POLITECNICO DI TORINO

Master of Science in
Data Science and Engineering

Master's Degree Thesis

A Study on Deep Learning Approaches for Visual Geo-localization

Design of a Benchmarking Framework
and Self-Attentive Sequence-Based Methods

Supervisor:
Prof. Barbara Caputo

Co-Supervisors:
Dr. Eng. Carlo Masone
Eng. Gabriele Moreno Berton

Candidate:
Riccardo Mereu

Academic year 2020 – 2021

Abstract

Many applications in Artificial Intelligence require the capability of an autonomous system to accurately and efficiently locate itself in the real world. The task of Visual Geo-localization (VG) can be formulated as the ability to recognize the geographical location of a picture, using only its visual information and comparing it to a database of geotagged images, which represent the previously visited places or the area under analysis.
In the last two decades, this field has seen rapid growth in interest and technical development from different communities. Consequently, the research landscape has become increasingly fragmented and dissociated. The first half of this thesis work consists of an extensive survey of Deep Learning methods and the development of a benchmarking framework. This effort aims to create a clear and fair evaluation protocol for VG methods, provide a complete and flexible training platform, and establish effective good practices for real-world applications. An exhaustive collection of experiments accompanies all the techniques under analysis. Their performances are evaluated on six well-established VG datasets to assess their generalization and robustness capabilities.
The second half of this work focuses on an extension to the classical VG task, in which the inputs for the system are short sequences of frames instead of single images. The aim is to explore the extension of current VG architectures with temporal and multi-view information. This approach is of particular interest for mobile robotics, autonomous vehicles and augmented virtual reality applications that inherently deal with visual data flows. In particular, this work investigates the use of architectures based on self-attention mechanisms and Vision Transformers. The focus is on the sequence-based VG problem formulated as matching a query sequence to a shortlist of database sequences depicting the same location.

Acknowledgements

In this section, I want to thank all the people who have been supporting me in these years and during this thesis work. Since not all of them are comfortable with English, this part will be in Italian.

Un particolare ringraziamento va alla professoressa Barbara Caputo, che mi ha concesso l'opportunità di lavorare con lei e il suo gruppo di ricerca per la realizzazione di questo lavoro. Vorrei ringraziare Carlo Masone e Gabriele Berton, per i loro preziosi consigli e il supporto durante questi mesi. Grazie anche a Gabriele, con cui, seppur a distanza, ho condiviso l'avventura di sviluppare di questo progetto. Insieme a voi ho imparato tanto in questi mesi e senza il vostro aiuto questo lavoro non sarebbe stato possibile.

Un ringraziamento dal profondo del cuore va a mia madre e mio padre, che hanno sempre creduto in me e hanno sostenuto ogni mia decisione durante questi anni. Grazie per essermi sempre stati vicini e avermi dato la possibilità di portare avanti i miei studi.

A Silvia, per essere sempre stata al mio fianco negli istanti felici così come nei momenti più bui, sai di avere un posto speciale nel mio cuore e ti ringrazio per aver vissuto insieme a me questi anni.

A mio fratello Cristiano e a zio Giorgio, con cui spesso durante questi anni anche solo scambiare poche parole al telefono mi ha fatto sentire a casa, vi voglio bene.

Ai miei nonni e il resto della mia famiglia, che ogni volta tra nel breve lasso di tempo che passava tra un "Quando sei arrivato?" e un "E quando riparti?" hanno sempre dimostrato di essere felici di potermi riabbracciare tutte le volte che tornavo a casa.

Una dedica speciale va a Nonno Pinuccio, che non ha potuto vedere la mia crescita, ma che spero sarebbe stato orgoglioso di vedermi ora.

A tutti gli amici in Sardegna e sparsi un po' in giro per l'Italia e l'Europa, grazie per tutti i sorrisi e i momenti passati insieme. Ormai abbiamo preso tutti strade più o meno diverse, viviamo in città o addirittura stati diversi, ma nonostante questo sento di poter contare su tutti voi. Grazie a Piolo, Colomo, Lucio, Lolli, Alessandro, Luca, Federico, Lorenzo e Antonio e tutti gli amici di Mogoro. Infine, un grazie a tutte le persone che ho incontrato in questi anni e mi hanno accompagnato in tanti momenti. Grazie a Marco, Angelo, Laura, Giuseppe, Giorgio, Alberto e Silvio.

Contents

Chapter 1

Introduction

> Given an image of a place, can a human, animal, or robot decide whether or not this image is of a place it has already seen?
>
> – Lowry et al. [1]

Humans and animals possess remarkable localization and navigation capabilities fundamental for their lives in the physical world. Psychology and neuroscience have investigated the mechanisms behind these abilities since the '30s, with the seminal studies of Edward C. Tolman [2, 3] about latent learning and, in particular, cognitive maps, i.e., the mental description that a person maintains of its environment. From a purely functional point of view, the cognitive map can be seen as a process that acquires observations of the subject environment and produces a description of the current location. Moreover, the information acquired can be further manipulated to answer route-finding and position-finding questions. This concept has inspired over the years not only psychologists and neuroscientists but also works in other fields such as urban planning [4], Artificial Intelligence (AI) and robotics [5, 6, 7]. In particular, these early works of Benjamin Kuipers aimed at equipping robots with *large-scale spatial knowledge*, which is the type of knowledge concerning a portion of physical space whose structure covers a larger area than the sensory horizon of the agent. Similar to humans, the robot travels in its environment to acquire, store, manipulate local information to infer global relationships. Despite the recent progress in the development of new techniques and more precise sensors, the task of creating an accurate map of a robot's environment and maintaining self-localization within it remains to this day a challenging open problem for the robotics community. Lowry et al. [1] describes a robot's visual place recognition system as the combination of three key components "an *image processing* module to interpret the incoming visual data, a *map* that maintains a representation of the robot's knowledge of the world, and a *belief generation* module, which uses the incoming sensor data in combination with the map to make a decision about whether the robot is in a familiar or novel place".

This formulation bounds the problem to the downstream robotic application, posing strict requirements for computational efficiency, real-time execution, and often implies the use of a variety of streams of heterogeneous data. However, robots are not the only ones that could benefit from this ability, other real-world applications that require it range from augmented reality to 3D reconstruction systems, photo-sharing services, and autonomous driving systems. Over the past two decades, this task emerged and attracted broad interest in different fields such as Computer Vision and Machine Learning, whose focus is on the image processing module and consequently on techniques manipulating appearance information. Although the importance of the contamination and exchange of ideas and knowledge from different areas, this spread across multiple scientific domains has led to a fragmented research landscape, making it tough to have a comprehensive and unified view of the entire field. Moreover, another source of confusion emerges from the different designations used for this task from paper to paper; the most common ones are *Visual Geo-localization* (VG) [8], *Visual Place Recognition* (VPR) [1, 9], and *Image-Based Geo-Localization* (IBL) [10].

In this work, this task is referred to with the name of Visual Geo-localization, and the main focus is on techniques that employ Deep Learning (DL) and cast it to an Image Retrieval setting. In this scenario, the input is an image (or a sequence of images), commonly referred to as *query*, and the goal is to estimate its geographic location by matching it to a collection of geotagged images, called *database* or *gallery*. The image retrieval setting is not the only one proposed in the literature; other approaches will be discussed later in Chapter 2.3.

1.1 Objectives

As evidenced by the number of papers published on this topic [1, 11, 12] and numerous dedicated workshops at prestigious international conferences on Computer Vision and Robotics, VG is a rapidly growing research field. Unfortunately, the broad interest coming from different research communities, each one with its own distinct goals, has generated a vast and complex landscape of works to study and understand properly. All these reasons contributed to increasing the dissociation within the entire field in terms of comparison between different methods and lack of standard definitions of essential concepts. For instance, the notion of place may change from dataset to dataset, and the same confusion arises when trying to understand the effective contribution of new techniques. In this sense, the differences in the type of setups used in each work, such as using different training datasets, data augmentation techniques, or weight initialization, limit the comparison of these methods significantly and make the state-of-the-art claims ambiguous and hard to interpret.

Another problem is that the researchers often focus mainly on increase the performance in terms of *recall@N*, i.e., the proportion of correct gallery images found in the top-N retrieved results. In this way, they often forget that an evaluation of the goodness of a VG algorithm must consider the balance of three key characteristics: memory footprint, computational requirements, and retrieval performance. Many papers lack completely or put low emphasis on the analysis of hardware and time requirements fundamental for the scalability of the proposed methods. However, these factors play a crucial role in developing and deploying real-world applications.

The issues described so far can be summarized into two main points:

1. **Lack of a uniform setting**: VG lacks a standard framework to evaluate and analyze methods' performances;

2. **Focus on the optimization of a single metric**: ignoring key aspects such as execution time, hardware requirements and scalability represents a criticality for the current state-of-the-art methods.

The first objective of this work is to address these issues by creating a flexible and complete framework for benchmarking VG methods. The main aim is to provide the VG community with a uniform environment to build, train and evaluate a wide range of commonly used state-of-the-art architectures. This framework could become a starting point for future research and industry projects because its modular design allows the modification and tuning of each element of the VG pipeline. Therefore, it enables the possibility to conduct an in-depth analysis of the actual influence of each component in the final performance. Moreover, the framework allows to effortlessly obtain information regarding the number of parameters, FLOPs, descriptor dimensionality, et cetera.

The second objective of this thesis is to build on the findings obtained in the single image setup examined in the benchmarking activity to extend these techniques to a multi-image scenario and explore the use of novel techniques based on the self-attention mechanism. In fact, although the importance of visual data flows in several domains that represent a downstream application for VG techniques, such as mobile robotics, autonomous driving vehicles and augmented reality, the literature in this area is limited. The current approaches, further detailed in Sec. 2.6, can be roughly divided into two main categories hand-crafted methods based on stochastic models and similarity matrices, and sequence representation techniques, that instead extract from the continuous stream of images effective representations used to capture meaningful information for localization. This thesis explores the possibility of applying for this task the recent models from Computer Vision, based on Transformer-based architectures and more broadly on the concept of self-attention, which represent a promising strategy still not studied by the Visual Geo-localization community. The development of techniques to effectively analyze this type of visual data flows to perform image-based localization, improving the accuracy and robustness of the systems, is of uttermost importance in a life-long Visual Geo-localization scenario. Additionally, the plethora of applications that could benefit from Sequence-based VG (S-VG) unlocks exciting future scenarios.

1.2 Contributions

Following the objectives set in the previous section, the contributions of this thesis went in two main directions. To address the flaws and lacks in the current benchmarking works in the VG literature, this thesis delves into an extensive analysis of the Deep Learning methods for VG that approach it as an image retrieval task. Based on the standard VG pipeline identified with this research, this work proposes a uniform and formal training and evaluation platform. The framework is designed with a modular structure to provide the users with a system that can replicate the state-of-the-art methods choosing different configurations. This software will be released as an open-source resource that could be expanded by introducing new techniques for each component in the pipeline.
This platform will also serve to analyze and understand the core contributions of the VG methods, avoiding the limitations of the current benchmarking frameworks. The analytic toolbox consists of a series of metrics to measure the performance in terms of recall@N and the complexity of the different configurations (FLOPs, model size, memory footprint, and others). The software also allows the user to download a suite of six well-known VG datasets that can be used to train models on different scenarios and evaluate their domain generalization capabilities. Moreover, it provides a helpful tool to pretrain common neural network backbones on datasets with images from domains close to the VG task. The platform was used to produce an exhaustive collection of experiments that span many configurations, providing valuable guidelines for VG applications in industry and research.

The work presented in the first half of this thesis is the outcome of a joint effort and is currently under submission to a top-tier Computer Vision conference. All the team members contributed to the development and design of the framework. In particular, my personal contributions in the development phase regard the implementation of different aggregation methods, part of the mining methods, the pre-processing module, and the development of pre and post-processing techniques of the VG pipeline; in the experimental phase, I took care of the experiments regarding kNN indexing techniques, data augmentation, pre-processing and part of the ones on backbones and mining methods.

The second part of the thesis extends the current Deep Learning VG techniques to a sequence-based approach. Among the architectures considered in this part, the focus is on models employing the self-attention mechanisms and, in particular, this work is the first attempt to introduce Transformer-based architectures in the VG literature. This more research-oriented investigation was conducted together with Gabriele Trivigno. First, we conducted a literature review of the proposed methods in this sub-field of VG and constructed the baselines. Then, we started to analyze several possible architectures to tackle this problem. Concerning the development and design of the proposed solutions, my contributions regard the implementation of different networks, i.e., ViT [13], CCT [14], and different versions of Non-Local layers [15], and the development of an alternative version of the common NetVLAD [16] layer, called SeqVLAD. Finally, the experimental phase of this part was equally subdivided, and then I took care of running half of the experiments described in this work.

Chapter 2

Related Works

This chapter aims to provide the reader with a general overview of the approaches to Visual Geo-localization and an introduction to the self-attention mechanism in Deep Neural Networks. The first section provides a theoretical introduction to the VG task and a description of its main drivers. Section 2.2 discusses the relationship between VG and similar research areas, while Section 2.3 contains a summary of the different approaches to the problem proposed in the literature. Then, Section 2.4 is focused on the methods and techniques employed when VG is cast to an Image Retrieval setting. Section 2.5 analyzes the previous benchmarking works in the literature and compares them with our proposed benchmark [8]. Finally, the last two sections, Sec. 2.6 and 2.7, are devoted to VG methods dealing with sequences of images, mainly coming from the robotic community, and the use of self-attention mechanism in vision problems, both in a modular form and the Transformer architecture.

2.1 Visual Geo-localization

Visual Geo-localization can be defined as the ability to coarsely estimate the geographical location of the place depicted in an input image given a collection of images of previously visited places. The problem from an abstract perspective can be thought of as a comparison of visual data that considers a positive match between two observations when the locations depicted show a meaningful visual overlap due to overlapping field-of-view of the underlying sensor. This implies that the two images need to be taken at the same location and that their viewpoint coincides (at least partially). The evaluation of the goodness of the visual overlap should consider the possible presence of appearance shifts in the environment and the requirements of the final application of the VG system.

The study of Garg et al. [9] identifies the three main drivers in the VG domain: the ***environment***, the ***agent*** and the ***downstream task***. These elements and

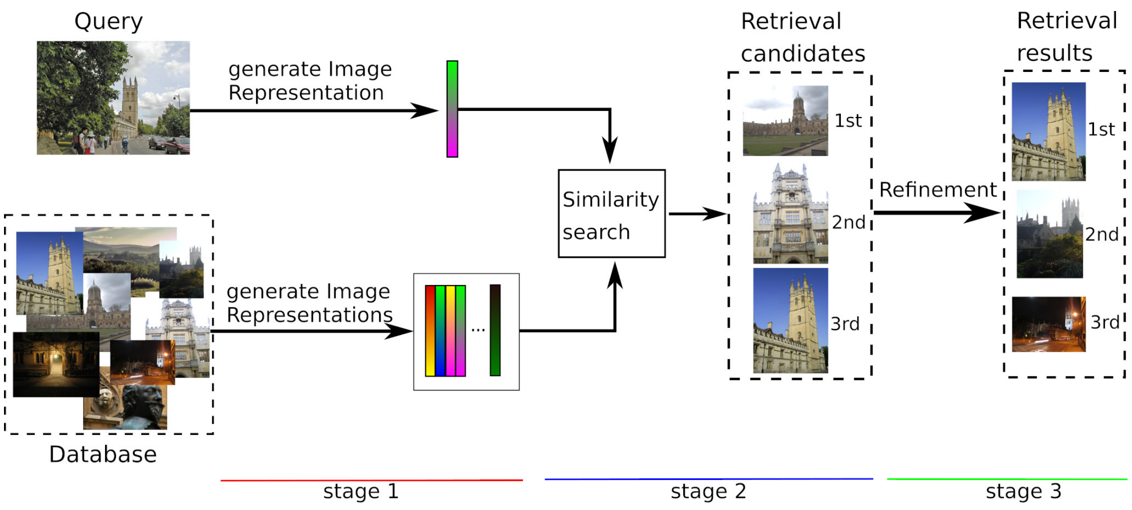


Figure 2.1: The abstract architecture of a VG system that approaches the problem as an Image Retrieval task. In the first stage of the pipeline, the system extracts from the query and the database images their feature representation (hand-crafted or deep-learned). The next step consists of a similarity search between query and database features, which produces a ranked list of potential candidates. The third (optional) stage employs post-processing methods to refine the final retrieval results. Figure from [11].

their interplay are fundamental to shaping the problem's definition, the solution design, and the evaluation procedure. The range of possible applications of VG can vary from mobile robots that need to be able to operate outdoor and indoor to servers of photo-sharing services or autonomous vehicles that operate in different types of streets (urban, suburban, or highways). These different operating environments influence the requirements, the datasets of choice, and the relevant metrics to evaluate. Another challenge related to the environment impacting VG systems' robustness is how to deal with the different appearance shifts that affect the physical world. The domain shifts can impact the point of view of the images; for instance, a realistic scenario may have gallery images collected from a specific type of source, e.g., sensors mounted in a car with changes mainly in the yaw direction, and queries from another source that alter the viewpoint, like photos from a smartphone. Moreover, a robust and solid VG system should deal effectively with changing lighting conditions at different day hours or weather conditions. In the context of long-term VG, the environment of choice will evolve due to seasonal changes and dynamic objects like cars and humans. In this sense, there could be methods that focus more on viewpoint-invariance, such as drones or robots, and others aiming at appearance-invariance or a trade-off between the two.

The other two drivers also impose significant constraints and require adequate design decisions. The different types of agents on which the VG method will be deployed have different computational resources and must be suited to the particular operation modality (real-time, near real-time, or even offline). The different types of downstream tasks could influence the performance requirements. For instance, a loop closure detector for SLAM needs high precision to avoid catastrophic results in the map building process, while some other applications have a larger tolerance margin.

The variety of approaches and techniques developed in the literature result from the different needs and the solutions arising from the interplay of these fundamental drivers. Regardless of the vast number of works in this field, there are still many open research problems whose solution will enhance the capabilities of current VG systems. For example, the researchers are developing new methods to address domain shifts in the visual data from the environment or take advantage of entire sequences of frames instead of single images. In the last years, the interest in VG would not have experienced this tremendous growth without the collection of numerous datasets covering the most disparate types of environments and applications [11]. Services like Google Street View or Mapillary, the diffusion of cars with visual sensors, and the spread of smartphones played a considerable role in making the collection procedure more straightforward. The datasets used in this work are described in detail later in Sec. 3.1 and 5.2.

This work deals with methods that approach VG as an Image Retrieval task (see Fig. 2.1) that employs deep learning models on urban and suburban datasets. The literature in this field investigated different problem settings and other techniques that do not use DL. An exhaustive and detailed analysis of the research studies VG can be found in these surveys [1, 11, 12]. The following sections try to give the reader a general overview of the research on VG with a strong focus on the themes related to this work.

2.2 Related Areas

2.2.1 Visual-Based Localization

Visual-Based Localization (VBL) is an image-based localization approach that shares some similarities with Visual Geo-localization but has different purposes and objectives. Within the VBL setting, the system aims at retrieving the pose of a visual query, i.e., both the position and the orientation of the sensor that captured the visual information. The result of this inference procedure is an accurate prediction of the 6-degrees-of-freedom pose of the camera with respect to a 3D map of the scene. The 3D representation of the environment can be a 3D model obtained with a Structure-from-Motion (SfM) algorithm using the local features of the database of images for the 2D-3D matching or an implicit representation learned with a machine learning algorithm. Even if the two tasks have different purposes, there are evident connections between VG and VBL methods in retrieving images from the same location. While VBL requires higher accuracy in the retrieval results, the VG methods are not that strict and, in some applications, could benefit from viewpoint-invariance (not necessarily a desirable property for VBL). For this reason, it is not strange to expect that over the years, solutions and ideas from the two areas have inspired each other.

In this direction, recently Pion et al. [17] propose in their benchmarking analysis to evaluate the performance of a set of VG methods when applied to three different VBL downstream tasks:

- *Pose approximation*, a straightforward extension of VG methods for VBL, where the estimate of the query pose is obtained as a linear combination of the top-k retrieved database images with different weighting schemes.

- *Pose estimation without a global 3D map*, composed of three steps (i) retrieve a set of images from various viewpoints of the same location represented in the query, (ii) use the retrieved elements to construct on-the-fly a small local SfM map of the scene, (iii) solving a perspective-n-point (PNP) problem to compute the camera pose.

- *Pose estimation with a global 3D map*, the closest setting to classical VBL, where a pre-built global SfM model is available and provides a 2D-3D correspondence between database images and the 3D model. The top-ranked images retrieved by the VG algorithm are used to obtain 2D-2D matches with the query, later translated into 2D-3D matches with the global map and used for the pose estimation.

As already mentioned, the task of VBL requires a different degree of precision in its predictions. A common practice in this area is to evaluate the performance of a method by considering correct all the pose estimates that fall within a given pair of error thresholds for position and absolute orientation error, i.e., $(X\ \mathrm{m}, Y°)$ [18, 19, 17]. The granularity of the thresholds can vary depending on the downstream task; the authors of [19] propose three different thresholds low $(5\mathrm{m}, 10°)$, medium $(0.5\mathrm{m}, 5°)$, and high $(0.25\mathrm{m}, 2°)$. For this reason, the datasets employed in VBL contain images labeled with 6-DOF poses and, optionally, the SfM 3D models. Some examples of commonly used datasets with these characteristics are Aachen Day-Night [20, 19], RobotCar Seasons [21, 19] and Baidu Mall [22]. For an in-depth analysis about this task, the reader is referred to [23, 19, 17].

2.2.2 Content Based Image Retrieval

Content-Based Image Retrieval (CBIR) is a well-known and extensively studied problem in Computer Vision [24, 25]. The task can be formulated as the process of

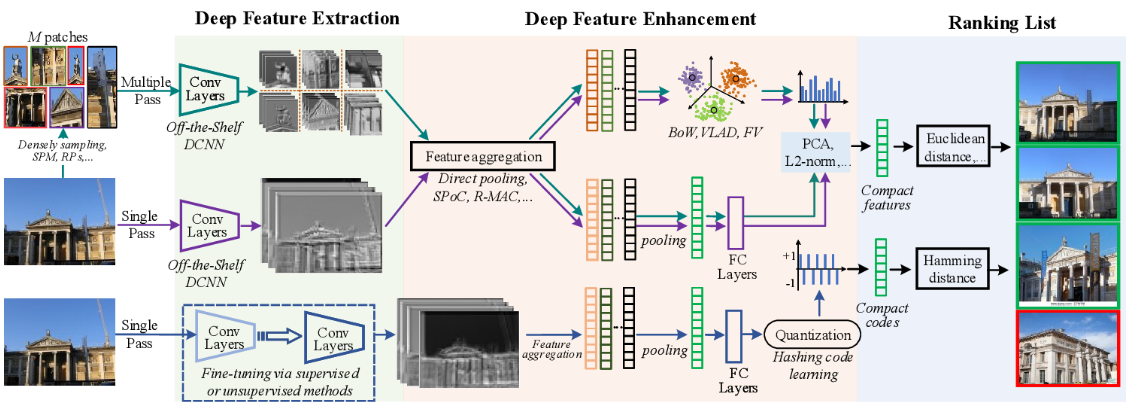


Figure 2.2: Reference CBIR pipelines employing different Deep Learning techniques. Figure from [24]

finding matches between a given query image and a database of images. Depending on the setting, a match can be considered successful when the retrieved images have a similar appearance, instance-level retrieval, or contain the same semantical information, category-level retrieval, e.g., objects not necessarily with the same appearance but that can be categorized together. From a general perspective, any image retrieval system follows four main steps:

1. *Features extraction:* which maps the raw images into a proper feature space for the task using hand-crafted or deep learning techniques. For efficiency reasons, the database features are usually computed offline, whereas the query features can be computed online.

2. *Features aggregation:* this step aims at rearranging the features extracted previously into a compact descriptor with better discriminative abilities suitable for the following step.

3. *Similarity search:* which is the algorithm that performs the comparisons between the compact descriptor of the query image and the ones of the database, retrieving the best ones according to a particular scoring function (e.g., Euclidean or Hamming distance).

4. *Candidates Re-Ranking:* consists of a set of methods that can be used to refine the ranking of potential positive matches obtained from the similarity search.

As will be further discussed in Sec. 2.3.1, the problem of VG is commonly framed as an Image Retrieval task; nevertheless, the underlying objectives of VG bring a completely different set of challenges and use-cases that drastically drift away from the ones of CBIR. In the CBIR setting, a single element is the object of interest of the retrieval process. Meanwhile, for Visual Geo-localization, the algorithm must deal with the complexity coming from the dynamic nature of the physical world.

The system should recognize any ordinary place or a region within the environment by combining a complicated collection of visual elements that characterize a particular location. Moreover, not all these elements may be informative for the VG task, like dynamic distractors, such as cars or pedestrians in an urban scenario, or appearance shifts as pointed out in Sec. 2.1. These additional requirements and objectives have contributed to shaping the research trends that identify VG as a research topic distinct from CBIR.

2.2.2.1 Landmark Retrieval

Another research topic that shares similarities with VG is Landmark Retrieval (LR), a particular case of instance retrieval task concerning only landmarks, i.e., relevant buildings, monuments, or other distinguishable objects. The objective of LR methods is to identify all the images in the database that contain the main object of interest depicted in the query. LR looks for matches in which the same landmark appears in both images regardless of whether they represent unrelated parts of the same landmark or if the pictures were collected in the same place. Compared to VG, the quality of the results is not affected by the geographical distance or the different viewpoints from which a particular location or building can be observed.

Another fundamental distinction between the two topics is that while VG datasets cover an entire geographical area with a certain degree of continuity, there is usually a discrete number of distinct landmarks in the datasets collected for LR. This allows the use of losses that can benefit from this additional supervised information, such as ArcFace [26] and CosFace [27] losses. These two classification losses were initially developed for facial recognition but also successfully used for LR [28]. Unlike contrastive and triplet loss, they do not require training techniques such as hard negative mining, which plays a significant role in the VG pipeline. On the other hand, even if some classification approach has been tried in VG, see Sec. 2.3.2, for the majority of the VG datasets, the only available information is the geographical coordinates of the database images.
The use of ArcFace and CosFace losses is also justified by the number of distinct landmarks in two of the most famous datasets, Google Landmarks Dataset v1 (GLDv1)[29] and v2 (GLDv2)[30], which have 30k and 200k landmarks, respectively. These large datasets contain landmarks spread all over the world; however, there exists also city-scale datasets like Paris 500k [31], or the two classic Paris [32] and Oxford [33], of which exists even a revised version ($\mathcal{R}$-Paris and $\mathcal{R}$-Oxford)[34]. The most common metric used in the LR literature is the mean Average Precision (mAP).

In the context of the benchmark developed in this thesis, we have investigated the performance of models pretrained on LR datasets on a Visual Geo-localization downstream task by using our framework.

2.3 Approaches to Visual Geo-localization

Although the predominant formulation for the Visual Geo-Localization problem establishes to approach it within an Image Retrieval setting, this has not stopped the researchers from considering and exploring different strategies. This section contains a summary of the different forms of approaches that have been proposed over these years.

2.3.1 Image Retrieval Approach

Nowadays, the dominant approach for Visual Geo-localization is to cast the problem as an image retrieval task. As already discussed in Sec. 2.2.2, CBIR aims to search images from a large database that possesses a similar appearance to a query. From an abstract point of view, the aim of both CBIR and VG is to extract an appropriate representation that can be used with a similarity research algorithm to decide if a couple of images can be or not a successful match. As for CBIR, the VG pipeline in this setting follows the four steps of extracting and aggregating the features from the raw images to produce a descriptor of the image. Then, a similarity search in the descriptor space is performed to retrieve the top-k matches. Finally, the last step consists of a re-ranking procedure to further refine the predictions, usually applied to obtain stronger performances at test time. For the VG problem, the database contains images annotated with their GPS coordinates. After the retrieval, this information is used to estimate the GPS location of the query image. The similarity between the two tasks is beneficial for both of them, and several techniques initially developed for CBIR have been successfully employed for VG over the years.

Although the procedure is very close to CBIR and LR problems, a VG method must encode an intrinsic spatial knowledge to overcome specific challenges that differentiate it from the other retrieval problems. The left side of Figure 2.3 illustrates the behavior of a CBIR system that matches the query image (blue border) with images only based on the visual content without considering distance and viewpoint information. Instead, on the right, the VG system only decides for a positive match between two images if taken in the exact location. The images with a red border are considered negative matches even if the building depicted is the same because they come from different places with respect to the query image. Because of the dynamic and evolving nature of the natural world, the datasets used in the VG setting must be various enough to represent as many environmental conditions as possible concerning illumination, shadows, or weather. Visual Geo-localization should be able to deal with scenes with common structures or elements that complicate building distinct image representations. Two common problems that arise for this reason are visual burstiness [35] and perceptual aliasing [1, 9]. Visual burstiness is when a given visual element appears in an image more frequently than expected by a statistically independent model, like a Bag-of-Words method, corrupting the similarity measure. Instead, perceptual aliasing describes the scenario in which two distinct places have a more similar appearance compared to the same place under different environmental conditions. This problem is frequent in indoor scenarios, which may have corridors or rooms with similar appearances, but can also appear in urban and natural vegetative environments. Compared to CBIR tasks, the images in VG datasets do not contain a clearly visible object to match with the database images, instead it is more likely to face the challenge of scenes cluttered with many non-informative distractors elements that are an integral part of the dynamic nature of the environment.

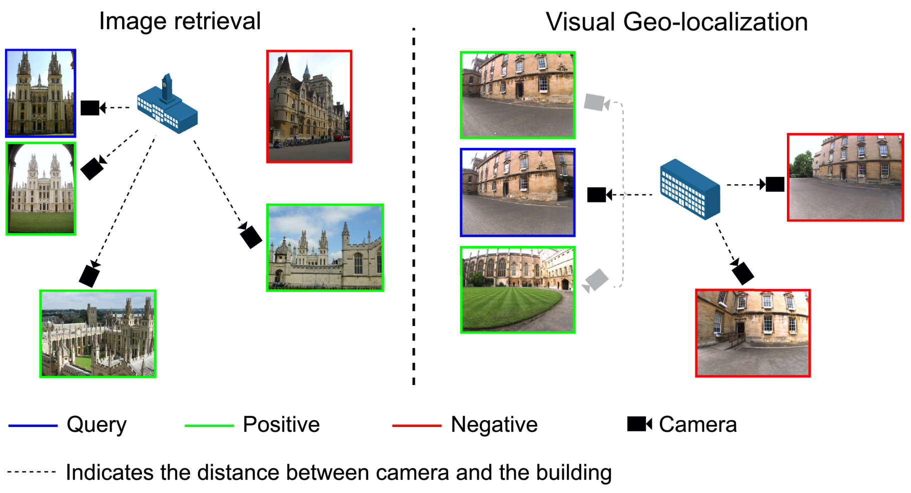


Figure 2.3: The figure portrays some of the differences between the Content-Based Image Retrieval and the Visual Geo-localization tasks. The orientation and the black camera's position in the drawing are indicative of the actual position and viewpoint of the real camera that took the pictures of the building. The grey cameras indicate pictures taken in the same position with different viewpoints from the black one. On the left, in the CBIR scenario, the query image (blue border) is matched to images that present a higher visual similarity in the database regardless of the camera's distance and viewpoint. On the right, in the VG scenario, the positive images (gree border) must consider the position from which the query was taken, even if they present drastic changes that reduce the visual similarity. In contrast, images with similar appearances must be regarded as negative if coming from different locations. Figure adapted from [12]

2.3.2 Classification Approach

An alternative formulation to the retrieval approach is to set the problem as a classification task that directly predicts the location depicted from the input image. As ambitiously stated in the pioneering work of Weyand et al. [36]: "[the] goal is to localize any type of photo taken at any location using just its pixels". The motivations behind this approach stem from two primary sources: the recent results of Deep Learning models in large-scale classification tasks and the observation that humans can roughly estimate the location of a photograph only using visual cues without the need to compare it with a massive dataset of similar images.

These methods rely on a partitioning scheme that divides the surface of the area of interest into a discrete number of non-overlapping cells used as classification labels for the training images, see Figure 2.4. The granularity used in the partitioning procedure represents a critical trade-off for the entire approach. The use of few large cells translates into a lower accuracy in the location estimate, but on the contrary, using a finer subdivision increases the number of classes and, consequently, the number of parameters of the model. The work of [37] proposes to overcome this issue by proposing a combinatorial partitioning technique, which unfortunately solves only the issues on the number of parameters but does not increase the performances under higher requirements of accuracy.

The strength of these methods over retrieval ones is that they do not require a massive reference database of images to produce the predicted location. As already mentioned, the shortcoming in this approach resides mainly in the partitioning scheme. Future research in this area, maybe combining classification and retrieval techniques, could unlock a promising path toward developing powerful global Visual Geo-localization systems.

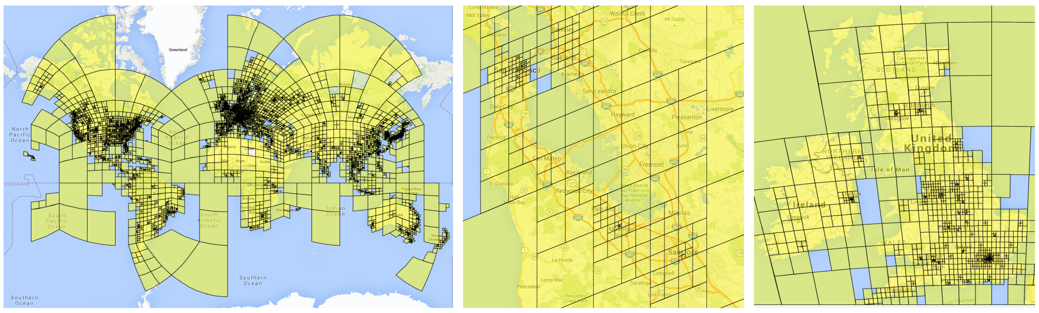


Figure 2.4: (Left) The adaptive partitioning scheme used by PlaNet to subdivide the world surface. The procedure divides the cells recursively until a threshold for the minimum number of images in the cell is met. (Center) Detail of the Bay Area. (Right) Detail on the British isles. Figures from [36]

2.4 Visual Geo-localization as Image Retrieval Task

The previous section described the variety of approaches proposed in the literature for the Visual Geo-localization problem. The focus of this thesis is on the Image Retrieval setting. The research on this area spans more than twenty years, and, for this reason, it has witnessed the different research trends that characterized the evolution of the communities interested in this task. From the early 2000s to 2012, Computer Vision's state-of-the-art methods relied on hand-crafted features and classic shallow Machine Learning algorithms. The success of AlexNet [38] in the ImageNet competition [39] in 2012 originated the rise of Deep Learning techniques. Following those trends, the VG literature comprises both methods relying on hand-crafted features and more recent deep learning techniques. This section aims to give an overview of the most influential methods from the VG literature.

2.4.1 Hand-Crafted Representations for VG

In the Image Retrieval setting, the most critical aspect of a VG system is constructing a mathematical representation that possesses discriminative and robust information to recognize places from the raw visual data. For a long period, hand-crafted representations were the de-facto standard for VG. The type of representations captured by the descriptors produced from these methods differentiates them into local and global features descriptors.

2.4.1.1 Local Descriptors

The methods based on the extraction of local features detect a set of small regions of interest in the input image, also called keypoints, composed of small groups of pixels and describe them with a set of vectors. This collection of features is then used to build a high-dimensional descriptor for the entire image. In more detail, the steps used to create an effective representation of the database images within this framework are the following:

1. *Local Feature Extraction*: that maps an image into a set of local features. It can be further divided into two intermediate steps:

    1.1 *Local detection*: a local detector, like the Hessian-affine detector [40] or MSER [41], identify a set of local regions of interest with stable properties under different imaging conditions;

    1.2 *Local description*: each keypoint is described with a d-dimensional descriptor, i.e. given N detected keypoints, they are mapped to a set of descriptors $\mathcal{D} = \{d_i\}_{i=1}^{N},\ d_i \in \mathbb{R}^n$.

2. *Codebook Training*: the set of local features extracted from the images of the entire training dataset are clustered into k clusters to create a codebook of visual words (or visual vocabulary).

3. *Feature Encoding*: the features representing each image and the codebook are used to obtain the final descriptor of the image, i.e., the set $\mathcal{D}$ of local features of an image is mapped to a feature embedding $g \in \mathbb{R}^m$ throughout this procedure.

The local features were particularly suited for the VG task because they own a set of valuable properties. They are repetitive, meaning that the same keypoints can be detected in different images, making them suitable to recognize distinctive elements necessary to identify a specific location. The local detectors look for local features invariant to translation, rotation, scale, and affine transformations. Together with their local nature, these attributes translate in strong robustness to occlusion and illumination changes.

Encoding the information detected in the local keypoints into features represents a crucial component of the pipeline described above. One of the most widely used algorithms is the Scale-Invariant Feature Transform (SIFT) [42] published and later patented by David Lowe in 1999, that extract from each region of interest a 128-dimensional vector. Further extensions to SIFT are PCA-SIFT [43], which reduces the dimensionality of the local features, and RootSIFT [44], which proposes an alternative normalization. Other local descriptors used in the literature are SURF (Speeded Up Robust Features) [45], BRIEF [46] and ORB (Oriented FAST and Rotated BRIEF) [47].

The idea of utilizing a codebook of visual words was initially proposed in the pioneering work of Sivic et al. [48]. They borrowed the Bag of Words (BoW) approach from the NLP community and adapted it to the Image Retrieval setting by using the local features extracted from images in place of word embeddings. Each of the local features is assigned to one of the visual words of the codebook. Then, the BoW representation is computed as a histogram of occurrences of visual words in the image that generates a k-dimensional vector, with k equals to the codebook dimension. Different weighting schemes can be used in this step, with the most popular being the term frequency-inverse document frequency (tf-idf). The resulting vector representations are L_2 normalized. During the retrieval stage, the database images are ranked based on the cosine similarity between their representation and the one of the query image.

The BoW vectors are usually high dimensional, $k = 20.000$ in the original implementation but in VG applications have been proposed methods using up to 100.000 dimensions [49]. The consequently sparse representation can benefit from inverted lists to implement an efficient search, but it also leads to several limitations in search times and memory requirements for large datasets. Other encodings have been proposed in the literature employing small-size codebooks, e.g., with dimension 64 or 128, to overcome these limitations; the most widely used are Fisher Vectors (FV) [50] and Vector of Locally Aggregated Descriptors (VLAD) [51, 52]. The former employs a Gaussian Mixture Model (GMM) to create a richer probabilistic visual vocabulary trained offline using Maximum Likelihood estimation. The output representation is the concatenation of the k d-dimensional gradients of the means of the Gaussian distributions that form the GMM. As for the BoW, the VLAD encoding learns a visual codebook by using the k-means algorithm. Then, each local descriptor is associated with its closer visual world. However, VLAD uses a different approach instead of storing only the number of occurrences. In fact, the sum of the differences between each visual word and its closer local features is accumulated, obtaining k d-dimensional vectors that are concatenated to obtain the final VLAD representation. It can be shown that VLAD is a simplified non-probabilistic version of the FV [53].

2.4.1.2 Global Descriptors

The previous section illustrated how the construction of image-level descriptors from local features requires two main steps: the detection of the image's local features and the feature embedding phase. These two stages are usually computationally expensive and build on the assumption that the image's content can be seen as a configuration of multiple elements. Another approach consists of considering the image as a whole. The descriptors that fall under this category are named global descriptors and follow the idea that the scene depicted in the input image can be described encoding the holistic properties of the entire scene without focusing on the individual details. These methods bypass the detection phase entirely by dividing the image into a grid and processing each cell regardless of its content. An example of a global descriptor used in several VG methods is GIST [54], which applies at each image block a set of Gabor filters with various orientations and frequencies. HoG [55] relies, instead, on the computation of histograms of occurrences of oriented gradients in each patch of the image. Some other works such as [56, 57] propose to interpret the grid subdivision of the images as a set of predefined keypoints on which apply local descriptors, with those two, in particular, using SURF and BRIEF, respectively. While being computationally lighter than local features, the drawbacks of global features arise in their less robust performances under changes in viewpoint and when the scene presents visual clutter or forms of occlusions.

2.4.2 Deep Learning Representations for VG

The last decade witnessed the rise of Convolutional Neural Networks (CNNs) [58] as one of the most popular and powerful tools in the Computer Vision community.

The convolution operation at the core of their functioning is based on three key ideas: sparse interactions, parameter sharing, and equivariant representations [59]. Following the success of AlexNet in large-scale image classification, the interest in this type of architectural design experienced enormous growth, which led to several studies on its application as an image representation generator on other visual tasks [60]. However, learning from scratch of CNNs parameters requires large datasets of annotated images; fortunately, the image representations learned on these datasets can also be successfully transferred to other tasks with a limited amount of data. Among the several visual tasks that profited from the introduction of CNNs, there is also Image Retrieval, with the excellent results obtained through the application of these Deep Learning techniques undermined the hand-crafted method dominance. The shift in the standard for CBIR also affected the Visual Geo-localization task.

This section aims to introduce the principal CNN-based methods proposed in the VG literature and how these methods are trained in the IR setting. The Deep Learning techniques discussed in this section do not include Transformer-based architectures, which will be discussed in Section 2.7. At the moment of writing this thesis, the only work that employs Transformers for Image Retrieval is [61], and one of the main contributions of this work is the introduction of this type of architecture for the VG task.

2.4.2.1 Fully Connected Representations

Around 2014 and 2015, several works in the VG literature [62, 63] investigated the use CNNs pre-trained on ImageNet by adopting as image representations the vectors of activations from the last Fully-Connected (FC) layer of these networks. In the following works, the researchers achieved better results by training these models directly for retrieval and employing metric learning losses, such as the triplet loss [64, 65, 66].

Unfortunately, the FC representations suffer from the same issues that characterize the hand-crafted global descriptors. They do not possess translation- and scale-invariant properties and are unstable when dealing with occlusions and distractors in the scene. At the time, the gap with classical local features based methods was consistent. Then, to circumvent the limitations above, other works tried to feed the CNNs with image patches and use patch-based representations for retrieval [67]. While finally closing the gap in performance with the state-of-the-art techniques of the time, FC representations cannot be compared to traditional methods in terms of memory footprint and computational costs.

2.4.2.2 Convolutional Representations

Instead of using the vector produced by the FC layer, the work of Babenko et al. [68] proposed to utilize directly the feature maps computed by the convolutional layers as image representation for the retrieval task. The authors proposed to flatten the feature maps into a vector and normalize them to produce the final image representation. This simple approach did not achieve superior results than FC representations but showed the potential in using the convolutional feature maps. Further works build on this idea and preserve the feature maps' tensor structure to benefit from their local intrinsic information. This line of research led to the current state-of-the-art methods in VG, which can be divided according to the computation performed on the feature maps into aggregation and pooling-based methods.

Aggregated Representations From a general perspective, the feature maps extracted by a convolutional layer are a $H \times W \times C$ tensor, where H, W and C are the height, the width, and the number of channels, respectively. This tensor can be interpreted as a set of C-dimensional feature vectors spatially located into an $H \times W$ grid. The early works applied traditional aggregation methods, as VLAD and BoW [69, 70]. Building on the observations on non-learned methods, it is possible to build a more robust and discriminative image representation by aggregating them.

Further studies aim to create aggregation techniques that can learn their parameters end-to-end from the data. The most popular and influential method in this family of learned aggregation-based methods is without any doubt NetVLAD[16]. Arandjelović et al. [16] proposed a differentiable generalized version of VLAD. The main idea resides in replacing the latter's hard assignment with a more suitable differentiable soft assignment approach. This design allows NetVLAD to learn the visual words of its codebook and the set of weights from the soft-assignment block, enabling greater flexibility than the original VLAD.

Between the further works that tried to enhance the performance of NetVLAD, there is the Contextual Reweighting Network (CRN)[10]. The authors proposed the introduction of a context modulation block to the original NetVLAD layer, which employs different multiscale context filters to produce a reweighting mask aiming at highlighting relevant structures and penalizing less relevant regions.

The main drawback in VLAD-inspired methods come from the high dimensionality of the output image descriptors, which is (kC)-dimensional vector where k is the number of visual words in the codebook, and C is the dimension of the features or in the case of CNN networks the number of channels of the feature maps. A common approach to obtain more compact and efficient descriptors is to apply dimensionality reduction techniques such as PCA or learned projection matrices.

Pooled Representations While the aggregation-based methods rely on an interpretation of the convolutional feature maps as dense C-dimensional features extracted from the input image. The alternative perspective, followed by the methods described in this section, is to see the output tensor as a stack of C activation maps of dimensions $W \times H$. Following the notation of [71], each activation maps is denoted by $\mathcal{X}_i \in \mathbb{R}^{H \times W}$, with $i = 1, \ldots, C$, Ω represents the set of spatial locations of this 2D tensors and $\mathcal{X}_i(p)$ is the value of the map i at position $p \in \Omega$. The output image descriptor is denoted by the vector $\mathbf{f}$. Compared to the feature aggregation of the traditional hand-crafted methods, the motivation behind this different approach comes from the argument that the features extracted by CNNs possess a higher discriminative capability, which can be manipulated using simpler schemes relying solely on their statistics. The pooling methods generally provide a low memory footprint and can outperform more complex hand-crafted representations. The work of Razavian et al. [72] propose to employ a max-pooling layer, named Maximum Activation of Convolution (MAC), i.e. computing the descriptor as $\mathbf{f} = [f_1, \ldots, f_C]^\top$ with $f_i = \max_{p \in \Omega} \mathcal{X}_i(p)$.

SPoC (Sum-Pooled Convolutional features) [73] utilizes instead a sum pooling layer, namely $\mathbf{f} = [f_1, \ldots, f_C]^\top$ with $f_i = \sum_{p \in \Omega} \mathcal{X}_i(p)$. This approach achieves better results than the max-pooling strategy, because it produces descriptors which are less sensitive to the presence of distractors.

The two previous approaches consider the feature maps in their entirety, R-MAC (Regional-MAC) [74] propose to apply the max-pooling operation on subregions of the feature map and then combine the resulting regional descriptors by summing them.

Finally, GeM (Generalized Mean aggregation) [71] propose a generalized pooling method that for particular values reduces to maximum and average pooling, i.e. it generalizes both MAC and SPoC. This method, together with R-MAC, is the current state of the art in the pooled-representation methods. The GeM descriptor is the vector $\mathbf{f} = [f_1, \ldots, f_C]^\top$ obtained as the concatenation of the parametric mean operation applied to each activation map i, $f_i = \left(\frac{1}{|\mathcal{X}_i|} \sum_{p \in \Omega} \mathcal{X}_i(p)^{m_i} \right)^{\frac{1}{m_i}}$. The parameters m_i can be learned end-to-end and also shared among the C feature maps.

2.4.3 Similarity Search

Regardless of the method adopted to extract the images representations, the following step in the Image Retrieval setting is to use these descriptors in order to match the input query with a shortlist of potential candidates in the database. The techniques in the VG literature most commonly use the Euclidean distance (or L_2 norm) to directly measure the dissimilarity between images in the descriptor space. Less diffused is instead the use of the cosine similarity.

The most similar elements in the database are retrieved using an exhaustive search with a k-Nearest Neighbors (kNN) algorithm, the primary technique used for CBIR and VG in the IR setting. Although the simple nature of this algorithm, the high dimensionality of the descriptor makes it quite expensive and can also lead to the disruptive phenomena known as the curse of dimensionality, i.e., the distances between the query and all the database instances become almost equal.

The dimensionality of the descriptors is for this reason regarded with uttermost importance for both memory requirements, especially in the case of large scale datasets, and to obtain effective discriminative image representations.

When the number of images in the database scales to millions of images or even more, exhaustive kNN becomes unsustainable and is replaced with Approximate kNN (ANN) algorithms that trade-off the the accuracy of the results with efficiency and memory footprint. Among the different strategies presented in the literature, the ones that have been employed in this work are Inverted File Index [48], Product Quantization [75], Inverted Multi-Index [76] and Hierarchical Navigable Small World graph [77]. Fortunately, efficient and highly optimized implementations of ANN algorithms are available; the library used for this work is the FAISS library [78].

2.4.4 Candidates Re-Ranking

The retrieved candidates obtained through the similarity search can be seen as a shortlist of possible locations of the query image. The quality of the potential candidates retrieved by a VG system can be worsened by different factors, such as the different disturbances introduced by the appearance modifications discussed in Section 2.1 or the use of the approximated similarity search algorithms discussed in Section 2.4.3. The VG literature has tried to overcome the presence of false positives in the final predictions by proposing a series of methodologies to post-process and refine the list of retrieved database images. These methods can be divided into the following categories:

- *Spatial (Geometric) Verification*: given a pair of images, the methods falling under this category first detect the feature-to-feature correspondence between them and then verify the reliability between the features matches by analyzing their consistency under different spatial transformations. The score achieved in this geometric verification is used to re-rank the candidates. The most popular example of these methods is RANSAC [33]. If the VG pipeline relies on hand-crafted sparse local features, they can be used to compute the image descriptor and for the re-ranking procedure. In the case of deep-learned descriptors, there are different choices:

  – the heuristic extraction of local descriptors from the CNN used to generate the image representation;

  – to employ two separate CNNs, one for the image representation and the other for the sparse local features;

  – hybrid CNNs to extract both image representation and local descriptors.

- *Non-Geometric Refinement*: all the methods that do not rely on the verification of geometric correspondences between local features, making them suitable for deep-learned representations, an example in [79] compare the query's MAC representation with the sets of R-MAC representations of the database images.

- *Query Expansion*: which refines the initial results by combing them with the query to produce an enriched representation of the depicted location used to perform a new search over the database. If the initial results are accurate enough, the second query will retrieve additional relevant candidates that improve the system's overall performance. The method was proposed for the first time in the visual domain by [80].

- *Diffusion*: which considers the manifold structure of the visual data to obtain a more accurate estimate of the similarity between images. This approach interprets the data as a graph where each node represents an image, and every edge represents the pairwise similarity among two database instances. The ranking score of each image with respect to the query is computed as a random walk over this graph.

2.5 Benchmarks in Visual Geo-localization

The first half of this work is devoted to developing a benchmarking framework for Deep Learning methods applied to the Visual Geo-localization task in its image retrieval formulation. The only benchmark designed explicitly for VG is the VPR-Bench framework presented in the work of Zaffar et al. [81]. Another work that evaluates the performances of VG methods is the previously mentioned work of Pion et al. [17], which has an entirely different focus on Visual Localization downstream applications.

In contrast to the approach proposed in this work, the authors of [81] compare the methods proposed in the literature treating them as off-the-shelf architectures. To clarify, the term "off-the-shelf" in this context indicates that these architectures are taken already trained from the different works. Since these methods are usually designed with different scopes and downstream applications (see the discussion of Sec. 2.1), they are trained on different types of datasets, possibly employing different mining techniques, losses, and a rather different combination of design choices in their training. This diverse set of underlying conditions lead to unstructured experiments and it is not straightforward to draw conclusions on the final results in this scenario.

Our work aims to provide a fair and clear evaluation protocol for VG methods that overcomes all the limitations of previous works. The central idea was to design a flexible framework that can be used to train and evaluate the various techniques from the literature. This framework enables combining and trying different elements in the VG pipeline, providing a powerful analytic tool for research or industry practitioners. Moreover, another problem addressed with this work is the necessity of a tool to investigate other practical requirements of these methods. Even if [81] provides valuable insights on descriptor dimensionalities and extraction times, our framework considers a broader and more general collection of hardware-agnostic statistics, such as FLOPS and size of the models, memory footprint, mining complexity, and parallelization capabilities of several VG techniques.

2.6 Sequence-Based Visual Geo-localization

Most of the methods studied for Visual Geo-localization are designed for single images. However, many applications of VG in the robotics community deal inherently with sequences of images. The most notable example is Visual SLAM, which stands for Visual Simultaneous Localization and Mapping, defined as the task for a robot to build a map of an unknown environment and perform self-localization using only visual information simultaneously. This task requires a robust and accurate loop closure detection module, which can identify areas that are observed again after a long period of the exploratory phase of the environment. When they are confidently detected, the loop closures provide correct data association for the whole SLAM system to obtain a consistent map. These methods find application also in robot relocation after entering in tracking lost state because of sudden motions, severe occlusions, or motion blur. The entire process of loop closure detection and construction of the internal representation of the environment entails a continuous stream of images that must be analyzed by the image processing module of the robot. As a result, this type of application could benefit from the development of methods able to extract as much information as possible from sequences of images in order to better understand the surrounding ambient.

Instead, in the Computer Vision community, only a few works explored the use of sequence-based techniques to extract temporal and multi-view information for VG. Nevertheless, a vast literature exists for spatio-temporal representation of video data with applications in action classification [82], activity recognition [83], person re-identification [84], dense video captioning [85], 3D semantic labelling [86], 3D shape completion [87], and dynamic scene recognition [88]. To learn better spatio-temporal representations, many of these works employ 3D convolutions on the video volume [89, 90]. However, most of these tasks only deal with a limited number of classes, while for VG the methods should acquire the ability to extract relevant features to discriminate ordinary places and regions of the physical world.

2.6.1 Current Approaches

In the robotics literature, building from the first methods that rely only on single images, like for FAB-MAP [91], we can identify a consistent line of work dealing with sequential score aggregation techniques. The methods following this approach aggregate the information from single images into descriptors of local navigation sequences, consisting of vectors storing differences of the hand-crafted features vectors of frames acquired in a small time window. Compared to traditional single-image methods, the works of [92], [93] and [94] showed to benefit from the use of sequence-based matching under extreme appearance variations. For instance, SeqSLAM [93] does not try to find the single descriptor that best matches the current image descriptor (global best match). Instead, such methods aim at finding all the descriptors within local neighborhoods of sequential images that best match the current one (local best match). Then, localization is achieved by recognizing coherent sequences of these "local best matches" through sequence-based matching. More in detail, the search for image sequences follows these steps. At first, the difference vectors between the local frames are computed and joined to form an image-matching matrix. Then, straight-line trajectories are projected from each possible template in the matrix to find the lowest-cost sequences. The final step involves applying a global cost threshold to determine which sequences are accepted or rejected.
The subsequent research in this area focused on improving these sequence score aggregation methods with the use of odometry [94], camera velocity-robust sequence searching [95, 96], hashing based match selection [97], handling different routes [98], using temporal information within a diffusion process [99] and trajectory-based attention learning for SLAM [100].
Another hand-crafted approach is DBoW [101], which uses a Bag of Visual Words for the BRIEF [102] with FAST keypoints [103] features extracted from the single images and incorporates a temporal consistency constraint. This constraint consists of grouping together images that depict the same place during the matching to prevent images collected in the same place from competing among them when the database is queried.
The aforementioned hand-crafted methods present several drawbacks:

- they cannot address changes in the sequence direction;
- they rely on long-term sequence matching, i.e., both query and database sequences must have many consecutive matching frames;
- they assume linear temporal correlation for sequence matching.

Moreover, these techniques are not learned end-to-end and operate on the matching scores obtained from the descriptors obtained from single images. Regarding this last point, the VG literature extensively studied the representation extracted from individual frames, while the use of temporal or sequential information to create a single compact representation for an entire sequence has received limited attention.

Few more recent works [104, 105, 106] try to address these issues proposing new methods that build on the previous research on both single-image VG and hand-crafted robotics techniques.
In the work of [104], the authors propose and evaluate three different deep network architectures. These architectures are composed of a backbone ResNet-50 [107] pre-trained on Imagenet, a fully connected layer, and three distinct aggregation methods to exploit multi-view and temporal information from small sequences of images. In order to include temporal information into the descriptors, the first approach consists of the concatenation of the descriptors of consecutive frames (Descriptor Grouping). However, this naive technique cannot learn to weigh differently the features extracted from different feature maps. Then, the second method, called Descriptor Fusion, improves the previous one by adding a fully connected layer that reduces the concatenated descriptors into a compact global descriptor for the whole sequence. Both Descriptor Grouping and Fusion do not exploit the sequential nature of the consecutive frames. In the third method, the authors replaced the fully connected layer of the second method with a Long Short Term Memory network (LSTM) [108].

The works of [105] and [106] aim to enhance the previous works based on the sequence similarity matrix. SeqNet [105] builds on SeqSLAM [93] proposing a hierarchical method that first filters the top-k matching candidate sequences and then performs sequence score aggregation. The input sequence is fed to an off-the-shelf descriptor extractor using NetVLAD [16], which is kept fixed during the training procedure. These descriptors are used from the downstream architecture to extract two other types of descriptors: sequence-level descriptors, which encode in a single vector the information of the entire sequence; frame-based descriptors, used to perform the search in the similarity matrix.
The first branch of the network extracts the sequence-level descriptors using a stack of three layers, i.e., a Temporal Convolution Network [109], a Sequence Average Pooling (SAP), and an L2-Normalization layer. The resulting descriptors are used to select from the gallery the top-k matching candidate sequences.
The other branch learns single image descriptors for each frame, which are a linear transformation of the input NetVLAD descriptors, and uses them to perform the sequential aggregation with the single-image descriptors of the top-k sequences obtained before obtaining the final match. Note that the use of the sequential search relies on the assumption of a one-to-one correspondence between the reference and query sequence, which could not always be the case in practice posing a consistent limitation to this method.

Finally, the work of [106] utilizes the sequential information to create a more robust descriptor, named Delta Descriptors, to contrast the influence of appearance changes due to different seasonal, atmospheric, and day/night conditions. This method works in an unsupervised setting and transforms the original input deep-learned image descriptors into a difference-based representation that considers the temporal differences across descriptors of different places observed throughout the sequence. The main reasons behind this work reside in how the sequences are managed in many applications of VG for mobile autonomous systems. In this scenario, the input images arrive as a continuous stream from the visual sensors and then are translated into high-dimensional descriptors. The resulting stream of descriptors can be seen as a multivariate time series. However, two sequences depicting the same succession of places under different conditions will present a substantial offset due to the appearance shift. The off-the-shelf descriptors alone cannot overcome this limitation. By mapping them into a difference-based representation, the authors aim at obtaining more robust and stable descriptors that ignore the domain shift focusing more on the actual place described. This approach requires medium or long length sequences, otherwise limiting the number of images leads to an inconsistent representation because of the high overlap between adjacent frames.

2.7 Self-Attention and Vision Transformers

2.7.1 The Attention Mechanism

The *attention mechanism* was initially developed in the Natural Language Processing (NLP) community in the context of neural machine translation [110] to help the models memorize the content of long input sentences. The traditional methods [111, 112, 113] in this area rely on an encoder-decoder architecture, where both components are recurrent neural networks (RNNs), usually LSTM [108] or GRU [114] units. The input sequence of source tokens is fed into the encoder, which compresses the information into a context vector of fixed length. This sentence embedding should summarize the meaning of the entire input sequence. The extracted context vector is used to initialize the hidden state of the decoder, which then generates the target tokens one after the other. Despite outperforming classical statistical methods, these methods presented two evident drawbacks. Working with long sequences, the RNNs forgets the old information after propagating it for many steps. Second, during the decoding phase, the model lacks an explicit world alignment between source and generated target sequence, leading to a scattered focus on the whole sequence.

Rather than building a single context vector, the authors of [110] maintain an RNN encoder but propose to let the decoder access the information of both encoder and decoder hidden states plus the alignment between the source and target tokens. At each decoding step j, an attention score α_{ji} is computed for each hidden state $\mathbf{h_i^{in}}$ of each input token and it is used to compute a dynamical context vector $\mathbf{c_j}$.

In formulas:

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

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

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

where $a(\cdot,\cdot)$ is the alignment function (or score function)which measures the similarity between two tokens, e_{ji} represent the attention score and α_{ji} is its normalized version. The hidden states generated at each step i by the encoder are combined together with a weighted average to obtain the current context vector $\mathbf{c_j}$. In [110] the alignment function was parameterized by a feed-forward neural network with a single hidden layer and trained with the rest of the network. This approach is commonly referred to as additive attention. Over the years, other score functions have been proposed, the most common ones are content-based attention [115], location-based attention [116], general attention [116], dot-product attention [116] and scaled dot-product attention [117]. In short, the attention mechanism allows the neural network to attend selectively to a subset of the input data to make its predictions.

2.7.2 Self-Attention

Self-attention is a type of attention mechanism for which the input sequence also represents the target sequence. The model estimates part of the input data utilizing other portions of the same input. This mechanism is conceptually similar to non-local means [118], which is a standard filtering algorithm for image denoising. In the NLP community, self-attention found applications in machine reading [119], abstractive summarization [120] and image description generation [121]. With the work of [117] that introduced the Transformer architecture, the use of self-attention has become ubiquitous first in NLP and more recently in the computer vision community.

Before the advent of transformers, the use of attention already fascinated Computer Vision researchers. The term *Visual Attention*, coined by [122], refers to the capability of the model to learn how to focus only on relevant regions and to encode long-range structural dependencies between areas of the input image. The literature related to the application of visual attention is vast and complex. For an exhaustive analysis of the topic, the reader is referred to [123, 124, 125]. Here we will restrict the focus to the methods employed in this work, further explained in detail in Sec. 5.3. In the literature, it is possible to identify two possible directions that aim at integrating the self-attention to vision tasks: propose self-attentive modules for CNNs or directly employ Transformer-based architectures. Before delving into the latter, the focus will be on summarising the most important methods that combine self-attention and CNNs.

In the work of [15], the authors take inspiration from the non-local means algorithm [118] and propose a general differentiable non-local operation module:

$$\mathbf{z}_i = W_z \mathbf{y}_i + \mathbf{x}_i \tag{2.4}$$

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j) \tag{2.5}$$

where $\mathbf{x}$ is the input image or its feature representation, $\mathbf{y}$ is the contribution from the non-local module that is combined with a residual connection with $\mathbf{x}$ to obtain the output $\mathbf{z}$. The function $f(\cdot,\cdot)$ computes a score between elements in position i and j, while $g(\cdot)$ extracts a representation of the input in position j. The general nature of the formulation becomes evident when inspecting the index i, which can identify a spatial, temporal, or spatio-temporal location. This flexibility allows overcoming the intrinsic spatial locality of CNNs, providing them with a mechanism to attend interactions between every couple of spatial locations in the input. Furthermore, it also let them operate in the temporal domain without the use of recurrent units.

However, the computational cost of the non-local module requires a quadratic computational complexity in the number of input feature maps. For this reason, the following work [126] tries to address this issue by proposing a criss-cross attention module. Instead of considering all the input features when computing $\mathbf{y}$, they account in the computation only the ones that are on the criss-cross path.

Self-attention was also used in Local Relation Net [127], which proposes a differentiable layer able to generate dynamical weights based on the compositional relationship of the input quantified in terms of similarity between features or pixels in a local window.

Finally, the work of [128] goes in the direction of fully replacing the convolutional layers with self-attention stand-alone modules that employ a 2D relative position encoding [129]. Although the competitive results were achieved with the self-attention-only modules, the authors got the best performances combining them with convolutional architecture.

2.7.3 Transformers

"Attention is all you need" [117] is the paper that in 2017 proposed the Transformer architecture. The network is based only on the self-attention mechanism, avoiding recurrent and convolutional elements. Moreover, their design does not imply the need for a strong inductive bias, typical of convolutional networks, making them suited for processing multiple modalities. Transformers showed an excellent capability to long model dependencies between input sequence elements with the advantage over standard RNNs to be highly parallelizable. This ability has proven particularly relevant for using this architecture outside the NLP tasks, achieving outstanding results in processing images, videos, audio, and graphs.

Compared to the models described in the previous section, Transformers relies totally on attention, particularly on the scaled dot-product self-attention. Given a query matrix Q, a key matrix K, and a value matrix V, whose columns are the learned projections of the original sequence tokens into three different spaces, the output of the self-attention module is a weighted sum of the value vectors. The weight for each position is dynamically determined based on the dot-product between each query and all the key vectors. In matrix form, the formula is the following:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.6}$$

where the scaling factor $1/\sqrt{d_k}$ is justified by the consideration that for large sequences the dot-product push the softmax function into regions with excessively small gradients.

The other important element in this architecture is the idea of Multi-Head Self-Attention (MHSA). Instead of relying only on a single attention block, the Transformers exploit the multi-head mechanism to split the input tokens into smaller vectors and compute the self-attention over each subspace in parallel. This approach captures multiple complex relationships among different elements in the sequence. Each head owns its set of learnable weight matrices $\{W_i^q, W_i^k, W_i^v\}$, with $i \in \{0, \ldots, h-1\}$ and h the number of heads, which is fixed to 8 in [117]. The output of the heads is concatenated into a single matrix and then fed again into another linear layer to obtain the final representation. The complete original Transformer architecture has an encoder-decoder structure and will be further detailed in Sec. 5.3.

2.7.4 Vision Transformers

Following [117], an increasingly extensive line of research in the NLP community focused on enhancing and developing Transformer-like models between them BERT (Bidirectional Encoder Representations from Transformers) [130], GPT (Generative Pre-trained Transformer) v1-3 [131, 132, 133], RoBERTa (Robustly Optimized BERT Pre-training) [134] and T5 (Text-to-Text Transfer Transformer) [135].

As already mentioned, the impact on NLP and the extreme flexibility of this architecture attracted the interest of the Computer Vision community, which in a short amount of time has applied them to a large variety of tasks (for a complete overview, please refer to [136, 125]). After the works on self-attention modules within CNN architecture, two subsequent works [137, 138] proposed to replace the convolutional layers with stand-alone single-head self-attention primitives completely. However, the turning point in the use of pure Transformers applied directly on image classification tasks arrived with the paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" [13]. Vision Transformers (ViT) became the first successful application of the pure Transfomer to image inputs. As the title of the paper suggests, this approach does not modify the architecture but proposes to handle 2D images by reshaping them into sequences of patches of 16x16 pixels that are then flattened as vectors and used as sequences of input tokens. Unfortunately, the high expressiveness of architectures using multi-head self-attention comes with a price: they must be trained or pre-trained on large datasets to achieve good performances. In the case of ViT, the model was initially pre-trained on JFT dataset [139], a proprietary dataset with 300 million images, before the fine-tuning on ImageNet. This requirement seems to restrict the applications of this family of architectures only to a restricted number of research centers with enough computational power and with the ability to collect these enormous datasets.

The next obvious step was to address this issue by developing models capable of generalizing well without using a massive amount of data. In this sense, the authors of DeiT [140] propose a distillation approach that allows training a vision Transformer effectively only on ImageNet by using a CNN as a teacher model. Another type of approach entails the combination of convolutional layers that inject an inductive bias in the pure-transformers. Some of the most common hybrid ViT architectures are the Convolutional vision Transformer (CvT) [141], LocalViT [142], LeViT [143] and Compact Convolutional Transformer (CCT) [14]. The latter is used in this work and will be further discussed in Sec. 5.3.

The last Transformer-based architecture described in this section is the TimeSformer [144], a work that proposes to use the ViT architecture in the context of video understanding. To model the spatio-temporal relationships of the video, the authors introduce a different self-attention scheme, named "Divided Attention". TimeSformer applies the temporal and the spatial attention separately in each encoder block, achieving state-of-the-art results on action recognition.

Part I

Deep Visual Geo-localization Benchmark

Chapter 3

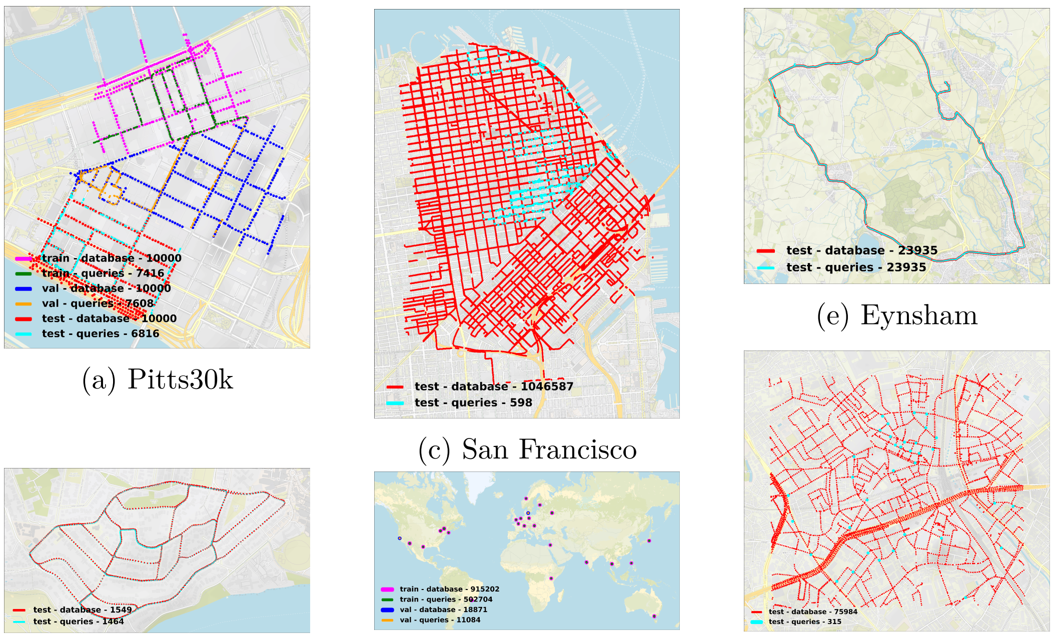
Benchmarking Framework

This chapter presents a discussion about the structure and the organization of the developed benchmarking framework. As already introduced in Section 1.1, the main motivation behind this work is the current lack in the VG literature of a standardized setting for training and evaluating models with a focus on both performances and hardware, and time requirements. Compared to similar works in the literature, the focus of this work is not only to evaluate the performance of an exhaustive collection of methods on several datasets considering only the standard Recall@N. Instead, the metrics considered and the experiments conducted for this work aim at providing a complete overview of the requirements and the different use case scenarios that may influence the choice of a particular method over another. This result was achieved by designing and developing the entire framework following a strict modular architecture that allows the analysis of each element of a VG pipeline. The software development, the execution of the experiments, and the analysis of the results is the outcome of a group effort. The paper produced from this work is currently under submission to a top-tier Computer Vision conference. For this reason, all the tables and figures in this and the following chapters cite our paper [8].

The development of an effective benchmark is based on three design elements: the datasets, the metrics, and the software implementation. The first section of this chapter provides a detailed description of the datasets utilized. The following sections explain the VG pipeline implemented in the framework and the methodology adopted to conduct the training and the evaluation of the VG models.

3.1 Datasets

An important element in the design of a benchmark is the choice of the datasets used to evaluate the methods under analysis. When considering a dataset, it is essential to be aware that it represents a specific problem with its challenges and



(a) Pitts30k (b) St Lucia (c) San Francisco (d) MSLS (e) Eynsham (f) Tokyo 24/7

Figure 3.1: Geographical coverage of the datasets used in the benchamark. Figure from [8].

limitations. From this perspective, it becomes evident that considering a broad collection of datasets with various characteristics is essential for the benchmarking scope. An ideal dataset for VG should have the following properties: dense, large-scale, with GPS labels, and with temporal variations for training. Among the VG datasets in the literature, six distinct datasets with heterogeneous properties have been selected: Pittsburgh 30k (Pitts30k) [16, 145], Mapillary Street Level Sequence (MSLS) [146], Tokyo 24/7 [147], Revisited San Francisco (R-SF) [148, 149], Eynsham [49] and St. Lucia [150]. This collection of datasets embraces a wide variety of Visual Geo-localization scenarios. Their choice was meant to provide a set of datasets with different geographical coverage scales (as depicted in Figure 3.1) and different image types ranging from panorama views, i.e., 360° degrees adequately tiled and projected, also used for pose estimation, to front-view and phone images. Other factors considered in this choice were the density and the number of images that compose the datasets. Further details on the characteristics of these datasets can be found in Tables 3.1 and 3.2.

The methods considered in the benchmark were trained using Pitts30k and MSLS for their complementary properties. These two datasets establish two very different training settings for the models. Pitts30k presents a homogenous selection of images collected in the city of Pittsburgh. The images are obtained from the

	# Train DB/Q	# Val. DB/Q	# Test DB/Q	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	-	-	75k / 315	4.0 GB	panorama	480×640	phone	variable
R-SF	-	-	1.05M / 598	36 GB	panorama	480×640	phone	variable
Eynsham	-	-	24k / 24k	1.2 GB	panorama	512×384	panorama	512×384
St Lucia	-	-	1.5k / 1.5k	124 MB	front-view	480×640	front-view	480×640

Table 3.1: Overview of the typology and the number of images for each dataset employed in the benchmark. The images are categorized in: "panorama", images tiled and undistorted from 360° panoramic views; "front-view", only forward-facing view available; "phone", collected with smartphones. Table from [8].
*MSLS also provde a limited amount of sideways images.
** A small group of MSLS images has a different resolution. They are resized to 480×640 as done by [146].

Google Street View API and, for this reason, are pre-processed 360° panorama-views. The locations appear under stable weather conditions and in the daytime. This dataset serves to mimic the model's performances on a small-medium urban dataset, with its 10k database and 7.4k query images for training (see Table 3.1).

On the other hand, Mapillary SLS represents a large-scale global dataset that includes urban and suburban locations, collected under different weather conditions and times of the day. With a total of 1.6 million images, it is the largest VG dataset to this date. For the most part, MSLS' images are front-view images collected by frontal cameras in cars and curated from the Mapillary mapping platform. The images come from 30 cities across the world. Each dataset split is composed of images from a disjoint set of towns with respect to the other splits. Unfortunately, the GPS ground truths for the test set are not available to the public because they use them to hold competitions, such as the Facebook Mapillary Visual Place Recognition Challenge at ECCV2020. Following what is already done in the literature [151], the results are computed on the validation set. Although the use of the test set would be preferable, given the geographical diversity between the training and validation set, in which images come from distinct cities, this approach can still be accepted as an overall result for evaluating the results obtained on MSLS.

As highlighted in the work of [16], a training dataset for VG applications must contain images collected over a large period in the order of years. The idea behind this remark is to provide the learning algorithm precious data that it can use to discern which features provide valuable information or not and learn to extract representations invariant to the appearance changes in images depicting the same location at different times. Both Pitts30k and MSLS provide images collected over several years. In particular, the Pitts30k employs the Time Machine functionality of Google Street View, while MSLS uses data collected for seven years by the Mapillary mapping platform.

The other four datasets, i.e., Tokyo 24/7, R-SF, Eynsham, and St. Lucia, are

	Area (km^2)	**Perimeter (km)**	**Environment**	**Day/night changes**	**Long-term variations**
Pitts30k	0.615	3.42	Urban	✗	✓
MSLS	-	-	Urban + Suburban	✓	✓
Tokyo 24/7	2.1	5.8	Urban	✓	✓
R-SF	13.6	14.0	Urban	✗	✓
Eynsham	-	-	Urban + Suburban	✗	✗
St Lucia	0.69	3.5	Suburban	✗	✗

Table 3.2: Summary of different qualitative information about the datasets. With "Long-term variations", we refer to images within a timespan larger than one year. Table from [8].

employed in the framework to assess the generalization capability of the methods trained on Pitts30k and MSLS. As better detailed in the following sections, this dataset group represents a helpful and variegated evaluation tool to analyze the strengths and weaknesses of the image representation learned by the different methods. Figure 3.2 contains a qualitative example of the differences between query and database images and how they differ across the selected datasets.

Other than the six datasets employed in the training and evaluation protocol of the benchmark, some experiments in this work also examined the importance of using different datasets for the pre-train of the backbones in place of the standard ImageNet [39]. Specifically, these datasets are the two versions of Google Landmarks Dataset [29, 30], from the Landmark Retrieval literature, and Places365 [152], which is a famous dataset for classification.

3.1.1 Mapillary SLS

Mapillary Street Level Sequence (MSLS)[146] has been introduced to facilitate the development of life-long VG applications. MSLS spans 30 cities across the globe in urban and suburban areas. One of its objectives is to reduce the data bias favoring highly populated cities in developed countries that usually affect the VG dataset; for this reason, it contains images from places like London and Paris as well as Goa, Nairobi, and Manila. It provides many variations in geographical diversity, seasonal changes, time of day, viewpoint, and weather conditions. In this work, it has been used for both training and evaluation of the models.

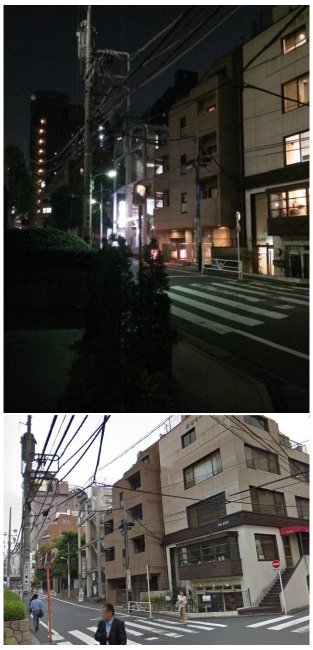
3.1.2 Pittsburgh 30k

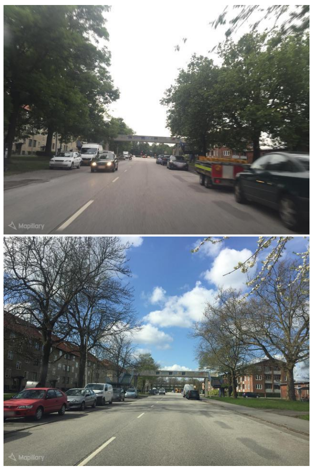
Pittsburgh 30k (Pitts30k) is a subset of the larger Pitts250k dataset from [145]. Pitts30k was proposed by Arandjelović et al. [16] as the training dataset of choice for their NetVLAD architecture. The images were collected from Google Street View by cropping the equirectangular panorama-view images into tiles and then applying

(a) Pitts30k

(c) San Francisco

(e) Eynsham


(b) Tokyo 24/7


(d) MSLS

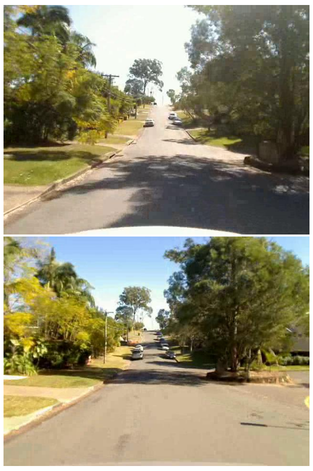

(f) St Lucia

Figure 3.2: Examples of query and positive images from the different datasets used in this work. Figure from [8].

a projection to alleviate the distortion. The dataset contains images gathered over a period of 2 years, but they do not contain other substantial domain shifts. In this work, Pitts30k was used for training and evaluation.

3.1.3 Tokyo 24/7

Tokyo 24/7 dataset was proposed by Torii et al. [147]. The medium-sized database is obtained from Google Street View. The query set contains a few hundred images manually collected with a smartphone that contain substantial variations in illumination and show structural changes in the scene. Some works [16, 153, 154] using this dataset employ the Tokyo Time Machine dataset (Tokyo TM) as a training dataset to overcome the limitation in the number of queries. However, in this work, Tokyo 24/7 is used only in the evaluation phase of the models.

3.1.4 Revised-San Francisco

The Revised-San Francisco (R-SF) is composed of a large-scale database of 1 million images collected with a mobile mapping vehicle equipped with a panoramic camera and a set of a few hundred queries collected with a phone camera [149]. Among the different studies in Visual Localization that provided queries' labels for this dataset, our choice for this work was the ones from [148].

3.1.5 Eynsham

The Eynsham dataset [49] is a traditional Visual Geo-localization dataset for robotics and autonomous vehicles. The grayscale images composing this dataset were collected by panoramic cameras placed on top of a car that ran across a circular loop passing through the city's streets and the countryside of Oxford. The whole dataset comprises images taken from two separate laps of the same path used as database and query photos in this work.

3.1.6 St. Lucia

St. Lucia [150], similarly to Eynsham, is an old-fashioned dataset collected traveling through the suburbs of Brisbane multiple times. The dataset is composed of front-view images taken by cameras positioned in the forward direction of the car. St. Lucia provides a high density per meter, then downsampled to one frame every 5 meters during our pre-processing step. The database and the query images are selected among the different laps provided. The images are all collected on a fine sunny day starting in the late morning and, for this reason, do not present substantial time or weather changes.

3.2 Methodology

3.2.1 Visual Geo-localization System

The benchmarking framework proposed in this work aims to create a standard, flexible and modular platform for evaluating and training Deep Learning methods for Visual Geo-localization tackled as an Image Retrieval task. To guarantee the flexibility needed to replicate several methods from the literature, the design of this system required a trade-off between the need for a straightforward setup and the possibility of integrating all the best-known practices.

The result of this design stage is the pipeline illustrated in Figure 3.3 that provides a wide variety of algorithmic and engineering choices. The architecture follows the abstract pipeline for an Image Retrieval approach, described previously in Section 2.3.1. The software developed mirrors the diagram. The idea behind the modular approach is to obtain a pipeline for which one can change each element without replacing the entire architecture. In this way, it is possible to pinpoint the impact of every switchable module at training and inference time. This framework allows to replicate several state-of-the-art methods from the literature [16, 10, 71, 74, 146, 155] and their training procedures [16, 146, 155].

Figure 3.3 highlights the different design choices that should be considered in the engineering of a VG application. During the training phase, the system selects

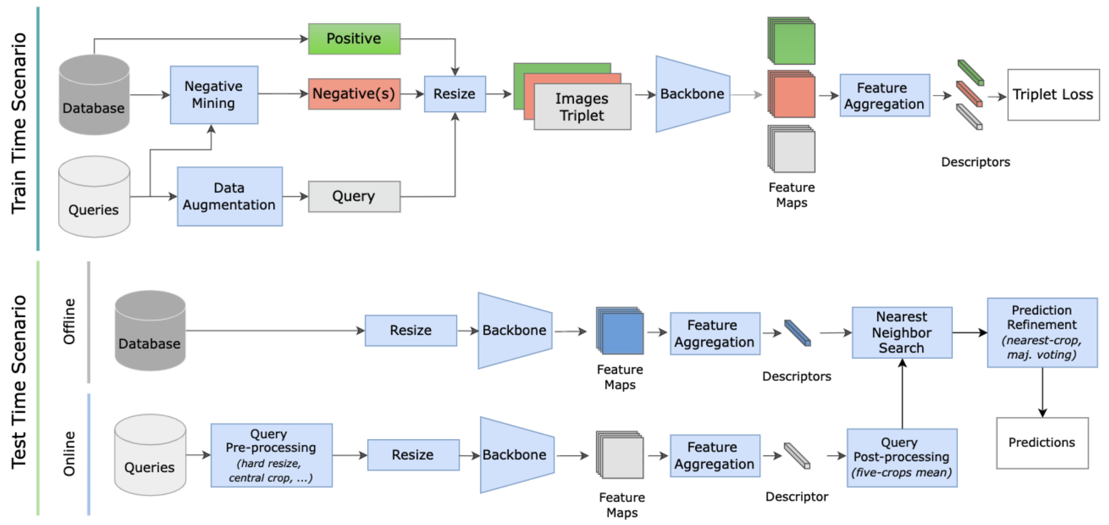


Figure 3.3: **Visual Geo-localization System.** The image represents the abstract VG pipeline adopted in the benchmarking framework. The diagram distinguishes between training and test time and describes all the elements involved in the system. All the light blue blocks are switchable elements that allow constructing different VG architectures. Figure from [8]

from the database a set of positive and negative images for each query through a **mining procedure** and proceeds with a learning procedure based on a triplet loss, further detailed in Sec. 3.2.2. For each triplet, composed of (i) the *query*, (ii) a set of *positive* images and (iii) a set of *negative* ones, the system extract image representation through a forward pass in a neural network **backbone** and a **feature aggregation layer**. The number of query images can be increased by using **data augmentation** techniques, and the entire triplet is subject to a **resize** operation.

At inference time, the VG system takes a new query with an unknown location, extracts its image representation, and then matches it with a shortlist of the most similar descriptors of the database's images. For performance reasons, the image representations of the database are computed offline and used for the similarity search with the descriptors of the query image for deployment. As for the training procedure, it is possible to select among different backbones and feature aggregation methods. Based on the hardware requirements, one can consider different image resizing strategies and the adoption of different **similarity search algorithms**. Most of the experiments presented in the following chapter rely on an exhaustive kNN, but it is possible to employ Approximate kNN algorithms that trade-off the accuracy of the results with computational efficiency and memory footprint. At test time, the proposed framework allows the use of **pre-processing** and **post-processing** methods that act to the query image and aim at improving the quality of the shortlist produced by the VG pipeline.

3.2.2 Experimental Protocol

The protocol adopted in the benchmark provides that the models are trained on MSLS and Pitts30k and then tested on all six datasets described in the previous section. All the architectures and methodologies evaluated in this work are trained by using a weakly supervised tripled loss [16, 146, 155]. The images in the datasets are labeled with their GPS location that is utilized as a form of weak supervision for the training algorithm. The GPS location gives no information on the orientation and perspective of the camera that took the picture.

The triplet loss aims at obtaining a model capable of extracting for pictures depicting the same location two descriptors that are close in the descriptor space; on the other hand, the descriptors of two images depicting different places should be adequately far away. At the same time, the triplet loss also wants to avoid the image representations collapsing into small clusters. Therefore, given a couple of images depicting the same place and the third one from another location, the idea behind this loss is to obtain a representation for the negative image that is farther away than a certain margin compared to the distance between the other two images. The mining procedure of the VG system produces a triplet composed of an anchor (i.e., the query), a positive, and a set of negative images. This procedure provides

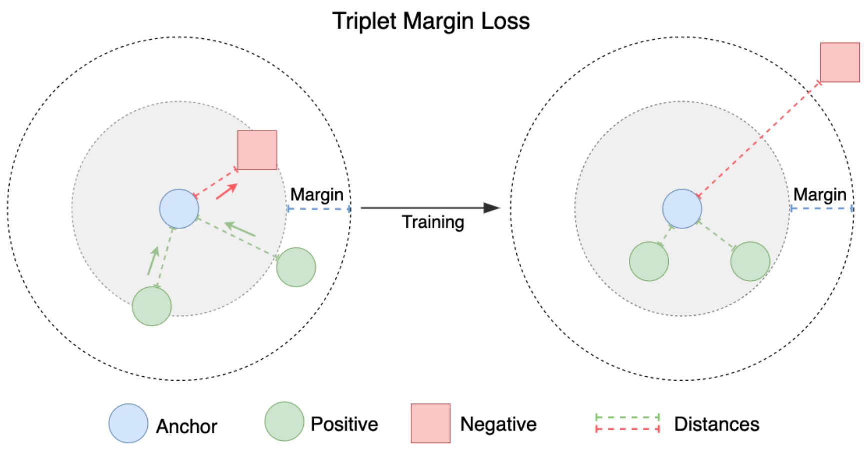


Figure 3.4: Illustration of the triplet loss.

the right and challenging examples that are fundamental to successfully training the model.

Based on the geographical distance obtained through the GPS information, the first step consists of producing for each query (i) a set of *potential positives* $\{p_i^q\}$, i.e., database images that are at most 10 m away from the query, and (ii) a set of *definite negatives* $\{n_j^q\}$, pictures taken further than 25 m. These two threshold distances represent the standard for the two training datasets adopted in this work [16, 146]. Let define the Euclidean distance in the descriptor space between two images as $d_\theta(I_1, I_2)$ and the query image as q. Then, the potential positives are further reduced to a single best matching positive image by selecting

$$p_{i*}^q = \underset{p_i^q}{\operatorname{argmin}}\, d_\theta(q, p_i^q), \quad \forall (q, \{p_i^q\}, \{n_j^q\}). \tag{3.1}$$

The set of definite negatives is composed only of a subset of the hardest negative images for the query. The "difficulty" is computed as the similarity with the query image in the descriptor space. The number of negatives is kept to 10 images throughout the entire work but can be modified as a hyperparameter. The ultimate goal of the triplet loss is to obtain an image representation for the elements of the triplet $(q, p_{i*}^q, \{n_j^q\})$ such that the distance between the positive image and the training query, $d_\theta(q, p_{i*}^q)$, is smaller than the distance between the query and all the hard negatives (see Fig. 3.4):

$$d_\theta(q, p_{i*}^q) < d_\theta(q, n_j^q), \forall j. \tag{3.2}$$

This idea leads to the definition of the weakly supervised triplet loss $\mathcal{L}_{triplet}$ for a training triplet $(q, p_{i*}^q, \{n_j^q\})$ as

$$\mathcal{L}_{triplet} = \sum_j \max\left(\left(\min_i d_\theta^2(q, p_i^q) + m - d_\theta^2(q, n_j^q)\right), 0\right) \tag{3.3}$$

where m represents the constant margin. Equation 3.3 is the sum of the individual hinge losses of each negative image n_j^q. When the query-negative distance is larger than the query-positive distance plus the margin, its contribution to the loss is zero; instead, if this condition is violated, it will contribute to the overall loss of the triplet proportionally.

The mining procedures adopted in this work are three (i) full, (ii) partial, and (iii) random mining, and they differ on the approach used to select the hard negatives for each query. Even for a small dataset like Pitts30k, considering all the database images for creating each triplet brings a prohibitive computational cost. The idea proposed by [16] is to introduce a caching mechanism that keeps in memory the image representations of the entire dataset for a certain number of iterations and updates the N_{neg} hard negatives for each query comparing them to 1000 random sampled database images. The value of N_{neg} is fixed to 10 for all the experiments. Unless otherwise specified, the training on Pitts30k is performed using the full database mining procedure, while for MSLS, it is used the partial mining for efficiency reasons. With this approach, instead of computing the descriptors for all the database images at each cache update, the algorithm samples 1000 random database images, computes their descriptors, and then identifies the negative examples for all the triplets.

Due to the different number of images in the training datasets, an epoch of the training procedure is defined as the forward pass of 5000 triplets. The number of epochs is not considered a hyperparameter because, after some preliminary experiments, the best option has been identified in using an early stopping strategy with a value of patience equal to 3 epochs based on the value of the Recall@5 on the validation set. The codebase gives the user the possibility to vary this value and also to choose among two different optimizers, Adam [156] and SGD. However, the experiments presented in the following chapter were conducted using Adam with a learning rate of $1 \cdot 10^{-5}$ and a batch-size of four triplets, i.e., the query image, a positive and ten negatives, for a total of 12 images per triplet. During the testing phase on the different evaluation datasets, the batch size for R-SF and Tokyo 24/7 is set to 1 to cope with the different resolutions of the images coming from phone cameras.

The metric used to measure the performances of the models under analysis on a particular dataset is the Recall@N (R@N), which computes the percentage of queries for which at least one of the top-N retrieved results is an image taken within a radius of a certain threshold distance, that for this work is set to 25 meters.

Chapter 4

Experiments and Results

This chapter provides a complete overview of the experiments conducted for the benchmarking activity. The analysis follows the architecture of the VG systems described in Sec. 3.2, for which an exhaustive set of configurations has been evaluated combining different choices for its modular elements. The experiments aim to reproduce most state-of-the-art methods in the Visual Geo-localization literature and provide the community with insights that highlight the good practices to follow under different scenarios in the development of VG applications.

The results presented in this chapter are grouped by the aspect of the VG pipeline under analysis. All the experiments are repeated three times and are reported with their standard deviation.

4.1 Backbones

All the VG systems relying on Deep Learning techniques adopt a neural network backbone for the extraction of highly discriminative features from the input images. In the context of the benchmarking analysis, we considered the approach followed by state-of-the-art methods which rely upon CNNs backbones. Their impact on the performance and the requirements of the overall system has been evaluated by trying different CNNs combined with the two most popular aggregation methods, i.e., GeM [71], which provides an extremely compact pooled representation, and NetVLAD [16], which instead produces a heavier aggregated representation with usually better performances. The works in the VG literature utilize a restricted number of CNN architectures (see [16, 10, 71, 74, 153, 146]), which can be identified in VGG-16 [157] and different versions of ResNet [107], i.e. ResNet-18, ResNet-50 and ResNet-101.

These different architectures provide different characteristics in terms of computational complexity that is measured in FLOPs, i.e., Floating-Point Operations. This metric is used to quantify the number of floating-point operations required to run a single instance of a given model. Other important factors to consider are the dimensionality of the image representations extracted by the combination of the backbone plus the aggregation layer and the model size. It is important to relate these factors with the performance in terms of R@1 over the different evaluation datasets considered to understand the importance of the backbone, but also which can be some relevant trade-offs that can lead to an efficient and robust VG system.

Table 4.1 contains the results obtained with the various combinations of backbones and aggregation layers. Note that all the models with a ResNet backbone do not utilize the full CNN architecture, but they are truncated to the `conv4_x` layer. This choice was motivated by the experimental results reported in Table 4.2, which demonstrate how this configuration represents the best trade-off in terms of the discriminative capability of the features extracted and computational complexity. The VGG-16-based models instead utilize all the convolutional layers of the architecture by removing the final pooling and fully connected layers.

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

Table 4.1: **Experiments with different backbones:** the Table contains the results and the requirements with different combination of CNN backbones and two aggregation layers (GeM and NetVLAD). Table from [8].

The outcomes of the experiments reported in Table 4.1 highlight some essential findings useful to make an informed choice for the backbone to employ in a VG system. In fact, a choice based only on the performance in terms of R@1 would suggest using the heavier and more complex ResNet-101. However, a better trade-off can be obtained by choosing the lighter ResNet-18 or ResNet-50. These two architectures may not obtain the highest performances in terms of R@1 but represent a considerably more suitable trade-off of performances and efficiency for a VG application.

Using a ResNet-18 backbone provides an exceptionally lightweight choice in terms of computational complexity with the lowest value of FLOPs and, at the same time, also leads to the most compact descriptor size compared to the other backbones. These properties are highly desirable for real-time VG applications. On the other hand, the ResNet-50 can be considered as a heavier alternative suitable for offline applications, where these requirements can be relaxed. For these reasons, these two architectures have been adopted as the two main backbones for the subsequent experiments.

The results of Table 4.1 are divided into two main blocks based on the dataset used to train the model. Observing the variability of the results among the different evaluation datasets, it becomes immediately evident how the training dataset plays a crucial role in the generalization capability of the models. As already pointed out, the two datasets have different characteristics that influence the retrieval performances over different datasets. The models using GeM are heavily penalized by the training with Pitts30k. For instance, using Pitts30k to train a ResNet-101 with GeM leads to a drop in the R@1 on St. Lucia of 30% compared to the same configuration trained on MSLS. These observations strengthen the argument that the discrepancy between training and evaluation datasets has a crucial role in the results obtained by the models. For this reason, it is necessary to remark how the use of different training settings cannot be neglected and leads to an unfair evaluation protocol when comparing directly different VG methods, as proposed in the work of [81]. Two possible counter-arguments to this statement could be: (i) that one may not be interested in the generalization capability of the models and (ii) that this uniform setting could be sub-optimal for certain techniques tailored to a specific setup. However, the flexibility offered by the proposed framework provides a level of granularity in the available choices that could be combined to create more specific setups. Moreover, it could also serve as a starting point for creating future automatic tuning systems for VG.

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

Table 4.2: **Analysis on ResNet Architectures:** the results show the impact on performances for Visual Geo-localization obtained truncating ResNet backbones up to `conv4_x`. Table from [8].

Throughout this work, all the models using the ResNet backbones refer to a truncated version of these architectures, up to the `conv4_x` layer, in place of the entire network, i.e., all the convolutional layers up to `conv5_x` (the nomenclature of the layers follows [107]). This choice is motivated by an experimental comparison of the two configurations exhaustively reported in Table 4.2. The results show how truncating the ResNet networks (both -18 and -50 variants), combined with the NetVLAD aggregation layer, produces the best R@1 performances overall. The cropping does not heavily affect the results when using the GeM layer, which instead achieves superior or at least comparable outcomes with the `conv5_x` version. Moreover, the feature maps produced in the `conv4_x` layer have half the number of channels of the entire network, leading to a more compact descriptor size that translates into a lower memory footprint and a reduced retrieval time for the VG systems.

4.2 Aggregation Methods

The literature on Deep Learning approaches for VG in the Image Retrieval setting proposed a large variety of aggregation techniques. These methods elaborate the feature maps extracted by the CNN backbones to produce an image representation used for retrieval. Compared to the previous section that employed NetVLAD and GeM with different CNN backbones, the objectives are (i) to evaluate a broader and exhaustive set of aggregation methods and (ii) to analyze the impact of descriptor dimensionality on the performances of the models.

The collection of aggregation techniques includes the Contextual Reweighting Network (CRN) [10] layer, which is a further development of NetVLAD. This aggregation layer performs a form of attention mechanism, different from the self-attentive approaches discussed in this work's later chapters. Using three convolutional layers with different kernel sizes, this method produces a reweighting mask for the soft-assigned features of a classic NetVLAD layer. Concerning the pooled representations, GeM is joined by MAC, R-MAC, and SPoC, all well-known techniques already introduced in detail in Section 2.4.2.2. Furthermore, the analysis also examines the recent Residual Retrieval Module (RRM) [158] layer, small residual fully connected block with LayerNorm [159] and L_2 normalization.

The choice between an aggregated representation and a pooled one impacts the dimensionality of the final descriptor. However, it is not unusual in the VG literature to encounter techniques that reduce or augment the descriptor sizes. Since the dimensionality of NetVLAD descriptors is in a range that spans from $16k$-dimensional vector for lighter architectures, as ResNet-18, to $65k$-dimensional for larger ones, even in the original paper [16] the authors propose the use of PCA learned on the training set. On the other hand, it is also possible to increase the descriptor size of compact ones by introducing a fully connected layer after the aggregation step, as done in the GeM's paper [71].

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 [73]	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 [72]	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	R-MAC [74]	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 [158]	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 [71]	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 [16]	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 [10]	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 [73]	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 [72]	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	R-MAC [74]	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 [158]	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 [71]	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 [16]	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 [10]	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 [73]	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 [72]	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	R-MAC [74]	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 [158]	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 [71]	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 [16]	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 [10]	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 [73]	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 [72]	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	R-MAC [74]	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 [158]	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 [71]	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 [16]	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 [10]	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**

Table 4.3: **Aggregation methods.** Full table of aggregation methods, grouped by backbone and features dimension. Table from [8].

Table 4.3 reports the exhaustive set of results obtained with all the aggregation methods mentioned before. The Table is organized into two blocks. The first half contains the results achieved with models trained on Pitts30k and the other half the ones trained on MSLS. Each of these parts is further subdivided into smaller sections based on the dimensionality of the descriptors to ease the comparison between models with similar requirements. The Table does not contain the FLOPs and model sizes since the effect of the different aggregation methods is negligible compared to the backbone. We can observe a recurrent trend in the block with pooling representation for each backbone-training dataset pair. In fact, among the pooled representations, the model with GeM performs better on all the evaluation datasets providing a more robust descriptor for the same dimensionality. Following this observation, GeM was used with an additional fully-connected layer to construct larger descriptors with size 2048.

Based on the backbone, we obtain different descriptor sizes. The pooling-based methods produce 256 and 1024-dimensional output vectors with ResNet-18 and ResNet-50, respectively. The same argument also applies to NetVLAD and CRN, for which, changing the CNN backbone, the descriptor sizes become 16384 and 65536. As already highlighted in the previous section's discussion, the training dataset for the models strongly influences their generalization abilities and overall retrieval performances. Using an FC layer to increase the GeM output size provides the best result for this type of configuration, especially when trained on MSLS since it resembles the large-scale datasets for which this method was initially proposed. One possible explanation for this behavior is the higher number of parameters introduced by the FC layer, leading to overfitting on smaller datasets such as Pitts30k. In this scenario, the models with NetVLAD produce more robust descriptors with and without PCA reduction. The reweighting mask proposed by CRN produces a model with a stronger domain generalization ability but requires an additional training step compared to NetVLAD. However, when training the models on the much larger MSLS dataset, the advantage of CRN and NetVLAD is outclassed by GeM on Tokyo and R-SF, probably due to the domain gap across the images in the training and test databases, which are panorama and phone-taking type, respectively.

Comparing NetVLAD+PCA and CRN+PCA performances with their vanilla counterparts, it is evident that the use of PCA, particularly when reducing the descriptor size considerably, yields a significant drop in retrieval performance. Furthermore, we must note that even if CRN obtains the most robust result across different evaluation datasets, it has the downside of requiring a two-stage training procedure, which requires two times the training time of a model with NetVLAD.

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

Table 4.4: **Memory Footprint Analysis.** The table shows a lower bound estimation of the memory occupation needed by a similarity search algorithm to store the test databases of the different datasets used in this work varying the descriptor sizes. Table from [8].

4.2.1 Memory footprint

An essential factor to consider in the development of a VG application is the dimensionality of the image descriptors. In the deployment phase of these systems, all the database descriptors are computed offline, kept in RAM, and efficiently indexed to perform the retrieval. The scale of the number of images in the database could then make some approaches prohibitively expensive in terms of memory footprint, and then it is critical to choose the right VG model that provides the best performance-scalability trade-off.

To get an idea, Table 4.4 contains the memory footprints of the test database of the six datasets varying the descriptor dimensionality. Using methods such as NetVLAD that with heavy backbones produce descriptors with a size of 65536, for large scale datasets like R-SF means a requirement of 256 GB of RAM. GeM or other pooled representations represent a much more suitable option for large-scale setup.

4.3 Mining Techniques

The training of the models used in this work is based on a weakly supervised triplet loss, see discussion in Sec. 3.2.2, that needs a triplet composed of an anchor (i.e., the query), positive and negative database images. The process required to generate those triplets is commonly referred to as mining. The quality of the images selected determines the positive outcome of the training procedure.

The positive images must possess two characteristics (i) their locations are within a predetermined distance radius from the query, and (ii) the pair of images must have a meaningful visual overlap. The first criterion can be addressed using GPS coordinates labels and fixing a distance threshold for the positive images. In this work, the threshold distance is set to 10 m, which represents the standard value applied in many works [16, 146]. The visual similarity of the images is considered by selecting the closer images in the descriptor space. While the lists of potential positives based on the distance are computed once and then kept in memory, the computation of the descriptors process requires the periodical update following the improvements obtained training the model.

The mining of negative examples presents numerous challenges. Following the definition of triplet's positive elements, all the negative images come from outside the distance threshold radius. The negative examples have a fundamental role in the learning of discriminative descriptors. If these images are "easy" for the model, it will be stuck in a "relatively good" representation still far from its true potential. In this case, most of the negatives will not violate the loss margin, and then it does not produce an effective training signal. Then, how to choose effectively and efficiently "hard" negative images? The authors of NetVLAD [16] answered this question by introducing a caching mechanism that works in this way. First, the mining starts by sampling 1000 random queries for which it should form the training triplets $(q, p^q_{i*}, \{n^q_j\})$; then computes the descriptors for all the database images and keep them in the cache. Then, for each one of the sampled queries and using the cached database image representations, it first computes the best positive (p^q_{i*}) and updates the hard negatives $\{n^q_j\}$ from a pool composed of (i) 1000 randomly selected database negatives and (ii) the previous ten hardest negatives. Since every 1000 iterations, this procedure computes the descriptors for the entire training database we refer to it as "*full database mining*". This mining method was initially designed for Pitts30k and, for this reason, it is perfectly suitable for medium/small-scale datasets. The main limitations are its efficiency and scalability issues.

The memory footprint and computational requirements needed are directly proportional to the database size. The authors of the MSLS dataset [146] address the mining of hard negatives by computing the descriptors only for a random sample of database images every 1000 iterations. This method is referred to as "*partial database mining*". The baseline for these two procedures is the "random database mining", which selects ten random negative database images for each query.

Table 4.5 presents the results obtained with these different mining methods, also varying backbone and aggregation methods. For the model trained with Pitts30k, the best outcomes are obtained with the use of full mining. It is interesting to notice how random mining produces only a 5% drop in R@1 over all datasets, while the gap between random and the other two procedures jumps to 12% when training on MSLS. This different behavior can be related to the substantial domain gaps in the latter dataset, compared to dense collected and homogeneous Pitts30k. The results reported on MSLS also highlights the issues in full database mining: (i) it was not possible to train models with high-dimensional descriptors, see the empty two rows with NetVLAD; (ii) even the models with GeM did not converge after five days of training on multi-GPU machines, compared to the other experiments requiring around 24 hours.

This analysis confirms that negative mining is a crucial element for training effectively a retrieval system with a metric learning approach. Moreover, the experiments also underline how the use of a full database mining approach is not the

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**

Table 4.5: **Mining methods.** Table from [8].

most suitable case for all the scenarios. On the other hand, partial mining can produce similar or even better retrieval performances requiring less computational time and memory resources.

4.4 Backbone Pre-Training

This section investigates the importance of the training and pre-training datasets for the VG systems by performing two analyses. The first one consists of using current pre-trained models for Landmark Retrieval for the VG task. Second, pre-training the networks for classification on Google Landmarks v2 [30] and Places 365 [152].

The first set of experiments employs the publicly available pre-trained state-of-art models from the repository [1] of [71]. These models use GeM and an FC layer to compute a 2048-dimensional descriptor. The datasets on which are pre-trained are Google Landmark v1 [29] and Sfm120k [160]. These models are compared to the same architectures trained on our usual training datasets. Note that higher R@1 values with the same descriptor size can be achieved using NetVLAD + PCA. The results of the evaluation on the six VG datasets are in Table 4.9, which clearly shows the benefit obtained by the highest number of images in the LR datasets

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

Source	Loss	Training Dataset	Backbone	Aggregation Method	R1 Pitts30k	R1 MSLS	R1 Tokyo 24/7	R1 R-SF	R1 Eynsham	R1 St Lucia
[71]	Triplet	GLDv1	ResNet-50	GeM + FC 2048	**84.1**	69.5	**77.8**	**76.4**	61.8	77.3
[71]	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**
[71]	Triplet	GLDv1	ResNet-101	GeM + FC 2048	**85.1**	72.4	**77.8**	**79.8**	61.6	83.4
[71]	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**

Table 4.6: **The role of the training dataset.** The table shows results with models trained on large scale landmark retrieval datasets. Table from [8].

Backbone	Aggregation Method	Dataset	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	ImageNet	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	GLDv2	Pitts30k	74.2 ± 0.4	30.9 ± 0.6	22.3 ± 1.9	20.4 ± 1.7	55.0 ± 2.0	43.3 ± 0.7
ResNet-18	GeM	Places 365	Pitts30k	**78.1** ± 1.0	**36.2** ± 0.9	31.8 ± 0.7	32.8 ± 1.6	**65.0** ± 2.1	**48.8** ± 2.1
ResNet-18	NetVLAD	ImageNet	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	GLDv2	Pitts30k	83.3 ± 0.5	39.9 ± 0.9	54.2 ± 2.3	41.1 ± 3.6	71.4 ± 2.6	46.8 ± 1.9
ResNet-18	NetVLAD	Places 365	Pitts30k	85.9 ± 0.4	**47.4** ± 0.6	57.9 ± 1.4	59.9 ± 3.2	**78.7** ± 0.7	50.4 ± 1.0
ResNet-50	GeM	ImageNet	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	GLDv2	Pitts30k	77.9 ± 0.5	35.2 ± 0.8	27.6 ± 2.1	37.2 ± 1.0	62.7 ± 1.6	48.4 ± 1.7
ResNet-50	GeM	Places 365	Pitts30k	**82.5** ± 0.4	**40.8** ± 0.3	41.3 ± 0.7	45.3 ± 0.6	**66.9** ± 1.3	**60.8** ± 1.6
ResNet-50	NetVLAD	ImageNet	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	GLDv2	Pitts30k	81.7 ± 0.6	43.5 ± 1.0	56.7 ± 0.9	54.1 ± 1.8	71.4 ± 0.6	42.3 ± 2.5
ResNet-50	NetVLAD	Places 365	Pitts30k	**86.2** ± 0.5	49.9 ± 2.0	66.3 ± 3.3	59.7 ± 3.5	75.4 ± 2.0	57.2 ± 5.5
ResNet-18	GeM	ImageNet	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	GLDv2	MSLS	60.7 ± 0.5	64.5 ± 0.7	30.9 ± 3.3	21.5 ± 0.8	79.2 ± 0.6	78.1 ± 1.0
ResNet-18	GeM	Places 365	MSLS	**71.6** ± 0.9	64.8 ± 1.1	36.6 ± 2.2	25.5 ± 0.3	80.1 ± 0.5	82.4 ± 0.6
ResNet-18	NetVLAD	ImageNet	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	NetVLAD	GLDv2	MSLS	73.3 ± 0.6	75.3 ± 0.3	53.4 ± 1.3	40.7 ± 2.9	86.1 ± 0.1	87.6 ± 0.9
ResNet-18	NetVLAD	Places 365	MSLS	79.7 ± 0.5	75.6 ± 0.2	61.5 ± 0.7	48.6 ± 1.5	86.5 ± 0.1	90.4 ± 0.4
ResNet-50	GeM	ImageNet	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	GeM	GLDv2	MSLS	71.1 ± 1.7	72.4 ± 0.2	47.6 ± 0.4	35.8 ± 1.6	84.0 ± 0.4	86.1 ± 1.1
ResNet-50	GeM	Places 365	MSLS	**78.2** ± 1.1	**72.7** ± 0.6	51.8 ± 2.7	41.8 ± 2.2	**84.4** ± 0.2	89.3 ± 0.8
ResNet-50	NetVLAD	ImageNet	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	NetVLAD	GLDv2	MSLS	74.7 ± 1.0	**77.4** ± 0.4	55.0 ± 1.7	45.4 ± 1.5	85.1 ± 0.5	87.7 ± 0.8
ResNet-50	NetVLAD	Places 365	MSLS	80.0 ± 1.1	75.6 ± 0.1	51.3 ± 3.3	44.8 ± 2.3	86.9 ± 0.1	91.3 ± 0.2

Table 4.7: **Pretraining the backbone on other datasets.** Table from [8].

allows the models from [71] to obtain robust image representations. As already discussed in Sec. 4.2, the use of Pitts30k to train a GeM+FC configuration is sub-optimal for its limited number of samples. MSLS alleviates this issue with its rich set of scenarios that provides the model with enough variability to obtain good generalization performances on Eynsham and St. Lucia. The same is not valid for Tokyo 24/7 and R-SF because their database images are collections of 360° panorama's pictures, and thus the front-view images from MSLS are not enough.

The objective of the second set of experiments is to analyze the impact of a different pre-training dataset for the backbones in place of ImageNet. An ad-hoc sub-module was developed to train various backbones for classification using GLDv2 and Places365. Given the high number of classes of the former dataset, the cross-entropy loss is replaced with the ArcFace loss [26]. However, the results of these experiments, shown in Table 4.7, are quite close to the one achieved with ImageNet, except for R-SF and Tokyo 24/7, where renouncing to the latter cause a drop in performance.

4.5 Approximate kNN and Inference Time

The time required by the VG system to retrieve the best matching candidates in the database starting from the input query image is referred to as *inference time* (t_i). From a computational perspective, this factor is the actual combination of (i) the *extraction time* (t_e), i.e., how long takes the extraction of the image descriptor, and (ii) the *matching time* (t_m), that is the time required by the similarity algorithm to retrieve the best database candidates.

Table 4.8 contains the values for the extraction times obtained varying different configurations of backbones and aggregation methods. The models that use GeM or NetVLAD without PCA show similar extraction times, with the second being slightly slower overall. Depending on the VG application, the downside of a simple NetVLAD layer lies in its large descriptor size, leading to a high memory footprint and longer matching times. However, introducing PCA to reduce the descriptors provides an additional overhead for the extraction compared to the plain version. One of the reasons behind this result is related to the `scikit-learn` implementation of PCA that runs only on CPU.

Aggregation	**VGG16**	**ResNet-18**	**ResNet-18**	**ResNet-50**	**ResNet-50**	**ResNet-101**	**ResNet-101**
		`conv4_x`	`conv5_x`	`conv4_x`	`conv5_x`	`conv4_x`	`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

Table 4.8: **Extraction time per image in milliseconds.** The table shows the time required to extract descriptors for each input image. Results are computed extracting descriptors from 10000 images and averaging the result. The images used have resolution of 480×640. Table from [8].

k-Nearest Neighbors is the principal similarity search algorithm used in the VG and IR literature. Considering the exhaustive kNN algorithm, the matching time per query depends on the database size, the descriptor dimensionality, and the number of queries to process in parallel. Figure 4.1 shows how the matching time depends on those factors. The image has a red dotted line that marks the extraction time required by a ResNet-101 with GeM. The matching time for a specific configuration becomes the bottleneck for the system when it surpasses that line.

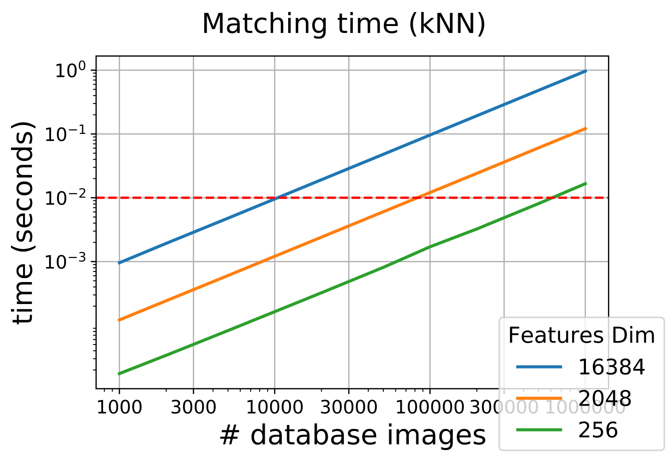


Figure 4.1: **Matching time for one query.** The plot shows the linear relationship between the number of database images and the matching time, varying the dimensionality of the descriptor. The red line in the graph indicates the extraction time required by a ResNet-101 with a GeM, i.e., ten milliseconds. When the matching time is higher than that value, then t_m becomes the system's bottleneck. As a rule of thumb, we can see that the exhaustive kNN is the bottleneck when the number of database images multiplied by the dimension of the descriptor exceeds 200M. Figure from [8].

The three plots obtained show that high dimensional descriptors can not satisfy many VG applications requirements for large-scale scenarios.

To address this issue, it is possible to adopt different indexing methods for the kNN, sacrificing some accuracy to obtain lower memory occupation and faster matchings. Among these various techniques, known as Approximate kNN (ANN), in this work, the following ones have been considered:

- *Inverted File Index* (IVF) [48]. This method aims to speed up the matching time by segmenting the descriptor space into several Voronoi cells. At test time, instead of comparing the query with all the entries in the database, the algorithm evaluates the cell with the most similar centroid and few neighbors. This method requires a training phase to compute the clusters. It also needs two additional hyperparameters: the number of Voronoi cells (fixed to 1000 here) and the number of neighboring cells to consider (1 and 10 cells in the experiments).

- *Product Quantization* (PQ) [75]. This algorithm performs a lossy compression of the vectors that reduces the memory footprint and approximates their distances. The product quantize decomposes each vector into smaller subvectors of dimension d (usually set to 8). Then the subvectors are assigned to the Voronoi cells of its subspace. The original vector is represented by a code composed of the index of the centroids of each subspace.

- *Composed Index IVF+PQ* (IVFPQ). ANN with PQ requires an exhaustive search among the database quantized descriptors. An inverted file computes the centroids for the quantized vectors and stores them in memory to reduce the number of instances considered during retrieval. This approach combines the lower memory footprint from PQ and the inexact search from IVF.

- *Inverted Multi-Index* [76].This approach uses PQ as a coarse quantizer, i.e., instead of splitting the database into Voronoi cells using the entire vector dimension, as done for IVF, it employs the multi-codebook idea from PQ. For IVF, there is a simple correspondence between each centroid and the list of database entries in that Voronoi cell. Instead, for Inverted Multi-Index, the set of centroids correspond to all possible tuples of codewords from the vector subspaces, leading to a denser subdivision of the search space.

- *Hierarchical Navigable Small World graphs* (HNSW) [77]. This approach utilizes a multi-layer graph representation of the vectors. The graph is explored during the matching time to find the nearest neighbors for the query. The number of neighbors considered in the construction of the graph is a tunable hyperparameter N_{neigh}.

All these indexes are available with the FAISS library [78] and can be easily applied in our VG pipeline. Figure 4.2 shows the results obtained varying the indexes and their hyperparameters for a model with ResNet-50 and GeM producing 1024-dimensional descriptors.

The most efficient indices in terms of the trade-off between matching time and accuracy are IVFPQ and Inverted Multi-Index, which provide a reduction in matching time of a factor of 20.
The most evident results come from IVFPQ applied to R-SF. This index reduces memory occupation of a factor of 64 (from 4GB to 64MB) and matching time by 98.5%. All these at the prices of a moderate drop in accuracy from 45.4% to 41.4%. The HNSW and IVF indices provide similar achievements as the inverted file index, with the first having a lightly slower computation but higher recall. Using larger values of the neighbors N_{neigh} leads to more accurate results but also uses more memory. The values used in the Figure for N_{neigh} are 4, 16, 64 and 256.
The simpler PQ index provides the same memory footprint as IVFPQ but has a lower accuracy-matching time trade-off.

This analysis provides valuable alternatives to the full kNN search that must be evaluated based on the downstream application. The experiments clearly show that using an IVF, possibly with a convenient number of neighbors, provides a good speedup without sacrificing the R@1 performance. For real-time applications or embedded applications, time and memory constraints are critical. For this reason, IVFPQ and the Inverted Multi-Index are preferable choices.

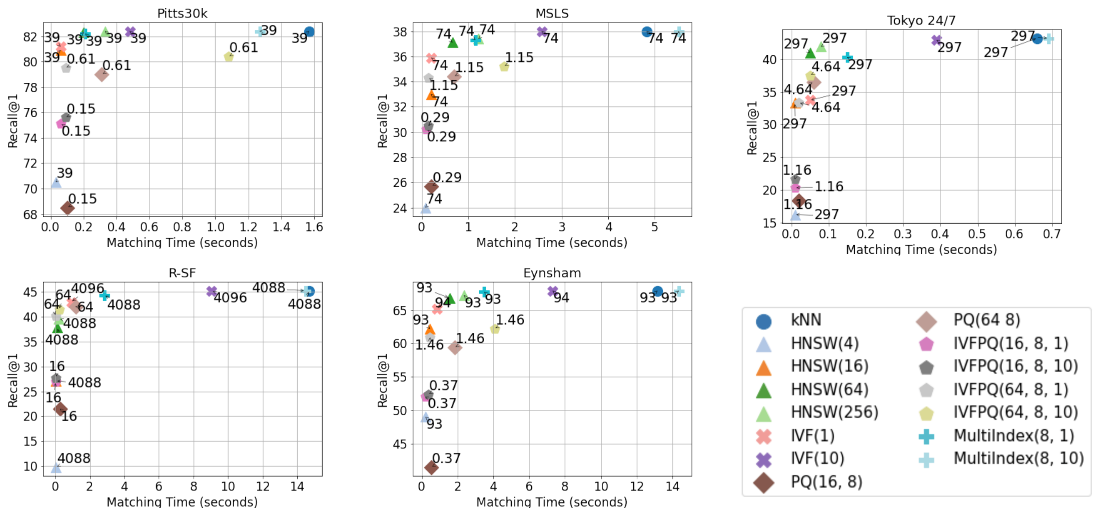


Figure 4.2: **Optimized kNN indexing.** The plots show several efficient kNN variants on different datasets, using 1024-sized descriptors extracted with a ResNet-50 + GeM trained on Pitts30k. The x-axis is the matching time in seconds for all test queries in each dataset, while the y-axis is the recall@1. The number near each dot is the RAM requirement for each method in MB. Besides exhaustive kNN, we employ Inverted File Indexes (IVF) [48], Product Quantization (with and without Inverted Indexes, respectively PQ and IVFPQ) [75], the Inverted Multi-Index (MultiIndex) [76] and Hierarchical Navigable Small-World graphs (HNSW) [77]. The legend shows the parameters for each method. The last parameter of IVFPQ, MultiIndex and IVF, which is either 1 or 10, represents the percentage of Voronoi cells to search, given that the search space has been split into 1000 Voronoi cells. Figure from [8].

4.6 Data Augmentation

Data augmentation is a simple but effective method widely used in Deep Learning to increase the number of training samples to train the models and therefore obtain better performances. Images are high dimensional and include an enormous variety of factors of variation that can be easily simulated, like alterations in brightness, contrast, or geometric transformations. In the context of VG, this technique is rarely taken under significant consideration in research papers. The goal of the experiments presented in this section is to analyze which could be the most effective techniques and if their application generalizes well over different datasets. For obvious reasons, the augmentations are applied in the training phase only to the queries, except for the random horizontal flipping applied to the entire triplet.

Figure 4.3 reports the results obtained with common data augmentation techniques used to train a model composed by a ResNet-18 with NetVLAD on Pitts30k.

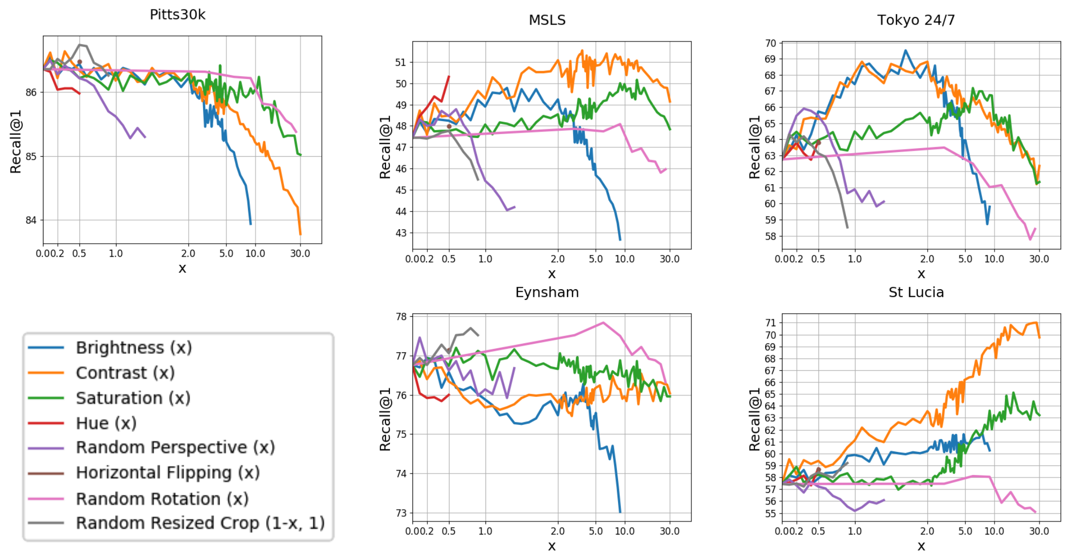


Figure 4.3: **Data Augmentation.** This Table contains the results obtained applying several popular data augmentation techniques during the training procedure. The methods used rely on PyTorch's transforms classes. The x values in the plot are scaled to have that for $x = 0$ the identity function is applied to the images. Augmentation techniques acting on brightness, contrast, saturation, and hue are performed with `ColorJittering()`. For `RandomResizedCrop()`, we use the value as $(1-x, 1)$ for the `scale` parameter and then the crops are resized to the original image resolution. In `RandomPerspective()` and `RandomRotation()`, the parameter refers to the `distortion_scale` and `degrees`, respectively. `HorizontalFlipping()` is applied with a probability of 0.5. The reader is referred to PyTorch's documentation for more information on the different functions. Figure from [8].

All the augmentations are implemented in the PyTorch library. The results on Pitts30k, which also represents the training dataset for these experiments, are not significantly influenced by augmentations, probably because of its homogeneous images. Even if they do not increase the performances for Pitts30k, they also do not worsen the results either. The difference in the models trained in this fashion becomes apparent when evaluating the model on unseen datasets. Looking at the Figure, it is interesting to notice how simple color jittering methods that vary at training time the contrast, brightness, and saturation of the queries can lead to more robust descriptors. For instance, random contrast modifications with its parameter set to 2 improve the R@1 performance of the base model of 3% for MSLS and 5% for Tokyo 24/7 and St. Lucia, with a slight drop for Pitts30k and Eynsham.

Summarizing, this analysis shows that there is not an augmentation technique that works better than the other alternatives for all the evaluation datasets. However, note that the random horizontal flipping (a single point in the graph at for x equal to 0.5, i.e., the flipping probability) and the random resized crop of the query images lead to improvements in the performances of all evaluation datasets. The color jittering techniques are most beneficial for Tokyo 24/7 and MSLS because they are the two datasets containing the highest number of domain variations. On the other hand, these methods are not helpful for Eynsham, which is composed of grayscale images.

4.7 Pre/Post Processing and Prediction Refinement

This section is devoted to analyzing various pre-/post-processing and prediction refinement techniques used in a VG pipeline to mitigate the approximation introduced by ANN algorithms or refine the shortlist of location hypotheses to improve the final performance. Another relevant scenario to address considering these methods are possible application scenarios where the dimension of the images at test time differs from the images on which the model was trained. Some works in the literature [16, 71, 74] when faced with datasets addressing this scenario, like R-SF [148, 149] or Tokyo 24/7 [147], address the problem by using a batch size of one image. This solution provides good results but requires a higher extraction time per query, which could be an issue for some downstream tasks. The experiments presented in this section use different engineering solutions that enable the computation of multiple queries in parallel and improve retrieval performance.

Based on the stage of the VG pipeline in which the method operates, we can classify these approaches into pre-processing, post-processing, and predictions refinement. The first group is composed of the following techniques that process the images before they are fed to the model. *Hard Resize* performs an anisotropic scale of the query image to resize the query to the same resolution of the database images. Note that there are datasets, like Pitts30k, Eynsham, and St. Lucia, where the resolution of database and query images match, for which this method does not apply any transformation to the images. For *Single Query*, the picture is resized, maintaining its aspect ratio such that the shortest side of the input image is equal to the smaller dimension of the database image resolution. In this way, the images cannot be fed into the network in batches if their original sizes differ. *Central Crop* obtains from the query a central crop of the size equal to the resolution of database images. If the query is smaller than this size, then it is adequately reshaped before cropping it. *Five Crops* extracts from the query five squared patches with side length equal to the shortest dimension of the database images.

The post-processing methods are the ones applied before the similarity search on the descriptors. With *Mean* the output descriptor is computed as the mean of the representations extracted from multiple patches of the query.

The last two methods covered are prediction refinement techniques. In the

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	✓	✗	✓	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	✓	✗	✗	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	✓	✗	✓	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	✓	✓	✓	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	✓	✓	✓	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	✓	✓	✓	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	✓	✗	✓	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	✓	✗	✗	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	✓	✗	✓	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	✓	✓	✓	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	✓	✓	✓	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	✓	✓	✓	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	✓	✗	✓	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	✓	✗	✗	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	✓	✗	✓	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	✓	✓	✓	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	✓	✓	✓	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	✓	✓	✓	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	✓	✗	✓	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	✓	✗	✗	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	✓	✗	✓	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	✓	✓	✓	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	✓	✓	✓	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	✓	✓	✓	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

Table 4.9: **Query pre/post-processing.** Results with different pre/post-processing methods applied at test time. The batch parallelization column indicates if images have to be processed one by one or if they can be stacked in a batch for parallel computation. Table from [8].

framework, they work together with the Five Crops method. The model extracts from the five patches their descriptors and for each one of them computes a shortlist of retrieved matching images. From these results, when using *Nearest Crop*, the predictions are generated by taking the images with the minimum distance from at least one patch. With *Majority Voting*, the final shortlist is obtained through a majority vote mechanism that depends on the distances of the top-20 retrieved elements for each input patch.

Table 4.9 contains the results obtained by using different configurations of backbones and aggregation methods trained on Pitts and tested with the techniques described before. As already explained, for Pitts30k, St. Lucia, and Eynsham, queries and database images have the same resolution. For this reason, the three pre-processing methods (Hard Resize, Single Query, and Central Crop) are the same and do not modify the input image. On the other hand, datasets like Tokyo 24/7 and R-SF that have images with different resolutions in their test queries on average benefit from the use of more complex techniques like Nearest Crop and Majority Voting. The Table shows how this trend is more evident with more robust models.
These methods should be evaluated with particular attention to the downstream application. For instance, the images will likely come from the same camera device during operation for robotics or autonomous driving scenarios, and then there is

no need to resize the queries. When instead the resolution of the images can take very different values, Single Query could be the simplest solution to use or used as a helpful baseline. A more robust solution could be Nearest Crop, which produces the highest R@1 performances and can process a batch of queries.

Part II

Sequence-Based Visual Geo-localization

Chapter 5

Sequence-Based Visual Geo-localization

All the methods and techniques discussed in the previous chapters addressed Visual Geo-localization working with single images. The VG system receives a query and produces a shortlist of database images that represent the candidate locations. In this type of setting, the queries and the database entries are single images. Another common ground among all the configurations analyzed is that they employ CNN backbones, the current state-of-the-art in VG literature.
This chapter aims to investigate two aspects still not entirely addressed by the VG research community: (i) the use and exploitation of multi-frame input data and (ii) to answer the question "can the new Transformer architectures provide interesting results in VG?". The use of short sequences of images depicting the same location or a small area is not new in VG, but the current approaches rely on traditional hand-crafted techniques or only partially integrate DL techniques. Moreover, last year, the Computer Vision community debated whether the visual Transformers could replace CNNs. This research trend originated in the NLP community, and following the development of different architectures has progressively interested a broader audience for their capability of model complex relationships among sequences. The idea behind this second contribution of this thesis is to develop and investigate new methods for Sequence-based VG (S-VG) and, at the same time, introduce and study the performances of visual Transformers and other self-attentive models for this field.

5.1 Sequence-Based VG

Many areas of application for VG, like mobile robotics and autonomous driving, have access to visual data streams. When moving within an environment, the same location can look very different, and also it is possible to identify some particular building or other landmarks that help in the localization process. As already introduced in Section 2.6, when dealing with video frames, the original VG task evolves into a continuous one in which the algorithms should be able to model the location of the single picture as well as the relationships between frames. Following the push from robotics applications, the community's efforts developed different families of hand-crafted approaches that use stochastic models, like FAB-MAP [91, 49], or similarity matrices, like SeqSLAM [93] and its derivative works. In the latter approach, the algorithm takes an input sequence of frames, i.e., the query, and builds a similarity matrix comparing each query frame with the database sequences. The resulting similarity matrix can be used to match a likely trajectory between the two sequences. A strong limitation of this family of methods is that they rely on the assumption that there is a one-to-one correspondence between the frames.

More recently, following the success of Deep Learning models for the single image task, Fácil et al. [104] propose to extract directly from the multiple frames a global sequence representation. This approach has been renewed with the introduction of the MSLS dataset by Warburg et al. [146]. The paper presents their new dataset and introduces the different settings in which it can be employed. They formalized three different scenarios that can be targetted with the introduction of sequential data. The authors define the following three approaches:

- *im2seq*, the query is a single image that should be matched with database sequences;
- *seq2im*, the reverse of the previous scenario, where the query is sequence and the database is composed of single images;
- *seq2seq*, both query and database are sequences of images.

Moving from the standard image to image (*im2im*) setting, used in the previous chapters, they also defined a positive match between sequences when they share at least one frame depicting the same location within a distance threshold of 25m at the test time and 10m during training [16, 146].

For this work, the analysis focuses on the ***seq2seq*** setting.The idea is to explore the potential improvements obtained by extracting sequence representations for queries and database sequences leveraging the multi-view information from each frame. However, for this setting, there is an important remark on the length of the sequences. Unlike other computer vision applications dealing with videos, the objective of a VG system is to estimate the location where the sequence was captured and use sequences too large and not densely sampled to cover a very extended area. For instance, if the sequence has images collected with a distance between each frame of 3 m, then using a sequence length of 15 frames covers an approximate distance of 50 meters, which is not ideal for the downstream application. Concerning future developments of the methods proposed in this work, it could be interesting for different downstream applications to modify this setting and the connected mining strategy to address different objectives. One example could be to predict the location of the last frame(s) of the sequence, i.e., to use a setting similar to the seq2im proposed by [146] but with a different definition of a positive match.

Before introducing the experimental setup and the architectures trained and evaluated for the S-VG task, the final remarks of this section aim to clearly state the objectives and motivations behind this second part of this thesis. The first question addressed is whether and how the introduction of sequential information affects the performance of the VG pipeline. As already mentioned, this approach to Visual Geo-localization is of interest for several applications that can exploit continuous sources of visual data and would take advantage of the hopefully more robust and precise sequence representation.

The second point is to introduce the Transformer architectures in the context of VG. In the short amount of time of the last year, several new networks welcomed the concepts and the ideas of this architecture and progressively became state-of-the-art not only in image recognition [161, 162, 163] but also in video understanding areas such as Action Recognition [144, 164]. There are no works in the VG literature that employ full self-attention models both in the traditional and sequence-based settings. Then, this work aims also to provide a guide and helpful insights for future works that will try to follow the same or similar approaches.

5.2 Dataset and Experimental Protocol

The topic of Sequence-based Visual Geo-localization as intended in this work is still not extensively studied in the literature. For this reason, also the datasets available for this task are limited. The vast majority of these datasets were proposed for robotics applications and used to train and evaluate the performances of hand-crafted or shallow methods. Among them, we find the Oxford Robotcar [165] and the Brisbane City Loop [166] datasets used in [105, 106]. Both datasets cover a limited geographical area of 10km around the streets of Oxford and 38km in Brisbane's area. None of these have geographical diversity, and since they are collected with the same camera-equipped cars, they do not have variations in the camera intrinsics. Moreover, the viewpoint, structural and weather changes are relatively small and do not provide enough variability for many downstream tasks. Another dataset used for this task by [104, 105] is the Nordland [167] dataset, which contains the images collected by the Norwegian public television (NRK) from front-camera places on trains that cross the Norwegian countryside, covering the same 182km journey four times, one for each season. The dataset is mainly composed of natural scenes and is unsuitable for training VG models on urban scenarios.

As anticipated in the previous section, the work of Warburg et al. [146] proposed the Mapillary Street-Level-Sequences (MSLS) dataset to contribute to the advancement of the VG community with a dataset instrumental in producing models that could achieve lifelong place recognition. In this sense, their dataset came with the possibility of generating training examples for the tasks described in the previous section, i.e., im2im, seq2im, and seq2seq. MSLS was extensively applied in Chapter 4 for the single image to image task proving to be fundamental for obtaining strong generalization capabilities for the models. MSLS was chosen as the dataset for conducting the experiments for this part of the work on S-VG.

With this dataset, it is possible to generate sequences of arbitrary length from 3 to 300 frames, offering great flexibility to evaluate the performances of the proposed models. Moreover, given the modular organization in different cities, it is possible to select only a limited number of towns. This allows performing training or testing of the performances in different scenarios, as done, for instance, by [105]. MSLS provides a large-scale domain variability that includes images collected from different cities across all the continents and variability in terms of illumination, weather, and time, all valuable characteristics to train robust and reliable models. The authors provide an open-source codebase implementing the partial database mining procedure for generating the triplets to train the models. The availability of this code should have helped in prototyping the S-VG pipeline; unfortunately, this work was not the case because the available code hid several issues that, in the end, required a complete reformatting of the entire dataset and its mining procedures.

Similar to the setting for the benchmarking experiments, the training procedure of the models used the weakly supervised triplet loss, requiring a single positive sequence and 10 negatives for each query, unless otherwise specified. A priori, this approach was justified by the results obtained in the single image architectures. As the experiments discussed in Section 6.9 will underline, the use of a high number of negatives is not mandatory for specific architectures, and future works in this task must exploit this advantage that considerably lowers the GPU memory requirements at training time. All the experiments of Chapter 6 were conducted using Adam [156] with a learning rate of $1 \cdot 10^{-5}$ and a batch-size of two triplets, each composed 12 sequences with F frames. The metric used to measure the models' performances under analysis is again the Recall@1 (R@1). This metric computes the percentage of queries for which the retrieved result is a sequence with at least one frame taken within a radius of 25 m from the frames of the query sequence.

5.3 Architectures

The aim of this section is to introduce and explain the architectures examined through this research analysis for seq2seq VG. Given the very few references in the literature and the absence of any previous work with many of these architectures for VG, this exploratory process resulted in several trials and errors. Not all the architectures produced the desired results, and some of them proved unsuitable for this task. However, after some iterations and following the helpful lessons learned in developing the benchmark framework, it finally led to satisfactory results and also highlighted some interesting insights for future developments.

5.3.1 Baseline Architectures

A comprehensive literature review was the initial step in the design process to produce an efficient and effective solution for the S-VG previously defined. However, the methods currently proposed in the VG literature are very few and follow two directions, or they mix DL approaches with hand-crafted techniques [105, 106] or rely on simple models that produce sequence descriptors [104]. The work of Fácil et al. [104] consists of straightforward adaptation of single image VG models to sequential input and is also already examined in the paper of MSLS [146]. As previously described in Section 2.6, they proposed three approaches to process the descriptors extracted with a ResNet-50 [107] backbone and a FC layer:

- *Descriptor Grouping*, which concatenates the descriptors produced from each image;

- *Descriptor Fusion* improves the previous method by adding a Fully connected layer that maps the concatenated descriptors to a more compact one;

- *Recurrent Descriptors* uses an LSTM [108] module to process the descriptors sequentially.

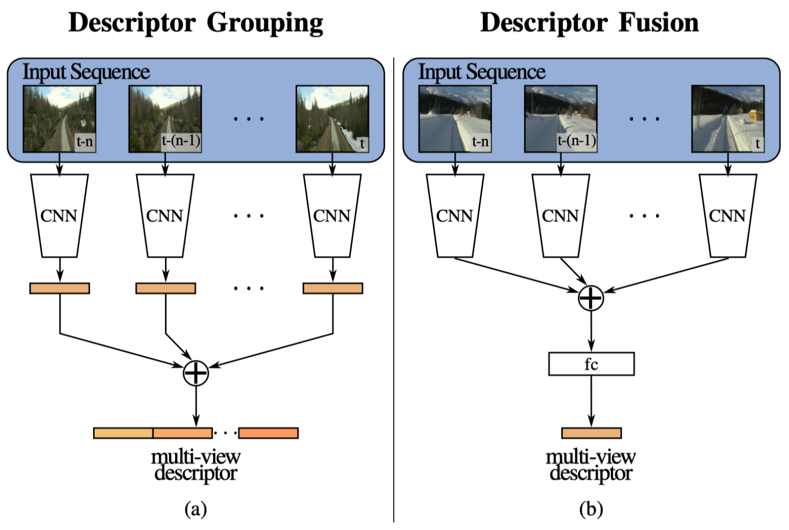


Figure 5.1: **MultiViewNet Architecture.** The two approaches, Descriptor Grouping and Fusion, employed as baselines for the seq2seq task. Figure from [104].

Building on the lessons learned in the development of the benchmark framework, instead of following the exact implementation proposed by [104], the choice on the backbone has gone to the ResNet-18, being the most efficient and lightweight architecture and, at the same time, able to produce state-of-the-art-results. These are fundamental characteristics given the high requirements in terms of memory footprint for processing many frames and the need for discriminative features to produce an accurate model. Then, the aggregation techniques employed for producing the descriptor for each image are NetVLAD and GeM. These configurations have been adapted to work on sequences.
In the analysis conducted in this work, the first two methods proposed by [104] are considered as baselines, i.e., Descriptor Grouping and Fusion. We decided not to include the third approach, which uses a recurrent neural network for multiple reasons. As pointed out by the paper's authors, this method performs worse than the other two methods in terms of recall@N. However, they keep this approach for its higher robustness when dealing with reverse or random order sequences. In these situations, simply stacking the descriptors of the sequence could be harmful due to the different order of the frames, then the use of an LSTM is partially beneficial. In this work, we preferred to focus on the best-performing methods because the aim is to achieve precise geo-localization accuracy, and therefore this model showing sub-optimal performances in standard scenarios does not reflect this purpose.

The other two works with a similar goal identified in Section 2.6 are Delta Descriptors [106] and SeqNet [105]. These two works rely on the use of handcrafted methods to process deep-learned image representations; in particular, they derive from the family of SeqSLAM [93] methods that use a similarity matrix search to match the sequences. These methods present different critical aspects because they rely on long-term sequence matching, i.e., two sequences must have many consecutive matching frames. Furthermore, they also assume a linear temporal correlation among the frames from the compared sequences.

For Delta Descriptors, the authors propose an interesting unsupervised adaptation method to obtain a more robust descriptor, starting from single frames and using a difference-based representation that incorporates information from the sequential nearest frames. After this process, the method employs the matrix similarity search to identify the positive matches. However, this method requires a high number of frames to achieve the proposed results, and then, given their different objectives, it has not been considered an appropriate baseline for this work.

On the other hand, the setting of SeqNet resembles the one adopted in this thesis. Despite this similarity, the method was not used as a baseline for two main reasons. First, their approach does not entirely align with the approaches considered in this thesis, i.e., end-to-end trained extractors of sequence representations. SeqNet can be seen as a two-stream prediction refinement technique that computes predictions based on off-the-shelf VG networks' descriptors. They use a pre-trained VGG-16 with NetVLAD from [16], which is neither trained nor finetuned but only used to produce the descriptors for each frame of the sequence. They rely on sequence-level descriptors, produced with Temporal Convolution [168], only for selecting a short-list of possible sequence candidates in the first stage of their architecture, later processed in the usual SeqSLAM-like approach with a similarity matrix search. Moreover, the authors do not provide the results on the MSLS validation set, but they only cherry-pick a restricted number of cities from this dataset. After a preliminary analysis, the results were comparable with the architectures described above, that couple single-frame descriptors with techniques proposed by [104], that therefore were considered enough representative.

5.3.2 Transformer Encoder

The Transformer architecture was initially proposed in the paper "Attention Is All You Need" by Vaswani et al. [117] for Neural Machine Translation, a sub-field of Natural Language Processing that aims at translating text between different languages. Besides producing significant improvements in translation quality, the authors provide a new architecture for many other NLP tasks, recently adapted also for Computer Vision.

The Transformer follows an encoder-decoder structure using stacking layers of

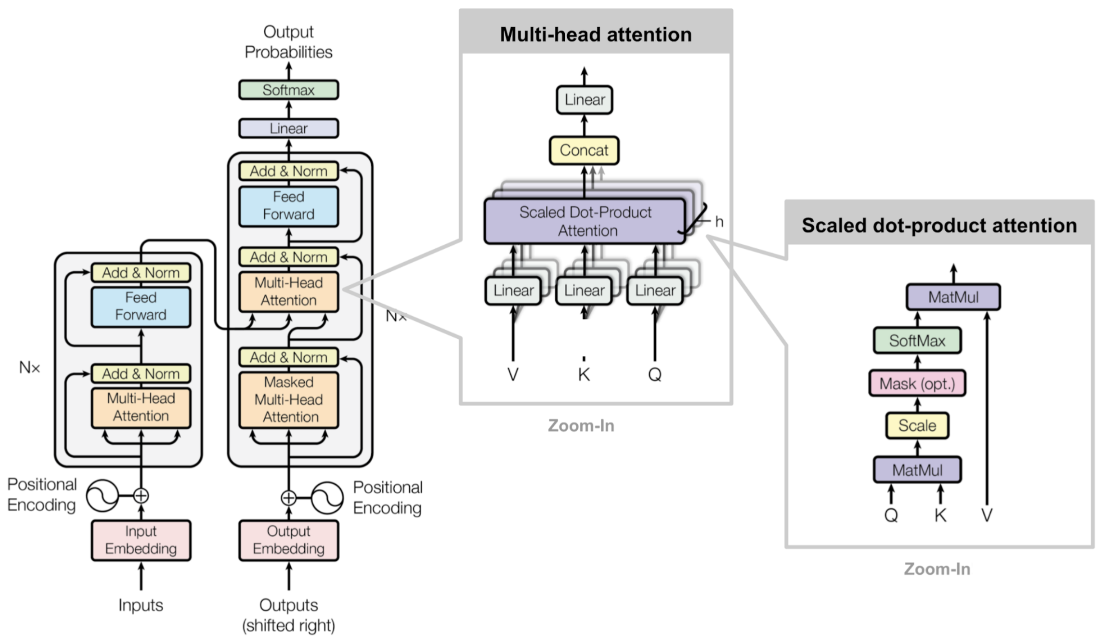


Figure 5.2: **Transformer Architecture.** The image depicts the complete Transformer architecture, with the encoder on the left and the decoder on the right. On the rightmost part, there are two zoom-in into the Multi-Head Self-Attention module and the scaled dot-product attention layer. Figure from [169].

Multi-Head Self-Attention (MHSA) and point-wise fully connected feed-forward networks (MLP), where point-wise means that the architecture applies the same linear transformation to each element in the sequence. The model is auto-regressive, meaning that it generates a new output token at each time step by consuming as input the previously produced symbols. Previous approaches in NLP use the last hidden state produced by an encoder composed of multiple stacked LSTM modules. Instead, the Transformer avoids sequential computations, typical of recurrent neural networks, and, instead of relying on single context vectors from the last hidden state of a recurrent encoder, creates shortcuts between the embedding representation and the entire source input, as shown in Figure 5.2.

The main innovative element introduced in this architecture is the Multi-Head Self-Attention mechanism. Given a sequence of n tokens $X = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n)^\top \in \mathbb{R}^{n\times d}$, where d is the dimension of each token, the self-attention mechanism models the interaction between all sequence instances. The idea is to compute a new embedding for each token that leverages the information of its interaction with the rest of the sequence, which translates into conveying the global context in the new sequence embedding. To achieve this objective, the first step consists of feeding the sequence X into a transformation layer that maps it into three different matrices Queries ($Q \in \mathbb{R}^{n\times d_q}$), Keys ($K \in \mathbb{R}^{n\times d_k}$), and Values ($V \in \mathbb{R}^{n\times d_v}$), where $d_q = d_k$ are the dimension of the queries and keys embeddings and d_v is the dimension of the values. These matrices are obtained by projecting the input sequence onto three learnable matrices:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{5.1}$$

where $W_Q \in \mathbb{R}^{d\times d_q}, W_K \in \mathbb{R}^{d\times d_k}, W_V \in \mathbb{R}^{d\times d_v}$. These embeddings are used in the *attention layer*, which performs the scaled dot-product attention. The output of this operation is a weighted sum of the value vectors, whose weights are computed on the fly as the dot product between the query and the key matrices, adequately scaled by $1/\sqrt{d_k}$:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V \in \mathbb{R}^{n\times d_v}. \tag{5.2}$$

Instead of relying on a single self-attention module, the Transformer architecture splits the inputs into smaller embedding vectors and computes the scaled dot-product attention in parallel over each embedding subspace. This approach allows the network to capture multiple complex relationships between the tokens that compose the sequence by attending to information that arises from the different representations subspaces at different positions. Finally, the output of the h self-attention heads are concatenated into a single matrix and projected onto an output weight matrix $W^O \in \mathbb{R}^{h\cdot d_v\times d}$. This operation can be formulated in the following way :

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{5.3}$$

$$\text{where head}_i = \text{Attention}(XW_i^Q, XW_i^K, XW_i^V) \tag{5.4}$$

where $i = 1, \ldots, h$ and the projection matrices are $W_i^Q, W_i^K \in \mathbb{R}^{d \times d_k/h}$ and $W_i^V \in \mathbb{R}^{d \times d_v/h}$.

The self-attention operation is permutation invariant, i.e., it processes the input sequence as a set without any notion of order. For this reason, Transformers require the injection of order information in the input sequence by the addition of positional encoding, i.e., a sequence of vectors with the same dimension of the input sequence ($E \in \mathbb{R}^{n \times d}$). The original Transformer implementation uses two types of positional encodings: the sinusoidal positional encoding constructed such that each dimension of E corresponds to a sinusoid of different wavelength; learned positional encoding, which is learned during training.

As previously mentioned, the overall architecture of the Transformer proposed by [117] is composed of two elements, i.e. the encoder and the decoder. In the original implementation, the encoder is a stack of six identical layers with a simple configuration of an MHSA layer and an MLP network connected with residual connections and Layer Normalization [159]. To facilitate the residual connections, the dimensionality of all the embeddings are the same, i.e. $d_k = d_q = d_v = d_{model} = 512$.
Again, the decoder is a stack of six layers, this time composed of two MHSAs and an MLP network that, like the encoder, are connected with residual connections and have the same dimension for the various embeddings. The first multi-head attention is masked such that the sub-layer cannot attend future tokens when predicting the current instance of the target sequence; the second, instead, takes as keys and values the tokens from the encoder and the query from the previous decoder sub-layer, in this way it can access directly in constant time the context of the entire source sequence. However, the complete Transformer architecture is not needed for this work, and therefore the decoder part is dropped to maintain only the encoder. The reason is that the task does not require to produce another different sequence as in the case of the original paper, but only to create a sequence representation that can be used in the following steps for retrieval.

The purpose of this section is twofold. First, introducing the Transformer encoder architecture is fundamental for understanding the functioning of the architectures discussed in the following sections. The second reason is that it was used directly as a component of one of the different architectures proposed for this work. Since this network can be fed only with sequences of vectors, i.e., the input tokens, it cannot work directly with images. Then, a vanilla transformer encoder was used in conjunction with a ResNet-18 and GeM to extract image descriptors from every single frame. In this way, the encoder receives a series of 256-dimensional vectors with the same length as the input sequence for each element of the triplet. The same type of configuration was not tried with NetVLAD because of the unfeasible memory requirements entailed by its high-dimensional descriptors. The vanilla Transformer Encoder tested follows the original configuration with six stacked encoder blocks, 16 attention heads per encoder, and produces a final descriptor with size 256 as the input GeM tokens.

5.3.3 Vision Transformer (ViT)

After gaining much popularity in many NLP tasks, the researchers in the Computer Vision community started to investigate the possibility of applying the Transformer architecture in this field. Even before the advent of Transformer-based architectures, the idea of attention and single-head self-attention was already investigated in some works; for instance, see the Non Local Neural Network [15] described in Section 5.3.6. Dosovitskiy et al. [13] show that it is possible to completely replace the convolutional operation in deep neural networks for large-scale image datasets with their model called Vision Transformers (ViT). The authors faced two main challenges: handling the input images directly and training this huge model effectively.

The self-attention mechanism requires a quadratic cost in the number of tokens, which means that it is unfeasible to use a naive approach to compute the self-attention between each pixel of the image. The solution introduced with ViT is to reshape the image $x \in \mathbb{R}^{H \times W \times C}$ into a sequence of flattened 2D patches $x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$, where (H, W) is the resolution of the image, C the number of channels, (P, P) the size of the image patch and $N = HW/P^2$ is the number of patches and, consequently, the length of the input sequence. The dimension of the tokens D is kept constant across all ViT layers, such that the architecture can process the images as sequences of D-dimensional tokens.

The second problem regards the training of the model. The MHSA mechanism of the Transformers has not the inherent inductive bias of the CNNs. The particular structure of the convolution operations provides these architectures with translation invariance and locality, which proved to be influential in the success of CNNs. On the other hand, Transformers-based models lack this type of inductive bias but can model more complex relationships in the data. Then, these models require more training data to learn how to deal with visual information to overcome this potential limitation. To achieve good results, the authors pre-trained the model on a large-scale image dataset, i.e., ImageNet-21k [170] and the proprietary JFT-300M [139] (4M and 300M images, respectively), and then fine-tune them on ImageNet-1k [39]. Unfortunately, the huge computational costs and the “data-hungry” nature of this architecture for training from scratch ViT are the two of the main issues that the subsequent works in this topic try to address.

The ViT architecture after mapping the input image into patches is the same as

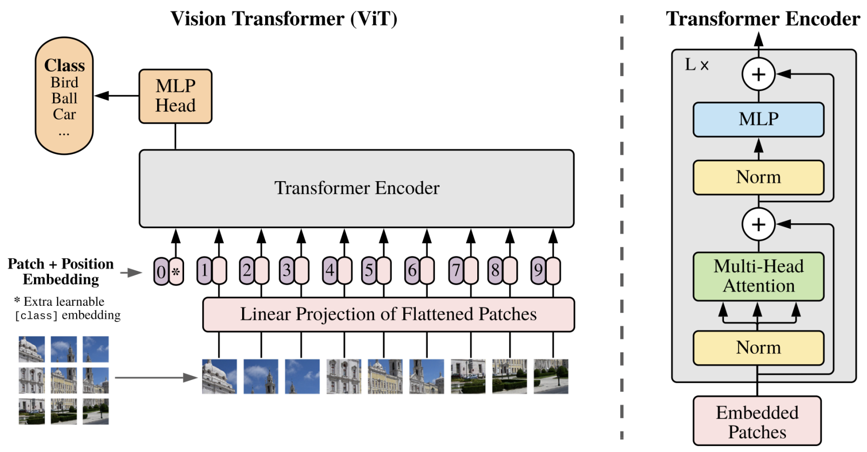


Figure 5.3: **ViT Architecture.** Model overview for the Vision Transformer architecture; the input image is split into fixed-size patches, later flattened and linearly embedded to process them as a sequence of tokens. A CLS token is prepended to this sequence, and then its output embedding is used for classification. The schematic representation of the layers composing an encoder block from the network is on the right. Figure from [13].

the traditional Transfomer encoder described in the previous section. Its output is a new sequence of tokens, then to obtain a compact representation, following what was previously done with the `[class]` token for BERT [130] in NLP, the authors prepend to the input sequence with a learnable vector, referred to as CLS token. The output state for this token is used as the image representation for the following downstream tasks, being classification in the original paper or retrieval in this work. The last remark on the ViT architecture concerns the positional embedding that, for this architecture, is a learnable 1D vector that considers the sequence of input patches in raster order, i.e., considering them one row after the other.

For the S-VG task, the ViT is employed as a frame-level representation extractor that uses the output state of the CLS token directly in place of other aggregation methods. Among the different configurations proposed in [13], the choice has gone to the Base-ViT, which uses 16×16 pixel patches, input images of resolution 224×224, and an output descriptor of dimension 768. To process the sequence of input images as for the traditional CNN backbones, following [104], the output representations are combined using concatenation or a fully connected layer.

5.3.4 Compact Convolutional Transformer (CCT)

The training from scratch of ViT models represents a significant obstacle in adopting these models for both research and industry practitioners that do not possess enough resources. In the paper of ViT, the authors argue that “Transformers lack some of the inductive biases inherent to CNNs, such as translation equivariance and locality, and therefore do not generalize well when trained on insufficient amounts of data"[13]. For these reasons, after the publication of ViT, many works attempt to come up with different solutions to address these limitations. For instance, the authors of DeiT [140] propose the use of distillation techniques that rely on a CNN teacher network to simplify the training of the Transformer model, but this approach requires the availability of CNN teacher models for the particular task.

Another line of research instead tries to inject in the architecture some inductive bias to avoid the model to learn it directly from the data. Among the various models that follow this path, there is the Compact Convolutional Transformer (CCT)[14], depicted in Figure 5.4, that aims to democratize the use of Transformer-based models by lowering the data and computational requirements using a different tokenization approach. CCT is a hybrid model that replaces the "image patching" and "embedding" techniques used by ViT with a Convolutional Tokenization. This module has a traditional CNN design comprising a convolutional layer, a ReLU activation, and a max-pooling operation. One or more of these blocks map the input image into a latent representation that offers more flexibility and efficiency than ViT. This approach reduces the number of model parameters and injects the inductive bias of the convolution operation that alleviates the need for a massive quantity of training data. The latent representation extracted by this module preserves the

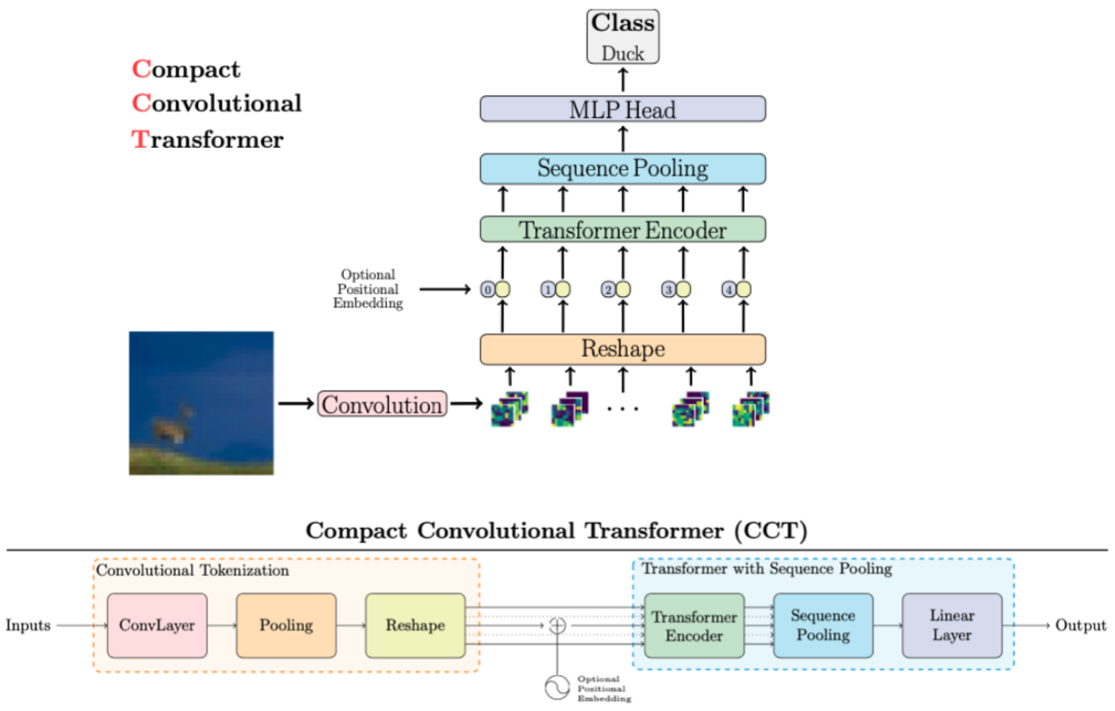


Figure 5.4: **CCT Architecture.** Figure from [14]

local spatial information and the relationship among close areas in the image. The convolutional filters share the same weights, then similar semantic elements in the picture will have a similar representation in the latent space. Furthermore, this tokenization makes positional encoding optional, allowing the flexibility of using an arbitrary number of tokens, with a limited impact on the final performances. This characteristic, paired with the fact that this approach does not require the input image to have a resolution multiple of the patch size, allows the model to process images with different resolutions without additional problems. After the Convolutional Tokenization layer, the CCT architecture follows the same architecture of a Transformer encoder as previously described for both the vanilla Transformer [117] and ViT [13]. CCT also introduces the use of a SeqPool layer that performs an attentive pooling over the output sequence of the Transformer, which avoids the need for the classification token.

In this work, CCT models have been employed as backbone networks to produce frame-level representations later processed with the SeqVLAD sequence-aggregator layer, introduced later in Section 5.3.7. Following the nomenclature used in [14], the configurations employed in this work are two versions of CCT-14/7x2, i.e., with a transformer encoder with fourteen layers and two convolutional blocks with kernel 7×7 for the tokenization. These models are pre-trained on ImageNet-1k [39] using two different image resolutions, i.e. 224×224, as ViT, and 384×384. To distinguish between them in the following chapter will be referred to as CCT-224 and CCT-384.

5.3.5 TimeSformer

ViT and CCT apply the concept of self-attention only to the spatial dimension of every single image that composes a sequence. However, as shown by Bertasius et al. [144] can be extended from the image space to space-time 3D volume of videos. Their work proposes a model called TimeSformer that adapts ViT to videos. This architecture process the video as a sequence of patches extracted from individual frames that are linearly mapped into an embedding representation and expanded with the additional positional encoding information. The resulting tokens sequence is then fed to a Transformer encoder that, compared to the traditional 2D or 3D CNNs, can model both global and local-range dependencies. However, the self-attention cannot consider the sequence entirely given the quadratic cost requirements. While for ViT in the Image Recognition scenario, this problem was addressed using image patches, in this case, the authors empirically evaluated different attention schemes that consider the tokens in both spatial and temporal axes with different granularities. These approaches are thoroughly described in [144], here the discussion focuses on the considerations that lead to the Divided Attention, which is the scheme that achieves the best results and offers a good tradeoff in terms of requirements.

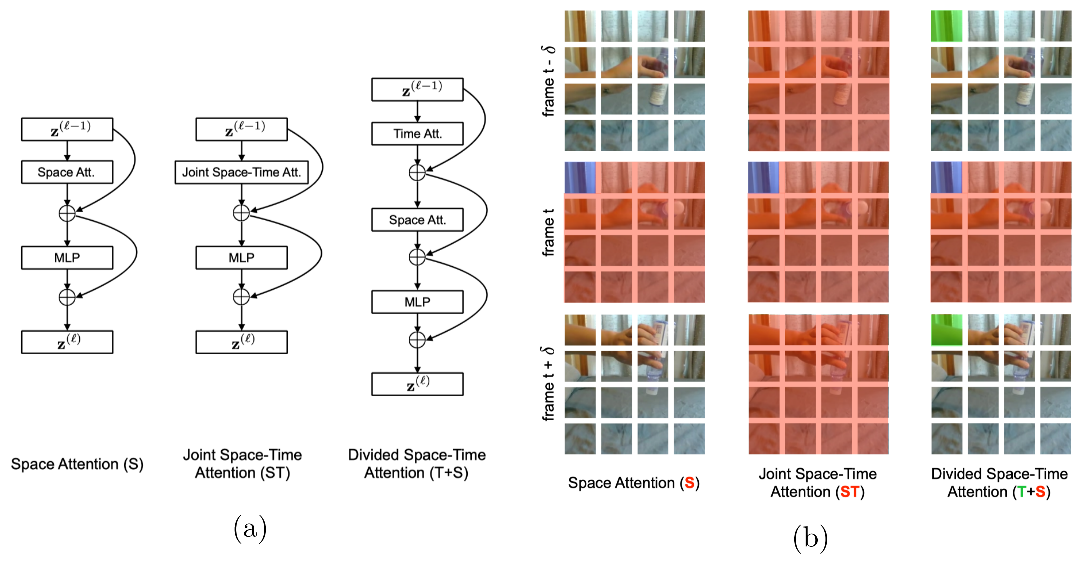

z^(ℓ−1)
Space Att.
MLP
z^(ℓ)
Joint Space-Time Att.
Time Att.
Space Attention (S)
Joint Space-Time Attention (ST)
Divided Space-Time Attention (T+S)
(a)
frame t - δ
frame t
frame t + δ
Space Attention (S)
Joint Space-Time Attention (ST)
Divided Space-Time Attention (T+S)
(b)

Figure 5.5: **TimeSformer Self-Attention Schemes.** Figure (a) contains a schematic representation of the different configurations used for the TimeSformers encoder blocks. Figure (b) depicts which patches are utilized by the different self-attention schemes using a short sequence of three frames as an example. Figures from [144].

Given an input video $X \in \mathbb{R}^{H\times W\times 3\times F}$, with F frames with resolution (H, W), each of them is divided into $N = HW/P^2$ patches of size $P \times P$. Each patch is identified by its position $(p = 1, \dots, N)$ within a frame and its temporal coordinate $(t = 1, \dots, F)$ in the input sequence. The flattened patch at position (p, t) is a vector $\mathbf{x}_{(p,t)} \in \mathbb{R}^{3P^2}$, that is then mapped into an embedding vector $\mathbf{z}^{(0)}_{(p,t)} \in \mathbb{R}^D$ by projecting it onto a learnable matrix $E \in \mathbb{R}^{D\times 3P^2}$ and summing the learnable positional encoding $\mathbf{e}^{pos}_{(p,t)} \in \mathbb{R}^D$:

$$\mathbf{z}^{(0)}_{(p,t)} = E\mathbf{x}_{(p,t)} + \mathbf{e}^{pos}_{(p,t)}. \tag{5.5}$$

The CLS token $\mathbf{z}^{(0)}_{(0,0)} \in \mathbb{R}^D$ is prepended to the generated sequence of vectors $\left\{\mathbf{z}^{(0)}_{(p,t)}\right\}_{\substack{p=1,\dots,N\\t=1,\dots,F}}$ to form the Transformer encoder's input. The encoder is composed of L stacked encoding blocks, at each block ℓ, for each embedding from the previous block $\mathbf{z}^{(\ell-1)}_{(p,t)}$ it computes the query, key and value vectors for the MHSA heads:

$$\mathbf{q}^{(\ell,a)}_{(p,t)} = W^{(\ell,a)}_Q \mathrm{LN}\left(\mathbf{z}^{(\ell-1)}_{(p,t)}\right) \in \mathbb{R}^{D_h} \tag{5.6}$$
$$\mathbf{k}^{(\ell,a)}_{(p,t)} = W^{(\ell,a)}_K \mathrm{LN}\left(\mathbf{z}^{(\ell-1)}_{(p,t)}\right) \in \mathbb{R}^{D_h} \tag{5.7}$$
$$\mathbf{v}^{(\ell,a)}_{(p,t)} = W^{(\ell,a)}_V \mathrm{LN}\left(\mathbf{z}^{(\ell-1)}_{(p,t)}\right) \in \mathbb{R}^{D_h} \tag{5.8}$$

where LN() denotes the LayerNorm [159], and $a = 1, \ldots, A$ is the index over the multiple attention heads. The latent dimension for each of MSHA is $D_h = D/A$. The formula to compute the attention weights, i.e., the result of the $\boldsymbol{\alpha} = \text{Softmax}(Q^\top K/\sqrt{D_h})$, it is possible to define different attention approaches, that are schematically represented in Figure 5.5. The weight vector obtained for each query patch (p,t) at layer ℓ in the attention head a is a vector $\boldsymbol{\alpha}^{(\ell,a)}_{(p,t)} \in \mathbb{R}^{NF+1}$. The self-attention weights computed considering jointly spatial and temporal patches can be written as:

$$\boldsymbol{\alpha}^{(\ell,a)}_{(p,t)} = \text{Softmax}\left(\frac{\mathbf{q}^{(\ell,a)}_{(p,t)}{}^{\top}}{\sqrt{D_h}} \cdot \left[\mathbf{k}^{(\ell,a)}_{(0,0)} \left\{\mathbf{k}^{(\ell,a)}_{(p',t')}\right\}_{\substack{p'=1,\ldots,N \\ t'=1,\ldots,F}}\right]\right) \tag{5.9}$$

This approach entails high computational costs and memory requirements that can be alleviated considering the computation of self-attention separately over the two axes. Following the notation used in this section, considering only the self-attention over the spatial dimension, which is the one used in ViT, can be expressed as:

$$\boldsymbol{\alpha}^{(\ell,a)space}_{(p,t)} = \text{Softmax}\left(\frac{\mathbf{q}^{(\ell,a)}_{(p,t)}{}^{\top}}{\sqrt{D_h}} \cdot \left[\mathbf{k}^{(\ell,a)}_{(0,0)} \left\{\mathbf{k}^{(\ell,a)}_{(p',t)}\right\}_{p'=1,\ldots,N}\right]\right) \tag{5.10}$$

This configuration does not consider the temporal dependencies among the frame that compose the sequence and therefore achieves the lowest results compared to the other attention schemes.

On the other hand, an attention scheme that considers only the temporal axis is:

$$\boldsymbol{\alpha}^{(\ell,a)time}_{(p,t)} = \text{Softmax}\left(\frac{\mathbf{q}^{(\ell,a)}_{(p,t)}{}^{\top}}{\sqrt{D_h}} \cdot \left[\mathbf{k}^{(\ell,a)}_{(0,0)} \left\{\mathbf{k}^{(\ell,a)}_{(p,t')}\right\}_{t'=1,\ldots,F}\right]\right) \tag{5.11}$$

The Divided Space-Time Attention scheme proposed with the TimeSformer model applies temporal attention from Eq. 5.11 and spatial attention from Eq. 5.10 one after the other. This scheme allows a more efficient approach, with only $(N+F+2)$ query-key multiplication per patch against the $(NF+1)$ required by the joint approach, and it also leads to better performances in the tests conducted by the authors on several Action Recognition datasets.

Applying the TimeSformer for Sequential Visual Geo-localization provides the advantage of using an architecture capable of directly dealing with sequences of frames without needing any sequence-level aggregator as for the previously described architectures. Even when using the Divided Space-Time self-attention, this architecture has the downside of requiring high computational and memory costs. Among the different configurations tried in this part of this work, the TimeSformer represents the heaviest model and, for this reason, it was possible only to apply it with resized 224×224 frames to fit in memory, given the high number of images required by the triples composing the training batches. However, the model produces a compact 768-dimensional sequence-level descriptor. Following the original implementation, the TimeSformer uses as initialization the weights from a Base-ViT pre-trained on ImageNet-21k, with 12 encoder blocks, and splits the image into 16×16 patches. In some experiments, the number of encoder blocks is reduced to 8 or 10 to evaluate the performances obtained with different network depths.

5.3.6 Non-Local Neural Networks

The work of Wang et al. [15] proposes a general definition for a differentiable non-local operation that can be introduced in any Computer Vision architecture to model long-range dependencies. This approach has many similarities with the self-attention mechanism proposed by Vaswani et al. [117], published around the same time, but its inspiration can be identified in the 2005's non-local means algorithm [118]. This technique is a classical de-noising filter for images that works with a mean of all the pixels composing the picture. This approach is in contrast with other algorithms developed for the same task, which instead consider only a group of neighbor pixels. Following the original non-local means operation, the authors of [15] propose a generic version for deep neural networks:

$$\mathbf{y}_i = \frac{1}{\mathcal{C}(\mathbf{x})} \sum_{\forall j} f(\mathbf{x}_i, \mathbf{x}_j) g(\mathbf{x}_j) \tag{5.12}$$

In contrast to the usual convolutional operation that relies on a local neighborhood of pixels of the input image or feature map, for the non-local operation, the output $\mathbf{y}$ at a generic position i relies on all the other elements of the input $\mathbf{x}$, note that input $\mathbf{x}$ and output $\mathbf{y}$ have the same dimensions. The function $f(\cdot,\cdot)$ that measures the relationship between two input elements and in Eq. 5.12 is used to compute the dependency of $\mathbf{x}_i$ with all the other elements in the input, i.e., for all other positions j. On the other hand, the function $g(\cdot)$ computes a representation of the input at position i.

The scaled dot-product self-attention is a special case of the general formula of Eq. 5.12. To see better this correspondence let use $f(\mathbf{x}_i, \mathbf{x}_j) = \theta(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$, i.e., the embedding version of the dot product similarity, and $g(\mathbf{x}_i) = \psi(\mathbf{x}_i)$, where the functions $\theta(\cdot)$, $\phi(\cdot)$ and $\psi(\cdot)$ are linear mappings that embed the input into queries, keys and values. These formulation allows different degrees of freedom, for instance, it can be modified using different pairwise similarity functions f and unary mappings g, as well as different normalization factors $\mathcal{C}(\cdot)$. The Transformer-based architectures differ from this approach for the introduction of MHSA and the use of only a particular form of attention.

This module represents an attractive alternative approach to Transformer-based models to introduce self-attention in the sequence-based VG pipeline. Its general

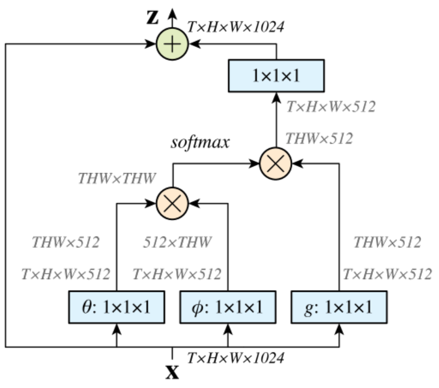


Figure 5.6: **Non-local module.** Schematic representation of the internal operations performed by a spatio-temporal Non-Local block. Figure from [15]

definition enables the formulation of several different self-attentive blocks. In particular, this layer was employed as a pluggable trainable element for capturing and modeling long-range dependencies with the ResNet-18 and ResNet-50 backbones and followed by different aggregation methods. In this work, three different non-local configurations have been tried:

- *Spatial Non-Local Layer*, this configuration works only with the spatial dimension, i.e., computing its non-local weights for each frame separately, exploiting the information from features extracted from different positions in the same image.

- *Temporal Non-Local Layer*, as the name suggests, this configuration uses for its computations for each element the features coming from the same spatial location across all the frames of the sequence.

- *3D Spatio-Temporal Non-Local Layer*, this last configuration relies on a 3D-convolution to encode in the internal computation for the self-attention mask together spatial and temporal information from the features extracted from the sequence.

5.3.7 SeqVLAD

This section introduces the SeqVLAD layer, an original contribution of this work that extends the capabilities of the widely known NetVLAD aggregator for sequences. The layer can also process both feature maps from CNNs and the output sequence embeddings produced by Transformer encoders. The S-VG task requires a sequence-level aggregation method tailored for the task. While the techniques presented in the previous sections rely on simplistic approaches as concatenation or fully-connected layers, or expensive approaches as the TimeSformer, SeqVLAD represents an ad-hoc layer that processes the entire sequence and has low memory requirements at the same time.

SeqVLAD arose from the necessity of a powerful aggregation technique that efficiently and effectively processes the information extracted by the different backbones. This layer exploits the idea of Vector of Locally Aggregated Descriptors (VLAD) proposed by Jégou et al. [51] that captures the information about the hand-crafted local descriptors of an image by computing the residuals of these descriptors with a set of visual words (or centroids or cluster centers). This approach produces a much richer representation than the previous Bag Of visual Words [48] methods.

The idea behind VLAD is that given N D-dimensional local descriptors $\{\mathbf{x}_i\}_{i=1}^{D}$ and a codebook with K visual words $\{\mathbf{c}_k\}_{k=1}^{K}$, the VLAD's output is a $(K \cdot D)$-dimensional vector. Let define the output as a matrix $V \in \mathbb{R}^{K \times D}$, that is flattened and normalized to produce the final VLAD descriptor. Then, each element of (j, k) of V is equal to:

$$V(j,k) = \sum_{i=1}^{N} a_k(\mathbf{x}_i)(x_i(j) - c_k(j)), \tag{5.13}$$

where $x_i(j)$ and $c_k(j)$ are the j-th elements of the i-th descriptor and the k-th visual word, respectively. The $a_k(\mathbf{x}_i)$ function denotes the hard-assignment of the local descriptor $\mathbf{x}_i$ to the k-th visual word. If the the k-th centroid is the closest one to the descriptor then the function is equal to one and it contributes to the residual computation, otherwise is equal to zero. As a result, each D-dimensional k-th column of V is the sum of all the residuals of the descriptors associated to that centroid. After the computation of all the residuals, the matrix V is intra-normalized (column-wise), flattened and then L_2 normalized again.

The authors of NetVLAD [16] proposed to extend the VLAD method to CNNs. In order to create an end-to-end trainable layer to process the extracted feature maps, they faced the challenge of modifying VLAD assignment to a differentiable operation avoiding the discontinuity of function $a_k(\cdot)$. They propose the following function that performs the soft-assignment with a softmax function:

$$\bar{a}_k(\mathbf{x}_i) = \frac{e^{-\alpha||\mathbf{x}_i - \mathbf{c}_k||^2}}{\sum_{k'} e^{-\alpha||\mathbf{x}_i - \mathbf{c}'_k||^2}} = \frac{e^{\mathbf{w}_k^\top \mathbf{x}_i + b_k}}{\sum_{k'} e^{\mathbf{w}_{k'}^\top \mathbf{x}_i + b_{k'}}}, \tag{5.14}$$

where the first equality assigns the weight of each local descriptor $\mathbf{x}_i$ to a cluster center $\mathbf{c}_k$ based also on its similarities with the other centroids. Computing the squares and setting $\mathbf{w}_k = 2\alpha\mathbf{c}_k$ and $b_k = -\alpha||\mathbf{c}_k||^2$, it is posible to separate the trainable parameters of NetVLAD into three independent sets $\{\mathbf{w}_k\}_{k=1}^{K}$, $\{b_k\}_{k=1}^{K}$, and $\{\mathbf{c}_k\}_{k=1}^{K}$. The formulation of NetVLAD can be written as

$$V(j,k) = \sum_{i=1}^{N} \bar{a}_k(\mathbf{x}_i)(x_i(j) - c_k(j)). \tag{5.15}$$

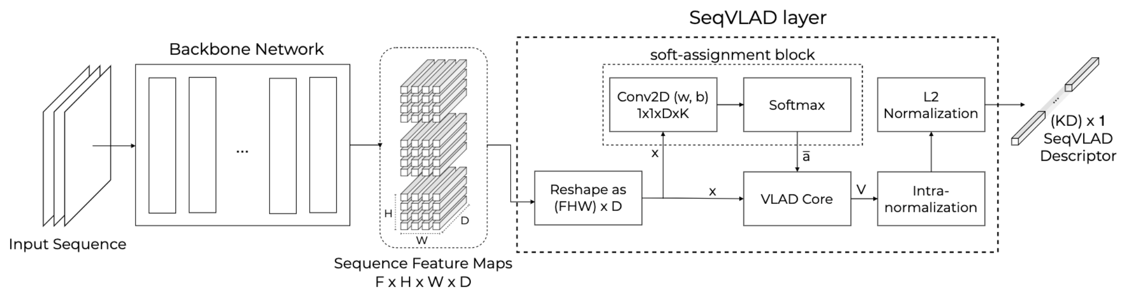


Figure 5.7: **SeqVLAD Layer.** The image shows the operations performed by SeqVLAD to produce its output descriptor. The VLAD core block is the operation described in Equation 5.15.

In this way, the layer learns not only from the data how to assign each feature vector to its centroids but also the centroids themselves producing a richer codebook.

When dealing with sequences, the CNN's feature maps output is a tensor of dimension $F \times H \times W \times D$, where F is the number of frames, D the number of channels, H and W the spatial locations of the D-dimensional features. Then SeqVLAD reshapes the tensors into $(F \cdot H \cdot W) \times D$ and processes all the features from the entire sequence together to produce the final sequence descriptor. Given the inner functioning of the VLAD core (Eq. 5.15), the dimensionality of the sequence descriptor is fixed to $(K \cdot D)$ regardless of the sequence length. Similarly, SeqVLAD can also be applied to the output embeddings of a Transformer-based architecture. In this situation, the network produces a tensor with dimensionality $F \times S \times D$, where S is the number of tokens produced for each frame. Then before computing the assignments, this sequence-wise tensor is reshaped to $(F \cdot S) \times D$ to let SeqVLAD consider all the embeddings extracted from all the frames composing the sequence.

SeqVLAD inherits from NetVLAD the soft-assignment operation and the learnable centroids that must be initialized before the training procedure. Another advantage of SeqVLAD over the other proposals described before is its remarkable flexibility that allows applying a model trained for a specific sequence length for other lengths. Moreover, this happens by producing a descriptor with a fixed dimension without introducing many computational and memory requirements, comparable to traditional NetVLAD.

This layer has been trained in conjunction with ResNet-18 and ResNet-50, with and without the use of Non-Local modules. Moreover, given the lower requirements, for the Transformer-based architectures, it was tested with CCT in order to be able to run more experiments in comparison with ViT.

Chapter 6

S-VG Experiments and Results

This chapter aims at discussing and showing the results obtained in the context of sequence-based Visual Geo-localization for the seq2seq task with the architectures described in Section 5.1. The analysis considers each technique's strengths and weaknesses, pointing out possible improvements that could impact its retrieval performance and lower the requirements in terms of memory footprint, training time, inference time, and computational cost. Given the limited number of works addressing the sequence-based VG, the experiments followed a trial and error approach to identify a possible architecture that could provide a satisfactory trade-off between results and requirements.
This chapter begins with a description of the results obtained with the baseline models that adapt the VG systems described in the first part of this thesis to the S-VG problem (Section 6.1). In order to introduce self-attention elements in the system to model the relationship among the semantic elements that emerge in the frames of the sequence, the first attempt is to apply a Transformer encoder as a sequence-level aggregator (Section 6.2). The following Sections 6.3 and 6.4 replace the CNN backbones with two Transformer-based architectures, ViT [13] and TimeSformer [144]. Given the high requirements of the previous networks, two different paths investigating the use of CNN in conjunction with Non-Local layers [15] (Sec. 6.5) and the hybrid approach of the Compact Convolutional Transformer (Sec. 6.6). These choices are the result of the considerations argued in Section 6.7 about the training costs of the models using multiple frames combined with the weakly supervised triplet loss [16] used for training. The following two Sections 6.8 and 6.9 inspect the performances provided by the use of models trained on the single-image task and the impact of different choices in the number of negative examples for the training triplets. Finally, Section 6.10 investigates the performances of different architectures modifying the order of the database sequences.

6.1 Baselines

Following the choice of the baseline architectures discussed in detail in Section 5.3.1, the results described here are produced by different configurations composed of a ResNet-18 cropped at layer `conv4_x` combined with GeM or NetVLAD to extract frame-level descriptors. They are later combined with a concatenation operation (CAT) or the use of a fully-connected layer, as proposed in the work of [104]. The experiments use short sequences with lengths 3, 5, and 7, justified by the idea of maintaining a reasonable geographical range for each place, and keep the images with their original resolution of 480×640 pixels.

Backbone	**Frame Aggregation**	**Sequence Aggregator**	**Descriptor Dim.**	**Train/Test Seq. Len.**	**R@1**
ResNet-18	GeM	CAT	768	3	73.6
ResNet-18	GeM	FC-512	512	3	71.6
ResNet-18	GeM	FC-2048	2048	3	73.9
ResNet-18	NetVLAD	CAT	49152	3	81.0
ResNet-18	NetVLAD	FC-512	512	3	68.9
ResNet-18	NetVLAD	FC-2048	2048	3	68.2
ResNet-18	GeM	CAT	1280	5	74.5
ResNet-18	GeM	FC-512	512	5	71.8
ResNet-18	GeM	FC-2048	2048	5	74.4
ResNet-18	NetVLAD	CAT	81920	5	80.8
ResNet-18	NetVLAD	FC-512	512	5	70.1
ResNet-18	NetVLAD	FC-2048	2048	5	66.9
ResNet-18	GeM	CAT	1792	7	75.2
ResNet-18	GeM	FC-512	512	7	72.9
ResNet-18	GeM	FC-2048	2048	7	74.3
ResNet-18	NetVLAD	CAT	114688	7	82.2
ResNet-18	NetVLAD	FC-512	512	7	66.8
ResNet-18	NetVLAD	FC-2048	2048	7	69.5

Table 6.1: **Baseline Results.** All the backbones are pre-trained on ImageNet. The number of negatives per triplet is 10, and the frame resolution is 480×640.

Table 6.1 contains the values of R@1 obtained with the different configurations. The main insights extracted from these results concern that GeM and NetVLAD keep the same performance gap evidenced in the benchmark experiments, with NetVLAD performing better but with the drawback of producing larger descriptors. Following this trend, the performances obtained with the larger descriptors using the concatenation approach also produce better results than the fully-connected layers. However, the use of fully-connected layers as sequence-level aggregation technique negatively impacts the retrieval performances of the configurations using NetVLAD if compared with the CAT approach. The configuration ResNet-18 at `conv4_x` and GeM produces a 256-dimensional descriptor for each frame. The concatenation of the single images' descriptors, even for the longer sequence with 7 frames, has a dimensionality below 2048, and the use of the fully-connected layer does not considerably affect the retrieval performance of the system. Another element is that the FC layers paired with NetVLAD, particularly for long sequences, require an increasing number of parameters that make the model heavier and more difficult to train.

The methods analyzed in this section are a straightforward adaptation of the single-image approaches that do not consider ad-hoc sequence-level aggregation techniques. As a result, they are not designed to capture complex relationships among the different frames and are not optimized to produce lightweight descriptors, which are desirable for real-time applications. These techniques should be considered a starting point towards more elaborate and tailored solutions that will be evaluated in the following sections.

6.2 Transformer Encoder

The first experiments considering a more elaborate sequence-level employ a configuration composed of ResNet-18 backbone, a GeM pooling layer, and a vanilla Transformer encoder to process the descriptors of the entire sequence. The output descriptor is the 256-dimensional output CLS embedding, obtained by prepending a learnable token to the input sequence composed of the frame-level descriptors. As shown in Table 6.2, this configuration represented a not encouraging starting point. The original idea was to propose a system relying on the lightweight CNN backbone with GeM to extract compact descriptors that are further processed with the Transformer encoder to exploit its self-attention mechanism to model inter-frame relationships.

Backbone	Frame Aggregation	Sequence Aggregator	Num. Encoder blocks	Train/Test Seq. Len.	R@1
ResNet-18	GeM	Transf. Encoder	3	3	18.8
ResNet-18	GeM	Transf. Encoder	3	5	18.1
ResNet-18	GeM	Transf. Encoder	6	5	16.8
ResNet-18	GeM	Transf. Encoder	8	5	18.3
ResNet-18	GeM	Transf. Encoder	3	7	18.0

Table 6.2: **Vanilla Transformer Encoder.** The results in this table are obtained using a vanilla Transformer encoder as a sequence-level aggregator for the 256-dimensional descriptors obtained with a ResNet-18 + GeM. Backbones are pre-trained on ImageNet, and the number of negatives per triplet is 10.

However, the experiments highlight that this configuration is not a viable approach and does not converge to satisfactory results. The first difference that may emerge comparing the overall system with other Transformer-based architectures is that the number of tokens used as input for the encoder is extremely low, i.e., equal to the number of input frames plus the CLS token. Inspired by the Convolutional Tokenization proposed by CCT [14], the idea was to circumvent this issue by increasing the number of tokens considering the feature maps produced by the ResNet-18 backbone. In particular, given the output features of dimension $F \times 256 \times H \times W$, where F is the sequence length, they are unrolled as $(F \cdot H \cdot W)$ tokens of dimension 256. Unfortunately, this solution required a lot of GPU memory for training and turned out with even worse results compared to the ones displayed in Table 6.2. Other possible motivations that could justify the failure of this configuration are that this encoder module requires higher amounts of training data since it started from a random initialization or that the triplet loss could not provide an adequate training signal. These results are reported for completeness and highlight the possible flaws in this attempt that must be considered in future investigations.

6.3 ViT

Taking a step back, the experiments described in this section, instead of addressing the sequence aggregation, focus on replacing the CNN backbone with the Vision Transformer (ViT) [13] architecture. Given the hardware constraints and the intrinsic memory requirements of the sequence-based VG, the ViT chosen for this task is the Base-ViT that works with 224×224 input images. They are further split into 16×16 pixels patches, later flattened and embedded into a sequence of 196 tokens plus the CLS token. The output descriptor size and the dimensionality of the internal embeddings are equal to $D = 768$. The model is pre-trained on ImageNet-21k [170].

Following the approach employed for the baseline models, the frame-level descriptors are combined using concatenation and an FC layer with an output dimension of 2048. As before, three different sequence lengths have been considered (3, 5, and 7).

The discussion of the results reported in Table 6.3 must be contextualized considering the pros and cons provided by this architecture. ViT is by far the most popular Transformer-based architecture for vision tasks, and then it seemed the ideal choice to begin an investigation of this family of networks. This architecture lacks the typical inductive bias of the CNNs and eliminates the need for a feature aggregation pooling layer. Unfortunately, ViT has a large number of parameters, i.e., 86 million parameters compared to the modest 11 million from a ResNet-18. On the one hand, this larger capacity leads the model to greater expressiveness and hypothetically more representative image descriptors. On the other hand, ViT has higher computational and memory requirements that hinder the number of experiments and usability in practical applications.

The results obtained with ViT, presented in Table 6.3, improve the ones produced by the GeM-CAT approach from the baselines. However, they achieve lower results than the baseline with NetVLAD plus CAT. For sequence lengths of 3 and 5 frames, the difference is around only 1-2%. The difference increases to 6% when considering a sequence length of 7 frames. However, in all these scenarios, it is

Backbone	Descriptor Dim.	Sequence Aggregator	Train/Test Seq. Len.	R@1
ViT	2304	CAT	3	76.0
ViT	3840	CAT	5	79.2
ViT	5376	CAT	7	76.5
ViT	2048	FC-2048	3	76.4
ViT	2048	FC-2048	5	79.6
ViT	2048	FC-2048	7	74.5

Table 6.3: **ViT Results.** The results obtained with Base-ViT with 16×16 patches and reshaping the input images to 224×224. All the results use 10 negative examples per triplet.

important to point out that ViT has a considerably lower dimensionality of the descriptors than the models with NetVLAD.
As for the baselines methods, these architectures suffer from the lack of a specialized sequence-level aggregator that could potentially provide an improvement on the retrieval performance, an issue that has been further investigated with the lightweight CCT architecture and the SeqVLAD layer in Section 6.6. As also emerged with other Transformer-based architectures in the following sections, future experiments should investigate the possibility of obtaining a gain in performance by cropping the ViT architecture and then reducing the number of encoder blocks.

6.4 TimeSformer

During the literature overview conducted on the available Transformer-based architectures, the TimeSformer [144] network stands out for its convenient design capable of handling both spatial and temporal information exploiting the Multi-Head Self-Attention mechanism. As already described in Section 5.3.5, this model derives from ViT, and, besides the empirical evaluation of the different self-attention schemes, the authors do not propose any particular design modification to lower the model requirements. For this reason, this approach is problematic for practitioners that do not have access to a massive quantity of GPU resources, and the experiments described in this section are limited only to the Divided Space-Time Attention scheme with an input frame resolution of 224×224 pixels. The TimeSformer model used in this section works with internal token embeddings of dimension 768, producing a sequence-level descriptor with the same size, and is composed of 12 encoder blocks with the Space-Time self-attention. The model uses as weight initialization the Base-ViT pre-trained on ImageNet-21k purposely adapted to the different attention scheme, which causes the downside effect of making the number parameters increase from 86M to 121M, making the models in this section the architectures with the larger capacity employed in this entire work. This reason, paired with the observation that for CNN models cropping the last layers of the architecture provides more expressive features, motivates the experiments investigating the possibility of truncating the architecture up to the 10th and the 8th encoder block.

Architecture	Truncated at block	Pretrain	Train/Test Seq. Len.	Margin	R@1
TimeSformer	-	ImageNet	3	0.1	78.9
TimeSformer	10	ImageNet	3	0.1	79.7
TimeSformer	-	ImageNet	5	0.1	81.2
TimeSformer	10	ImageNet	5	0.1	81.0
TimeSformer	-	ImageNet	7	0.1	82.0
TimeSformer	8	ImageNet	7	0.1	81.5
TimeSformer	10	ImageNet	7	0.1	85.2
TimeSformer	-	ImageNet	8	0.1	83.7
TimeSformer	-	ImageNet	8	0.5	82.7
TimeSformer	-	K400	8	0.1	82.8
TimeSformer	-	K400	8	0.5	83.8
TimeSformer	-	K600	8	0.1	80.6
TimeSformer	-	K600	8	0.5	83.5

Table 6.4: **TimeSformer Results.** Results obtained with the TimeSformer architecture with the Divided Space-Time self-attention scheme [144]. The experiments were conducted with 10 negative examples per triplet and resizing the input frames to 224×224. All the models produce 768-dimensional descriptors.

The model is trained and evaluated for different sequence lengths, i.e., the 3, 5, 7, and 8 frames. The latter is employed to compare the results obtained with the models initialized only with Base-ViT weight for ImageNet-21k with other TimeSformer from the GitHub repository [1] of [144] trained on standard Action Recognition datasets, Kinetics-400 (K400) [171] and Kinetics-600 (K600) [172].

Table 6.4 contains the results obtained with the experiments using the TimeSformer architecture. Comparing the results with the baselines from Section 6.1 it emerges that TimeSformers have a better retrieval performance for sequence lengths of 5 and 7 frames compared with the baseline. In particular, for 7 frames, it improves by 3% (R@1 85.2%) over the best baseline method. However, for shorter sequences of 3 frames, it reaches an R@1 of 79.7%, which is lower than the 81% obtained with NetVLAD+CAT.

The experiments using the weights of finetuned for Action Recognition task use sequence with 8 frames and unfortunately does not improve the results obtained with the Image-Net initialized models. Then, different values of the margin of loss

[1]https://github.com/facebookresearch/TimeSformer

have been tried to require the sequence representations of the positive pairs closer and push them further away from the representations of negative examples. This attempt led to higher results for the models pre-trained on K400 and K600 but still lower than the results obtained with the TimeSformer pre-trained on ImageNet. Regarding the latter, the same adjustment in the margin was tried for these models, but using a higher margin only worsened the final performances.

To summarize, the pros of using the TimeSformer model reside in the fact that (i) it does not require any aggregator layer, (ii) using its CLS output embedding produces a compact 768-dimensional sequence-descriptor, and (iii) it provides good retrieval performances. However, they come at the cost of a high requirement in terms of computational costs and memory requirements and cannot be used for different sequence lengths from the one on which the model is trained. As a result, this limited the number of experiments with this architecture and motivated the need for lighter models. Another possible track to follow in future developments could be to modify the original architecture of the TimeSformer with the introduction of a Convolutional Tokenization block, as proposed in the CCT architectures, that will produce a hybrid model. A model with these characteristics could benefit from the inductive bias of the CNNs and the flexibility of the Transformer architecture. Using the low-level features in place of image patches could also considerably lower the hardware requirements. In this case, the initial pre-train of the architecture could be addressed following the same approach of TimeSformer that uses a ViT architecture trained on ImageNet and then modifies its positional encoding to adapt it for the different self-attention scheme.

6.5 Non-Local Blocks

The results from the previous sections underline two main issues. The models that rely on simple sequence-level aggregators cannot completely exploit the information coming from the multi-frame setup. However, the use of a complex architecture, like the TimeSformer, demands high amounts of resources. Then, the following steps were motivated by two factors to reduce the overall requirements and design an adequate sequence-level aggregator that could provide a good trade-off in performance and versatility.

Given the high requirements of pure-Transformer architectures, two main paths relying on hybrid convolutional and self-attentive models were assessed, i.e., the Non-Local [15] modules of this section and the CCT [14] of the following. However, both alternatives do not solve the other need for an adequate sequence-level aggregator and then are coupled with the SeqVLAD layer introduced in Section 5.3.7.

The three Non-Local modules evaluated in this section are introduced in the S-VG pipeline right after the CNN backbone to operate on the extracted feature maps to capture relationships within the single images or across an entire sequence. Then, the output features of the module are fed to the SeqVLAD layer, that for ResNet-18 `conv4_x` and using 64 centroids produces a fixed descriptor of dimensionality 16384, similarly to the single-image descriptors produced by NetVLAD for the same backbone.

The Non-Local modules provide different degrees of flexibility by tuning the number of channels in the internal computation and the weight to put in the attention mask in the residual connection. This layer allows the introduction of the expressiveness of a self-attentive layer in a traditional CNN, with a low-cost impact on the overall system.

Backbone	Channel Bottlenek	Residual Weight	Non-Local Block Type	Sequence Aggregator	Train/Test Seq. Len.	R@1
ResNet-18	-	-	-	SeqVLAD	3	82.6
ResNet-18	32	1	2D	SeqVLAD	3	83.2
	64					82.7
	128					82.6
	256					82.3
ResNet-18	32	1	2D + T	SeqVLAD	3	82.7
	64					82.0
ResNet-18	32	1	3D	SeqVLAD	3	83.5
	64					82.9
	128					82.6
	256					81.8
ResNet-18	32	2	3D	SeqVLAD	3	81.8
	64					81.4
	128					80.9
	256					80.5
ResNet-18	-	-	-	SeqVLAD	5	85.2
ResNet-18	32	1	2D	SeqVLAD	5	84.9
	64					84.3
	128					83.6
ResNet-18	32	1	3D	SeqVLAD	5	85.7
	64					84.6
	128					83.7

Table 6.5: **Non-Local Blocks.** Results obtained with ResNet-18 backbone paired with different versions of the Non-Local layers, i.e., spatial (2D), spatial plus temporal attention (2D+T), and the 3D spatio-temporal attention (3D). The sequence-level aggregator is SeqVLAD for all the configurations with a descriptor of size 16384. The backbones are pre-trained on ImageNet. For training, the triplet used 10 negative examples with input frames of resolution 480 × 640.

Table 6.5 contains the results obtained for sequences of length 3 and 5 with the base ResNet-18+SeqVLAD model and then applying the various Non-Local modules. In the Table, the Spatial Non-Local Module is indicated as 2D attention, the 3D Spatio-Temporal NL goes under the name 3D, and 2D + T indicates the TemporaL NL used together with the Spatial one.

The first consideration is that looking at the outcomes, the use of ResNet-18 with SeqVLAD produces promising results and that the introduction of NL modules did not improve this baseline significantly.

Inspecting the results, we can see that the best results were obtained with the Spatial and 3D Spatio-Temporal NL modules when reducing the channels feature from the input value of 256 to 32, with a downscale factor of 8. The combination of Spatial and Temporal NL modules does not improve the results of the baseline.

The results obtained with 2D and 3D Non-Local layers slightly improve compared to the ResNet-18+SeqVLAD baseline on short sequences of 3 frames. More importantly, the methods presented in this section improve by 3% in R@1 the NetVLAD+CAT baseline and the TimeSformer models with sequences composed of 3 frames. The improvement when considering sequences of 5 frames is even greater for the 3D NL module, which achieves 5% more with respect to the baseline of the previous sections.

Unfortunately, the Non-Local modules do not provide the desired performance boost if compared with ResNet-18+SeqVLAD alone. A possible strategy to improve the final R@1 could be to insert the NL layer at different points in the CNN backbone. However, properly tuning this choice requires an exhaustive collection of experiments, that given the limited availability of computational resources, are postponed for future investigations.

6.6 CCT

The authors of [14] proposed the Compact Convolutional Transformer (CCT) to purposely offer a Transformer-based architecture that requires a considerably lower number of parameters compared to ViT to reduce its computational demands. The CCT configuration employed in this section has a Convolutional Tokenization block with two convolutional layers with 7×7 kernels that produce a feature map more compact than the flattened image patches obtained from the input frames. These feature maps have 384 channels, which, as a result, is also the dimensionality of the tokens and the embeddings manipulated by the network. This model has a number of parameters around 22M, comparable with the ResNet-18 architecture of the baselines. The output encodings of the model are processed using SeqVLAD, which aggregates the output embeddings into a sequence descriptor of dimensionality equal to $384 \cdot 64 = 24576$. The different configurations tested in this section investigate the impact of cropping the stack composed of 14 encoder layers and the effect of freezing the Convolutional Tokenization layer with or without the first encoder blocks.

Table 6.6 shows the beneficial effect for the results in retrieval performance obtained by reducing the number of encoder blocks to use the output embedding sequence from lower layers. The same positive outcome also emerges when freezing the Tokenization layer or the first encoder blocks with the weights learned on ImageNet. The majority of the results from Table 6.6 are obtained with the CCT architectures that take as input 224×224 images, then named CCT-224. A few

Backbone	**Sequence Aggregator**	**Truncated at block**	**Freeze up to block**	**Train/Test Seq. Len.**	**R@1**
CCT-224	SeqVLAD	10	_	3	83.9
CCT-224	SeqVLAD	10	Conv. Tok.	3	83.7
CCT-224	SeqVLAD	10	TE 2	3	82.5
CCT-224	SeqVLAD	8	-	3	83.5
CCT-224	SeqVLAD	8	Conv. Tok.	3	84.7
CCT-224	SeqVLAD	8	TE 2	3	84.6
CCT-224	SeqVLAD	6	-	3	81.4
CCT-384*	SeqVLAD	10	Conv. Tok.	3	86.4
CCT-384*	SeqVLAD	8	Conv. Tok.	3	87.9
CCT-224	SeqVLAD	10	-	5	85.1
CCT-224	SeqVLAD	10	Conv. Tok.	5	86.0
CCT-224	SeqVLAD	8	-	5	85.6
CCT-224	SeqVLAD	10	-	15	89.6
CCT-224	SeqVLAD	8	-	15	91.2

Table 6.6: **CCT Results.** This Table contains the results obtained with CCT models and SeqVLAD. CCT-224 and CCT-384 identify the two resolutions used for the input frames. The descriptor dimensionality is 24576 for all the models.
* CCT-384 models are trained with 6 negative examples.

results using larger resolutions of 384 × 384 pixels (CCT-384) were obtained by lowering the number of negative examples from 10 to 6. As already mentioned in Section 5.3.7, the sequence-aggregator layer can process the output embeddings of a Transformer-based architecture by using a 1-dimensional convolutional layer in place of the usual 2-dimensional one. The layer receives as input a tensor of shape $F \times S \times 384$, where F is the number of frames in the sequence and S is the number of embeddings produced by the CCT. The next step is reshaping this tensor as $(F \cdot S) \times 384$ and applying the usual soft-aggregation.

Compared to the results obtained with ResNet-18+SeqVLAD of the previous section, CCT is able to achieve better results with its CCT-224 versions. In particular, it improves by 2% R@1 with the sequences of 3 frames and 1% over the 5-frames sequences. However, it requires a larger descriptor than the one obtained with ResNet-18 because of the bigger embeddings of CCT compared to the 256-channel features of the CNN model.

The use of CCT-384, on the one hand, improves the R@1 performance considerably, achieving a result of 87.9% with 3-frames sequences, but with the downside of requiring more GPU memory for training, given the number of tokens embeddings.

6.7 Training and Inference Costs for S-VG

The previous sections deal with several architectures that have different requirements. This section summarizes and discusses all these factors to provide a complete picture of the different architectural choices investigated in this part of the thesis. Tables 6.7 and 6.8 contain the feature dimensionalities, inference times, and training GPU-memory requirements for models applied to sequences with lengths of three and seven frames. As previously explained in Section 4.5, the inference time is the result of two components, extraction and matching time, that are reported singularly and combined in the Inference Time column. The extraction time is correlated with the complexity of the model, i.e., a higher extraction time means a higher number of computations required to extract the sequence descriptor. The matching time is obtained as the time required by a forward pass of 1000 query sequences matched against a database of 10^5 elements divided by 1000 to mimic a realistic S-VG scenario. Note that, at inference time, the system works directly on the sequence-descriptors previously computed off-line. All the timings reported in the Tables were obtained using a machine equipped with an NVIDIA GTX Titan GPU and an Intel i7-5930K CPU.

Architecture	**Descriptor Dim.**	**Extraction Time [ms]**	**Matching Time [ms]**	**Inference Time [ms]**	**R@1**	**GPU Memory per triplet**
ResNet-18 - GeM - CAT	768	12	2	15	73.6	3.0 GB
ResNet-18 - NetVLAD - CAT	49152	20	145	164	81.0	3.8 GB
ResNet-18 - SeqVLAD	16384	19	48	67	82.1	4.0 GB
ResNet-18 - Attn. 2D - SeqVLAD	16384	23	48	71	83.2	4.3 GB
ResNet-50 - SeqVLAD	65536	72	191	262	83.5	13.2 GB
Base-ViT - CAT	2304	31	7	38	76.0	8.3 GB
Timesformer	768	48	2	50	79.7	11.0 GB
CCT-224 - SeqVLAD	24576	11	72	83	86.8	3.2 GB
CCT-384 - SeqVLAD	24576	35	72	107	85.9	12.0 GB

Table 6.7: **Requirements for Sequence Length 3**. This Table contains a summary of the costs and requirements for the different models when dealing with sequences with 3 frames sequences. All the times are reported in milliseconds and obtained as the average time required for 1000 images. The matching times consider a database of 10^5 elements. The GPU memory is the lower bound needed for the forwarding of a single triplet through the model at training time.

This section does not consider an analysis of different kNN indexes that would require an ad-hoc tuning procedure for each method and, instead, utilizes an exhaustive kNN approach. Given the direct proportionality between the sequence descriptor size and the matching time required, the architectures with the smaller descriptors also have the lowest time to retrieve the shortlist of candidates. In the case of ViT-CAT and TimeSformer models, the former produces relatively short descriptors for both sequence lengths, while the second produces a fixed-size descriptor regardless of the frames in the sequence. However, the complexity of these models demands a long extraction time, leading to an overall inference time comparable with a ResNet-18 paired with SeqVLAD. Comparing the configuration composed of ResNet-18 and SeqVLAD with the CCT-224 and the same sequence aggregator,

Architecture	**Descriptor Dim.**	**Extraction Time [ms]**	**Matching Time [ms]**	**Inference Time [ms]**	**GPU Memory per triplet**
ResNet-18 - GeM - CAT	1792	29	5	35	6.0 GB
ResNet-18 - NetVLAD - CAT	114688	46	339	385	8.4 GB
ResNet-18 - SeqVLAD	16384	45	48	93	8.6 GB
ResNet-18 - Attn. 2D - SeqVLAD	16384	54	48	101	10.0 GB
ResNet-50 - SeqVLAD	65536	168	191	359	28.0 GB
Base-ViT - CAT	5376	71	15	86	16.2 GB
Timesformer	768	108	2	110	22.0 GB
CCT224 - SeqVLAD	24576	25	72	98	7.1 GB
CCT384 - SeqVLAD	24576	80	72	153	24.0 GB

Table 6.8: **Requirements for Sequence Length 7**. This Table contains a summary of the costs and requirements for the different models when dealing with sequences with 7 frames sequences. All the times are reported in milliseconds and obtained as the average time required for 1000 images. The matching times consider a database of 10^5 elements. The GPU memory is the lower bound needed for the forwarding of a single triplet through the model at training time.

the latter architecture has a remarkable 40% lower extraction time but counterbalance it producing a larger sequence descriptor that leads to an overall comparable result in term of inference time but with considerable higher R@1 performances.

Given the available resources to conduct the experiments, the GPU memory requirement is one of the most critical factors. The weakly supervised triplet loss [16] adopted in this work when using a number of negatives equal to 10 examples and a single positive, each triplet is composed of 12 sequences of F frames each. This means that for sequence lengths equal to 3 and 7 frames, the triplet is composed of 36 and 84 images, respectively, and considering only batches with a single triplet, whereas a larger batch size of at least two would be a better choice for training the model. These simple considerations have the practical implication of large memory requirements at training time that make the resources for training architectures as Timesformer, ViT, and CCT-384 very difficult or impossible to obtain. However, this problem is not limited to Transformer-based architectures but also affects deeper and heavier CNNs. For instance, using a ResNet-50 with SeqVLAD, using images with full resolution 480 × 640, requires a similar GPU memory demand. This section underlines that the main problem encountered in this scenario is the overwhelming training needs of the adopted metric learning approach. A possible future line of research could involve the use of different procedures that require a lower memory footprint by default. In the following sections, two alternative approaches are considered to lower the memory requirements with the triplet loss (i) to use methods trained on the im2im and (ii) to reduce the number of negative samples in each triplet.

6.8 Single-Image Models for S-VG

The experiments presented in this section aim to circumvent the issues related to the high GPU memory requirements highlighted previously, exploiting the flexibility of the SeqVLAD layer. The idea is to train the models on an im2im setup that drastically reduces the memory impact and then evaluate the performances on various sequence lengths.

This approach cannot be exploited using TimeSformers or configurations with fully connected layers because these methods can be applied only on sequences with the same length as the training ones. On the other hand, concatenating the descriptors requires very large descriptors when using NetVLAD, then only the results obtained with the light GeM with CAT configuration are reported. The techniques analyzed in this section focus on ResNet and CCT architectures paired with SeqVLAD with different sequence lengths up to 15 total frames.

Backbone	**Decriptor 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**
ResNet-18	$F \cdot 256$	GeM - CAT	65.7	75.8	77.5	78.9	79.4	83.0
ResNet-18	16384	SeqVLAD	73.9	82.1	83.1	84.0	85.4	90.4
ResNet-18 + Attn. 2D		SeqVLAD	74.9	82.1	83.7	85.0	86.3	91.8
ResNet-50	65536	SeqVLAD	75.9	83.5	85.6	86.5	88.0	92.0
CCT-224 (tr8)	24576	SeqVLAD	74.6	83.4	85.1	84.9	86.8	91.1
CCT-224 (tr10)			74.5	83.8	85.5	86.0	87.7	90.4
CCT-384 (tr8)	24576	SeqVLAD	77.5	86.4	87.3	87.8	89.6	94.7
CCT-384 (tr8-fz2)			**79.8**	**87.7**	88.9	88.8	90.3	**94.8**
CCT-384 (tr10)			79.4	**87.7**	**89.2**	**89.4**	**90.5**	93.6

Table 6.9: **Models trained on the im2im task applied to S-VG.** The table contains the results obtained evaluating models trained on the im2im task employing the SeqVLAD layer, which provides enough flexibility to adapt to different sequence lengths, maintaining a fixed-length descriptor. The backbones are pre-trained on ImageNet. The entire pipeline is finetuned on MSLS, with 10 negatives per triplet. The parameters indicated for CCT models are the encoder block at which the architecture is cropped (tr$N$) and the layers frozen during the training (fz$N$).

Table 6.9 contains the obtained outcomes and the dimensionalities of the different descriptors. Note that for GeM-CAT, the dimensionality changes with the number of frames (F) as $256 \cdot F$, while SeqVLAD always produces a fixed-length descriptor. The outcomes obtained with these experiments give interesting insights into two aspects. In contrast with the initial intuition, the model trained for im2im paired with SeqVLAD can produce high retrieval performances. A complex architecture as the TimeSformer employs the self-attention mechanism to model the relationships between the semantic elements in the images both within a single frame and across the sequence to capture the scene's evolution. This property is desirable for Action Recognition but could not be mandatory for Sequence-based VG, or at least with the MSLS dataset used in this work. Inspecting the results obtained with im2im trained models, this hypothesis can be relaxed in favor of a simpler sequence-level aggregator as SeqVLAD. In fact, this layer relies on the NetVLAD's ideas of creating a descriptor by interpreting the output embeddings from CCTs or the feature maps from CNNs, as dense local features and aggregates them computing their residuals with a learnable codebook. The training using the im2im setup leads CCT-224 models to a drop in performance around 1%, consistent over the different sequence lengths. However, this approach allows experimenting with CCT for larger image resolutions of 384×384 without impacting the GPU memory requirements. While the requirements for one triplet with 10 negatives and sequence length 3 required over 12GB, with this approach, the memory needed for one triplet is lower than 8GB. Combining the higher resolution for the input images with the representational capability of the Transformer-based architecture of the CCT provides better results than the CNNs counterpart.

6.9 Variations with Number of Negatives

The second approach proposed for reducing the GPU memory requirements at training time is to reduce the number of negative samples is set to 10, following the indications of [16]. However, when considering sequences together within this metric learning setup, the total number of images for each triplet increases quickly and, as a consequence, also the required memory. This section proposes a series of experiments to test lower values for the training triplets' negatives. Due to the limited time and computational resources, the experiments are not exhaustive, but they give precious hints for future works. The configurations tried in this collection of experiments are ResNet-18, CCT-224, and CCT-384, all paired with SeqVLAD, trained on single frames or with a sequence length of three frames. Table 6.10 contains the outcomes obtained training these models with 10, 5, and 2 negative examples per triplet and then evaluating them with different sequence lengths.

The choice of ResNet-18 with SeqVLAD is meant to have a baseline model for both single image and 3-frames sequences. When using single images, the SeqVLAD with CNN backbones is equivalent to NetVLAD, and then it is possible to assess how the use of different negatives affects this ”traditional” pipeline. Inspecting the outcomes reported in the Table, for this configuration, the retrieval performance when using 2 negatives is consistently lower than with 10 negatives for all the different sequence lengths. Instead, reducing the number to 5 negatives does not impact the results, which are always comparable or even slightly superior. When training the architecture on sequences of 3 frames, lowering the number of negatives has a positive impact on the performances. For this scenario, the best results are obtained using 2 negatives, but with 5 the results are very close, too.

Backbone	**Training Seq. Len.**	**Aggregation**	**Num. Neg.**	**R@1 Seq. 1**	**R@1 Seq. 3**	**R@1 Seq. 5**	**R@1 Seq. 7**	**R@1 Seq. 9**	**R@1 Seq. 15**
ResNet-18	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
CCT-224 (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
CCT-384 (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**
ResNet-18	Seq. 3	SeqVLAD	10	73.7	83.5	83.9	85.0	86.2	90.3
			5	75.0	83.8	85.5	86.1	87.5	91.6
			2	75.6	84.0	85.8	86.8	88.1	91.8
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
CCT-384 (tr8-fz2)	Seq. 3	SeqVLAD	10	77.1	87.1	88.7	89.3	90.7	94.3
			5	78.9	87.8	89.8	90.1	91.5	95.0
			2	**80.2**	**88.8**	**90.4**	**90.5**	**92.1**	**97.1**

Table 6.10: **Number of Negatives.** Results obtained training the models with different numbers of negatives. For CCT models, the two parameters indicate the layer at which the network is cropped (tr$N$) and the frozen blocks during training (fr$M$). Note that ”Conv” indicates the Convolutional Tokenization block. The best result for each architecture is underlined, while the best results for each training sequence length are in bold.

The results obtained with CCT architecture are of particular interest. While, as expected, the results obtained with CCT-224 and CCT-384 are higher for the -384 version, the use of single images and 3-frames sequences provides similar results, with the latter being slightly better. In both cases, the best results are obtained using 2 negative examples, which provide a considerable boost in performance between 1% and 4% with the various sequence lengths. The results reported in this section are the highest ones on the S-VG task. A possible motivation behind these surprising results can be found following these considerations. The setup of [16], where the authors do not use the partial database mining as done in MSLS [146] but the full mining procedure. Then, in that scenario, the negatives are selected considering the entire database, and it is more likely that all the 10 negatives are more visually similar to the query than the set of 10 negatives obtained from 1000 randomly sampled elements of the partial mining. Using a high number of negatives leads to a harmful effect on the loss because, for its formulation, when the examples fall very far in the descriptor space, they produce a null contribution and the overall effect of shrinking the final value of the loss. As a result, the optimization steps are shorter, leading to local optima with lower retrieval performances. The takeaway message obtained from the results discussed in this section is that the number of negative examples can be an essential hyperparameter to tune for S-VG. Besides the increase in R@1 performance, the use of a lower number of negatives in the S-VG setting is extremely beneficial since it substantially reduces the images in each triplet. This translates into lower GPU memory requirements and allows to train previously untractable methods.

6.10 Reverse and Random Sequence Order

This section proposes a series of experiments that modify the order of the database sequences to analyze how these variations impact the performances of the different models. In particular, the database sequences at test time were reversed and shuffled to assess the robustness of the predictions. In this sense, it is essential to recall that the MSLS dataset consists of images from long sequences captured from front-view cameras mounted on cars. Those sequences do not have a 360° degree view and have a defined direction. The utility that generates the training sequences use the forward direction and selects different portions of the original sequence of the right length. By reversing or shuffling its order, the model operates on the backward sequence or entirely at random, providing a challenging situation for the S-VG system.

Table 6.11 contains the results of this analysis. The models are trained following the usual approach but are then tested using the right order for the test queries and different orderings for the database sequences.

As expected, the results show that aggregation methods relying on concatenation and fully-connected layers are affected by the different ordering. This is the case for ResNet-18 with NetVLAD-CAT and GeM-CAT, and the ViT-FC2048. Considering the reverse ordering, the performances decrease for the two CAT-based pipelines in a range between 5% and 14%, with higher drops for longer sequences; while shuffling the frame order has a more contained impact, probably because of the short length of the sequences for which the final sequence can be close to the original one. The ViT model has a drop of 4.7% for the reverse order and 2.3% for the shuffled sequences, which is lower than the equivalent CAT models. Unfortunately, the use of a fully-connected layer prevents the evaluation of this architecture on different sequence lengths.

The TimeSformer architecture again cannot process sequences with different lengths, but it proves to be robust to the different order of the frames. Unlike the previous methods, it builds a sequence descriptor by considering the sequence as a whole entity, which inhibits the potentially detrimental effects of a different frame order.

A similar argument can be applied to the SeqVLAD layer, that instead of considering the frames composing the sequence as separate entities, works on the features/embeddings treating them as an unordered set of elements that must be considered together to describe the whole sequence. From this order-invariant property of SeqVLAD, it follows that the results obtained with this sequence-aggregator are independent of the frame order.

Architecture	Training Seq. Len.	Aggregation	Frame Order	R@1 Seq. 3	R@1 Seq. 5	R@1 Seq. 7	R@1 Seq. 9	R@1 Seq. 15
ResNet-18	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
ResNet-18	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
ResNet-18 + Attn. 2D	Seq. 3	SeqVLAD	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
ResNet-18 + Attn. 3D	Seq. 3	SeqVLAD	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	FC-2048	Normal	-	79.2	-	-	-
			Reverse	-	74.5	-	-	-
			Shuffle	-	76.9	-	-	-
Timesformer	Seq. 7	-	Normal	-	-	83.7	-	-
			Reverse	-	-	83.7	-	-
			Shuffle	-	-	83.8	-	-
CCT-224 (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

Table 6.11: **Experiments with different sequence orders.** This Table contains the results obtained by shuffling or reversing the order of the database sequences and keeping the original order for the queries. A configuration has been evaluated for each of the architectures proposed in this chapter. All the backbones are pre-trained on ImageNet, and trained on MSLS with 10 negative examples per triplet. The parameters indicated for CCT models are the encoder block at which the architecture is cropped (tr$N$) and the layers frozen during the training (fz$N$).

Chapter 7

Conclusion and future works

The path followed to achieve the set objectives of this thesis led to an in-depth analysis of all the most important and widespread Deep Learning techniques used for Visual Geo-Localization. The first goal of this work was to produce a fair and comprehensive benchmark that could allow the researchers and industry practitioners to analyze the performances and the requirements of the methods proposed in the literature. The process of design and development of this framework followed a series of progressive steps. The first phase was an exhaustive literature survey to identify the most popular and effective techniques adopted in VG. This study was pivotal to prototype a modular system that could include these techniques and allow their training and evaluation. For training and evaluating the models, six heterogeneous datasets covering various real-world scenarios have been selected to build the evaluation suite of the framework. After this preliminary phase, the development of the VG system proceeded with the implementation of all the selected techniques composing the different stages of the VG pipeline. Finally, we conducted extensive experiments using our framework to analyze the contributions and costs of the different choices in a VG system. The analysis of the results points out general guidelines to follow in developing and deploying a Visual Geo-localization application. This massive work resulted from a group effort and is currently under review for publication.

The contribution of the first half of this thesis is the introduction of a modular framework to build, train and evaluate several popular Visual Geo-localization architectures. Compared to similar works, it allows to inspect and analyze the enhancements provided by every single component because its modular design provides the flexibility of changing the various elements composing the VG pipeline and evaluating them in a standardized environment. Moreover, the exhaustive collection of experiments provides precious guidelines on the different engineering choices at training and inference time to calibrate the VG system based on the performances and resources required by its downstream applications.

The experiments show that among the different CNN architectures used in the

Method	**Descr. Dim.**	**R@1 Pitts30k**	**R@1 Pitts250k**	**R@1 Tokyo 24/7**
VGG16 + NetVLAD + PCA [173]	4096	85.2	86.5	68.9
VGG16 + NetVLAD [173]	32768	-	84.1	60.0
SRALNet (ICRA21) [174]	4096	-	87.8	72.1
SRALNet (ICRA21) [174]	32768	85.1	85.8	68.6
APPSVR (ICCV21) [173]	4096	87.4	88.8	77.1
APPSVR (ICCV21) [173]	32768	-	86.6	68.3
ResNet-18 + NetVLAD + PCA (Ours)	4096	86.8	87.9	72.2
ResNet-18 + NetVLAD (Ours)	16384	87.2	88.1	73.7

Table 7.1: **Comparison with state-of-the-art models.** This table shows the results obtained comparing a simple architecture composed of ResNet-18 + NetVLAD following the insights gathered from the benchmarking experiments. The model's architecture was selected between the best-performing ones and trained on Pitts30k using data augmentation and majority voting on the tests for Tokyo 24/7.

VG literature, the ResNet-50 provides a good compromise between requirements and performance, and ResNet-18 represents the best option when looking for a lighter alternative. Concerning the aggregation layers, CRN provides the best results but has the downside of requiring a significant training cost compared to the other techniques. Then, two alternatives are NetVLAD, which reaches similar performances but requires half of CRN's training time, and the GeM pooling layer, which shows a good generalization capability when trained on large-scale heterogeneous datasets.

An important aspect when approaching VG as a retrieval task within the metric learning setup is the choice of an effective and efficient mining procedure, which is required to identify the negative examples of the training triplets. Full database mining proved to be an impracticable option for large-scale datasets while using partial mining achieves a proper trade-off between performances and computational costs.

One of the primary motivations for this benchmarking framework is the lack in the literature of a common platform to evaluate the VG methods fairly. Other works, as [17, 81], compare the performance results of architectures trained on different datasets; however, the outcomes of the experiments highlighted how the training sets matter considerably on the generalization performances of the different techniques. This result points out that a framework that adequately addresses this harmful habit is fundamental to comprehend the actual contributions of VG methods.

Furthermore, the analysis also proposes a study of optimized kNN indexing techniques, showing that they can play a crucial role in reducing the memory requirements and the inference time in deploying the models. These techniques must be adequately tuned to the final application needs with a careful trade-off between the inference time, the memory footprint of the database, and the impact on retrieval performance. The choice of the right kNN indexing allows the use of techniques with larger descriptors that otherwise, with plain kNN, could not be deployed.

To prove the importance of the insight gathered with this work, the results in Table 7.1 show that using a simple architecture optimized following these guidelines can achieve comparable results with more complex and recent state-of-the-art methods. Once more, these results underline the importance of a standard platform for training and evaluating VG models.

Currently, the framework presents some limitations that will be addressed in future releases. For instance, compared to [81] it lacks an in-depth analysis of viewpoint and invariance analysis of the VG methods, and the focus is only on outdoor urban environments. Moreover, some current state-of-the-art methods like [175, 173] and other losses [153] are not yet fully integrated into the framework, but they will be added in the future. The current plan is to continuously update the benchmark's website [1] with new results and to expand the pool of available methods.

The second goal of the thesis was to extend the Deep Learning methods for VG analyzed in the first part exploiting the information provided by sequences of frames. Sequence-based VG has roots in the robotics community and several possible downstream applications, from autonomous driving to augmented reality. The development of methods able to capture and model the relationship of the semantic elements can enhance the retrieval ability of the systems and produce more robust descriptors. For this task, the focus was shifted from traditional CNN architectures to explore the possibility of using the emerging Transformer-based architectures. This investigation represents an original contribution of this thesis aimed at exploiting their self-attention mechanism capable of modeling complex spatial and temporal relationships in the sequences. Approaching Sequence-based VG within a metric learning setup presented several challenges in finding a good trade-off between the proposed methods' requirements and their retrieval performances.

S-VG can be formulated in different ways, and this work embraces the seq2seq formulation, where both the queries and the database instances are sequences of frames. The objective is to develop models that extract discriminative and robust sequence-level descriptors to select the potential database sequences captured in the same location as the query one. The analysis focuses on small sequences between 3 and 15 frames, with the idea of preserving a meaningful and accurate concept of geographic location.

Throughout the investigation, several forms of self-attentive techniques were employed. They include different Non-Local modules configurations used as pluggable attention layers for CNN networks and the use of Transformer-based architectures,

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

as Vision Transformers (ViT), TimeSformers, and Compact Convolutional Transformers (CCT). The latter presents a hybrid approach that exploits the advantages of both worlds of convolutions and self-attention operations. ViT and TimeSformer rely on a patching technique to transform the single frames into sequences of vectors. This simple approach has the downside of requiring higher computational resources that limit their applicability for this work. Given the need for a reliable sequence-level aggregation layer, we introduced SeqVLAD, which adapts NetVLAD, i.e., one of the best-performing methods identified in the benchmarking analysis, to the S-VG task. This layer paired with CCT produced the best results, proving the richer expressiveness of the Transformer-based architectures and the benefit of using an ad-hoc sequence aggregation layer for this task.

Given the considerable amount of resources required by many of these models, a section has been dedicated to analyzing training and inference time costs. Following this analysis, we considered two possible alternatives to lower the models' requirements: (i) reducing the number of negative examples per triplet and (ii) training models with SeqVLAD on the single image task and then adapting them with different sequence lengths. Finally, the last section investigates the behavior of the models when faced with database sequences with shuffled or reversed frame order.

The results obtained for the S-VG task highlight the potential of exploiting sequences to achieve better performances in the Geo-Localization task and to produce more robust descriptors. This task benefits from using ad-hoc techniques to model spatio-temporal relationships between the elements in the frames. Moreover, the experiments show how the introduction of Transformer-based architectures produces compelling results that could be the basis for future works. Other possible research directions emerged during this work. One possibility is to use a different formulation of the task, that instead of mapping query sequences to database sequences, produces a prediction for the last frame of the sequence and uses the previous frames as past information of the path followed to get to that specific location. The insights generated on the number of negative examples for the training triplets give space for other experiments with models that otherwise could not be evaluated. Additionally, new hybrid models combining the CCT's Tokenization block and the self-attention scheme from TimeSformer could be tried to gain the positive aspects coming from these two architectures.

Bibliography

[1] 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 (pages 1, 2, 8, 12).

[2] E. C. Tolman and C. H. Honzik. «Degrees of hunger, reward and non-reward, and maze learning in rats». In: (page 1).

[3] Edward Chace Tolman. «Cognitive maps in rats and men.» In: *Psychological review* 55 4 (1948), pp. 189–208 (page 1).

[4] Kevin Lynch. *The image of the city.* Cambridge, Massachusetts: MIT Press, 1960 (page 1).

[5] Benjamin Kuipers. «Modeling Spatial Knowledge». In: *IJCAI.* 1977 (page 1).

[6] Benjamin Kuipers and Yung-Tai Byun. «A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations». In: *Robotics Auton. Syst.* 8 (1991), pp. 47–63 (page 1).

[7] Benjamin Kuipers. «The Spatial Semantic Hierarchy». In: *Artif. Intell.* 119 (2000), pp. 191–233 (page 1).

[8] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. *Deep Visual Geo-localization Benchmark.* 2021 (pages 2, 6, 32–36, 38, 43, 44, 46, 48, 50–53, 55, 56, 58).

[9] Sourav Garg, Tobias Fischer, and Michael Milford. «Where is your place, Visual Place Recognition?» In: *ArXiv* abs/2103.06443 (2021) (pages 2, 6, 12).

[10] 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 (pages 2, 19, 38, 42, 45, 46).

[11] Carlo Masone and Barbara Caputo. «A Survey on Deep Visual Place Recognition». In: *IEEE Access* 9 (2021), pp. 19516–19547 (pages 2, 7, 8).

[12] 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` (pages 2, 8, 13).

[13] 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: *ArXiv* abs/2010.11929 (2021) (pages 4, 29, 70–73, 80, 83).

[14] 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]` (pages 4, 30, 72, 73, 82, 86, 88).

[15] 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 (pages 5, 28, 70, 76, 77, 80, 86).

[16] Relja Arandjelović, Petr Gronát, Akihiko Torii, Tomás Pajdla, and Josef Sivic. «NetVLAD: CNN Architecture for Weakly Supervised Place Recognition». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (2018), pp. 1437–1451 (pages 5, 19, 25, 33–35, 37–42, 45, 46, 48, 49, 57, 62, 66, 78, 80, 91, 93, 94).

[17] 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 (pages 9, 22, 98).

[18] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew W. Fitzgibbon. «Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images». In: *2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 2930–2937 (page 9).

[19] 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 (page 9).

[20] Torsten Sattler, Tobias Weyand, B. Leibe, and Leif P. Kobbelt. «Image Retrieval for Image-Based Localization Revisited». In: *BMVC.* 2012 (page 9).

[21] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. «1 Year, 1000km: The Oxford RobotCar Dataset». In: *The International Journal of Robotics Research* (2017) (page 9).

[22] 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` (page 9).

[23] 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 (page 9).

[24] Wei Chen, Yang Liu, Weiping Wang, Erwin M. Bakker, Theodoros Georgiou, Paul W. Fieguth, Li Liu, and Michael S. Lew. «Deep Image Retrieval: A Survey». In: *ArXiv* abs/2101.11282 (2021) (pages 9, 10).

[25] Liang Zheng, Yi Yang, and Qi Tian. «SIFT Meets CNN: A Decade Survey of Instance Retrieval». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (2018), pp. 1224–1244 (page 9).

[26] Jiankang Deng, J. Guo, and Stefanos Zafeiriou. «ArcFace: Additive Angular Margin Loss for Deep Face Recognition». In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 4685–4694 (pages 11, 52).

[27] H. Wang, Yitong Wang, Zheng Zhou, Xing Ji, Zhifeng Li, Dihong Gong, Jingchao Zhou, and Wenyu Liu. «CosFace: Large Margin Cosine Loss for Deep Face Recognition». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 5265–5274 (page 11).

[28] Shuhei Yokoo, Kohei Ozaki, Edgar Simo-Serra, and Satoshi Iizuka. «Two-stage Discriminative Re-ranking for Large-scale Landmark Retrieval». In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2020), pp. 4363–4370 (page 11).

[29] Hyeonwoo Noh, Andre F. de Araújo, Jack Sim, Tobias Weyand, and Bohyung Han. «Large-Scale Image Retrieval with Attentive Deep Local Features». In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 3476–3485 (pages 11, 35, 50).

[30] Tobias Weyand, Andre F. de 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 (pages 11, 35, 50).

[31] Tobias Weyand, Jan Hendrik Hosang, and B. Leibe. «An Evaluation of Two Automatic Landmark Building Discovery Algorithms for City Reconstruction». In: *ECCV Workshops*. 2010 (page 11).

[32] James Philbin, Ondřej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. «Lost in quantization: Improving particular object retrieval in large scale image databases». In: *2008 IEEE Conference on Computer Vision and Pattern Recognition* (2008), pp. 1–8 (page 11).

[33] James Philbin, Ondřej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. «Object retrieval with large vocabularies and fast spatial matching». In: *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), pp. 1–8 (pages 11, 21).

[34] Filip Radenović, Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondřej Chum. «Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018), pp. 5706–5715 (page 11).

[35] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. «On the burstiness of visual elements». In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 1169–1176 (page 12).

[36] Tobias Weyand, Ilya Kostrikov, and James Philbin. «PlaNet - Photo Geolocation with Convolutional Neural Networks». In: *ArXiv* abs/1602.05314 (2016) (page 14).

[37] Paul Hongsuck Seo, Tobias Weyand, Jack Sim, and Bohyung Han. «CPlaNet: Enhancing Image Geolocalization by Combinatorial Partitioning of Maps». In: *ECCV*. 2018 (page 14).

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger. Curran Associates, Inc., 2012, pp. 1097–1105. URL: `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf` (page 15).

[39] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, K. Li, and Li Fei-Fei. «ImageNet: A large-scale hierarchical image database». In: *CVPR*. 2009 (pages 15, 35, 70, 73).

[40] Krystian Mikolajczyk and Cordelia Schmid. «An Affine Invariant Interest Point Detector». In: *ECCV*. 2002 (page 15).

[41] Jiri Matas, Ondřej Chum, Martin Urban, and Tomás Pajdla. «Robust Wide Baseline Stereo from Maximally Stable Extremal Regions». In: *BMVC*. 2002 (page 15).

[42] David G. Lowe. «Object recognition from local scale-invariant features». In: *Proceedings of the Seventh IEEE International Conference on Computer Vision* 2 (1999), 1150–1157 vol.2 (page 16).

[43] Yan Ke and Rahul Sukthankar. «PCA-SIFT: a more distinctive representation for local image descriptors». In: *CVPR 2004*. 2004 (page 16).

[44] Relja Arandjelović and Andrew Zisserman. «Three things everyone should know to improve object retrieval». In: *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), pp. 2911–2918 (page 16).

[45] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. «Speeded-Up Robust Features (SURF)». In: *Comput. Vis. Image Underst.* 110 (2008), pp. 346–359 (page 16).

[46] Michael Calonder, Vincent Lepetit, Mustafa Özuysal, Tomasz Trzciński, Christoph Strecha, and Pascal Fua. «BRIEF: Computing a Local Binary Descriptor Very Fast». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (2012), pp. 1281–1298 (page 16).

[47] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. «ORB: An efficient alternative to SIFT or SURF». In: *2011 International Conference on Computer Vision* (2011), pp. 2564–2571 (page 16).

[48] Josef Sivic and Andrew Zisserman. «Video Google: a text retrieval approach to object matching in videos». In: *Proceedings Ninth IEEE International Conference on Computer Vision* (2003), 1470–1477 vol.2 (pages 16, 21, 53, 55, 78).

[49] Mark Joseph Cummins and Paul Newman. «Appearance-only SLAM at large scale with FAB-MAP 2.0». In: *The International Journal of Robotics Research* 30 (2011), pp. 1100–1123 (pages 16, 33, 37, 62).

[50] Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier. «Large-scale image retrieval with compressed Fisher vectors». In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), pp. 3384–3391 (page 17).

[51] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. «Aggregating local descriptors into a compact image representation». In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010), pp. 3304–3311 (pages 17, 78).

[52] Relja Arandjelović and Andrew Zisserman. «All About VLAD». In: *2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 1578–1585 (page 17).

[53] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. «Aggregating Local Image Descriptors into Compact Codes». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34 (2012), pp. 1704–1716 (page 17).

[54] 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 (2004), pp. 145–175 (page 17).

[55] Navneet Dalal and Bill Triggs. «Histograms of oriented gradients for human detection». In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* 1 (2005), 886–893 vol. 1 (page 17).

[56] Niko Sünderhauf and Peter Protzel. «BRIEF-Gist - closing the loop by simple means». In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2011), pp. 1234–1241 (page 17).

[57] Hernán Badino, Daniel F. Huber, and Takeo Kanade. «Real-time topometric localization». In: *2012 IEEE International Conference on Robotics and Automation* (2012), pp. 1635–1642 (page 17).

[58] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. «Backpropagation Applied to Handwritten Zip Code Recognition». In: *Neural Computation* 1 (1989), pp. 541–551 (page 17).

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016 (page 18).

[60] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. «Learning and Transferring Mid-level Image Representations Using Convolutional Neural Networks». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1717–1724 (page 18).

[61] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Herv'e J'egou. «Training Vision Transformers for Image Retrieval». In: *ArXiv* abs/2102.05644 (2021) (page 18).

[62] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan 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 (page 18).

[63] Jixiang Wan, Dayong Wang, Steven C. H. Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. «Deep Learning for Content-Based Image Retrieval: A Comprehensive Study». In: *Proceedings of the 22nd ACM international conference on Multimedia* (2014) (page 18).

[64] Kilian Q. Weinberger and Lawrence K. Saul. «Distance Metric Learning for Large Margin Nearest Neighbor Classification». In: *NIPS*. 2005 (page 18).

[65] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. «Learning Fine-Grained Image Similarity with Deep Ranking». In: *2014 IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1386–1393 (page 18).

[66] Ruben Gomez-Ojeda, Manuel López-Antequera, Nicolai Petkov, and Javier González. «Training a Convolutional Neural Network for Appearance-Invariant Place Recognition». In: *ArXiv* abs/1505.07428 (2015) (page 18).

[67] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. «Multi-scale Orderless Pooling of Deep Convolutional Activation Features». In: *ECCV*. 2014 (page 18).

[68] Artem Babenko, Anton Slesarev, Alexander Chigorin, and Victor S. Lempitsky. «Neural Codes for Image Retrieval». In: *ArXiv* abs/1404.1777 (2014) (page 18).

[69] Mattis Paulin, Matthijs Douze, Zaïd Harchaoui, Julien Mairal, Florent Perronnin, and Cordelia Schmid. «Local Convolutional Features with Unsupervised Training for Image Retrieval». In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 91–99 (page 19).

[70] Eva Mohedano, Amaia Salvador, Kevin McGuinness, Ferran Marqués, Noel E. O'Connor, and Xavier Giro-i-Nieto. «Bags of Local Convolutional Features for Scalable Instance Search». In: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval* (2016) (page 19).

[71] Filip Radenović, Giorgos Tolias, and Ondřej Chum. «Fine-Tuning CNN Image Retrieval with No Human Annotation». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41 (2019), pp. 1655–1668 (pages 19, 20, 38, 42, 45, 46, 50, 51, 57).

[72] Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. «Visual Instance Retrieval with Deep Convolutional Networks». In: *CoRR* abs/1412.6574 (2015) (pages 20, 46).

[73] Artem Babenko and Victor S. Lempitsky. «Aggregating Deep Convolutional Features for Image Retrieval». In: *ArXiv* abs/1510.07493 (2015) (pages 20, 46).

[74] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. «Particular object retrieval with integral max-pooling of CNN activations». In: *CoRR* abs/1511.05879 (2016) (pages 20, 38, 42, 46, 57).

[75] 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` (pages 21, 53, 55).

[76] Dmitry Baranchuk, Artem Babenko, and Yury Malkov. *Revisiting the Inverted Indices for Billion-Scale Approximate Nearest Neighbors*. 2018. arXiv: `1802.02422 [cs.CV]` (pages 21, 54, 55).

[77] 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]` (pages 21, 54, 55).

[78] Jeff Johnson, Matthijs Douze, and Hervé Jégou. «Billion-scale similarity search with GPUs». In: *arXiv preprint arXiv:1702.08734* (2017) (pages 21, 54).

[79] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. «Particular object retrieval with integral max-pooling of CNN activations». In: *CoRR* abs/1511.05879 (2016) (page 22).

[80] Ondřej Chum, James Philbin, Josef Sivic, Michael Isard, and Andrew Zisserman. «Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval». In: *2007 IEEE 11th International Conference on Computer Vision* (2007), pp. 1–8 (page 22).

[81] 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 (pages 22, 23, 44, 98, 99).

[82] 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 (page 23).

[83] 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 (page 23).

[84] 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 (page 23).

[85] 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 (page 23).

[86] 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 (page 23).

[87] 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 (page 23).

[88] 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 (page 23).

[89] 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 (page 23).

[90] 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 (page 23).

[91] 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 (pages 24, 62).

[92] Kin Leong Ho and Paul Newman. «Detecting Loop Closure with Scene Sequences». In: *International Journal of Computer Vision* 74 (2006), pp. 261–286 (page 24).

[93] 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 (pages 24, 25, 62, 66).

[94] 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 (page 24).

[95] Tayyab Naseer, Luciano Spinello, Wolfram Burgard, and C. Stachniss. «Robust Visual Robot Localization Across Seasons Using Network Flows». In: *AAAI*. 2014 (page 24).

[96] Olga Vysotska, Tayyab Naseer, Luciano Spinello, Wolfram Burgard, and C. Stachniss. «Efficient and effective matching of image sequences under substantial appearance changes exploiting GPS priors». In: *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), pp. 2774–2779 (page 24).

[97] Olga Vysotska and C. Stachniss. «Relocalization under Substantial Appearance Changes using Hashing». In: 2017 (page 24).

[98] Olga Vysotska and C. Stachniss. «Effective Visual Place Recognition Using Multi-Sequence Maps». In: *IEEE Robotics and Automation Letters* 4 (2019), pp. 1730–1736 (page 24).

[99] Xiwu Zhang, L. Wang, Yan Zhao, and Yan Su. «Graph-Based Place Recognition in Image Sequences with CNN Features». In: *Journal of Intelligent & Robotic Systems* (2019), pp. 1–15 (page 24).

[100] 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 (page 24).

[101] 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 (page 24).

[102] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. «BRIEF: Binary Robust Independent Elementary Features». In: *ECCV*. 2010 (page 24).

[103] Edward Rosten and Tom Drummond. «Machine Learning for High-Speed Corner Detection». In: *ECCV*. 2006 (page 24).

[104] José M. Fácil, Daniel Olid, Luis Montesano, and Javier Civera. «Condition-Invariant Multi-View Place Recognition». In: *ArXiv* abs/1902.09516 (2019) (pages 25, 62, 63, 65–67, 71, 81).

[105] 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 (pages 25, 63–66).

[106] 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` (pages 25, 63, 65, 66).

[107] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. eprint: `arXiv:1512.03385` (pages 25, 42, 45, 65).

[108] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* 9 (1997), pp. 1735–1780 (pages 25, 26, 65).

[109] 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) (page 25).

[110] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. «Neural Machine Translation by Jointly Learning to Align and Translate». In: *CoRR* abs/1409.0473 (2015) (pages 26, 27).

[111] Nal Kalchbrenner and Phil Blunsom. «Recurrent Continuous Translation Models». In: *EMNLP*. 2013 (page 26).

[112] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. «On the Properties of Neural Machine Translation: Encoder–Decoder Approaches». In: *SSST@EMNLP*. 2014 (page 26).

[113] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. «Sequence to Sequence Learning with Neural Networks». In: *NIPS*. 2014 (page 26).

[114] 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 (page 26).

[115] Alex Graves, Greg Wayne, and Ivo Danihelka. «Neural Turing Machines». In: *ArXiv* abs/1410.5401 (2014) (page 27).

[116] Thang Luong, Hieu Pham, and Christopher D. Manning. «Effective Approaches to Attention-based Neural Machine Translation». In: *EMNLP*. 2015 (page 27).

[117] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2017. arXiv: `1706.03762 [cs.CL]` (pages 27–29, 67, 69, 73, 76).

[118] 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 (pages 27, 28, 76).

[119] Jianpeng Cheng, Li Dong, and Mirella Lapata. «Long Short-Term Memory-Networks for Machine Reading». In: *EMNLP*. 2016 (page 27).

[120] Alexander M. Rush, Sumit Chopra, and Jason Weston. «A Neural Attention Model for Abstractive Sentence Summarization». In: *EMNLP*. 2015 (page 27).

[121] 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 (page 27).

[122] Volodymyr Mnih, Nicolas Manfred Otto Heess, Alex Graves, and Koray Kavukcuoglu. «Recurrent Models of Visual Attention». In: *NIPS*. 2014 (page 27).

[123] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. «An Attentive Survey of Attention Models». In: *ArXiv* abs/1904.02874 (2019) (page 27).

[124] 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) (page 27).

[125] 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) (pages 27, 29).

[126] 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 (page 28).

[127] 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 (page 28).

[128] 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 (page 28).

[129] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. «Self-Attention with Relative Position Representations». In: *NAACL*. 2018 (page 28).

[130] 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]` (pages 29, 71).

[131] Alec Radford and Karthik Narasimhan. «Improving Language Understanding by Generative Pre-Training». In: 2018 (page 29).

[132] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. «Language Models are Unsupervised Multitask Learners». In: 2019 (page 29).

[133] Tom B. Brown et al. «Language Models are Few-Shot Learners». In: *ArXiv* abs/2005.14165 (2020) (page 29).

[134] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. «RoBERTa: A Robustly Optimized BERT Pretraining Approach». In: *ArXiv* abs/1907.11692 (2019) (page 29).

[135] 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) (page 29).

[136] 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 (page 29).

[137] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. «Stand-Alone Self-Attention in Vision Models». In: *NeurIPS*. 2019 (page 29).

[138] 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 (page 29).

[139] 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 (pages 30, 70).

[140] 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 (pages 30, 72).

[141] 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) (page 30).

[142] Yawei Li, K. Zhang, Jie Cao, Radu Timofte, and Luc Van Gool. «LocalViT: Bringing Locality to Vision Transformers». In: *ArXiv* abs/2104.05707 (2021) (page 30).

[143] 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) (page 30).

[144] 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 (pages 30, 63, 73, 74, 80, 84, 85).

[145] 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 (pages 33, 35).

[146] 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 (pages 33–35, 38–40, 42, 48, 49, 62, 63, 65, 94).

[147] 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 (pages 33, 37, 57).

[148] 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 (pages 33, 37, 57).

[149] 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 (pages 33, 37, 57).

[150] 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 (pages 33, 37).

[151] 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 (page 34).

[152] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. «Places: A 10 million Image Database for Scene Recognition». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017) (pages 35, 50).

[153] Dongfang Liu, Yiming Cui, Liqi Yan, Christos Mousas, Baijian Yang, and Yingjie Chen. «DenserNet: Weakly Supervised Visual Localization Using Multi-scale Feature Aggregation». In: *Proceedings of the AAAI Conference on Artificial Intelligence* (May 2021), pp. 6101–6109 (pages 37, 42, 99).

[154] 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 (page 37).

[155] Liu Liu, Hongdong Li, and Yuchao Dai. «Stochastic Attraction-Repulsion Embedding for Large Scale Image Localization». In: *IEEE International Conference on Computer Vision*. 2019 (pages 38, 39).

[156] Diederik Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: *International Conference on Learning Representations* (Dec. 2014) (pages 41, 64).

[157] Karen Simonyan and Andrew Zisserman. «Very Deep Convolutional Networks for Large-Scale Image Recognition». In: *CoRR* abs/1409.1556 (2014). URL: `http://arxiv.org/abs/1409.1556` (page 42).

[158] Giorgos Kordopatis-Zilos, Panagiotis Galopoulos, S. Papadopoulos, and Y. Kompatsiaris. «Leveraging EfficientNet and Contrastive Learning for Accurate Global-scale Location Estimation». In: *ACM International Conference on Multimedia Retrieval* (2021) (pages 45, 46).

[159] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. «Layer Normalization». In: *ArXiv* abs/1607.06450 (2016) (pages 45, 69, 75).

[160] Filip Radenović, Giorgos Tolias, and Ondřej Chum. «CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples». In: *ECCV*. 2016 (page 50).

[161] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. «Scaling Vision Transformers». In: *ArXiv* abs/2106.04560 (2021) (page 63).

[162] Zihang Dai, Hanxiao Liu, Quoc V. Le, and Mingxing Tan. «CoAtNet: Marrying Convolution and Attention for All Data Sizes». In: *ArXiv* abs/2106.04803 (2021) (page 63).

[163] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. «Scaling Vision with Sparse Mixture of Experts». In: *ArXiv* abs/2106.05974 (2021) (page 63).

[164] Ze Liu, Jia Ning, Yue Cao, Yixuan Wei, Zheng Zhang, Stephen Ching-Feng Lin, and Han Hu. «Video Swin Transformer». In: *ArXiv* abs/2106.13230 (2021) (page 63).

[165] William P. Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. «1 year, 1000 km: The Oxford RobotCar dataset». In: *The International Journal of Robotics Research* 36 (2017), pp. 15–3 (page 63).

[166] S. Garg M. Milford and J. Mount. *P1-007: How automated vehicles will interact with road infrastructure now and in the future*. Tech. rep. iMove, QUT and Queensland Government, 2020. URL: `https://imoveaustralia.com/wp-content/uploads/2020/02/P1-007-Milestone-6-Final-Report-Second-Revision.pdf` (page 63).

[167] Niko Sünderhauf, Peer Neubert, and Peter Protzel. «Are We There Yet? Challenging SeqSLAM on a 3000 km Journey Across All Four Seasons». In: 2013 (page 63).

[168] 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) (page 67).

[169] Lilian Weng. «The Transformer Family». In: *lilianweng.github.io/lil-log* (2020). URL: `https://lilianweng.github.io/lil-log/2020/03/27/the-transformer-family.html` (page 67).

[170] T. Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. «ImageNet-21K Pretraining for the Masses». In: *ArXiv* abs/2104.10972 (2021) (pages 70, 83).

[171] Will Kay, João Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Apostol Natsev, Mustafa Suleyman, and Andrew Zisserman. «The Kinetics Human Action Video Dataset». In: *ArXiv* abs/1705.06950 (2017) (page 85).

[172] João Carreira, Eric Noland, Andras Banki-Horvath, Chloe Hillier, and Andrew Zisserman. «A Short Note about Kinetics-600». In: *ArXiv* abs/1808.01340 (2018) (page 85).

[173] Guohao Peng, Jun Zhang, Heshan Li, and Danwei Wang. «Attentional Pyramid Pooling of Salient Visual Residuals for Place Recognition». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* Oct. 2021, pp. 885–894 (pages 98, 99).

[174] Guohao Peng, Yufeng Yue, Jun Zhang, Zhenyu Wu, Xiaoyu Tang, and Danwei Wang. «Semantic Reinforced Attention Learning for Visual Place Recognition». In: *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021.* IEEE, 2021, pp. 13415–13422 (page 98).

[175] Yixiao Ge, Haibo Wang, Feng Zhu, Rui Zhao, and Hongsheng Li. «Self-supervising Fine-Grained Region Similarities for Large-Scale Image Localization». In: *Computer Vision – ECCV 2020.* Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Cham: Springer International Publishing, 2020, pp. 369–386 (page 99).