism, each patch in the video is identified by the tuple $(p, t)$, $p \in \{1..N\}$, $t \in \{1..F\}$, N being the number of patches in each frame and F the number of frames. Each of the patches is multiplied, in the process of self-attention, by the weights $\mathbf{Att}_{(p,t)}^{(l,a)} \in \mathcal{R}^{NF+1}$. The +1 in the dimensionality is associated with the additional instance of computation due to the aforementioned CLS token that is used in this architecture much like in [98, 106], to contain the final representation, and it is represented by $\mathbf{Att}_{(0,0)}^{(l,a)}$. The notation $(l, a)$ instead, represents the $a$-th attention head of the $l$-th layer.
In the most basic of the proposed versions, the computation fall back in the spatial-only attention, requiring only N+1 query-key comparisons for each

token, thus removing the temporal axis:

$$\mathbf{Att}_{(p,t)}^{(l,a)space} = Softmax\left(\frac{\mathbf{q}_{p,t}^{(l,a)T}}{\sqrt{d_h}} \cdot [\mathbf{k}_{(0,0)}^{(l,a)}\{\mathbf{k}_{(p',t)}^{(l,a)}\}_{p'=1..F}]\right) \tag{5.10}$$

The previous case is of course not so interesting as it processes images like ViT without considering the temporal axis, and in fact in the results of [113] it performs substantially worse than its counterparts as it totally lacks the concept of temporal dependency. The joint spatiotemporal version, which performs full attention across both axes is the following, and presents quadratic complexity for each of the tokens, accounting to a cubic cost overall:

$$\mathbf{Att}_{(p,t)}^{(l,a)joint} = Softmax\left(\frac{\mathbf{q}_{p,t}^{(l,a)T}}{\sqrt{d_h}} \cdot [\mathbf{k}_{(0,0)}^{(l,a)}\{\mathbf{k}_{(p',t')}^{(l,a)}\}_{p'=1..N,t'=1..F}]\right) \tag{5.11}$$

This is of course not ideal and results in unsustainable requirements both in terms of GPU-memory footprint and computational time as well, provided that it can fit in memory. Therefore the version that also their authors deem as most promising, which is more efficient and provides the best results as well, is what they call "*Divided Space-Time Attention*" [113]. The idea is to still consider full spatial attention, and additionally to consider all patches across the temporal axis, but only in the same spatial position $p$. The two attentions are applied separately, and the formula below represents the temporal step as the spatial one stays unchanged:

$$\mathbf{Att}_{(p,t)}^{(l,a)} = Softmax\left(\frac{\mathbf{q}_{p,t}^{(l,a)T}}{\sqrt{d_h}} \cdot [\mathbf{k}_{(0,0)}^{(l,a)}\{\mathbf{k}_{(p,t')}^{(l,a)}\}_{t'=1..F}]\right) \tag{5.12}$$

Such an approach requires only N + F + 2 query-key comparisons, and therefore if the sequence length is contained in magnitude the overall complexity can be approximately considered the same as the one of ViT. This approach not only prevents the TFLOPs required from exploding like in the case of full joint-attention but remarkably is also the one that achieves the best results in the tests of [113] for the Action Recognition Task.

Fig. 5.6 shows a nice representation of the different kinds of attention, depicting three consecutive frames and highlighting in red all the patches involved in the computation of the context vector for a single patch.

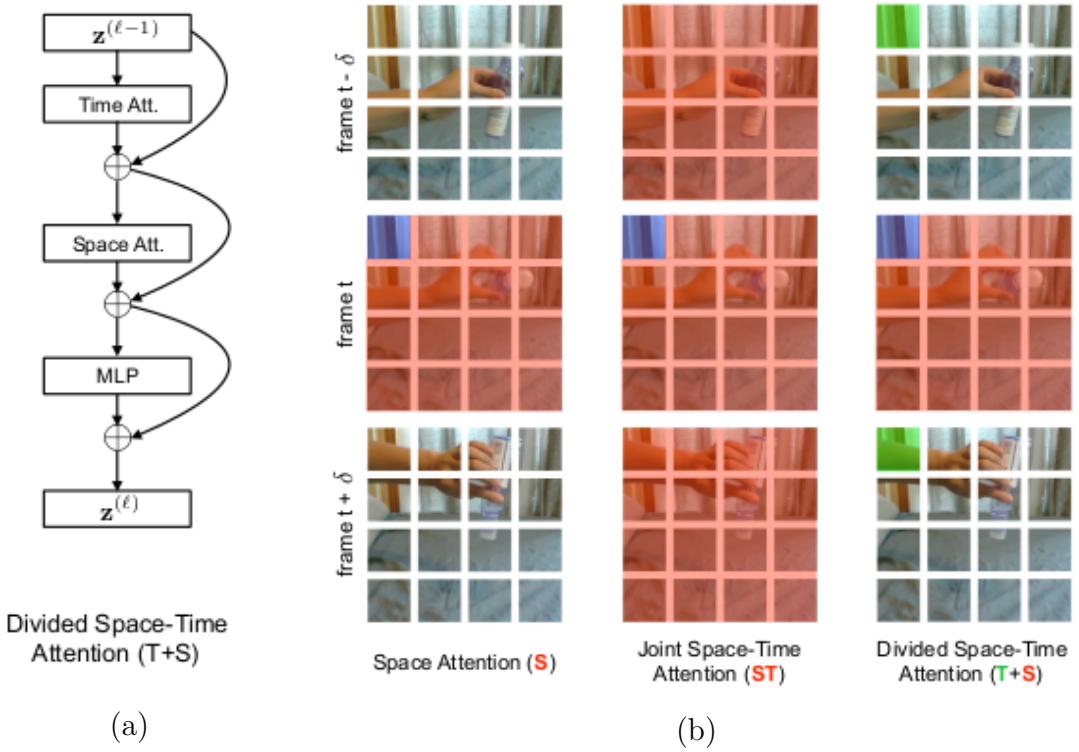



Figure 5.6: **Representation of the self-attention employed by Timesformer** architecture.
(a) Schematics of an encoding block using the divided space-time attention
(b) Representation of which patches of which frames are considered by (left) space-only attention, (middle) full space and time attention, (right) divided time and space attention, which turns out to perform best. Credits to [113].

Specifically for the S-VG task at hand, the advantage provided by such an architecture is to explore the representational capability of transformer extended across the temporal axis, therefore representing an appetible approach. With respect to the methods presented before, this is the only one that allows an all-in-one processing of the sequence, without requiring for separate procedures one of which would be forced to extract features from single-frames, without leveraging contextual information about the temporal evolution of semantic elements across the frames. It needs also to be specified that, even though its optimized version *Divided Space-Time* attention, this architecture still represents the heaviest model considered, and due

to its memory requirements it has been used in its smaller version, taking 224x224 images in order to fit in memory. As already discussed, the training procedure of S-VG tasks is rather demanding in terms of memory and computational requirements due to the presence of triplets which inflate the size of single batches. Such a model outputs a sequence descriptor of size 768, which is a rather compact dimensionality and it can be applied to any input sequence length.
In its proposed version it leverages 12 encoder blocks and it takes 16x16 patches as input. An additional note regards the input embedding, which is performed with a convolutional layer with a 16-sized kernel instead of a Fully Connected layer, in order to partially reduce the already consistent number of parameters. In some experiments it has also been truncated after 8 or 10 blocks in order to test the quality of the extracted descriptors at those layers, utilizing fewer parameters in case it was overfitting.

### 5.4.5 CCT - The compact Transformer

Whereas the main direction of the research trying to apply with good results the transformer architecture to Computer Vision task has mainly been towards large-scaled datasets over the hundreds of millions of images, as it was found that at such scales the inductive bias of convolution starts to lose ground against the superior capacity in tracking dependencies and representational capacity of transformers. The work of [112] goes instead in the opposite direction and poses itself against the belief that the characteristics of transformers cannot be useful in small (relatively small, not huge) datasets. Such a conception, has rightfully pointed out by the authors, can have the negative effect of cutting out from the wave of Transformers and their potentiality a part of the community, if they do not have access to either large data lakes, or the computational resources which are the often neglected aspect of transformers, that require elevated amounts of memory and computational capabilities.
The contribution of [112] consists in providing a new type of Transformer that eliminates the need for large-scale datasets, thanks to a drastic reduction in the number of parameters which can be as low as $0.28M$, whereas, to give a reference, ViT for example in its smaller version contains $86M$ parameters [106]. Moreover, they propose a number of variants addressing many possible use cases. The variant that has been adopted for the task at hand is the Compact Convolutional Transformer, which has a series of modifications that make it a much more flexible architecture, suited to work also with small

datasets (even though this last condition is not the case of this thesis).
In order to reduce the number of parameters, and re-introduce an inductive bias in the early layers, the CCT architecture uses two convolutional layers that are responsible for the tokenization. This modification not only greatly reduces the number of parameters, but the implicit assumptions adopted by the convolutional layer allows to obtain a much less "data-hungry" architecture, that can generalize well even without having seen an enormous amount of data. Moreover, such convolutional encoding is able to preserve spatial local information, and also maintains relationships among patches since the weights of the kernels are shared and therefore semantic elements of the same category in different areas will have the same representation. This design makes optional the use of positional embeddings, which is an engineering choice that often requires a trial-and-error process as it is not clear what kind of encoding might work best in general. Using the CCT architecture, is it possible to omit at all this step with only a minor hit in performance.
Another characteristic of the CCT architecture is given by the introduction of a *SeqPool* [112] layer, that performs a pooling over the tokens at the output of the last encoder block, thus removing the need for the additional CLS token.
Furthermore, another aspect that makes CCT a much more flexible model is given again by the Convolutional block used as a tokenizer; thanks to the properties of convolutional layers, it removes the requirement of having the image size to be a multiple of the patch size as it is the case in ViT. Additionally, another important fact that makes CCT by a long shot the most efficient transformer implementation is that varying the size of the input image does not affect the number of parameters, thanks to the property of convolutional layers; however it does increase the number of tokens, as it is inevitable, and therefore memory and computational requirements do grow together with image size. Fig. 5.7 shows a representation of the described architecture.

Regarding the application of this architecture to the S-VG task at hand, it, unfortunately, does not provide the capability, unlike the Timesformer presented earlier, to process all at once temporal and spatial data. However, as it will be discussed in the following Sec. 5.5, the Timesformer comes at a very high computational cost; therefore alternatives like this one have been explored. In particular, it is interesting how this model combines the most powerful properties of both convolutional and transformer layers in a clever way, that supposedly should be able to achieve when applied on a large-scale

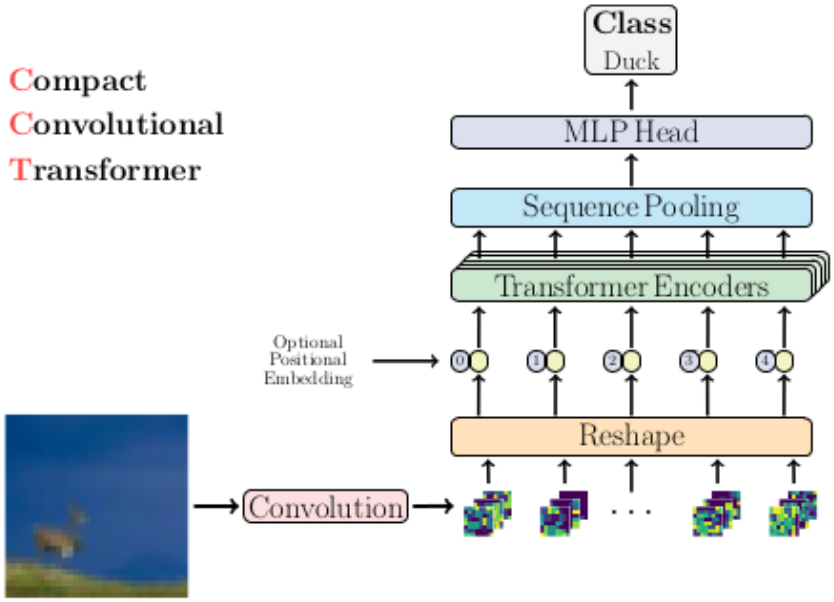



Figure 5.7: Scheme depicting the main concepts of the Compact Convolutional Transformer. It replaces linear layers with convolutional ones to build the starting embedding, greatly reducing the number of parameters, and moreover it also renders optional the use of positional encoding. Credits to [112].

dataset such as MSLS, a superior representational capability with respect to the commonly used ResNet-18. This is thanks to the efficiency of the early convolutional layers in extracting low-level features while preserving spatial relationships, and, subsequently, the high capacity of transformer encoders to track complex and long-range dependencies across the tokens.
For now, this is all just a hypothesis whose validity will be tested in the following Section 5.5; however, for the exposed reasons it will be interesting to combine such an extractor with a specifically tailored aggregation method suited to build a robust sequence-level descriptor, comparing its performances with the one obtainable with a ResNet. More details on such an aggregator can be found in the following Sec. 5.4.7.

### 5.4.6 Non-local blocks

As Transformer-based architectures started to gain popularity after their introduction in 2017 by [84], as already mentioned they generated an interest

thanks to their remarkable results, in the Vision community. Before the introduction of ViT in 2020 [106], many other researchers had been experimenting with the concept of attention in order to insert into convolutional networks the ability to go beyond the local spatial neighborhood that their kernels are able to treat.
An interesting approach inspired by the concept of attention can be found in [92]. The authors note how both convolutional and recurrent networks, due to their architecture design, are forced to perform their computation only on local data (either spatially or temporally local); such characteristic makes it so that in order to analyze dependencies spanning distances longer than the treated neighborhood, they need to apply repeatedly the same operations, which causes many practical problems both for computational cost and for optimization. What makes the self-attention of Transformer innovative from this point of view, is to obtain the weighted contribution of all the data in a single shot.
Reasoning on these concepts, the authors of [92] propose a generic non-local layer in which attention takes place; so instead of having attention as the main operation, performed identically in every layer like Transformers, here the idea is of having dedicated trainable layers with this capability, pluggable anywhere in any kind of network, allowing for a more flexible approach, and possibly a lower computational requirement.
That motivation that makes this contribution interesting are manifold; first of all it introduces into otherwise local-only networks like convolutional, the ability to compute learned contextual weighting taking into account features from all spatial location. This allows to exploit the potentiality of both approaches, as the inductive bias of convolutional kernels is still without a doubt rather useful when it comes to detecting visual features, and with this added layer it becomes possible to build representation accounting for relationships between independently extracted features. Moreover, unlike an FC layer used as an aggregator (which by the way also introduces much more parameters) that uses learned weights specifically associated with each spatial location, the non-linear computation is not computed in function of the data fed to the network, but is rather a response on the relationships detected between the extracted features at different locations. As reported in [92], the formulation for the generic non-local layer is the following:

$$\mathbf{y}_i = \frac{1}{C(x)} \sum_{\forall j} f(x_i, x_j) g(x_j) \tag{5.13}$$

In this general formula there are many degrees of freedom; for example the

function $f$ which needs to be a similarity, and it is traditionally the *softmax* and g can be any unary function used similarly to the **Value** weight matrix in the Transformer.

One of the greatest properties of this block is that its application really has no limit, and it can also be stacked multiple times. Specifically, what makes it interesting for the S-VG task is the fact that its application is not limited to 2D feature maps, but it can be applied even in cases that have an additional temporal dimension. Fig. 5.8 shows an example of such blocks applied to a sequence of images of length T, where each frame has a 1024 channels-wide representation.

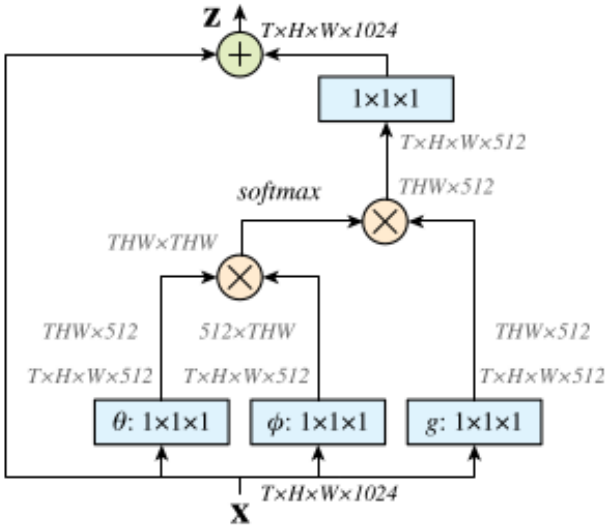



Figure 5.8: Scheme of the computations in a Non-Local block, that allows to include in the embedded features the contribution of distant spatial locations. Credits to [92].

At this point, it should be clear which are the reasons that make the block presented by [92] of interest for the task at hand. Its flexibility allows for a number of different approaches; in particular, it was used in the following ways:

- **2D Non-Local Attention Module**: One possible implementation sees such module plugged on top of any convolutional extractor, computing its contextual re-weighting only based on the spatial features. In this

sense it has been tried on top of ResNet-18 and ResNet-50 as backbones, to analyze whether or not such attention module could augment the robustness of the descriptors extracted by a convolutional backbone. In particular, this pipeline has been tested compared with Transformer-based backbones extractors that already include the self-attention mechanism in all their encoding blocks, like CCT or ViT;

- **3D Spatiotemporal Non-Local Attention Module**: Another interesting approach could be to use the Non-Local block in its 3D version (Fig. 5.8), to be fed with the feature extracted from single frames of the sequences, stacked along the temporal axis. This could be an interesting way of building a more robust sequence-level descriptor, by detecting relationships across spatial features extracted in different frames and therefore resulting in a more informative descriptor;

- **Temporal only Non-Local Attention Module**: Finally, since the high flexibility of such module gives really no limit to its applicability, it has been tested as a temporal-only attention module, evaluating the relationship between the same spatial location, across the time dimension.

### 5.4.7 SeqVLAD

This section describes an aggregation technique called SeqVLAD, which is an original contribution proposed with this thesis. It is basically a modified implementation of NetVLAD, that transforms it into a layer able to receive concatenated features extracted from any arbitrary number of frames, and return a fixed-dimensionality vector that represents the whole sequence. The considerations that brought to the idea for this method stemmed from the fact that all the alternatives analyzed up to this point suffered from two main issues. One being the defect of treating frames individually without a powerful aggregation approach able to combine the single-image-level information, summarizing the relationship of semantic elements across the whole sequence. Indeed, simple aggregation methods like concatenation, or use of FC are not sophisticated enough to grasp the concepts mentioned, and the resulting descriptor will mostly depend on the order with which frames show up, as results in Sec. 5.5.9 will prove. Actually, a method that in theory could have such characteristic was the combination of ResNet-18 + GeM extractor from frames, and the Transformer Encoder described in Sec. 5.4.2 as an aggregator, but unfortunately as the next section 5.5.2 will show such an

approach was not even close to provide meaningful results.
Contrarily, methods that are indeed able to treat the sequence in its entirety, as it is the case of the described Timesformer (Sec. 5.4.4, [113]) end up having heavy computational requirements due to the costly attention mechanism extended through the temporal axis. To give a practical example, training such an architecture with even sequences as short as 3 frames, requires over 12 *Gb* of GPU memory to use a batch size of 2, and in practice, to be able to train over sequences of 5 or more frames, a cluster of GPUs with 32 *Gb* memory is required.
As for the Non-Local blocks presented in the previous section, they do represent a method that is able to create a contextual re-weighting of the features based on their relationships, and it is available both in its spatial-only and spatiotemporal version as well; however applying it on top of any backbone is not sufficient to obtain a complete descriptor and still requires an aggregation method (in practice in the case at hand the Non-Local block is only applicable after convolutional backbones as it would not make sense to try to perform attention in a transformer-based backbone that already employs self-attention).

Therefore, the need for SeqVLAD. The foundation of the idea comes from the following facts. NetVLAD relies on the wisdom of traditional retrieval pipelines, in which it was established that local descriptors, as described in Sec. 2.4.1.1, provide the useful property of invariance to many distractors due to the inductive bias of the process ( see [29, 24, 39]); and finally aggregate the statistics of such local descriptors with tf-idf counts ( [28] ), or in the famous VLAD approach store sum of residuals with respect to a set of centroids that represent *Visual words* [29]. Therefore NetVLAD [39] cleverly implements this same idea using a differentiable convolutional layer with a 1x1 kernel to produce soft cluster assignments, and finally evaluate residuals, and consider the sum of residuals corresponding to each of the channels of the feature maps extracted by the backbone as the final descriptor.
Following this chain of thoughts, a possible approach to adapt this framework to a sequence rather than a single image, would be to use as list of local descriptors to be aggregated in visual words not only the $HxW$ spatial locations across the feature maps obtained from a single frame, but to include as well the feature maps from all the frames. In practice this is implemented by concatenating the feature maps, and evaluating the soft assignment as well as the residual on the whole set of sequence-wide features. A representation of this modification with respect to NetVLAD, that allows the proposed

layer to handle any number of frames is reported in Fig. 5.9 and 5.10, both adapted from the scheme in [39].

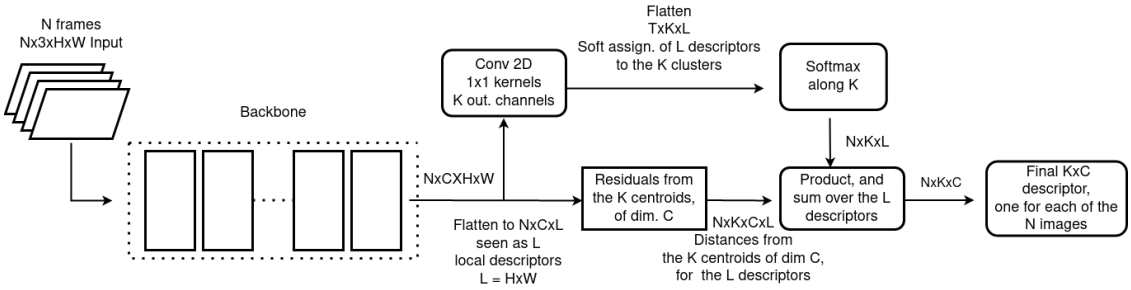



Figure 5.9: Working scheme of **NetVLAD**. When fed with N frames, it outputs N separate representations. Adapted from [39].

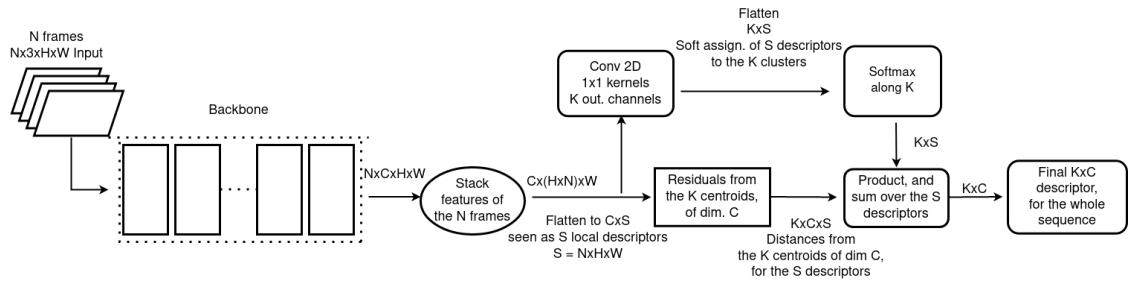



Figure 5.10: Working scheme of the proposed **SeqVLAD** layer. It shows the customization applied to NetVLAD that enables the computation of a single sequence-level descriptor. When fed with N frames, it outputs a single representations. Adapted from [39].

Summarizing the main idea is to exploit the powerful properties of robustness of local descriptors, and instead of computing their statistics on single frames as NetVLAD does, they are evaluated considering the whole sequence. Therefore the whole set of local descriptors extracted from the backbone is used to compute the sum of residuals from each visual word, and the result is a single sequence descriptor, of fixed dimensionality regardless of the length of the input sequence, which is a remarkable property.
The resulting layer has a number of interesting properties, besides the main point that it allows to compute directly sequence-level descriptors while keeping descriptor dimensionality fixed to $KxD$, where $K$ is the number of clusters (i.e. *visual words*) and $D$ the number of channels of the feature maps extracted from the backbone, which is seen as the dimensionality of the series

of local descriptors. It also presents some nice properties that make it the most flexible approach among the ones presented. Firstly, it allows to use a model that was trained on any sequence length for inference on any arbitrary sequence length as well. Results in Sect. 5.5 will show that it is able to generalize to different sequence lengths with good results. This, intuitively, is due to the fact that, given a large enough training set, the learnt visual words and weights for soft-cluster assignment are based on the local descriptors extracted from semantic elements in the frames, therefore if the training set is large enough the model will see a broad variety of domains and semantic categories and will be able to recognize such elements if present in any number of frames. Coming back to the two issues to be addressed exposed in the beginning of this section, SeqVLAD certainly addresses the limitations of CAT and FC aggregation that did not have the ability to organically look at the semantic elements present in the whole sequence, without relying on the ordering of frames. As for the computational issue that affect architectures like the Timesformer, SeqVLAD does not increase the memory requirements of the architectures of which it plugs itself on top, of as all the features of the various frames are already loaded in memory, and with respect to CAT it even makes them lower as it reduces the dimensionality of the final descriptor to a fixed value. Also in terms of computational cost, it is not higher than applying NetVLAD for each frame, which is already lightweight, and it ends up being potentially even lower as now the computation is batched together for the whole sequence.

Such layer has been tested on top of all the feature extractors that are able to treat only single-frames and need an aggregation step, those being the ResNet-18 and ResNet-50, tested both in order to evaluate the impact of the different number of channels on the final result. It has also been tested the use of Non-Local blocks on top of a convolutional backbone to augment the representational capacity of such network by taking into account contextual information.
As for the transformer-based backbone, it has been tested on CCT rather than ViT because of the greater flexibility and lower requirements of the former that allowed to perform a lot more experiments than ViT would have, considering how computational resources are never easy to acquire.

## 5.5 Experiments

This final section of this chapter is devoted to analyzing the performances of all the methods described in Sec. 5.4, highlighting strength and weaknesses of each of them while trying to relate the obtained results to the specific characteristics and property that were discussed when presenting each of the approaches. The discussion starts from the adopted Baselines; and continues using the vanilla Transformer-Encoder as a sequence-level aggregator on the single-frame features. It then moves on to explore transformer-based backbones, different attention schemes and also the application of the proposed SeqVLAD layer. Finally, a discussion on the generalization capabilities of models trained on single image is presented, as well as the ability of models to still obtain robust features when presented with reverse-ordered frames, or shuffled, and the impact of using a lower number of negative examples for the triplet loss.

### 5.5.1 Baselines

As detailed in Sec.5.4.1, the most representative architectures to be used as baselines are constituted by a ResNet-18 backbone as feature extractor, either GeM or NetVLAD as frame-level aggregation, and finally CAT or FC to obtain the final sequence-level descriptor. Results are reported in Tab.5.1 using $3, 5, 7$ as sequences length, since as explained in the objectives section 5.2 such short-sequences allow to keep the concept of place-specific localization, and of improving the results obtainable using single frames.

Looking at the table, it appears that the performance gap between GeM and NetVLAD is comparable to the one found in the single-image case, and that however the sheer recall figure is higher on sequences than it on single images as reported in Tab. 4.3, even though obviously such numbers are not directly comparable. The results also show two additional facts: the basic technique of CAT aggregation seems to be more effective than to use a FC layer. This consideration, paired with the fact that increasing the dimensionality of the FC-layer increases the recall figure, brings immediately to understand how the final dimensionality plays an important role, and that simple techniques like FC layers are not sophisticated enough to capture a meaningful representation that is representative of the whole sequence. It is noticeable infact how using FC-512 on top of NetVLAD considerably worsens the results, due to the incapability of such a simple technique to condense in a compact 512 descriptor a meaningful representation. Even with a bigger

Table 5.1: **Baseline results** obtained using ResNet-18 + NetVLAD/GeM as single frame descriptor extractor, and CAT or FC layer as aggregators over the sequence frames.
All backbones pretrained on ImageNet, N. neg 10, Input images size 480x640.

| Backbone | Feature Dim | Pooling | Aggregator | train/test SeqLen | R@1 |
|---|---|---|---|---|---|
| r18l3 | 768 | GEM | CAT | 3 | 73.6 |
| r18l3 | 1280 | GEM | CAT | 5 | 74.5 |
| r18l3 | 1792 | GEM | CAT | 7 | 75.2 |
| r18l3 | 512 | GEM | FC-512 | 3 | 71.6 |
| r18l3 | 512 | GEM | FC-512 | 5 | 71.8 |
| r18l3 | 512 | GEM | FC-512 | 7 | 72.9 |
| r18l3 | 2048 | GEM | FC-2048 | 3 | 73.9 |
| r18l3 | 2048 | GEM | FC-2048 | 5 | 74.4 |
| r18l3 | 2048 | GEM | FC-2048 | 7 | 74.3 |
| r18l3 | 49152 | NetVLAD | CAT | 3 | 81.0 |
| r18l3 | 81920 | NetVLAD | CAT | 5 | 80.8 |
| r18l3 | 114688 | NetVLAD | CAT | 7 | 82.2 |
| r18l3 | 512 | NetVLAD | FC-512 | 3 | 68.9 |
| r18l3 | 512 | NetVLAD | FC-512 | 5 | 70.1 |
| r18l3 | 512 | NetVLAD | FC-512 | 7 | 66.8 |
| r18l3 | 2048 | NetVLAD | FC-2048 | 3 | 68.2 |
| r18l3 | 2048 | NetVLAD | FC-2048 | 5 | 66.9 |
| r18l3 | 2048 | NetVLAD | FC-2048 | 7 | |

FC layer, FC-2048, results are much lower than with the simple CAT, and it needs also to be pointed out that using a FC-2048 layer on top of the huge descriptors obtained concatenating the single-frame features, yields a layer with more or less (varying the number of frames) $1M$ parameters! This fact make it evident how there is the need for specialized techniques to deal with sequences. Due to the huge number of parameters deriving from the use of FC-2048 aggregation, it has not been tested on Seq7, with NetVLAD.
In the case of GeM, where the CAT and FC representation have similar dimensionality, it can be seen that results are also quite close, but also sub-optimal.

Summarizing, the point is that if the chosen aggregation method does not have the capability to compute a meaningful representation that accounts for the semantic elements present in the various frames, the result is a dimensionally inefficient method that requires very large descriptor spaces (see NetVLAD-CAT) to improve results. Moreover, as regards the methods presented in this table they are likely to be relying mostly on the ordering of the frames and their feature content, as the aggregation is naturally built with this concept and how results in Sec. 5.5.9 will show.
An additional consideration can be drawn with respect to the analysis carried out during the benchmarking work in Sec. 4.2.5, in a real-life situation in which a VG system of this kind is deployed, a fundamental factor for time efficiency is constituted by the descriptors dimensionality; such considerations are even more important in systems that need to operate in real-time. Therefore the picture that this table paints, using this baseline methods, is of pipelines that either have satisfactory results with unsustainably large descriptors (NetVLAD-CAT), or sub-optimal results (GeM-CAT, FC).

### 5.5.2 Transformer Encoder

Transformer Encoder, aka: a not so exciting starting point...
After the evaluation of the results obtainable with baseline methods, it is clear that there is the need for more specifically tailored methods, in order to provide more efficient and accurate approaches for the task. The first idea to be implemented, was to use a vanilla Trasformer Encoder, following the principles explained in Sec. 5.4.2.
Basically the resulting pipeline would use a ResNet-18 and a GeM aggregator to generate 256-dimensional tokens, one for each frame of the input sequence, for the encoder layers to process and output the final sequence descriptor using the artificially appended CLS-token, thus keeping the fixed dimensionality of 256 thanks to the tranformer properties. Tab. 5.2 shows the obtained results on sequences of lengths 3,5,7 for the same reasons explained in the baselines section.

This pipeline could have had the advantages of keeping a very low feature dimensionality, while relying as well on a lightweight backbone, and also the ability to leverage contextual information from the whole sequence thanks to the attention mechanism of transformers operating on the local descriptors in each token extracted by the backbone.
However, the experimental reality shows that it just does not work, at all. The reasons for such disappointing results can be understood in light of a few

Table 5.2: Results using a vanilla **Transformer Encoder** as aggregator on the concatenation of features extracted on single frames by a Resnet-18 + GeM pipeline.
Backbones pre-trained on ImageNet. N. neg 10, Feature Dimensionality 256 in all cases.

| Backbone | Pooling | Aggregator | Num. Encoder blocks | train/test SeqLen | R@1 |
|---|---|---|---|---|---|
| r18l3 | GeM | Transf. Encoder | 3 | 3 | 18.8 |
| r18l3 | GeM | Transf. Encoder | 3 | 5 | 18.1 |
| r18l3 | GeM | Transf. Encoder | 6 | 5 | 16.8 |
| r18l3 | GeM | Transf. Encoder | 8 | 5 | 18.3 |
| r18l3 | GeM | Transf. Encoder | 3 | 7 | 18.0 |

facts regarding the experimental setup, the main one being the rather limited number of tokens available. In fact, for example Vision Transformers like ViT [106] or CCT [112] use 196 token for 224x224 input images, a number that goes up to 596 if the input size is 384. Even in NLP applications, usually the order of magnitude of the number of used tokens is of the hundreds. Therefore a possible explanation could be that the attention mechanism is not able to infer enough contextual information from such a limited number of tokens (up to 7).
To try and work around this issue many variants have been tried, like by-passing the pooling layer so to have the $256xHxW$ feature maps and use them as $HxW$ 256-dimensional tokens (therefore in the same way that NetVLAD treats them), so to consistently increase the number of tokens available, but still results turned out to be even worse than the one in Tab. 5.2.
Another possible issue could be the fact that 256-sized tokens could be too small, as ViT [106] for example operates on 768-dimensional spaces and CCT [112] uses 384, even though the dimensional difference does not seem to be so large to justify the results.
Finally, one of the most likely explanations is that using almost a complete ResNet-18 (it is truncated at *conv4*) injects too much of its inductive bias in the already high-level features that can be found at that point, and training from scratch a Transformer encoder starting from tokens constructed in this fashion yields a difficult optimization procedure that hardly converges to an accurate model to the task.
Therefore this results have been reported for the sake of completeness, and

the lesson learnt has been exploited to develop the alternatives that are subsequently presented.

### 5.5.3 ViT

This section explores the use of the most popular Vision Transformer as a backbone to process single frames. The architecture in question is ViT ( see Sec. 5.4.3, [106]) in its base version operating on input images downsized to $224x224$ across 12 stacked Encoder layers. The choice of this version of the network was due to its memory requirements, as in fact (this is true for any vision transformer) it grows quadratically with the input size divided by patch size ($16x16$). Its token dimensionality, that coincides due to its architectural properties with the output descriptor size, is 768. It has been tested with the same aggregation methods as the baselines, CAT and FC layer, and results are reported in Tab. 5.3.

Table 5.3: Results using **ViT as a feature extractor** on single frames, and CAT or FC layer as aggregators over the sequence frames.
ViT pretrained on ImageNet, N. neg 10, Input images resized to 224x224.

| Backbone | Feature Dim | Aggregator | train/val/test SeqLen | R@1 |
|---|---|---|---|---|
| ViT | 2304 | CAT | 3 | 76.0 |
| ViT | 3840 | CAT | 5 | 79.2 |
| ViT | 5376 | CAT | 7 | 76.5 |
| ViT | 2048 | FC-2048 | 5 | 79.6 |
| ViT | 2048 | FC-2048 | 7 | 74.5 |
| ViT | 2048 | FC-2048 | 15 | 80.9 |

The usage of the ViT architecture, with respect to a standard convolutional backbone, should provide a few advantages. Firstly, this transformer-based architecture is able to provide a much more compact descriptor, eliminating the need for a pooling step like GeM or NetVLAD, allowing to learn a meaningful representation in an end-to-end fashion.
Moreover, lacking the inductive bias of convolutional networks, it could in principle, exploiting the larger capacity given by the higher number of parameters (86 $M$ vs. 11 $M$ for ResNet-18) and the ability to consider contextual information, allow this model to extract more representative descriptors.

Of course there are also downsides, and they reside mainly in the higher computational and memory requirements. As Tab. 5.8 will show, the higher computational time required to extract features is mitigated by the fact that the compact descriptor that it is able to provide yield very low retrieval time; the main issue remains the memory requirement, which depending on the hardware available, can determine the usability of this architecture.

This being said, looking at the results, they more or less reflect the expectations even though there is not a consistent gain with respect to baselines. Overall, results in terms of the mere recall figure are comparable with the ones obtained by NetVLAD-CAT, however if the feature dimensionality is added to the equation it is clear that ViT has a better recall-to-dimensionality ratio, meaning that it is able to provide better results with smaller descriptors. As regards the use of FC aggregation, it outperforms all the baselines, (except, weirdly, for Seq 7 in which it is equivalent to baselines) with satisfactory results. Unlike previous cases, it has been tested also on longer sequences, up to 15 frames, only with FC and not with CAT aggregation as to try to keep a reasonable feature dimensionality. The ratio of this experiments was to test the ability of such pipeline to handle longer sequences, and it seems like it is the case. However it needs to be specified that in practice it may not be of interest to use such long sequences, and also that it drives up the training time quite a lot, as well as evaluation time.
Another consideration is that this approach shares with the presented baselines, the defect of lacking a more sophisticated aggregation approach, as the techniques of CAT and FC suffer of the issues described for the baselines in Sec. 5.5.1. A possible path for further experiments could be to try and truncate the number of used encoder layers, or to freeze some of the first ones.

### 5.5.4 Timesformer

Methods presented up to this point have lacked the presence of a sophisticated sequence-wide evaluation of the semantic elements present in the various frames. Either due to the simplicity of the aggregation (CAT, FC), or in the case when the aggregation should in principle have had this characteristic, it did not work in practice ( vanilla transformer encoder). This section discusses the performances obtained using the Timesformer [113] architecture, which is capable to exploit contextual information from all the frames in a sequence, as well as spatial information in single frames. This

is achieved thanks to the *Divided Space-Time Attention* proposed by [113] and explained in Sec. 5.4.4. The latter is also the only multi-dimensional attention version among the many proposed by [113] that has been tested, since even though it is the least computationally demanding, its requirements both for memory and computation are substantial.
Like in the case of ViT, it has been fed with images resized to $224x224$; it is also made up of 12 space-time attentive encoder layers, and it too has a token/output dimensionality of 768. Even though its number of layers and dimensionality is the same as ViT, due to the additional attentive mechanism over the temporal axis, its number of parameters rises to 121 $M$, whereas ViT as mentioned only contains 86 $M$, making it the model with the highest capacity tested throughout this work. The model is so heavy that it even switched from the FC layers used by ViT to tokenize the image to a single convolutional layer with $14x14$ kernels, in order to reduce the number of parameters in this initial stage.
It has been tested both in its full version, and truncated after 8 or 10 encoder layers. Also, as the authors of [113] released a pre-trained version on some popular Action Recognition datasets (Kinetics400, Kinetics600) suche pre-trained models have been used as a starting point. Moreover, given the huge representational capacity of this model, it has been tried with a higher margin on the triplet loss, which is equivalent to ask that queries and its negative examples be pushed further away.
Tab. 5.4 displays the obtained results.

The main selling point, for this approach, is the fact that it is able to provide directly the sequence-level descriptor, without needing any pooling or aggregation steps. It provides a compact 768-dimensional descriptor, and its self-attention mechanism allows it to obtain a final representation that accounts fro contextual information extracted from the whole sequence; as the attention score for each token is evaluated considering all the other token from the same frame, and, additionally also all the tokens from the rest of the frames, from the same spatial position, thus accounting for the evolution of semantic elements in the images across the frames.
Looking at the results, it outperforms all of the approaches presented up until now. Only in the case of sequences of length 3, the sheer number is below the figure obtained with NetVLAD-CAT by 1%, however it is able to do so with a descriptor that is around 50 times smaller, which shows the remarkable capacity of this models and proves that the compact descriptors obtained are quite rich of meaningful information.
In terms of the layer structure, it seems that leaving the last 2 layers out of

Table 5.4: Results using **Timesformer** architecture with *Divided Space-Time attention* [113] which is able to compute directly a sequence descriptor from all the frames without the need for an additional aggregation step.
N. Neg: 10, Input images resized to 224x224. Feature Dimensionality 768 in all cases.

| Architecture | Truncated at block | Pretrain | train/test SeqLen | R@1 | Margin |
|---|---|---|---|---|---|
| Timesformer | - | - | 3 | 78.9 | 0.1 |
| Timesformer | 10 | - | 3 | 79.7 | 0.1 |
| Timesformer | - | - | 5 | 81.2 | 0.1 |
| Timesformer | - | - | 7 | 82.0 | 0.1 |
| Timesformer | - | - | 8 | 83.7 | 0.1 |
| Timesformer | - | - | 8 | 82.7 | 0.5 |
| Timesformer | 8 | - | 8 | 81.5 | 0.1 |
| Timesformer | 10 | - | 8 | 85.2 | 0.1 |
| Timesformer | - | K400 | 8 | 82.8 | 0.1 |
| Timesformer | - | K400 | 8 | 83.8 | 0.5 |
| Timesformer | - | K600 | 8 | 80.6 | 0.1 |
| Timesformer | - | K600 | 8 | 83.5 | 0.5 |

the equation improves performances, as it happens also with convolutional backbones, the retrieval task often benefits from less refined feature representations.
Another consideration can be done on the relationship of the margin on the triplet loss and the pre-training. It seems infact that only pre-trained versions of the model benefit from a higher margin for metric learning; and this can be understood by thinking that when training from scratch, asking the network to push further away negative examples can lead to too big optimization steps, whereas the pre-trained version that can benefit from a better starting point in terms of sequence representation extracted, and can therefore better manage the higher margin requested.
Overall, this approach yields satisfactory results outperforming the ones presented up to this point; and it does so by exploiting temporal information and not only single-frame processed features. However it comes with an elevated computational cost, especially in terms of GPU-memory required, and

therefore the following methods will be more focused on efficiency.

### 5.5.5 Non-Local blocks

This section focuses on the applicability of Non-Local blocks to augment the ability of convolutional backbones to take into account contextual information from both the entirety of the frame, and other frames as well. The ratio behind the choice of this path at this point, is the following. The first experimental section have showed that models that process separately each frame, having then to rely on too simplistic aggregations like CAT or FC do not perform very well and do not exploit contextual information from the whole sequence. On the other hand, methods that do exploit such more complete information, like Timesformer, end up having a very high computational cost.
Therefore the concept motivating the architectures presented in this section and the following, is to find an efficient way to exploit contextual, sequence-wide information, while also keeping the descriptor dimensionality contained. The idea of Non-Local blocks, described in Sec. 5.4.6, greatly addresses the first purpose, as it provides with high flexibility a low-cost method to re-weight the feature representation based on the relationship of the extracted features from the rest of the sequence. It even allows to regulate its computational cost by increasing or reducing the channels bottleneck on which the attention contextual vector is computed. However, even adding such a layer there still remains the need of a proper aggregation method, more sophisticated than CAT or FC, able to deal with multi-frame data.
This are the circumstances in which the idea for SeqVLAD stemmed. As described in more details in Sec. 5.4.7, this modified version of NetVLAD provides a solution to the exposed problem, allowing to compute a fixed-dimensional representation of the sequence obtained by putting in relationship the sequence-wide features with a clustering of *visual words*.
Therefore on this two pillars stands a new pipeline, made up of a convolutional backbone, augmented with a Non-Local block providing a 2D or 3D attention, or even separately 2D + temporal. Finally, SeqVLAD takes the feature maps to output the final descriptor with the same dimensionality that NetVLAD would yield, 16384 in the case of a Resnet-18 as a backbone. Tab. 5.5 reports this results, varying as well the channels bottleneck and the residual weight given to the attentive mask.

The table reports, for sequences of length 3 and 5, first the result using the backbone + SeqVLAD without the Non-Local block, and then the different

Table 5.5: Results using **Non-Local blocks** on top of a ResNet-18 backbones, concluding the pipeline with the proposed SeqVLAD aggregator. The Non-Local block has been tested in 3 different versions: spatial, spatial + temporal separately, and 3D.
All backbones pretrained on ImageNet, N. neg 10, Input images size 480x640, and Feature Dimensionality is 16384 in all cases.

| Backbone | Channels bottlenek | Attention residual weight | Attention type | Aggregator | train/test SeqLen | R@1 |
|---|---|---|---|---|---|---|
| r18l3 | — | — | — | SeqVLAD | 3 | 82.6 |
| | 32 | 1 | 2D | | | 83.2 |
| | 64 | | | | | 82.7 |
| | 128 | | | | | 82.6 |
| | 256 | | | | | 82.3 |
| r18l3 | 32 | 1 | 2D + T | SeqVLAD | 3 | 82.7 |
| | 64 | 1 | | | | 82.0 |
| r18l3 | 32 | 1 | 3D | SeqVLAD | 3 | 83.5 |
| | 64 | | | | | 82.9 |
| | 128 | | | | | 82.6 |
| | 256 | | | | | 81.8 |
| r18l3 | 32 | 2 | 3D | SeqVLAD | 3 | 81.8 |
| | 64 | | | | | 81.4 |
| | 128 | | | | | 80.9 |
| | 256 | | | | | 80.5 |
| r18l3 | — | — | — | SeqVLAD | 5 | 85.2 |
| | 32 | 1 | 2D | | | 84.9 |
| | 64 | | | | | 84.3 |
| | 128 | | | | | 83.6 |

versions varying the bottleneck and attention type. The first fact that can be noticed is how using a higher weight for the attention mask (2), yields the worst results across the board.
Another consideration that can be drawn out is that using a lower number of channels to compute the attention mask yields better results than keeping the original number of channels (256), and in fact it seems that both with 2D and 3D attention there is an inverse proportionality between the recall figure and the number of channels. Using the combination of 2D and temporal attention does not seem to improve results with respect to the non-attentive version.
Contrarily, both the 3D and the 2D versions do provide a slight improvement,

of respectively 0.9 % and 0.6 %, on Seq3. The important point, however, is that such a method outperforms of about 3 % both its base version of NetVLAD CAT, which has 3 times bigger descriptors, and the Timesformer as well, who has smaller descriptor, but a much more costly model. A more precise evaluation of the inference time will be provided in the following sections.
As regards results using 5 as number of frames, there are unfortunately not enough results, due to time and computational resources constraints, and it seems that here the 2D attention does not improve results; however it might be that 3D attention or more channels could be useful. An important point is the 6 % boost with respect to the base version of NetVLAD CAT, which in this case has 5 times bigger descriptors.

Summarizing, the pipeline proposed in this section has proven to achieve better results than the ones presented before, while also keeping low the computational cost. However, the Non-Local block does not seem to provide a significant boost, and therefore in the next section a different backbone will be tested to try to implement differently the attention mechanism.

### 5.5.6 CCT

From the analysis carried out up to this point, after having verified the satisfactory results achievable with the Timesformer architecture, the focus has shifted towards finding a less demanding approach that was still able to exploit useful information from all the available frames. Testing the proposed SeqVLAD layer on the ResNet-18, as it was done in the previous chapter, turned out to improve results with respect to any other method presented; however the use of a Non-Local block did not provide a significant performance boost. A possible explanation for this fact is that the attention performed by the Non-Local layer is applied only on top of the backbone, therefore on a set of already higher level features. It could of course be applied at any point of the convolutional network, but to find an optimum would require an exhaustive set of experiments on the number of Non-Local blocks to place, their position, the channel bottleneck... Since time and computational resources are always limited, a different approach is hereby proposed.
As pointed out in Sec. 5.5.6 describing the CCT architecture, it is among the less computationally demanding Vision Transformer in the literature, with its many variants, and it was specifically the purpose of the authors in [112] to provide a Transformer-based Vision-oriented architecture with

significantly less parameters in order to make the use of such architectures more computationally accessible to everyone. It is by far the less demanding among the approaches tested during the work for this thesis. The version used is among the bigger ones, that uses 2 Convolutional layers with $7x7$ kernels as a learnable tokenizer, obtaining two advantages. Firstly, a rather lightweight (parameter-wise) model, as the two convolutional layers yield tokens of dimensionality 384 (half as much as ViT and Timesformer). The second advantage and this approach has also the advantage of exploiting the inductive bias of convolutional layers to extract the low-level features, which are the ones for which such bias is the most suited to.
Moreover this tokenization approach reduces by far the amount of data needed for the model optimization to converge to a good approximation of the data distribution.

As for the rest of the architecture, the Transformer Encoder layer can rely on the robust low-level features extracted by the convolutional tokenizer, and from there onwards the superior representational capability of the Transformer Encoder can do the rest, and this is the point of interest for the task at hand. In particular, the argument is that thanks to the combination of properties described, using CCT as a backbone instead of a ResNet-18 can provide a richer descriptor, thanks to the higher capacity of the model, combined with the lack of inductive bias and the self-attention mechanism. The hypothesis is that these two facts together can bring to have at the output of the CCT encoder layers, token whose features are able to capture semantic elements and more importantly their relationship inside the whole image in a way that a CNN may not be able to reproduce.
Tab.5.6 reports the results of the experiments with the described backbone, that will tell whether or not this hypothesis can be true. This backbone has been combined only with SeqVLAD layer, as it has proved to be the most effective one to use in combination with backbones that work on single frames. The way in which it is used is to by-pass the SeqPool layer introduced by [112] (see Sec. 5.5.6, and to get the whole set of $Tx384$ tokens for each of the $N$ frames. Then in order to feed them to SeqVLAD, coherently with the interpretation of the features that characterizes SeqVLAD, the tokens are reshaped $384x(NxT)$. Therefore the values of all token in a particular position are seen as a local descriptor, and the local descriptors obtained in this fashion for each frame are stacked along the frames. This formulation makes the interpretation identical to the one adopted with convolutional backbones. Technically, the only difference is that in this case the convolutional kernel

used to compute the soft-cluster assignment a 1-D kernel is used instead of a 2-D, however mathematically the result is the same as this is only due to the fact that with a convolutional backbone the feature maps were not flattened and were kept to $(NxH)xW$. The different configurations tested vary in terms of the sequence length, and as well in the number of layers used (originally it is made up of 12 layers) and the number of frozen ones.

Table 5.6: Results using **CCT transformer** as a backbone, followed by the proposed SeqVLAD aggregator; test truncating and freezing the encoders block of CCT at different points.
Backbones all pretrained on ImageNet. N neg 10, Feature Dimensionality 24576 in all cases. CCT224 means that it is receiving images resized to 224x224. The * for CCY384 means that it has been trained with 6 negatives per triplet, due to memory constraints.

| Backbone | Aggregator | Truncated at block | Freeze up to block | train/test SeqLen | R@1 |
|---|---|---|---|---|---|
| CCT224 | SeqVLAD | 10 | _ | 3 | 83.9 |
| CCT224 | SeqVLAD | 10 | Conv block | 3 | 83.7 |
| CCT224 | SeqVLAD | 10 | 2 | 3 | 82.5 |
| CCT224 | SeqVLAD | 8 | _ | 3 | 83.5 |
| CCT224 | SeqVLAD | 8 | Conv block | 3 | 84.7 |
| CCT224 | SeqVLAD | 8 | 2 | 3 | 84.6 |
| CCT224 | SeqVLAD | 6 | _ | 3 | 81.4 |
| CCT384 * | SeqVLAD | 10 | Conv block | 3 | 86.4 |
| CCT384 * | SeqVLAD | 8 | Conv block | 3 | 87.9 |
| CCT224 | SeqVLAD | 10 | _ | 5 | 85.1 |
| CCT224 | SeqVLAD | 10 | Conv block | 5 | 86 |
| CCT224 | SeqVLAD | 8 | _ | 5 | 85.6 |
| CCT224 | SeqVLAD | 10 | _ | 15 | 89.6 |
| CCT224 | SeqVLAD | 8 | _ | 15 | 91.2 |

Results show how it is beneficial to truncate the network as to rely on less-refined descriptors, and also how freezing at least the convolutional tokenizer pre-trained on ImageNet is almost always convenient, understandably.
The presented results are all obtained with input images resized to 224x224, which is less than 50% of the original one, and nevertheless recall is 2% higher with respect to a ResNet-18 + SeqVLAD with a sequence length of 3, and 1

% higher with sequences of 5 frames. With images truncated to $224x224$, this model is so efficient that it is actually faster than a ResNet-18 in computing the single-frames descriptor; unfortunately, the fact the final dimensionality is higher (24576 vs 16384) due to the number of local descriptor being 384 instead of 256, makes it so that retrieval is faster with the ResNet-18. A dedicated section evaluating inference time using the same framework as the one presented in Sec. 4.2.5 will be discussed later on. Using a higher input size of the images, identified in the table as 'CCT384', the memory footprint due to the higher number of tokens increases substantially, and due to the temporarily unavailable high-memory hardware used for ViT and Timesformer, it has been trained with 6 negatives only. However, results show that the reducing too much the input resolution can indeed cause hits in performances, as the CCT384 backbone achieves a 3% boost.
This fact has brought to consider the cost of training, and to explore more lightweight procedures, that the next section will discuss.

### 5.5.7 On the cost of multi-frame training

Having discussed a set of different alternatives, with a high variability in terms of computational requirements, this section has the purpose of clarifying what are the relationships among the presented methods in terms of both computational and GPU-memory requirements. Tab. 5.7 and Tab. 5.8 report such requirements for sequences of length respectively 3 and 7. It reports for each of the evaluated methods, feature dimensionality, inference time and memory requirements. The value of inference time is averaged over the forward of a 1000 queries, in milliseconds, and it is more related to the performances of a deployed model, as it does not include the time for gradients update. However the embedding time does also provide a measure of how costly it is to use a certain model to compute features, and paired with the number of parameters it is somewhat related to the training time, which was not reported as the models were trained on different machines and it would not therefore constitute a fair comparison. The inference times were all evaluated using a machine with a GTX Titan and an i7-5930K. A more detailed explanation on why the inference time is divided in embedding + retrieval time, and in which situations it matters the most, refer to Sec. 4.2.5.

From the table, a number of important considerations can be drawn. First of all, in terms of inference time, the table reports results evaluated with a database size of $10^5$. Such a figure is not at all unrealistic, and in all applications that aim at large-scale VG is a rather standard setting, and in can

Table 5.7: Table reporting **Cost vs Recall using a sequence length of 3** for the main methods analyzed. For retrieval, a database size of $10^5$ has been considered. The times are in milliseconds, reported as a per-query average over 1000, and same goes for embedding time. The memory requirements indicates the minimum GPU memory required in training to forward a single triplet through the model.

| Architecture | Feature Dim | Embed. Time [ms] | Retrieval Time [ms] | Inference Time [ms] | R@1 | GPU Memory per triplet |
|---|---|---|---|---|---|---|
| r18l3 - GeM - CAT | 768 | 12 | 2 | 15 | 73.6 | 3.0 Gb |
| r18l3 - NetVLAD - CAT | 49152 | 20 | 145 | 164 | 81 | 3.8 Gb |
| r18l3 - SeqVLAD | 16384 | 19 | 48 | 67 | 82.1 | 4.0 Gb |
| r18l3 - Attn 2D - SeqVLAD | 16384 | 23 | 48 | 71 | 83.2 | 4.3 Gb |
| r50l3 - SeqVLAD | 65536 | 72 | 191 | 262 | 83.5 | 13.2 Gb |
| ViT-224 - CAT | 2304 | 31 | 7 | 38 | 76 | 8.3 Gb |
| Timesformer-224 | 768 | 48 | 2 | 50 | 79.7 | 11 Gb |
| CCT224 - SeqVLAD | 24576 | 11 | 72 | 83 | 86.8 | 3.2 Gb |
| CCT384 - SeqVLAD | 24576 | 35 | 72 | 107 | _ | 12.0 Gb |

Table 5.8: Table reporting **Cost vs Recall using a sequence length of 7** for the main methods analyzed. For retrieval, a database size of $10^5$ has been considered. The times are in milliseconds, reported as a per-query average over 1000, and same goes for embedding time. The memory requirements indicates the minimum GPU memory required in training to forward a single triplet through the model.

| Architecture | Feature Dim | Embed. Time [ms] | Retrieval Time [ms] | Inference Time [ms] | R@1 | GPU Memory per triplet |
|---|---|---|---|---|---|---|
| r18l3 - GeM - CAT | 1792 | 29 | 5 | 35 | 75.2 | 6.0 Gb |
| r18l3 - NetVLAD - CAT | 114688 | 46 | 339 | 385 | 82.2 | 8.4 Gb |
| r18l3 - SeqVLAD | 16384 | 45 | 48 | 93 | _ | 8.6 Gb |
| r18l3 - Attn 2D - SeqVLAD | 16384 | 54 | 48 | 101 | _ | 10 Gb |
| r50l3 - SeqVLAD | 65536 | 168 | 191 | 359 | _ | 28 Gb |
| ViT-224 - CAT | 5376 | 71 | 15 | 86 | 76.5 | 16 Gb |
| Timesformer-224 | 768 | 108 | 2 | 110 | 82 | 22 Gb |
| CCT224 - SeqVLAD | 24576 | 25 | 72 | 98 | _ | 7.0 Gb |
| CCT384 - SeqVLAD | 24576 | 80 | 72 | 153 | _ | 24 Gb |

even be higher in some cases. Of course in applications which rely on smaller databases, the retrieval time becomes no longer a bottleneck and it should not be considered as a relevant issue (refer again 4.2.5 for a more detailed evaluation varying database size).
Coming back to the results, it is clear how the ability of the different architectures of keeping compact the descriptors dimensionality is fundamental to

obtaining a system without long delays at inference time. It is remarkable, in fact, how the model with the heaviest embedding cost, that is the Timesformer, ends up having a very similar total inference time to a ResNet-18 - SeqVLAD, and the same goes for the use of ViT. It is also noticeable how the CCT architecture, when fed with $224x224$ images, mantains the expectation of providing a lightweight architecture, whose embedding time is around 40% lower even than a ResNet-18! However, due to the higher dimensionality obtained when combining CCT224 and SeqVLAD, the inference time ends up being comparable with the ResNet-18 - SeqVLAD. All aspects considered, CCT224 - SeqVLAD remains the better alternative considering that it has as well the better recall scores, and lower memory requirements as well.

While inference times are for sure an important aspect to evaluate, as it is the one that matters when the model is actually deployed, it should not be underestimated the footprint in terms of GPU memory that is required to train a model, as the memory available on the hardware at hand can often turn out to be the actual bottleneck. In fact, perhaps one the most important consideration to be drawn from this table lies in the relevance of memory requirements, that especially Tab.5.8 shows how as soon as the sequence length increases to as low as 7 frames, the memory required can pretty much explode, for methods like Timesformer, ViT and CCT384, that in fact were not trained on such sequence lengths due to this high memory requirements (note that also the r50l3, mainly due to the input images kept at 480x640, and the high feature dimensionality, yields a rather unfeasible option). The difference between computational requirements and memory footprint, is that one might be able to overlook the fact that a training procedure lasts longer, as long as it converges in reasonable time, as it is after all a one-time procedure; whereas a memory footprint higher than the one available on the hardware at hand makes it simply impossible to train a model. Remember also that the table reports the memory required for a single triplet, and to obtain better results one would like to use a batch size of at least 2.

Following this considerations, since the issue that these table show is that the main obstacle is the cost of the training procedure, the following section explores whether it is possible to train models on single frames, and then test their generalization capabilities to sequences of different lengths. This is of course possible only using aggregation methods that provide the flexibility to adapt to different sequence lengths, as it is the case only for CAT and the

proposed SeqVLAD layer.

### 5.5.8 Testing models from single-image task

This section is perhaps the one containing the most unexpected results. Its purpose is to test whether models trained on the traditional single-image task with an aggregator flexible enough that it can be used for inference with any arbitrary sequence length can be a valid alternative to training directly on multi-frame sequences. The table reports the use of CAT aggregator on GeM for comparison, whereas NetVLAD-CAT has not even been evaluated as even if training on single-images were to provide an improvement, the huge dimensionality obtained ($10^5$ already at Seq 7) would not make it of interest anyway, as instead this section focuses on the practical advantages obtainable exploiting the single-image task. Tab.5.9 reports the obtained results evaluating the models trained on single image on sequences of various length up to 15 frames.

Table 5.9: Testing different models trained on the **im2im** task. Only models with the capability of being adapted to multiple sequence length were tested. Note that when trained on single image, SeqVLAD is equivalent to NetVLAD, however the ability to treat different sequence lengths comes from the SeqVLAD implementation only.
All backbones were pretrained on ImageNet and full models trained on MSLS single image task. N. neg 10. cct224 or 384 indicates the size to which input images were resized to 224x224 or 384x384, whereas the *_tr N* indicates they were truncated at N-th block, and *_fz* M it was freezed up to M-th block. Each columns is color coded to highlight highest values.

| Backbone | Feat. Dim | Aggregation | R@1 Seq 1 | R@1 Seq 3 | R@1 Seq 5 | R@1 Seq 7 | R@1 Seq 9 | R@1 Seq 15 |
|---|---|---|---|---|---|---|---|---|
| r18l3 | _ | GEM - CAT | 65.7 | 75.8 | 77.5 | 78.9 | 79.4 | 83.0 |
| r18l3 | 16384 | SeqVLAD | 73.9 | 82.1 | 83.1 | 84.0 | 85.4 | 90.4 |
| r18l3+Attn2D | | SeqVLAD | 74.9 | 82.1 | 83.7 | 85.0 | 86.3 | 91.8 |
| r50l3 | 65536 | SeqVLAD | 75.9 | 83.5 | 85.6 | 86.5 | 88.0 | 92.0 |
| cct224_tr8 | 24576 | SeqVLAD | 74.6 | 83.4 | 85.1 | 84.9 | 86.8 | 91.1 |
| cct224_tr10 | | | 74.5 | 83.8 | 85.5 | 86.0 | 87.7 | 90.4 |
| cct384_tr8 | 24576 | SeqVLAD | 77.5 | 86.4 | 87.3 | 87.8 | 89.6 | 94.7 |
| cct384_tr8_fz2 | | | 79.8 | 87.7 | 88.9 | 88.8 | 90.3 | 94.8 |
| cct384_tr10 | | | 79.4 | 87.7 | 89.2 | 89.4 | 90.5 | 93.6 |

For GeM the feature dimensionality is not reported as it is the only with

CAT aggregator and therefore the dimensionality increases with the number of frames multiplied by 256.
Looking at table, already at a first glance emerge the surprising results obtained with this models. In particular, the most important results concern CCT using $384x384$ images. In fact, results obtained with CCT224 are remarkable as well, as they are just about 1% below the results obtained training the same model on Seq 3 and Seq 5. Nevertheless, the most important result is constituted by the performances obtained by CCT384, as it represents the very point of this section. It was not feasible to train this architecture directly on sequences, as even with sequences of length 3 it requires a cluster of GPUs with more than $12Gb$ of memory, which are quite hard to obtain.
Instead, the training on single frames is feasible on GPUs with as little as $8Gb$ of memory, rendering it much more accessible, and these outstanding results make it a successful choice.
The reason of this remarkable results, in terms of the comparison between CCT384 trained on single image, with respect to CCT224 trained directly on the sequence, can be the following. In both cases the backbone is trained to extract features on single frames, therefore the SeqVLAD aggregator can perform the soft-assignment in terms of *Visual words* collecting the local descriptors; in the case of multi-frame training these descriptors are collected over the entire sequence. However, such local descriptors are in anycase the product of the backbone, that does not have the concept of sequence, so the SeqVLAD layer learns how to cluster this extracted feature, but the nature of these features is the same whether the model is being trained on 1 or more frames, because of the way the backbone treats frames. Therefore the result is that when using the same architecture, for example CCT224, either trained on single-images or multi-frames, the final recall obtained is very similar. Whereas using CCT384, which it is important to remember has the same number of parameters as CCT224, it can rely on a higher input resolution, extract more tokens, and experimentally the results is that the backbone is able to extract richer representations.
It may seem weird that a model trained on single image is able to outperform methods specifically tailored to work with sequences, like Timesformer with its dedicated temporal attention. There can be many explanations for this phenomenon; remembering that the Timesformer architecture was developed for Action Recognition, in which it is important to have a temporal attention in order to tract the evolution of features across frames. Contrarily, in the task of VG, based on these results it seems that this factor is less relevant, and that to robustly represent a sequence it is enough to look at features

individually in the frames, and collect them organically like SeqVLAD does. The explanation can also derive from the used dataset; e.g. it can be than in MSLS query and database frames are quite similar and it is not necessary to embed information about the evolution of the semantic elements across the sequence. Exploring this possibility on more dataset is certainly among the future works. Moreover, another explanation can be simply that given the large number of parameters presented by the Timesformer architecture, it is harder to fine-tune the training parameters, the early-stopping criterion and so on. This can only be answered with a more exhaustive set of experiments; the main obstacle is, again, the computational requirements.

Another important answer that this table gives, is the confirmation of the superior representational capability of the CCT architecture with respect to a ResNet-18, thanks to its clever combination of convolutional tokenizers and transformer-encoder layers. Such advantage of CCT was hypothesized in Sec. 5.4.5, and partially confirmed by results in Sec. 5.5.6. However, up until now it remained to be examined whether the advantage in pure recall that it provides could exclusively depend on the higher feature dimensionality obtained. This table that brings into the comparison a ResNet-50 - SeqVLAD as well, which presents an even higher dimensionality, proves that the use of Self-Attention in the encoding blocks allows the network to extract richer features, that ultimately lead to better recalls.

### 5.5.9 Reversing frames order

This section is oriented towards the analysis of the learning capabilities of each of the presented methods, in order to understand if the extracted features depend strictly on the information contained inside the frames of a sequence, or if the model is concentrating also on other factors like the specific ordering of frames, which should not affect the representation of a sequence. In fact, in the case of a dataset specifically built for the task such as MSLS, it happens that each query sequence and its positive matchings are all sequences whose orientation is the same; in other words, if the query sequence goes from point A to point B, its positive match in the database may be skewed as that it starts a bit before or after point A, but it still goes towards B in the same direction.
In a real application this might (and most likely will not) be the case, and queries will be received containing different ordering of the frames with respect to the one present in the sequences collected in the database. Moreover,

based on the specific system that utilizes a VG system, it can also happen due to processing errors, network optimization procedures, or simply an unordered collection, the user of the deployed VG system (note that the user can be an automated system itself) can submit sequences whose frames do not a follow a particular order and are therefore shuffled.

For these reasons, Tab. 5.10 is devoted to the analysis of the behavior of all the presented approaches when the sequences available in the gallery get shuffled or reversed in the order of their frames, whereas the query do not get modified.

Table 5.10: Comparison of results **reversing/shuffling the frame order** in the database. The table compares results of the same methods in the standard case, reversing the frame order and shuffling it. A method for each of the categories considered in this work has been tested
All backbones were pretrained on ImageNet. N. neg 10. cct384 indicates input images were resized to 384x384, whereas the *_tr N* indicates they were truncated at N-th block, and*_fz* M that it was freezed up to M-th block. Each columns is color coded to highlight highest values.

| Architecture | SeqLen in training | Aggregation | Frame order | R@1 Seq 3 | R@1 Seq 5 | R@1 Seq 7 | R@1 Seq 9 | R@1 Seq 15 |
|---|---|---|---|---|---|---|---|---|
| r18l3 | Seq 5 | GEM + CAT | Normal | 72.7 | 74.5 | 75.9 | 77.6 | 81.1 |
| | | | Reverse | 65.9 | 64.8 | 65.8 | 66.6 | 66.5 |
| | | | Shuffle | 69.5 | 70.7 | 72.1 | 73.7 | 75.9 |
| r18l3 | Seq 3 | NetVLAD + CAT | Normal | 81.0 | 82.3 | 83.4 | 83.5 | 84.7 |
| | | | Reverse | 77.7 | 75.9 | 75.2 | 75.0 | 73.6 |
| | | | Shuffle | 79.4 | 80.4 | 80.7 | 81.0 | 80.6 |
| r18l3 | Seq 3 | SeqVLAD + Attn 2d | Normal | 83.1 | 84.5 | 85.5 | 86.8 | 90.1 |
| | | | Reverse | 83.1 | 84.5 | 85.5 | 86.8 | 90.1 |
| | | | Shuffle | 83.1 | 84.5 | 85.5 | 86.8 | 90.1 |
| r18l3 | Seq 3 | SeqVLAD + Attn 3d | Normal | 83.3 | 85.1 | 85.7 | 87.4 | 91.1 |
| | | | Reverse | 83.3 | 85.1 | 85.7 | 87.4 | 91.1 |
| | | | Shuffle | 83.3 | 85.1 | 85.7 | 87.4 | 91.1 |
| ViT | Seq 5 | FC2048 | Normal | | 79.2 | | | |
| | | | Reverse | | 74.5 | | | |
| | | | Shuffle | | 76.9 | | | |
| Timesformer | Seq 7 | _ | Normal | | | 83.7 | | |
| | | | Reverse | | | 83.7 | | |
| | | | Shuffle | | | 83.8 | | |
| CCT224 tr8_fz2 | Seq 3 | SeqVLAD | Normal | 86.8 | 88.0 | 88.1 | 89.1 | 92.9 |
| | | | Reverse | 86.8 | 88.0 | 88.1 | 89.1 | 92.9 |
| | | | Shuffle | 86.8 | 88.0 | 88.1 | 89.1 | 92.9 |

Looking at the results in the table, a few interesting considerations can be discussed. As it was to be expected, methods whose aggregation is simply made up of the CAT of the single frame features, take a hit in performances ranging from 5 to 15% if the database against which the queries are matched presents a different frame ordering. However the drop in recall is not as much to make the predictions completely useless; and the reason for this can be the fact that overall the sequences are quite short, and therefore the appearances in consecutive frames is not so different, and given that the middle frame stays the same, the highest differences occur probably only at the extreme frames. As for the situation with a shuffled database, the behavior is the same and the performances hit is generally lower, which is understandable as especially on short sequence shuffling can end up being closer to the original ordering than a reverse. As it is understandable in light of the explanation discussed, the gap in performances with respect to the 'standard' setting increases as the number of frames increases.
The same is true for methods using FC aggregation, as it is the case of ViT, as the FC layer does not have any mechanism forcing it to learn meaningful representations independently from the order of the stacked frame features, and therefore there is a bias towards the specific order of the features that shows up in the results as well. Moreover the use of the FC does not allow to test on sequence lengths different than the training one.

Regarding the Timesformer architecture, it too does not have the flexibility to treat different sequence lengths. However, it shows that it is able, unlike previously discussed methods, to build a sequence descriptor that only depends on the semantic elements present inside the various frames, as in fact its recall score is identical regardless of ordering of the frames considered. This result shows how a method specifically tailored to treat sequences can yield a representation which truly reflects the information available in the frames, and it is not just a stacking of single-frame features.
A similar consideration can be made for the use of SeqVLAD, being it paired either with an Attention-augmented ResNet-18 or with the novel CCT architecture. Even though the backbones in these two pipelines only treat singularly the frames to extract their representation (with the exception of the 3D version of the Non-Local block), the proposed SeqVLAD aggregator considers the set of local descriptors in its entirety only based on their values, regardless of the specific position inside the stack of descriptors. According to this explanation it follows that the results using such aggregator are not at all dependent on the visiting order of the frames.

### 5.5.10 Number of Negatives

Considering how it has been highlighted that one of the most influent factors that make training a VG system, especially in a multi-frame setting is the number of negatives examples for the triplet loss, this section explores whether or not the value of 10 traditionally adopted by the literature since [39] is really the only alternative. Due to limited time and computational resources, the set of performed experiments on this topic is not completely exhaustive, however it is enough to give source for some interesting considerations. The pipelines tested are the overly populare ResNet-18 - SeqVLAD trained on single images, and the two variants of CCT followed by SeqVLAD, trained either on single frames or on Seq 3. Note that SeqVLAD, when trained on single frames, is equivalent to NetVLAD, and in the table it is reported as SeqVLAD only because to obtain those result it is required to exploit the implementation of SeqVLAD which allows for different number of frames. Tab. 5.11 contains the obtained results. ResNet-18 + SeqVLAD is reported as it represents a common baseline, and the others represent the best performing alternatives. The comparison is made repeating the training procedure using different number of negatives, ranging from to 2 to the traditional value of 10. Finally, the trained models have been tested on different sequences length.

This analysis shows as well some surprising results. Whereas for the baseline it seems that 2 negatives cause the performances to take a small hit, it appears that halving the number of negative examples does not cause the model to learn less robust representation; on the contrary in some cases there is a slight advantage.
Furthermore, the most surprising results come from the other tested methods. It would appear, infact, that for all the version of CCT tested reducing the number of negatives from 10 to 2 yields a consistent performance boost ranging from 1% up to even 4% in some cases!
This unexpected results can be explained in light of the following considerations. First of all, based on the behavior of the baseline it appears that actually 10 negative examples are not really needed, and therefore the value utilized by [39] turns out to be more of an upper bound rather than a lower bound estimate. It needs to be remembered, however, that in the original paper [39] used a different mining technique, caching the feature representation of the whole dataset. For a complete discussion on the mining techniques, refer to Sec. 4.2.3. Therefore when applying partial mining, it seems that what happens is that since negative examples are randomly sampled, after

Table 5.11: Analysis of the impact of **Number of negative** examples in the triplets used for metric learning.
Backbones are all pre-trained on ImageNet. CCT384 indicates input images were resized to 384x384, whereas the *_tr N* indicates they were truncated at N-th block, and *_fz* M that it was freezed up to M-th block. *fzConv* indicates that only the convolutional block used as tokenizers are freezed. Each columns is color coded to highlight highest values.

| Backbone | SeqLen in training | Aggregation | N. Negs | R@1 Seq 1 | R@1 Seq 3 | R@1 Seq 5 | R@1 Seq 7 | R@1 Seq 9 | R@1 Seq 15 |
|---|---|---|---|---|---|---|---|---|---|
| r18l3 | Seq 1 | SeqVLAD | 10 | 73.9 | 82.1 | 83.1 | 84.0 | 85.4 | 90.4 |
| | | | 5 | 73.9 | 82.5 | 83.7 | 84.8 | 86.1 | 90.9 |
| | | | 2 | 72.7 | 80.4 | 81.7 | 82.7 | 84.3 | 89.6 |
| CCT224 tr10_fzConv | Seq 1 | SeqVLAD | 10 | 74.9 | 84.3 | 86.3 | 86.7 | 87.9 | 90.9 |
| | | | 5 | 76.2 | 85.1 | 87.4 | 87.9 | 89.4 | 92.0 |
| | | | 2 | 78.4 | 87.1 | 88.8 | 88.8 | 90.3 | 93.8 |
| CCT384 tr8_fz2 | Seq 1 | SeqVLAD | 10 | 79.8 | 87.7 | 88.9 | 88.8 | 90.3 | 94.8 |
| | | | 5 | 80.5 | 88.3 | 89.5 | 90.0 | 91.4 | 95.7 |
| | | | 2 | 81.3 | 89.2 | 90.5 | 90.1 | 91.5 | 96.2 |
| CCT224 tr8_fz2 | Seq 3 | SeqVLAD | 10 | 74.1 | 84.6 | 86.4 | 86.6 | 88.0 | 92.7 |
| | | | 5 | 75.2 | 85.2 | 87.2 | 86.8 | 88.3 | 93.1 |
| | | | 2 | 76.9 | 86.8 | 88.0 | 88.1 | 89.1 | 92.9 |

the closest examples (after 5, for the baseline) they stop being useful. It is important to remember that negative examples are chosen with the criteria of being as hard as possible for the network to distinguish and are therefore sorted according to an ascending distance in the feature space with respect to the query. Therefore if after 5 examples or so this examples are too far away from the query, they do not represent useful examples for learning and their only effect is to bring down the average value of the triplet loss, which in turn results in shorter optimization steps than they could have been, and in the end the result is that performances are lower.
As for the CCT-based methods, for which reducing negatives is a straightforward performance boost, a similar explanation holds. Specifically, since the CCT has a higher representational capacity than the ResNet (as discussed in Sec. 5.5.8) it is more likely to be able to distinguish more easily the negative examples, and as the table shows even 2 are sufficient, and any number above that will only dilute the optimization steps. In other words, for this kind of models, using less negative examples for metric learning yields a faster and more accurate optimization procedure. Another information to support this

affirmation is the fact that training the CCT384 - SeqVLAD with 10 negatives took 40 epochs to converge, whereas it only took 31 in the case with 2 negatives only.

This last result are quite important as they show that even hyperparameters like the number of negatives which is often overlooked and taken for granted, can be a relevant factor to tune. In particular, the most important aspect beside the few percentual points of recall that can be gained, is that cutting drastically this number (e.g. from 10 to 2) can substantially reduce as well the memory required and training time too; which especially in the case of multi-frame training can make the difference between a feasible and an unfeasible training.

# Chapter 6

# Conclusions and future works

This thesis has thoroughly explored the field of Visual Geo-localization (VG), by providing an extensive benchmark of all the most popular and effective methods that have been proposed in the literature over the years, for each of the specific aspects of a general VG pipeline. For each specific step, a number of alternatives have been evaluated through an extensive set of experiments, and a discussion on the conclusions drawable regarding the application of the considered methods, taking into account real-world constraints and different possible use-cases. This considerable amount of work has been a group effort whose outcome has been also a submission to NeurIPS Benchmark and Datasets 2021 track.

The second part of the thesis presented a more experimental settings, in which the objective was to provide an innovative contribution to the field of Sequence-based Visual Geo-localization (S-VG), with a specific focus on accurate localization using short sequences ranging from 3 to 15 frames.
The novelties analyzed in this setting include the application of the concept of Self-Attention in various forms; through Transformer Encoder layers used either as feature aggregator and as backbone. Additionally, a hybrid backbone that uses a mixed approach to combine the most powerful aspects of both convolutional and Transformer-based backbone has been tested, proving that it is able to provide richer feature embeddings compared to a standard ResNet. An ulterior alternative implementation of attention mechanism used has been the one of Non-Local blocks, a flexible layer pluggable inside every

kind of network to provide a contextual re-weighting based on the relationship between features across the whole feature-maps in case of a convolutional network.
Moreover, the usage of an extended version of the Self-Attention has been tested, to take into account both spatial and temporal information in the computation of the contextual attention vector. This creates an architecture able to process the sequence in its entirety exploiting temporal information as well, eliminating the need for a dedicated aggregation steps.
A further contribution is represented by the introduction of the SeqVLAD layer, which is a modification to the popular NetVLAD aggregator that allows to collect organically the features extracted from all the frames in the sequence, yielding a single sequence descriptor of fixed dimensionality that represents a statistic of all the relevant features extracted by the backbone with respect to a set of *Visual Words*, that are, in other words, the semantic elements useful to the task.
An additional section has been dedicated to investigate the capacity of models of learning the distinguishing semantic elements in a sequence a-priori, independently from the specific order in which frames were encountered, showed that specifically tailored methods guarantee a robust representation unlike the considered baselines.

Finally, to complement the analysis of this methods a discussion on their computational cost and memory requirements has been presented, offering potential alternatives to speed-up training, and bring down the memory footprint in some cases in order to make models trainable on standard hardware. Such alternatives considered the reduction of the number of negative examples for the metric learning, and the use of flexible models trained on single images with the capability of processing sequences of any number of frames, showing that good generalization properties can be achieved with respect to the multi-frame evaluation settings.

Overall, this work has showed both the usefulness of having a literature-wide benchmark on the most effective techniques, as it helps practitioners and researchers who first approach the field get a practical sense of which approaches work best in which situations. Secondly, this acquired knowledge has been used as a baseline to extend the task to work with sequences which are often available in many practical applications. This second part has shown how sequences require tailored methods that are able to capture the

variety of information present across the frames, and how including attention mechanisms in the feature extraction pipeline can yield more robust and representative descriptors. Therefore the results and experiments presented throughout the thesis can be of interest for many practical applications as real-world constraints and requirements of the different methods have been discussed. In many sectors of application, in fact, such systems need to operate in real-time (autonomous driving, SLAM) and therefore an evaluation of the delays compared with the localization performance is crucial.
There are many possible future developments following the findings of this work. First of all, extending the experiments performed on different datasets, analyzing the impact of a more dense database distribution on the performances; another interesting path to follow could be to use a different formulation of the task, in which the interpretation of a match is of finding the arrival point of the sequence of frames. Furthermore, the analysis in the last sections showed that there is room for further improvements in performances while also rendering the training procedure lighter, by reducing the number of negatives, or training on single-frames. Therefore this results are the source for many possible experiments, testing models that without these considerations was not possible to use due to their requirements. Additionally, the possibility to use models pre-trained on single-frames as a starting point for a further fine-tuning on the specific sequence length desired can be explored.

# Bibliography


[1] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. *Deep Visual Geo-localization Benchmark*. 2021. URL: https://openreview.net/forum?id=at5mq2EJebX.

[2] Carlo Masone and Barbara Caputo. «A Survey on Deep Visual Place Recognition». In: *IEEE Access* 9 (2021), pp. 19516–19547.

[3] Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. «Visual Place Recognition: A Survey». In: *IEEE Transactions on Robotics* 32.1 (2016), pp. 1–19.

[4] Sourav Garg, Tobias Fischer, and Michael Milford. «Where Is Your Place, Visual Place Recognition?» In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Survey Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4416–4425. DOI: 10.24963/ijcai.2021/603.

[5] Xiwu Zhang, Lei Wang, and Yan Su. «Visual place recognition: A survey from deep learning perspective». In: *Pattern Recognition* 113 (2021), p. 107760. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2020.107760. URL: https://www.sciencedirect.com/science/article/pii/S003132032030563X.

[6] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. «Large-Scale Image Retrieval with Attentive Deep Local Features». In: *IEEE International Conference on Computer Vision*. 2017. URL: https://arxiv.org/abs/1612.06321.

[7] Tobias Weyand, A. Araújo, Bingyi Cao, and Jack Sim. «Google Landmarks Dataset v2 – A Large-Scale Benchmark for Instance-Level Recognition and Retrieval». In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 2572–2581.

[8] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. «Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases». In: *IEEE Conference on Computer Vision and Pattern Recognition*. June 2008.

[9] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. «Object retrieval with large vocabularies and fast spatial matching.» In: *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2007. ISBN: 1-4244-1179-3. URL: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2007.html#PhilbinCISZ07.

[10] Nathan Piasco, Desire Sidibé, Cédric Demonceaux, and Valérie Gouet-Brunet. «A survey on Visual-Based Localization: On the benefit of heterogeneous data». In: *Pattern Recognit.* 74 (2018), pp. 90–109.

[11] Noé Pion, Martin Humenberger, Gabriela Csurka, Yohann Cabon, and Torsten Sattler. «Benchmarking Image Retrieval for Visual Localization». In: *2020 International Conference on 3D Vision (3DV)*. 2020, pp. 483–494.

[12] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. «1 Year, 1000km: The Oxford RobotCar Dataset». In: *The International Journal of Robotics Research* (2017).

[13] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and Tomas Pajdla. «Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8601–8610.

[14] Xun Sun, Yuanfan Xie, Pei Luo, and Liang Wang. «A Dataset for Benchmarking Image-Based Localization». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5641–5649. DOI: 10.1109/CVPR.2017.598.

[15] Nam N. Vo, Nathan Jacobs, and James Hays. «Revisiting IM2GPS in the Deep Learning Era». In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2640–2649.

[16] Tobias Weyand, Ilya Kostrikov, and James Philbin. «PlaNet - Photo Geolocation with Convolutional Neural Networks». In: (2016). DOI: 10.1007/978-3-319-46484-8_3. eprint: arXiv:1602.05314.

[17] James Hays and Alexei Efros. «Large-Scale Image Geolocalization». In: *Multimodal Location Estimation of Videos and Images* (Oct. 2015), pp. 41–62. DOI: 10.1007/978-3-319-09861-6_3.

[18] Youji Feng, Lixin Fan, and Yihong Wu. «Fast Localization in Large Scale Environments Using Supervised Indexing of Binary Features». In: *IEEE Transactions on Image Processing* 25 (Nov. 2015), pp. 1–1. DOI: 10.1109/TIP.2015.2500030.

[19] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. *From Coarse to Fine: Robust Hierarchical Localization at Large Scale*. 2018. eprint: arXiv:1812.03506.

[20] M. Aubry, B. Russell, and J. Sivic. «Painting-to-3D Model Alignment Via Discriminative Visual Elements». In: *ACM Transactions on Graphics* (2013). Pre-print, accepted for publication.

[21] Tsung-Yi Lin, Serge Belongie, and James Hays. «Cross-View Image Geolocalization». In: June 2013, pp. 891–898. DOI: 10.1109/CVPR.2013.120.

[22] Nam Vo and James Hays. *Localizing and Orienting Street Views Using Overhead Imagery*. 2016. eprint: arXiv:1608.00161.

[23] Mayank Bansal, Harpreet Sawhney, Hui Cheng, and Kostas Daniilidis. «Geo-localization of street views with aerial image databases». In: Nov. 2011, pp. 1125–1128. DOI: 10.1145/2072298.2071954.

[24] David Lowe. «Distinctive Image Features from Scale-Invariant Keypoints». In: *International Journal of Computer Vision* 60 (Nov. 2004), pp. 91–. DOI: 10.1023/B:VISI.0000029664.99615.94.

[25] R. Arandjelovic. «Three things everyone should know to improve object retrieval». In: June 2012, pp. 2911–2918. DOI: 10.1109/CVPR.2012.6248018.

[26] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. «Speeded-up robust features (SURF)». In: *Computer Vision and Image Understanding* 110 (June 2008), pp. 346–359. DOI: 10.1016/j.cviu.2007.09.014.

[27] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. «Predicting Good Features for Image Geo-Localization Using Per-Bundle VLAD». In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1170–1178. DOI: 10.1109/ICCV.2015.139.

[28] Josef Sivic and Andrew Zisserman. «Video Google: A Text Retrieval Approach to Object Matching in Videos.» In: *ICCV*. IEEE Computer Society, 2003, pp. 1470–1477. ISBN: 0-7695-1950-4. URL: http://dblp.uni-trier.de/db/conf/iccv/iccv2003-2.html#SivicZ03.

[29] Hervé Jégou, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. «Aggregating Local Image Descriptors into Compact Codes». In: *IEEE transactions on pattern analysis and machine intelligence* 34 (Dec. 2011). DOI: 10.1109/TPAMI.2011.235.

[30] H. Jégou and Andrew Zisserman. «Triangulation Embedding and Democratic Aggregation for Image Search». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 3310–3317.

[31] Aude Oliva and Antonio Torralba. «Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope.» In: *International Journal of Computer Vision* 42.3 (2001), pp. 145–175. URL: http://dblp.uni-trier.de/db/journals/ijcv/ijcv42.html#OlivaT01.

[32] Antonio Torralba, Rob Fergus, and Yair Weiss. «Small codes and large image databases for recognition». In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587633.

[33] Adel Fakih Charbel Azzi Daniel Asmar and John Zelek. «Filtering 3D Keypoints Using GIST For Accurate Image-Based Localization». In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Edwin R. Hancock Richard C. Wilson and William A. P. Smith. BMVA Press, Sept. 2016, pp. 127.1–127.12. ISBN: 1-901725-59-6. DOI: 10.5244/C.30.127. URL: https://dx.doi.org/10.5244/C.30.127.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[36] A. Razavian, Hossein Azizpour, J. Sullivan, and S. Carlsson. «CNN Features Off-the-Shelf: An Astounding Baseline for Recognition». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2014), pp. 512–519.

[37] Ruben Gomez-Ojeda, Manuel Lopez-Antequera, Nicolai Petkov, and Javier Gonzalez-Jimenez. *Training a Convolutional Neural Network for Appearance-Invariant Place Recognition*. 2015. arXiv: 1505.07428 [cs.CV].

[38] Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. *Neural Codes for Image Retrieval*. 2014. arXiv: 1404.1777 [cs.CV].

[39] Relja Arandjelovic, Petr Gronát, Akihiko Torii, Tomás Pajdla, and Josef Sivic. «NetVLAD: CNN Architecture for Weakly Supervised Place Recognition.» In: *CVPR*. IEEE Computer Society, 2016, pp. 5297–5307. ISBN: 978-1-4673-8851-1. URL: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2016.html#ArandjelovicGTP16.

[40] A. Razavian, J. Sullivan, A. Maki, and S. Carlsson. «Visual Instance Retrieval with Deep Convolutional Networks». In: *CoRR* abs/1412.6574 (2015).

[41] Artem Babenko and Victor Lempitsky. «Aggregating Deep Convolutional Features for Image Retrieval». In: *ICCV* (Oct. 2015).

[42] Arsalan Mousavian and Jana Kosecka. *Deep Convolutional Features for Image Based Retrieval and Scene Categorization*. 2015. arXiv: 1509.06033 [cs.CV].

[43] A. Gordo, J. Almazan, J. Revaud, and D. Larlus. «End-to-end Learning of Deep Visual Representations for Image Retrieval». In: *IJCV* (2017).

[44] F. Radenović, G. Tolias, and O. Chum. «Fine-tuning CNN Image Retrieval with No Human Annotation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).

[45] Jeff Johnson, Matthijs Douze, and Hervé Jégou. «Billion-scale similarity search with GPUs». In: *arXiv preprint arXiv:1702.08734* (2017).

[46] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. «Product Quantization for Nearest Neighbor Search.» In: *IEEE Trans. Pattern Anal. Mach. Intell.* 33.1 (2011), pp. 117–128. URL: http://dblp.uni-trier.de/db/journals/pami/pami33.html#JegouDS11.

[47] Dmitry Baranchuk, Artem Babenko, and Yury Malkov. *Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors*. 2018. arXiv: 1802.02422 [cs.CV].

[48] Yu. A. Malkov and D. A. Yashunin. *Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs*. 2018. arXiv: 1603.09320 [cs.DS].

[49] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. «Object retrieval with large vocabularies and fast spatial matching». In: June 2007. DOI: 10.1109/CVPR.2007.383172.

[50] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. *Particular object retrieval with integral max-pooling of CNN activations*. 2015. eprint: arXiv:1511.05879.

[51] Mubariz Zaffar, Sourav Garg, Michael Milford, Julian Kooij, David Flynn, Klaus McDonald-Maier, and Shoaib Ehsan. «VPR-Bench: An Open-Source Visual Place Recognition Evaluation Framework with Quantifiable Viewpoint and Appearance Change». In: *International Journal of Computer Vision* 129.7 (2021), pp. 2136–2174.

[52] Rohit Girdhar, Deva Ramanan, Abhinav Gupta, Josef Sivic, and Bryan C. Russell. «ActionVLAD: Learning Spatio-Temporal Aggregation for Action Classification». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 3165–3174.

[53] Ahmad Jalal, Yeonho Kim, Yong-Joong Kim, Shaharyar Kamal, and Daijin Kim. «Robust human activity recognition from depth video using spatiotemporal multi-fused features». In: *Pattern Recognit.* 61 (2017), pp. 295–308.

[54] Yehao Li, Ting Yao, Yingwei Pan, Hongyang Chao, and Tao Mei. «Jointly Localizing and Describing Events for Dense Video Captioning». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 7492–7500.

[55] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. «Bags of Spacetime Energies for Dynamic Scene Recognition». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 2681–2688.

[56] Angela Dai, C. Qi, and Matthias Nießner. «Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6545–6554.

[57] Shuran Song, Fisher Yu, Andy Zeng, Angel X. Chang, Manolis Savva, and Thomas A. Funkhouser. «Semantic Scene Completion from a Single Depth Image». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 190–198.

[58] Lin Wu, Yang Wang, Ling Shao, and M. Wang. «3-D PersonVLAD: Learning Deep Global Representations for Video-Based Person Reidentification». In: *IEEE Transactions on Neural Networks and Learning Systems* 30 (2019), pp. 3347–3359.

[59] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. «Large-Scale Video Classification with Convolutional Neural Networks». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1725–1732.

[60] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. «Learning Spatiotemporal Features with 3D Convolutional Networks». In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 4489–4497.

[61] Michael Milford and Gordon Wyeth. «SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights». In: *2012 IEEE International Conference on Robotics and Automation* (2012), pp. 1643–1649.

[62] Edward Pepperell, Peter I. Corke, and Michael Milford. «All-environment visual place recognition with SMART». In: *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 1612–1618.

[63] Kin Leong Ho and Paul Newman. «Detecting Loop Closure with Scene Sequences». In: *International Journal of Computer Vision* 74 (2006), pp. 261–286.

[64] Mark Joseph Cummins and Paul Newman. «FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance». In: *The International Journal of Robotics Research* 27 (2008), pp. 647–665.

[65] Tayyab Naseer, Luciano Spinello, Wolfram Burgard, and C. Stachniss. «Robust Visual Robot Localization Across Seasons Using Network Flows». In: *AAAI*. 2014.

[66] Olga Vysotska and C. Stachniss. «Effective Visual Place Recognition Using Multi-Sequence Maps». In: *IEEE Robotics and Automation Letters* 4 (2019), pp. 1730–1736.

[67] Emilio Parisotto, Devendra Singh Chaplot, Jian Zhang, and Ruslan Salakhutdinov. «Global Pose Estimation with an Attention-Based Recurrent Network». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2018), pp. 350–35009.

[68] Dorian Gálvez-López and Juan D. Tardós. «Bags of Binary Words for Fast Place Recognition in Image Sequences». In: *IEEE Transactions on Robotics* 28 (2012), pp. 1188–1197.

[69] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. «BRIEF: Binary Robust Independent Elementary Features». In: *ECCV*. 2010.

[70] Edward Rosten and Tom Drummond. «Machine Learning for High-Speed Corner Detection». In: *ECCV*. 2006.

[71] Jose M. Facil, Daniel Olid, Luis Montesano, and Javier Civera. *Condition-Invariant Multi-View Place Recognition*. 2019. arXiv: 1902.09516 [cs.CV].

[72] Sourav Garg and Michael Milford. «SeqNet: Learning Descriptors for Sequence-Based Hierarchical Place Recognition». In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 4305–4312.

[73] Sourav Garg, Ben Harwood, Gaurangi Anand, and Michael Milford. «Delta Descriptors: Change-Based Place Representation for Robust Visual Localization». In: *IEEE Robotics and Automation Letters* 5.4 (Oct. 2020), pp. 5120–5127. ISSN: 2377-3774. DOI: 10.1109/lra.2020.3005627. URL: http://dx.doi.org/10.1109/LRA.2020.3005627.

[74] K. He, X. Zhang, S. Ren, and J. Sun. «Deep Residual Learning for Image Recognition». In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[75] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9 (1997), pp. 1735–1780.

[76] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. «An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling». In: *ArXiv* abs/1803.01271 (2018).

[77] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. «Neural Machine Translation by Jointly Learning to Align and Translate». In: *CoRR* abs/1409.0473 (2015).

[78] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. «Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation». In: *EMNLP*. 2014.

[79] Nal Kalchbrenner and Phil Blunsom. «Recurrent Continuous Translation Models». In: *EMNLP*. 2013.

[80] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. «On the Properties of Neural Machine Translation: Encoder–Decoder Approaches». In: *SSST@EMNLP*. 2014.

[81] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. «Sequence to Sequence Learning with Neural Networks». In: *NIPS*. 2014.

[82] Alex Graves, Greg Wayne, and Ivo Danihelka. «Neural Turing Machines». In: *ArXiv* abs/1410.5401 (2014).

[83] Thang Luong, Hieu Pham, and Christopher D. Manning. «Effective Approaches to Attention-based Neural Machine Translation». In: *EMNLP*. 2015.

[84] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. «Attention is All you Need». In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[85] Jianpeng Cheng, Li Dong, and Mirella Lapata. «Long Short-Term Memory-Networks for Machine Reading». In: *EMNLP*. 2016.

[86] Alexander M. Rush, Sumit Chopra, and Jason Weston. «A Neural Attention Model for Abstractive Sentence Summarization». In: *EMNLP*. 2015.

[87] Ke Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. «Show, Attend and Tell: Neural Image Caption Generation with Visual Attention». In: *ICML*. 2015.

[88] Volodymyr Mnih, Nicolas Manfred Otto Heess, Alex Graves, and Koray Kavukcuoglu. «Recurrent Models of Visual Attention». In: *NIPS*. 2014.

[89] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. «An Attentive Survey of Attention Models». In: *ArXiv* abs/1904.02874 (2019).

[90] Abdul Mueed Hafiz, Shabir A. Parah, and Rouf Ul Alam Bhat. «Attention mechanisms and deep learning for machine vision: A survey of the state of the art». In: *ArXiv* abs/2106.07550 (2021).

[91] Salman Hameed Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. «Transformers in Vision: A Survey». In: *ArXiv* abs/2101.01169 (2021).

[92] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. «Non-Local Neural Networks». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[93] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. «A non-local algorithm for image denoising». In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* 2 (2005), 60–65 vol. 2.

[94] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, Humphrey Shi, and Wenyu Liu. «CCNet: Criss-Cross Attention for Semantic Segmentation». In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 603–612.

[95] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Ching-Feng Lin. «Local Relation Networks for Image Recognition». In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 3463–3472.

[96] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le. «Attention Augmented Convolutional Networks». In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 3285–3294.

[97] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. «Self-Attention with Relative Position Representations». In: *NAACL*. 2018.

[98] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: `1810.04805 [cs.CL]`.

[99] Alec Radford and Karthik Narasimhan. «Improving Language Understanding by Generative Pre-Training». In: 2018.

[100] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. «Language Models are Unsupervised Multitask Learners». In: 2019.

[101] Tom B. Brown et al. «Language Models are Few-Shot Learners». In: *ArXiv* abs/2005.14165 (2020).

[102] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. «Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer». In: *ArXiv* abs/1910.10683 (2020).

[103] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, Zhaohui Yang, Yiman Zhang, and Dacheng Tao. «A Survey on Vision Transformer». In: 2020.

[104] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. «Stand-Alone Self-Attention in Vision Models». In: *NeurIPS*. 2019.

[105] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. «Exploring Self-Attention for Image Recognition». In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 10073–10082.

[106] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. «An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale». In: *International Conference on Learning Representations*. 2021. URL: `https://openreview.net/forum?id=YicbFdNTTy`.

[107] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. «Revisiting Unreasonable Effectiveness of Data in Deep Learning Era». In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 843–852.

[108] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herv'e J'egou. «Training data-efficient image transformers & distillation through attention». In: *ICML*. 2021.

[109] Yawei Li, K. Zhang, Jie Cao, Radu Timofte, and Luc Van Gool. «LocalViT: Bringing Locality to Vision Transformers». In: *ArXiv* abs/2104.05707 (2021).

[110] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Herv'e J'egou, and Matthijs Douze. «LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference». In: *ArXiv* abs/2104.01136 (2021).

[111] Haiping Wu, Bin Xiao, Noel C. F. Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. «CvT: Introducing Convolutions to Vision Transformers». In: *ArXiv* abs/2103.15808 (2021).

[112] Ali Hassani, Steven Walton, Nikhil Shah, Abulikemu Abuduweili, Jiachen Li, and Humphrey Shi. *Escaping the Big Data Paradigm with Compact Transformers*. 2021. arXiv: 2104.05704 [cs.CV].

[113] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. «Is Space-Time Attention All You Need for Video Understanding?» In: *Proceedings of the International Conference on Machine Learning (ICML)*. July 2021.

[114] Frederik Warburg, Soren Hauberg, Manuel Lopez-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera. «Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition». In: *IEEE Conference on Computer Vision and Pattern Recognition*. June 2020.

[115] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. «Patch-NetVLAD: Multi-Scale Fusion of Locally-Global Descriptors for Place Recognition». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 14141–14152.

[116] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. «24/7 Place Recognition by View Synthesis». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.2 (2018), pp. 257–271.

[117] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. «City-scale landmark identification on mobile devices». In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2011, pp. 737–744. DOI: 10.1109/CVPR.2011.5995610.

[118] A. Torii, Hajime Taira, Josef Sivic, M. Pollefeys, M. Okutomi, T. Pajdla, and Torsten Sattler. «Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization?» In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43 (2021), pp. 814–829.

[119] M. Cummins and P. Newman. «Highly scalable appearance-only SLAM - FAB-MAP 2.0». In: *Robotics: Science and Systems*. 2009.

[120] Michael Milford and G. Wyeth. «Mapping a Suburb With a Single Camera Using a Biologically Inspired SLAM System». In: *IEEE Transactions on Robotics* 24 (2008), pp. 1038–1053.

[121] A. Torii, J. Sivic, M. Okutomi, and T. Pajdla. «Visual Place Recognition with Repetitive Structures». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.11 (2015), pp. 2346–2359. DOI: 10.1109/TPAMI.2015.2409868.

[122] Jun Yu, Chaoyang Zhu, Jian Zhang, Qingming Huang, and Dacheng Tao. «Spatial Pyramid-Enhanced NetVLAD With Weighted Triplet Loss for Place Recognition». In: *IEEE Transactions on Neural Networks and Learning Systems* 31.2 (2020), pp. 661–674.

[123] Diederik Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: *International Conference on Learning Representations* (Dec. 2014).

[124] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. «Learned Contextual Feature Reweighting for Image Geo-Localization». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 3251–3260.

[125] Karen Simonyan and Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». In: *International Conference on Learning Representations*. 2015.

[126] Liu Liu, Hongdong Li, and Yuchao Dai. «Stochastic Attraction-Repulsion Embedding for Large Scale Image Localization». In: *IEEE International Conference on Computer Vision*. 2019.

[127] María Leyva-Vallina, Nicola Strisciuglio, and Nicolai Petkov. «Generalized Contrastive Optimization of Siamese Networks for Place Recognition». In: (Mar. 2021).

[128] Gabriele Moreno Berton, Valerio Paolicelli, Carlo Masone, and Barbara Caputo. «Adaptive-Attentive Geolocalization From Few Queries: A Hybrid Approach». In: *Proc. IEEE Winter Conf. on Applications of Computer Vision*. Jan. 2021, pp. 2918–2927.

[129] B. Cao, A. Araujo, and J. Sim. «Unifying Deep Local and Global Features for Image Search». In: *European Conference on Computer Vision*. Springer Int. Publishing, 2020, pp. 726–743. ISBN: 978-3-030-58564-8.

[130] Albert Gordo, Jon Almazán, Jérôme Revaud, and Diane Larlus. «Deep Image Retrieval: Learning Global Representations for Image Search». In: *ECCV*. 2016.

[131] Jérôme Revaud, Jon Almazán, R. S. Rezende, and César Roberto de Souza. «Learning With Average Precision: Training Image Retrieval With a Listwise Loss». In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 5106–5115.

[132] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. «1 Year, 1000km: The Oxford RobotCar Dataset». In: *The International Journal of Robotics Research (IJRR)* 36.1 (2017), pp. 3–15. DOI: 10.1177/0278364916679498. eprint: http://ijr.sagepub.com/content/early/2016/11/28/0278364916679498.full.pdf+html. URL: http://dx.doi.org/10.1177/0278364916679498.

[133] Niko Sünderhauf, Peer Neubert, and Peter Protzel. «Are we there yet? challenging SeqSLAM on a 3000 km journey across all four seasons». In: *Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA)* (Jan. 2013), p. 2013.

[134] Łukasz Kaiser and Samy Bengio. *Can Active Memory Replace Attention?* 2017. arXiv: 1610.08613 [cs.LG].

[135] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. *Neural Machine Translation in Linear Time*. 2017. arXiv: 1610.10099 [cs.CL].

[136] HuggingFace documentation. «Transformers-based Encoder-Decoder Models.» In: 2020. URL: https://huggingface.co/blog/encoder-decoder.