POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Matematica

Tesi di Laurea Magistrale

Beyond the Standard Domains of Visual (Geo)Localization

Relatori
Prof. Carlo Masone
Gabriele Berton
Gabriele Trivigno

Candidato
Enrico Chiavassa

Anno Accademico 2023-2024

Abstract

Domain Adaptation (DA) is the task of making a model function effectively on domains that differ from the one(s) it was trained on. Visual (Geo-)Localization, on the other hand, is the task of estimating the position or the pose in which a query image has been taken by comparing it to a large labeled database. The objective of this thesis is to investigate two scenarios where this paradigm is deployed in challenging domains which, without proper countermeasures, would cause severe degradation in performances. In particular, the first one regards the deployment of a large scale model in a different city from the one that it has been trained on. This is done through a carefully picked domain adaptation strategy that has then been accurately tuned to best fit the problem. The second scenario questions the performances of state-of-the-art methods in complex indoor environments and presents a simple yet effective fine-tuning procedure that significantly boosts their performances of all models across all the relevant benchmarks.

Contents

Chapter 1

Introduction

One of the most important trends of this century has been the exponential rise of Deep Learning. Although the concept traces its roots back almost a century, it is in the last few years that advancements in hardware capabilities and efficiency have fueled its widespread adoption across academic and industrial worlds. Recently it has even become a mainstream topic and it is often a point of discussion in traditional, non-technical media. This is, at least in part, due the development of *ChatGPT*: for the first time a deep learning model could emulate a human-like level of grammatical, lexical and semantic prowess. Beyond language processing, however, the impact of this paradigm has been pervasive across many domains, such as Computer Vision, Speech Recognition and Robotics. But why is that?

The general aim in the field of Artificial Intelligence is to build intelligent systems capable of comprehending information coming from the world, processing and converting it into knowledge, storing it, and utilizing it for reasoning and interacting again with the world. Each step has its own challenges: first of all, information may originate from diverse sources that have to be interpreted in different ways. Secondly, external inputs are often chaotic and noisy in nature, therefore processing information poses significant challenges: the system must be able to retain only pertinent and useful information and convert it into actual knowledge. Thirdly, knowledge needs to be stored efficiently, as this allows for faster reasoning. Lastly, the way that the model interact with the world is extremely varied. Some systems may be build for specific tasks, while others may be designed to be as general as possible.

The deep learning paradigm has become the state-of-the-art solution to many of this problems. Very generally, the system (usually called model) receives a set of parameters and uses them to elaborate external inputs and to create an output. This is used to evaluate a specific function, referred to as *loss*, which is then utilized to adjust the system parameters through the *back propagation* process. After many repetitions, the quality of the output improves significantly as the system learns how to effectively perform its task. Please note that during this procedure, called *training* there is no explicit human intervention: for this reason deep learning is incredibly powerful and versatile. It does, however, come with some severe and structural issues. The performance of a system is correlated with the amount of data available, while the speed of the training depends on the capacity of the hardware. As the model size grows, both performances and requirements tend to increase. This trend has lead to growing emphasis on aggressive data collection, raising notorious privacy concerns, and to the extensive utilization of powerful yet energy-intensive hardware. The way researchers will tackle this challenges will shape the future of deep learning.

An additional concern with this paradigm is *domain shift*. The data used during the training phase may be different from the inputs given to the deployed model (or, in general, at test time). If this discrepancy is not properly addressed, it may cause heavy degradation in performances. This happens because the model learns to associate some features to the particular conditions encountered during training. Any deviation during testing can result in a loss of efficiency. Several factors contribute to domain shift in *visual* data, including:

- The data source changes. If the device used to create the training set is not the same as the one used for test images, there could be significant difference in resolution, orientation and overall quality. Many training dataset that will be mentioned are filled with pictures taken by industry-grade cameras, but the corresponding test images are taken using a commercial smartphone;

- The type of illumination changes. The lightning conditions play an heavy role on how scenes are perceived in an picture. If a model is accustomed to well-lit images, it will struggle when the scenario changes. In outdoor-related tasks, some important illumination factors are the weather, the season and the time of the day.

- The demographic changes. If the underlying task concerns humans but only a specific demographic group is represented in the training dataset, then the model will not generalize effectively to all people. This is an especially dangerous scenario, as the outputs of the system may be biased in favor or against specific age groups or ethnicities.

The tasks that aim at fixing domain shift are commonly referred to as *domain adaptation* and *domain generalization*. The main idea is to encourage the model to extract information that is invariant to any transient domain, such as the ones mentioned above, effectively making it blind to those changes. In this thesis a range of domain adaptation techniques will be introduced and applied in multiple real-world scenarios, with the scope of enhancing state-of-the-art models in non-standard domains.

Within the many subfields of artificial intelligence, Computer Vision is the one responsible of enabling computers to interpret and understand visual information from the world. Its ultimate goal is to find efficient ways for machines to gain some sort of knowledge from all sorts of visual data, ranging from simple photos and videos to medical and microscopy imaging. For this reason countless computer vision tasks are being studied. Some are more straight forward, such as Object Detection, and some are more structured and complex, such as Egocentric Vision. In this thesis the focus will be on Visual Place Recognition and Visual Localization. The former is the task of estimating the geographical position where a given photo was taken by comparing it with a large database of images of known locations. In most cases the metric of reference is the GPS or UTM coordinates. The latter is the task to determine the pose of a camera in a given picture, which means estimating the spatial position of the camera as well as its orientation. This six degrees of freedom are represented by a three dimensional vector and three rotation angles. The metrics employed are usually meters and degrees with respect to an arbitrary reference system. These two tasks are inherently related to each other, yet they serve two very distinct purposes and have to deal with unique challenges. To illustrate, Visual Place Recognition models operate on a broader scale, as state-of-the-art algorithms work over entire metropolis. In contrast, Visual Localization models find practical applications in indoor environments with stricter thresholds for error.

The remaining chapters will be structured as follows. Chapter two will delve into both Visual Place Recognition and Visual Localization in details, with particular attention to the state-of-the-art approaches and most popular datasets. Chapter three will shift its focus to domain adaptation, starting from the mathematical theory that has been developed and covering all relevant techniques. Chapter four will present the first experimental scenario. The goal is to enhance the performances of a Visual Place Recognition model when deployed in cities that differ from the one used during training. Finally, the fifth chapter will discuss the second experimental scenario, where the focus is on indoor environments. These are notoriously hard and the localization pipeline is composed of multiple steps and model. The results of extensive testing will be shown and discussed, with the goals of providing the widest possible overview of the performances of current methods and finding useful insight into possible improvements.

Chapter 2

Visual Place Recognition and Visual Localization

In this chapter the central tasks of this thesis will be presented, following this general structure. First of all, there will be a broad discussion on the most relevant approaches that, over the years, have proven to be more successful. Then a more detailed analysis will explore some the current state-of-the-art models and many of the most used datasets. Since a full coverage of both tasks would be unfeasible, let alone pointless, the focus will mostly be on methods and datasets that will be used in at least one of the experimental scenarios. For the same reason, there will not be an in-depth explanation of many general concepts of deep learning. However, a reference to useful learning material will be given when they are first mentioned.

2.1 Visual Place Recognition

Visual Place Recognition (VPR), also known as Visual Geo-Localization (VG) or image localization, is the task of recognizing the place depicted in a given image by comparing it to a large dataset of images. This definition is extremely broad on purpose. Indeed, VPR is also studied in communities outside of computer vision, each providing a unique prospective. The most prominent one is robotics, where this task is usually implemented using streams of heterogeneous data and with a particular focus on computational efficiency and real-time execution. The word "place" itself could have different meaning depending on the objective at hand. It may refer to the name of a particular landmark or the 6 degree of freedom (DoF) of the camera pose. In this work the former interpretation is associated with the *landmark retrieval* task, while the latter with the *visual localization* task. In this work the concept of place in VPR always refers to the GPS or UTM coordinates associated with an image, putting the emphasis on the *geo*-localization aspect. In a dynamic and rich research panorama, it is difficult to have such a strict nomenclature. The partition above is quite clearly an over-classification, but is to be intended as an introductory description of VPR and some of its existing instances.

Visual Geo-Localization is generally formulated as an image retrieval problem: a deep model (usually a convolutional neural network) is used to extract a vector of descriptors from each database image. These contains all the information that the network considers relevant in the picture. Then, the same procedure is applied to all query images and similarity scores are computed between queries and database descriptors. This score can be as simple as the *L2* norm distance or a more complex cosine similarity. For each query, the k best scoring images are considered and their coordinates are used as estimates for the query position. If they are within a fixed error threshold, then the prediction is considered correct. The most common metric is *recall@k*, which represents the percentage of queries for which one of the k estimates is correct. Generally the most relevant k is 1. In order to improve the performances, some approaches add a re-ranking step after the retrieval. Most of them are based on geometric verification, which is notoriously computationally intensive.

The retrieval paradigm is applied in many other fields of deep learning, but in VG it faces some unique challenges strictly related to the nature of the task:

1. The scene in which a place appears are complex and there is not a single, identifying object that can be exploited. In most cases the information is scattered across some of the many elements of the image. Some of it may even be hidden behind occlusions and not available to the model.

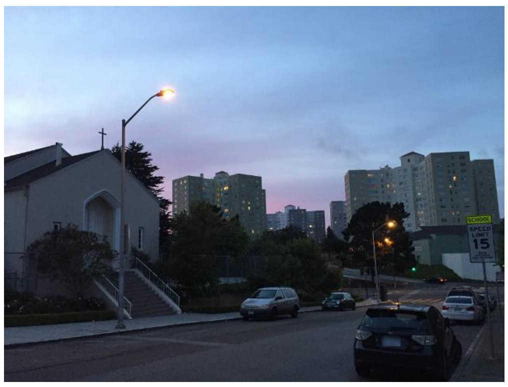


Figure 2.1: In this picture there are multiple elements of interest, from the small house to the left to the tall buildings in the background.

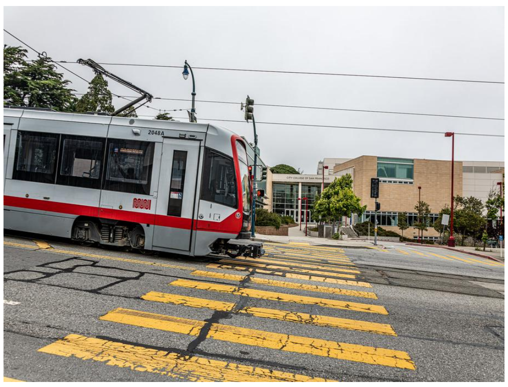


Figure 2.2: Most of this picture is taken up by the street, which is not very informative, and a tram, which possibly hides useful building in the background.

2. In real world scenarios the scenes are extremely dynamic and can change rapidly. This happens not only because of natural illumination variations, such as the day/night cycle, seasons and weather, but also because of the presence of physical objects, such as temporary construction sites or billboards.

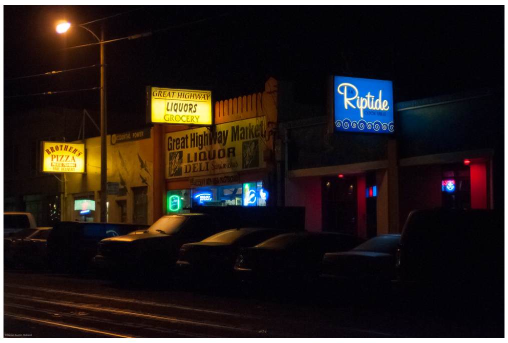


Figure 2.3: A picture taken as night. Note that the lack of natural light highlight the presence of artificial light sources, that are much more predominant than in the daytime.

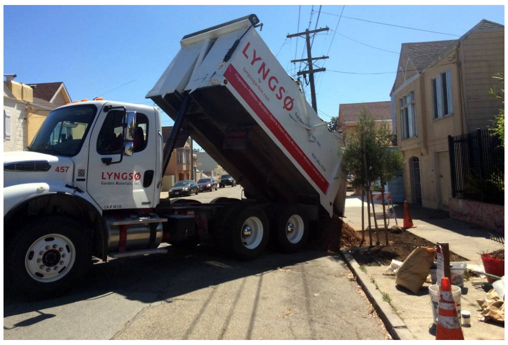


Figure 2.4: Here an example of both a transient obstruction and long term artificial changes are shown. The truck is obscuring most of the current image, but the curated trees will be part of the scenario going forwards.

3. A single scene can be depicted from many different points of view and the resulting images have different appearances. There is no guarantee that the database contains a positive match with the same viewpoint, especially if the images source is a camera with a fixed pose (e.g. a front facing camera on a car). Even if that is the case, an invariance to the viewpoint shift can easily increase the performances of the model.

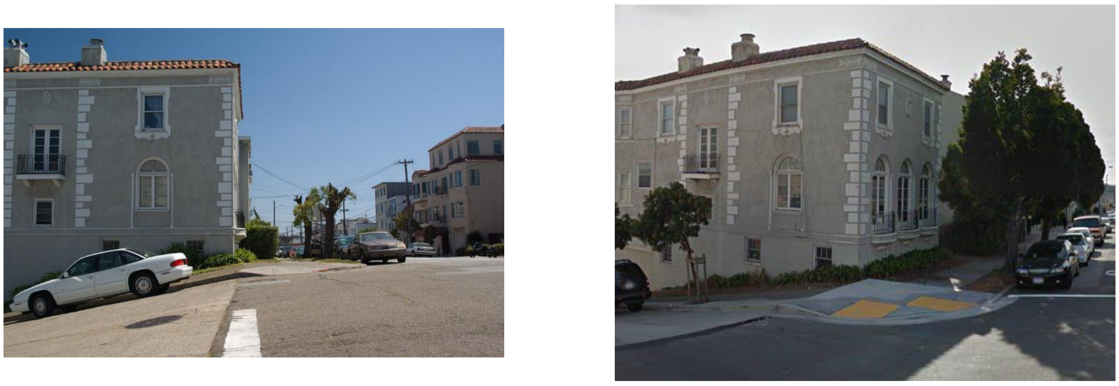


Figure 2.5: Two of pictures representing the same intersection from different viewpoint. The one on the left is a query, while the one on the right is part of the dataset. The shift in viewpoint adds more noisy elements such as trees and a new facade. It also changes the geometric properties of the building and its elements.

Images from Figure 2.1 to Figure 2.4 are all taken from the SF_XL dataset (Berton et al. [2022a]).

2.1.1 Geo-Localization methods over the years

Over the years a variety of different approaches have been proposed. Originally, visual geo-localization algorithms were based on hand-crafted representations of images using local descriptors. All of them uses a keypoints detector to identify interesting points of the image. The patches containing such points are then fed to a local descriptors extractor, such as SIFT (Lowe [1999]) or SURF (Bay et al. [2008]), that generates multiple numerical arrays describing each patch. Each method is characterized by the way it uses this descriptors. Since a full comparison of all database image is unfeasable over large scales, Sivic and Zisserman [2003] takes inspiration from Natural Language Processing and proposes a Bag of Words (BoW) approach where the local descriptors are clustered with respect to a codebook of visual words. Each image is then represented by the histogram of the assignment to visual words of all image descriptors, weighted by the "term frequency – inverse document frequency" (tf-idf). This index is designed to promote visual words that appear frequently in an image but not frequently overall.
A clear improvement over BoW was introduced in Jégou et al. [2010]. The main idea was to not just use the assignment of each local descriptor to the visual word, but to compute the difference between each of their components. This way a lot more information is preserved and exploited during the matching phase, and performances improve significantly. Even before the advent of convolutional neural networks there has been some proposed approaches using global descriptors i.e. descriptors that do not focus on single patches of scene, but rather encode its holistic properties (Dalal and Triggs [2005], Oliva and Torralba [2006]). This approach is generally less robust to the issues mentioned above, but it does not require the extraction and matching of the local descriptors, thus being significantly faster over large scale datasets. A more thorough investigation of shallow approaches can be found at Lowry et al. [2016]

In order to overcome the limitations of global descriptors, while retaining their computational advantages, researchers tried to use Convolutional Neural Networks (CNNs) to generate image representations. Their experimentations proved that, after some proper processing, the information extracted from the convolutional layers of a CNN were akin to global descriptors (Babenko et al. [2014]). In particular, the tensor output of this kind of layers has shape $H \times W \times C$, where H is the height, W is the width and C is the number of channel. In order to convert it into a one dimensional array, a simple flattening is not enough, as it loses all spatial information contained in the tensor. There are two possible approaches to this objective:

- **Aggregated Representations**: the output tensor can be seen as a set of densely extracted local descriptors (specifically, a $H \times W$ grid of C-dimensional descriptors). These can then be aggregated using classical encodings such as BoW and VLAD (Sivic and Zisserman [2003] and Jégou et al. [2010] resp.) or using more advanced modules with learnable parameters. The most important one, which is still competitive today, is NetVLAD (Arandjelović et al. [2016]). The idea is to replace the hard, non differentiable assignment to each cluster center with a softmax operation, which is differentiable and exploitable for backpropagation. This module can then be easily plugged on top of a CNN backbone and the parameters of the softmax assignment are learned during training. This increases significantly the flexibility of the method and its performances.

- **Pooled Representations**: the output sensor is aggregated and compared without any embedding process. The argument is that the convolutional features have an higher discriminatory power than hand-crafted ones, so they can be pooled together following a simple scheme, such as:
  - Max/Mean/Sum-Pooling: using a sliding window of fixed size, the values of each feature map are pooled together with a max/mean/sum operation. Mousavian and Kosecka [2015] observes that max-pooling is more invariant to a change in scale, while sum-pooling is more robust against noisy features;
  - R-MAC: Tolias et al. [2016] max pooling is applied to a number of patches randomly sampled from the feature maps generated by the convolutional network. The results can be seen as *regional* descriptors. These are finally summed and $L2$-normalized, in order to keep their dimensionality low. This whole procedure was improved multiple times: one notable example is Gordo et al. [2017], that introduced a differentiable sampling procedure so that R-MAC could be used in top of a CNN for end-to-end training. Other more complex variations are still extremely competitive today;
  - GeM: this simple layer was introduced by Radenović et al. [2018] and implements the generalized mean operator. This is differentiable, hence the mean parameter can be learned during training, offering state-of-the-art performances.

When discussing and selecting models later in the experimentations chapters, a lot of weight will be given to the dimensionality of the descriptors. This characteristic is extremely important for efficiency (and thus scalability) purposes, as larger descriptors will require more space in memory and will lead to heavier computations. Even a simple K Nearest Neighbours algorithms scales linearly with their size. For this reasons many new models that have been introduced over the years are designed to perform well even with limited descriptors dimension.

2.1.2 Training a model

Since it became evident that deep learned models would outperform any shallow counterpart, researched questioned what training procedure would best fit the Geo-Localization task. Their efforts can be partitioned into three different approaches.

- **Classification Loss**: the first CNN-based representations ever used came from model that were trained for classification, generally on ImageNet, and not fine-tuned for VPR. This simple approach had decent performances, as the featured extracted by a convolutional network are somewhat transferable across tasks, but was promptly superseded by more task specific methods. Generally speaking, the issue of using classification for training purposes is that it enhances the robustness of

the descriptor against intra-class variability. This is generally not a desired property for image retrieval, where it is important to distinguish between particular objects even if they belong to the same semantic class Gordo et al. [2017].
For many years the focus shifted away from classification and towards contrastive, triplet or ranking losses, which will be presented briefly. Recently the trend seems to have shifted, as classification techniques have been further investigated. For example Cao et al. [2020], Yokoo et al. [2020] and Berton et al. [2022a] use the ArcFace loss (Deng et al. [2022]) for the training of a network using only image level labels. The main reasons for this resurgence are two: firstly, the descriptors produced by such methods are compact. The importance of this property has been already discussed. Secondly, only image level labels are needed. This means that there is no need for the expensive mining of negative samples, which is the main drawback of constrastive learning.

- **Contrastive learning**: most studies regarding image retrieval for VPR use either the constrastive or the triplet loss. Both follow the idea of metric learning, i.e learning to extract descriptors that represent well the similarity under a distance function. This is done by feeding the model both positives and negative examples for each training sample. To be clear, in the context of VPR the former are images taken in the same position, while the latter are taken in a different place. The network is therefore encouraged to create a representation of the sample that is close to those of positives samples and distant to those of negative ones. Since this paradigm has been mainstream for many years numerous variations on the theme have been published. Some examples can be found in Arandjelović et al. [2016] and Ong et al. [2017].
  As it was already mentioned, the main drawback of this approach is the process of finding the appropriate positive and negatives examples. The former can be extracted (or *mined*) offline by exploiting their labels (if available) or 3D models Radenović et al. [2018]. The latter are especially tricky, as there is not really an obvious way to define them. One of the most popular approaches is mining "hard" negatives, i.e images of a different place that have similar descriptors. Some example are shown in fig. 2.6 In general it is important to strike a balance between examples that are too easy, and therefore not effective for learning, and too difficult, which could lead to overfitting and/or local minima of the loss function (Radenović et al. [2016]). A full rundown of many of the losses and mining variations can be found in Masone and Caputo [2021].

- **Listwise Ranking**: other than sample mining, constrastive learning has another significant limitation. *Mean average precision (mAP)* is generally considered as an appropriate metric for a model performance, but the losses used (both triplet and constrastive) are not related to it in a satisfactory way. Indeed it has been proven by Liu [2009] that they are simply upper bounds on mAP. The idea of the listwise loss is to modify the computation of the average precision to make it differentiable. This is done by implementing histogram binning and a differentiable soft assignment Revaud et al. [2019a].

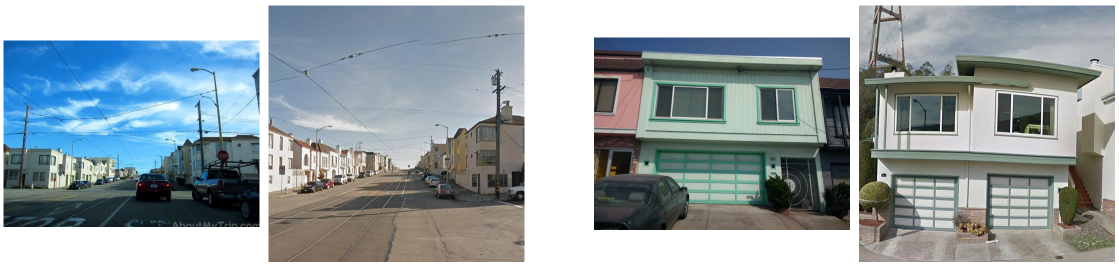


Figure 2.6: The first and the third images are training samples, while the second and fourth ones are examples of hard negatives. They are of similar appearances, but taken in different places. These hard examples have been found by using a pre-trained VPR model and selecting two of its wrong predictions.

2.2 Visual Localization

Visual Localization (VL) is the task of estimating the position and the orientation of an image. Generally it is studied within the robotics community as part of the broader Simultaneous Localization And Mapping (SLAM) paradigm. Multiple deployment scenarios require different approaches, which Chen et al. [2023] summarize as:

- Incremental Motion Estimation: given a stream of frames from a sensor, estimate the variation in translation and rotation between two consecutive frames. For example, a camera mounted on the front of a car can be used to determine its position and orientation in real time;

- Global Relocalization: given a known scene and some prior knowledge, retrieve the pose of some sort of agent by matching the input image with a map of the environment. This can alliviate the pose drift, i.e. the difference between the current estimated position and the actual one, or can be used to tackle the kidnapped robot problem, which arises when the agent position changes without it being able to keep track.

- Mapping: build a consistent model of the surrounding environment. This can then be used for various purposes, generally to provide information to human operators or for robot tasks, including the one mentioned in *global relocalization*.

- Loop Closing: when mapping an environment, an agent may end up in a place it already visited. Is is important for it to be able to recognize that this is happening and to therefore "close the ring" in its mapping, by properly connecting the two different parts of the map.

Each scenario is tackled in its own way and it would be vain to describe all of them. Instead, the focus of this work will be limited to *global relocalization in a 3D map*, where the task is to recover the camera pose of a 2D image with respect to a 3D prebuilt scene model. In most cases the map of the space is built using structure-from-motion (SfM)

algorithms and database of images is available. The general idea is to first establish 2D-3D correspondences between the pixels of a given image (*query*) and the points of the 3D model, and then use those matches to estimate the camera pose by solving a perspective problem.
The first step can be solved by establishing 2D-2D correspondences between the query and k database images, which are then matched with the 3D map (this last computations can be done offline, thus speeding up the whole procedure). The final result is the desired set of 2D-3D matches. The correspondences are computed using a keypoints detector and a local features extractor. The former identifies some elements of interest within the image and the latter generates the corresponding vectors of descriptors. We discussed some proposed methods in the previous section. Different approaches change the order of this two procedures, even attempting to do them simultaneously (Revaud et al. [2019b]).
It is particularly important to choose the best possible value of k. At least in theory, it is enough to retrieve a single relevant image, but a larger k makes it more likely to happen. On the other hand, since 2D-2D matches have to be computed online, local feature matching is the bottleneck of the whole algorithm. Moreover, a larger k will also introduce more noisy matches that may hinder the performances of the second step. The value of k should therefore be set to be as low as possible.
But how are those database images selected? They are then predictions made by a VPR algorithm applied to the queries and the dataset. The quality of this predictions is strictly correlated to the optimal value of k, as a better model would allow to lower it while guaranteeing the retrieval of at least a single relevant image. Visual Localization has been studied in a wide array of domains, but the datasets that will be used in this thesis are all from indoor environment. In order to design an effective VPR model for the retrieval in this context, additional challenges have to be kept in mind:

- Repetitive or non informative elements. Many indoor scenes may contain multiple instances of very similar objects or structures, which can easily trick a visual localization model. Another issue is reflective or non informative surfaces, such as windows or plain walls, that should be ignored. All these elements are not as predominant in urban datasets, so most models (that are trained for urban environment) are not trained to deal with them. An example is shown in Figure 2.7

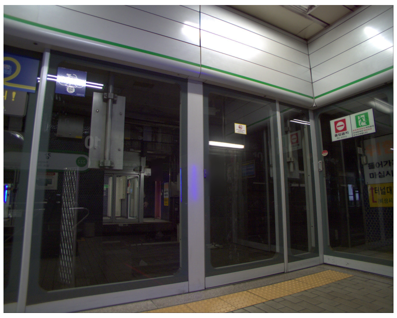


Figure 2.7: Example of both repetitive and reflective structure.

- When images are taken in crowded public environment, the likeliness of human obstructions is very high. Again, this is not the case for large outdoor datasets. An example is shown in Figure 2.8

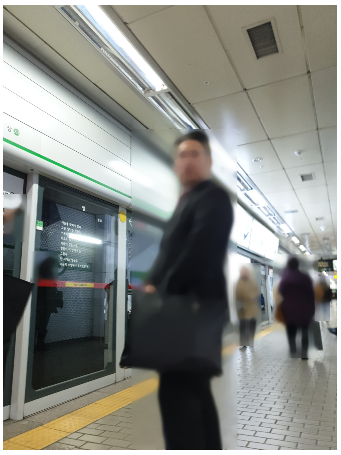


Figure 2.8: Example of prevalent human obstruction.

- All indoor datasets have limited size, and thus can only be used for testing and not for training. To be more specific, the indoor datasets mentioned in this thesis contain a few thousand images, while the outdoor ones have millions. The second experimentation scenario has been designed to tackle this exact problem.

The second step is generally solved by combining a Perspective-n-Problem (PnP) algorithm with RANSAC (Fischler and Bolles [1981]), in order to compensate for noise and outliers. A full discussion on this topics is out of the scope of this work. A more insightful analysis can be found in Humenberger et al. [2022].

Chapter 3

Domain Adaptation

When introducing both Visual Place Recognition and Visual Localization, one concept that has been mentioned multiple times is the *domain* as a general description of the environment in which the pictures are taken. For example, in Section 2.1 the day-night shift was mentioned, while in Section 2.2 the focus was on outdoor-indoor differences. Since deep learning models requires large amounts of data to be trained, the sources of the training images are generally limited. On the other hand, there is no such limitation regarding the deployment environment which can be extremely varied. This creates an incoherence between what the model learns and what it has to deal with at test time, which may lead to a significant decrease in performances. This difference is generally referred to as *domain shift* and *domain adaptation* is the task that aims at mitigating the effects of such gap. This chapter will contain a complete introduction to the theoretical fundamentals of Domain Adaptation, followed by a presentation of the methods that have some relevance within the scope of this thesis.

3.1 The theoretical background

Domain Adaptation theory has received plenty of attention over the years thanks to the simplicity in which it can be described in mathematical terms, starting from the definition of domain itself.

Definition 3.1. *A domain D is defined as a quadruple $(X, Y, \mathbb{P}(X), \mathbb{P}(X|Y))$, where:*

- *X is the feature space;*
- *Y is the label space;*
- *$\mathbb{P}(X)$ is the marginal distribution probability of X;*
- *$\mathbb{P}(X|Y)$ is the conditional probability distribution of Y given X.*

In most of Machine Learning applications, the object is to learn $\mathbb{P}(X|Y)$ from a set of observations and labels $\{(x_i, y_i)\}$. More specifically, this is referred to as supervised learning. In an ideal scenario both training and testing data would be originated from the same domain, however this is often not the case.

Definition 3.2. *The source domain D_S represents the origin of the training data, while the target domain D_T the one of the test data. They are denoted respectively as:*

$$D_S = (X_S, Y_S, \mathbb{P}_S(X), \mathbb{P}_S(X|Y)) \quad D_T = (X_T, Y_T, \mathbb{P}_T(X), \mathbb{P}_T(X|Y))$$

The notation will be lightened in the following way:

$$D_S = (X_S, Y_S, S(X), S(X|Y)) \quad D_T = (X_T, Y_T, T(X), T(X|Y))$$

Please note that the amount of data from the target domain is generally not enough for an effective training of the model, as this would already be an effective solution to the domain adaptation problem.

How can D_S and D_T differ from each other? In the wider context of *transfer learning* there are countless way as there could even be a difference in the underlying task, as illustrated in Pan and Yang [2010], but in domain adaptation it is common to assume that $Y_S = Y_D$ and that the task is the same in both domains. Having said that, Kouw and Loog [2021] identify three different ways in which D_S and D_T may differ, starting from the fact that $\mathbb{P}(X, Y) = \mathbb{P}(Y|X)\mathbb{P}(X) = \mathbb{P}(X|Y)\mathbb{P}(Y)$

- *prior shift*: the conditional distribution remains the same, but the prior is different i.e. $S(X|Y) = T(X|Y)$ but $S(Y) \neq T(Y)$;
- *covariate shift*: the posterior distribution remains the same, but the data distribution is different. i.e. $S(Y|X) = T(Y|X)$ but $S(X) \neq T(X)$. This translates to a bias in the sample selection;
- *concept shift*: it the opposite case of covariate shift, as the data distribution is the same but the posterior is different i.e. $S(X) = T(X)$ but $S(Y|X) \neq T(Y|X)$.

This classification is not often reflected in real world applications, as multiple of those cases can be true at the same time. However, most of the efforts made are towards the covariate shift case.

Now that the possible differences between domains have been presented, it is crucial to discuss a quantitative way to represent them starting from some notation:

- from now on δ will always represent the desired confidence that any bound holds, as they will always be a Probably Approximately Correct (PAC) bound;
- all the theory that will be presented was developed in the case of classifiers, specifically for scenario in which the number of candidates is finite. The set that contains all the considered classifiers (or *hypothesis*) is called $\mathcal{H}$ and an arbitrary element is referred to as h;

The first relevant equation was introduced in Ben-David et al. [2010] and describes the upper bound on the cross-domain generalization error of a classifier trained on the source data and deployed in the target domain with respect to the optimal classifier. Before presenting that equation, it is useful to introduce the following:

Definition 3.3. *The symmetric difference hypothesis divergence $d_{\mathcal{H}\Delta\mathcal{H}}(S,T)$ is the quantity that evaluates to what extent two classifiers disagree with each other on both domains.*

$$d_{\mathcal{H}\Delta\mathcal{H}}(S,T) = 2 \sup_{h,h' \in \mathcal{H}} \left| \, \mathbb{E}_{x\sim D_S}\left[\left|h(\mathbf{x}) - h'(\mathbf{x})\right|\right] - \mathbb{E}_{x\sim D_T}\left[\left|h(\mathbf{x}) - h'(\mathbf{x})\right|\right] \right| \tag{3.1}$$

Definition 3.4. *The error of the ideal joint hypothesis is:*

$$err^*_{S,T} = \min_{h\in\mathcal{H}} \left[err_S(h) + err_T(h)\right] \tag{3.2}$$

Now that all the preliminary elements have been defined, it is possible to present the PAC bound on cross-domain generalization error.

Theorem 3.5. *The deviation between the target generalization error of a classifier trained on the source data, $err(\hat{h}_S)$ and the target generalization error of the optimal target classifier $err(\hat{h}_T)$ can be defined as:*

$$err(\hat{h}_S) - err(\hat{h}_T) \leq err^*_{S,T} + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(S,T) + C(\mathcal{H},\delta) \tag{3.3}$$

where $C(\mathcal{H},\delta)$ is a constant that depends on complexity of the classifiers in $\mathcal{H}$ and the desired confidence δ (Vapnik [2000]).

As it will be discussed later, domain adaptation methods try to minimize the first two terms of Equation (3.3) by building a classifier that simultaneously works well on the source domain and whose features are invariant with respect to the domain of the data. The latter is achieve by making sure that the induced source and target distributions in the representation space are as close as possible.

The main issue with the $d_{\mathcal{H}\Delta\mathcal{H}}(S,T)$ divergence is that it does not show a way to practically extract domain invariant features. In particular, it is not obvious what it means for two distributions to be close. Several functions have been proposed for the purpose of effectively quantifying the distance between two domains, and the most relevant ones will be now presented.

Maximum Mean Discrepancy

One of the most popular distribution distance measures in domain adaptation is the Maximum Mean Discrepancy, introduced in Borgwardt et al. [2006]. The formal expression for MMD is:

$$D_{MMD}(S,T) = \sup_{f\in\mathcal{F}} \left(\mathbb{E}_{\mathbf{x}\sim T(X)}[f(\mathbf{x})] - \mathbb{E}_{\mathbf{x}\sim T(X)}[f(\mathbf{x})]\right) \tag{3.4}$$

where:

- $\mathbb{E}_{\mathbf{x}\sim D(X)}[\cdot]$ is the expectation under the distribution $D(X)$;
- $\mathcal{F}$ is the set of functions that transform the data.

Before discussing why the MMD is so useful for DA, it is necessary to introduce the well-renowned concept of Reproducing Kernel Hilbert Space (RKHS).

Definition 3.6. *A reproducing kernel Hilbert $\mathcal{H}$ space is defined as a Hilbert space of real-valued functions in which all evaluation functionals can be be represented as an inner product between the argument and a particular function $K_x \in \mathcal{H}$. To be more specific, given an arbitrary set X on which the functions are defined, an evaluation functional is:*

$$L_x : \ f \to f(x) \quad \forall f \in H \tag{3.5}$$

If $\mathcal{H}$ is a RKHS, then it exists $K_x \in \mathcal{H}$ so that

$$L_x(f) = f(x) = \langle f, K_x \rangle \quad \forall x \in X \tag{3.6}$$

Since $K_x \in \mathcal{H}$ $\forall x \in X$, it is possible to write:

$$L_y(K_x) = K_x(y) = \langle K_x, K_y \rangle \tag{3.7}$$

and define the reproducing kernel of $\mathcal{H}$ as:

$$K : \ x, y \to \langle K_x, K_y \rangle \tag{3.8}$$

By imposing that $\mathcal{F}$ is the set of functions in the unit ball in a RKHS, it is possible to prove (Gretton et al. [2012]) that the MMD is null if and only if the source and target distribution are equal, which is the exact desired property.
Another important quality of the MMD is that its empirical estimate is simple to compute. Given a set of n labeled observations from the source domain and another one of m from the target domain, the formal expression would be:

$$D_{MMD}(X_S, X_T) = \| \frac{1}{n} \sum_{i=1}^{n} \phi\left(x_i^S\right) - \frac{1}{m} \sum_{i=1}^{m} \phi\left(x_i^T\right) \| \tag{3.9}$$

where ϕ is the mapping to a RKHS $\mathcal{H}$. However, Gretton et al. [2012] proved that it can be expressed in terms of kernel function values by using the *kernel trick*. This means that it is possible to compute the distance between the sample means in a high dimensional feature space with requiring the explicit mapping function.

f-Divergences

A different approach to the matter is offered by f-divergences. In this scenario, probability distributions are considered as elements of a Riemannian manifold. In particular, given an arbitrary set X and a family $p(x|\theta)$ of probability density functions parameterized by θ on X, the space $\{p(x|\theta) | \theta \in \mathbb{R}^d\}$ is a Riemannian manifold, usually referred to as the statistical manifold. This concepts are discussed at length in Baktashmotlagh et al. [2017]. An f-divergence is nothing but a way in which the metric on the manifold, called Fisher-Rao, can be approximated. Its general expression is:

$$D_f(S\|T) = \int f\left(\frac{S(x)}{T(x)}\right) T(x)\, dx \tag{3.10}$$

where f defines the specific instance of divergence.

Kullback-Leibler and Jensen-Shannon divergences

A particular example of f-divergence is the **Kullback-Leibler** divergence, where $f(x) = xlog(x)$. The full expression is:

$$D_{KL}(S\|T) = \int S(x)\, log\left(\frac{S(x)}{T(x)}\right) dx \tag{3.11}$$

This particular formulation has some issues, as it is not symmetric and does not satisfy the triangular inequality. For this reason the **Jensen-Shannon divergence** was introduced:

$$D^2_{JS} = \frac{1}{2} D_{KL}(S\|M) + \frac{1}{2} D_{KL}(S\|M) \tag{3.12}$$

where M is a mixture distribution of S and T. *This quantity can be thought of as the square of the mutual information between a random variable X, associated to a mixture distribution between S and T, and a binary indicator Z used to switch between S and T to produce the mixture* (Csurka et al. [2022]). One of the most important property has been proved by Goodfellow et al. [2014], and is the fact that it is equal to the loss used to train Generalized Adversarial Models, minus a constant. This gives a strong theoretical foundation to the adversarial paradigm in domain adaptation, which will be built upon in the next chapter.

Wasserstein Distance

The last distance that will be discussed is the Wasserstein distance. Given the set Γ of all the joint distribution whose marginals are S and T, the p-th Wasserstein distance is defined by:

$$W_p(S,T) = \left(\inf_{\gamma \in \Gamma(S,T)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}[d(\mathbf{x},\mathbf{y})^p]\right)^{1/p} \tag{3.13}$$

The most used instance of distance is the so-called *Earth Mover's*, which requires $p = 1$, because its computation can be efficiently solved as an optimal transport problem. Many modern DA approaches are related to this distance, as they solve optimal transportation problems to align the distributions of the two domains.

3.2 Proposed Methods

Over the last decades an astounding number of domain adaptation methods have been proposed in the literature. In this section the most important ones are presented and their relationship with the theoretical background is discussed.
The introduction of the deep learning paradigm had a significant impact, as a models properly trained could extract robust and general features without any domain adaptation. For this reason the focus of the proposed methods shifted heavily towards end-to-end training and fine-tuning procedures. Since this work is focused on deep learning, shallow methods are largely ignored and only mentioned when they have lead to deep-learned counterparts.

All methods can generally be divided into two categories, based on their scope:

- **aligning data representations**: the ultimate goal is to align the distributions of the features extracted by the backbone of the model.
- **aligning data distributions**: instead of focusing on the representations, these approaches aim at directly bridging the gap between the distributions of the data i.e. making it so that $S(X) = T(X)$.

The underlying idea is the same in both categories, and revolves around minimizing the second term in the right hand side of Equation (3.3). The difference is simply on the level on which they act, feature and image respectively. Moreover,

Aligning Data Representations

This category contains a plethora of different ideas, many of which were the foundation of shallow methods. One of the most relevant is called *CORAL* and aims at aligning the second order statistics of the features from the two domains, i.e. aligning the covariance matrices. In a shallow context this could be done by computing a transformation matrix Sun et al. [2016], while the deep counterpart utilizes the squared Frobenius norm between the covariance matrices Sun and Saenko [2016]. The idea behind *CORAL* was further exploited to create more sophisticated approaches.
There are many more other methods that deserve to be mentioned, but since both experimental chapter will discuss methods that do not belong in this category, they will be skipped. More information about them can be found in Csurka et al. [2022]

Aligning Data Distributions

The first batch of methods that will be introduced follow the same general procedure: a Siamese network with two streams, one for each domain, is trained using a task specific loss on the source branch and a distribution alignment loss defined on both the source and the target samples and employed on the last activation layer before the soft-max. The latter loss is generally chosen to reproduce one of the distances presented in Section 3.1. For example it is possible to implement an empirical Maximum Mean Discrepancy based loss:

$$D_{MMD}(X_S, X_T) = \| \frac{1}{|X_S|} \sum_{i=1}^{|X_S|} \phi\left(x_i^S\right) - \frac{1}{|X_T|} \sum_{i=1}^{|X_T|} \phi\left(x_i^T\right) \| \tag{3.14}$$

where X_S and X_T are the activations of the network in the source and target branches respectively. In the spirit of deep learning, the means are not computed over all the dataset but rather on the current training batch. Further improvements have been proposed in Long et al. [2015] and Li et al. [2020]. Alternatively Zhuang et al. [2015] proposes a loss based on the Kullback-Leibler divergence while Damodaran et al. [2018] explores the utilization of the Wasserstein Distance.

The other batch of methods follow instead a different paradigm, called *adversarial learning*, where two connected network work against each other in an attempt to improve the robustness and domain-invariance of the extracted features. All these methods fall under the name of Generalized Adversarial Networks and, as mentioned above, it has been proven that they minimize the Jensen-Shannon divergence. All these approaches rely on two elements:

- A domain classifier θ_d, usually called *discriminator*, which is trained to classify each sample to its domain.
- The main model, which is trained to extract features from samples of both domains.

The adversarial idea is in the fact that, while θ_D tries to learn to discriminate between domains, the main model learns to fool it by creating features that are domain-invariant. Those are then passed onwards and used for the downstream task, for example to another classifier.

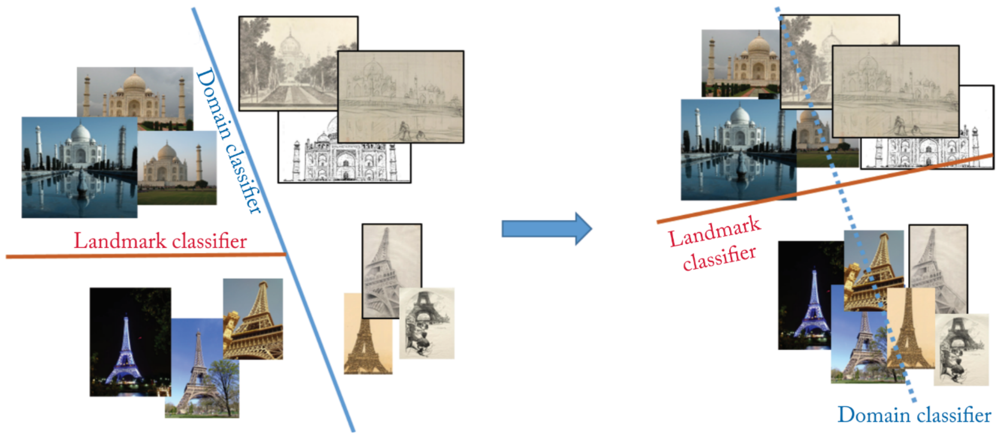


Figure 3.1: A visual representation of the effect of adversarial domain adaptation to the image representations. On the left the two domains are well separated but the main model, a landmark classifier in this example, currently works only on the "photo" domain. On the right, the domains are much closer to each other while maintaining the same class separation. This allows the main classifier to work effectively on the "sketch" domain. Image taken from Csurka et al. [2022]

One of the first adversarial domain adaptation approaches was proposed in Ganin and Lempitsky [2015] and Ganin et al. [2016]. The authors presents the following scanerio and architecture:

- The origin domain is known for all data, however only source data is labeled;
- There is a main model that extract features from the data, which is represented by the mapping G_f;
- There is a label predictor that, during training, receives only features vectors of source data. It is represented by the mapping G_y and is associated with an arbitrary loss L_y. There are no limitations on which loss can be used;

- There is the aforementioned domain classifier that takes all feature vectors as input. It is represented by the mapping G_d and is associated with a binary loss L_d;
- Each part of the model contains learnable parameters that will be referred to as θ_f, θ_y and θ_d.

The goal is to find the optimal θ_f, θ_y so that L_y is minimized and L_d is maximized, while simultaneously find the optimal θ_d so that L_d is minimized. In Figure 3.2 it is possible to see a clear representation of this model.

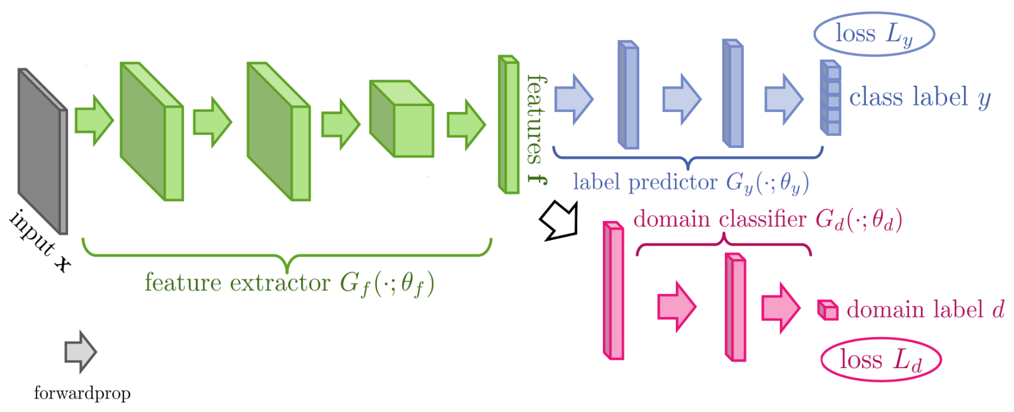

loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
domain classifier $G_d(\cdot;\theta_d)$
input x
feature extractor $G_f(\cdot;\theta_f)$
domain label d
loss L_d
forwardprop

Figure 3.2: A visual representation of the full architecture, data flow and losses. Image taken and modified from Ganin et al. [2016]

.

Formally, the training is formulated as the min-max problem:

$$\left(\hat{\theta}_f, \hat{\theta}_y\right) = \underset{\theta_f, \theta_y}{\arg\min}\, E\left(\theta_f, \theta_y, \hat{\theta}_d\right) \tag{3.15}$$

$$\hat{\theta}_d = \underset{\theta_d}{\arg\max}\, E\left(\hat{\theta}_f, \hat{\theta}_y, \theta_d\right) \tag{3.16}$$

where

$$E\left(\theta_f, \theta_y, \theta_d\right) = \frac{1}{|X_S|} \sum_{\mathbf{x} \in X_S} L_y(\mathbf{x}) - \lambda \left(\frac{1}{|X_S|} \sum_{\mathbf{x} \in X_S} L_d(\mathbf{x}) + \frac{1}{|X_T|} \sum_{\mathbf{x} \in X_T} L_d(\mathbf{x}) \right) \tag{3.17}$$

Please note that this formula contains some notation abuses, as the losses function should formally be written as:

- $L_y\left(\mathbf{x}\right) \sim L_y\left(G_y\left(G_f\left(\mathbf{x};\theta_f\right);\theta_y\right)\right)$;
- $L_d\left(\mathbf{x}\right) \sim L_d\left(G_d\left(G_f\left(\mathbf{x};\theta_f\right);\theta_d\right)\right)$;

but one may see how confusing it would be to use it every time.

Now that the problem has been defined, it is time to present a way to solve it. This is done in an extremely simple and elegant way: by introducing the *Gradient Reversal Layer*. This new element operates as an identity function during the forward pass and as a negative multiplicative coefficient during the backpropagation step. It is set between the feature extractor and domain classifier (please note the white forwardprop arrow in Figure 3.2 and Figure 3.3). By doing so, the contribution of the loss L_d to the learning of θ_d and θ_f is opposite. The former will aim at actually minimizing the loss, while the former will maximise. Formally, the complete loss used to train the model is the pseudo-function of θ_f, θ_y and θ_d:

$$\begin{aligned}\tilde{E}\left(\theta_f, \theta_y, \theta_d\right) = & \frac{1}{|X_S|} \sum_{\mathbf{x} \in X_S} L_y\left(G_y\left(G_f\left(\mathbf{x}_i; \theta_f\right); \theta_y\right)\right) \\ & - \lambda \frac{1}{|X_S|} \sum_{\mathbf{x} \in X_S} L_d\left(G_d\left(R\left(G_f\left(\mathbf{x}_i; \theta_f\right)\right); \theta_d\right)\right) \\ & - \lambda \frac{1}{|X_T|} \sum_{\mathbf{x} \in X_T} L_d\left(G_d\left(R\left(G_f\left(\mathbf{x}_i; \theta_f\right)\right); \theta_d\right)\right)\end{aligned} \tag{3.18}$$

where R is the gradient reversal layer. By minimizing this loss it is possible to solve the problem presented in Equation (3.15) and Equation (3.16) and obtain a feature extractor that is robust to the shift in domain from S to T. In Figure 3.3 it is possible to see a representation of the whole training procedure.

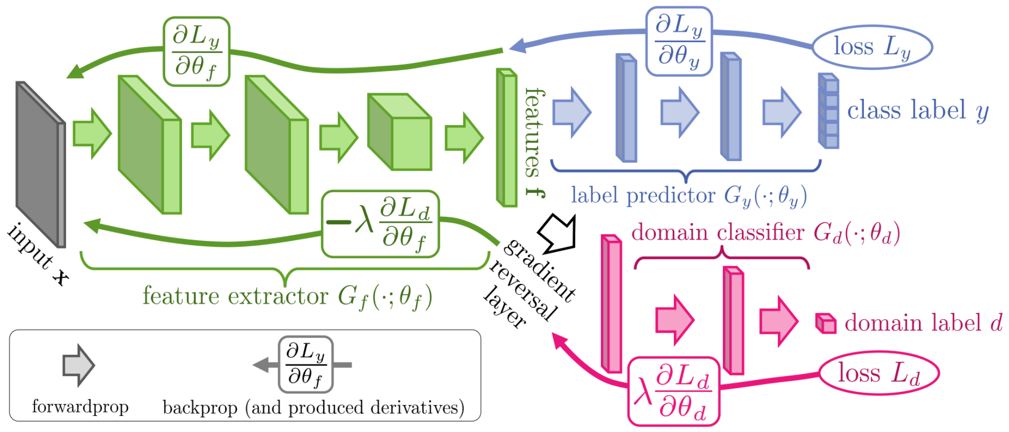

$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
features $\mathbf{f}$
class label y
label predictor $G_y(\cdot; \theta_y)$
$-\lambda \frac{\partial L_d}{\partial \theta_f}$
input $\mathbf{x}$
feature extractor $G_f(\cdot; \theta_f)$
gradient reversal layer
domain classifier $G_d(\cdot; \theta_d)$
domain label d
$\lambda \frac{\partial L_d}{\partial \theta_d}$
loss L_d
forwardprop
$\frac{\partial L_y}{\partial \theta_f}$
backprop (and produced derivatives)

Figure 3.3: A full representation of the training procedure of the model. The key element is the change in sign of the gradient when it flows from the domain classifier, in pink, to the feature extractor, in green. Image taken from Ganin et al. [2016]

.

Relation with distribution distance measures

Now that the method has been fully presented, it is interesting to see how it relates to the theoretical measures and bounds presented in Section 3.1. First of all it will be shown how the adversarial training paradigm is related to the Shannon-Jensen divergence (Equation (3.12)).
In a scenario where there is a feature extractor F and a domain classifier D, assume that the features of the source data $\mathbf{x_S}$ and of the target data $\mathbf{x_T}$ follows two different distributions, called p_S and p_T. During training, D learns to predict the origin domain of the features, while F learns to minimize $\log\left(1 - D\left(F\left(x_T\right)\right)\right)$. This procedure can be written in the form of a min-max problem:

$$\min_F \max_D \left(\mathbb{E}_{F(\mathbf{x})\sim p_S}[\log D\left(F\left(\mathbf{x}\right)\right)] + \mathbb{E}_{F(\mathbf{x})\sim p_T}[1 - \log D\left(F\left(\mathbf{x}\right)\right))]\right) \tag{3.19}$$

Theorem 3.7. *The min-max problem presented in Equation (3.19) has global optimum of $-\log 4$ for $p_S = p_T$.*

Proof. Fixed any F, the training criterion for the domain classifier D is to maximize:

$$V\left(F, D\right) = \int_{\mathbf{x}} p_S\left(\mathbf{x}\right) \log\left(D\left(F\left(\mathbf{x}\right)\right)\right) + p_T\left(\mathbf{x}\right) \log\left(1 - D\left(G\left(\mathbf{x}\right)\right)\right) dx \tag{3.20}$$

For each fixed $\mathbf{x}$, the maximum value of the integrating function is achieved for $D\left(F\left(\mathbf{x}\right)\right) = \frac{p_S(F(\mathbf{x}))}{p_S(F(\mathbf{x}))+p_T(F(\mathbf{x}))} = \frac{p_S(\mathbf{x})}{p_S(\mathbf{x}))+p_T(\mathbf{x})}$. The second equality is only a simpler notation.

For this reason it is possible to define the virtual training criterion $C\left(F\right)$ as:

$$C\left(F\right) = \mathbb{E}_{F(\mathbf{x})\sim p_S}\left[\log \frac{p_S\left(\mathbf{x}\right)}{p_S\left(\mathbf{x}\right) + p_T\left(\mathbf{x}\right)}\right] + \mathbb{E}_{F(\mathbf{x})\sim p_T}\left[\log \frac{p_T\left(\mathbf{x}\right)}{p_S\left(\mathbf{x}\right) + p_T\left(\mathbf{x}\right)}\right] \tag{3.21}$$

In order to minimize it, one could note that each term could be rewritten as:

- $\mathbb{E}_{F(\mathbf{x})\sim p_S}\left[\log \frac{p_S(\mathbf{x})}{p_S(\mathbf{x})+p_T(\mathbf{x})}\right] = \mathbb{E}_{F(\mathbf{x})\sim p_S}\left[\log \frac{2p_S(\mathbf{x})}{p_S(\mathbf{x})+p_T(\mathbf{x})}\right] - \log 2$
- $\mathbb{E}_{F(\mathbf{x})\sim p_T}\left[\log \frac{p_T(\mathbf{x})}{p_S(\mathbf{x})+p_T(\mathbf{x})}\right] = \mathbb{E}_{F(\mathbf{x})\sim p_T}\left[\log \frac{p_T(\mathbf{x})}{2p_S(\mathbf{x})+p_T(\mathbf{x})}\right] - \log 2$

Please note that the first terms on the right hand side are Kullback-Leibler divergences from p_S and p_T respectively to the mixture $\frac{p_S+p_T}{2}$. Their sum is the definition of the Jensen-Shannon, so it possible to redefine $C\left(F\right)$ as:

$$C\left(F\right) = D^2_{JS}\left(p_S \| p_T\right) - \log 4 \tag{3.22}$$

It can be easily proven that the Jenson-Shannon divergence is always non-negative and is zero if and only if the distributions are equal. Therefore, if $p_S = p_S$, the global minimum of $C\left(F\right)$ is obtained and it is $-\log 4$. □

This theorem is of extraordinary importance for the adversarial framework, as it provides a solid theoretical foundations to this idea.

The GRL approach has another interesting property that relates its training to the symmetric difference hypothesis divergence Definition 3.3. Suppose that $\mathcal{H}_y$ and $\mathcal{H}_d$ are the hypothesis sets of the label and domain classifiers respectively and that X_d includes the symmetric difference hypothesis set of $\mathcal{H}_y$, i.e.:

$$\mathcal{H}_d \supseteq \mathcal{H}_y \Delta \mathcal{H}_y = \{h | h = h_1 \oplus h_2, h_1, h_2 \in \mathcal{H}_y\} \tag{3.23}$$

This set can be used to rewrite Equation (3.1) as:

$$d_{\mathcal{H}\Delta\mathcal{H}}(S,T) = 2 \sup_{h \in \mathcal{H}\Delta\mathcal{H}} | \, \mathbb{E}_{x \sim D_S}[h(\mathbf{x})] - \mathbb{E}_{x \sim D_T}[h(\mathbf{x})] \, | \tag{3.24}$$

It will now be shown that, by exploiting this inclusion, the training of the GRL model actually relates to the minimization of the $\mathcal{H}\Delta\mathcal{H}$-divergence and thus of the upper bound on the generalization error:

$$\begin{aligned} d_{\mathcal{H}\Delta\mathcal{H}}(S,T) &= 2 \sup_{h \in \mathcal{H}\Delta\mathcal{H}} | \, \mathbb{E}_{x \sim D_S}[h(\mathbf{x})] - \mathbb{E}_{x \sim D_T}[h(\mathbf{x})] \, | \\ &\le 2 \sup_{h \in \mathcal{H}_d} | \, \mathbb{E}_{x \sim D_S}[h(\mathbf{x})] - \mathbb{E}_{x \sim D_T}[h(\mathbf{x})] \, | \\ &= 2 \sup_{h \in \mathcal{H}_d} | \, \mathbb{P}_{x \sim D_S}[h(\mathbf{x}) = 1] - \mathbb{P}_{x \sim D_T}[h(\mathbf{x}) = 1] \, | \\ &= 2 \sup_{h \in \mathcal{H}_d} |1 - \mathbb{P}_{x \sim D_S}[h(\mathbf{x}) = 0] - \mathbb{P}_{x \sim D_T}[h(\mathbf{x}) = 1] \, | \\ &= 2 \sup_{h \in \mathcal{H}_d} |1 - \alpha(h) \, | \end{aligned} \tag{3.25}$$

The optimal domain classifiers maximizes the value of $\alpha(h)$ and creates an upper bound on the divergence. During the training, however, the reversed gradient modifies the representation space created by the feature extractor in a way that reduces the effectiveness of the optimal domain classifier and thus the bound on generalization error presented in Equation (3.3).

Chapter 4

First Scenario: Dealing with Domain Shift in Large Scale Visual Place Recognition

As discussed in Section 2.1, most current state of the art methods requires a huge amount of data to be trained properly. The best source to procure the necessary images is Google Street View, which not only offers a large collection of panoramic images from all over the world, but also allows to access photos from old Street View collections. This is incredibly useful not only because it increases the amount of available images of the same places, but using the outdated one may be helpful to build a model that is robust to long term changes in urban landscapes. This source provides two other extremely useful advantages:

- All images have GPS coordinates available which can be used as labels;
- Even when ignoring images from old collections, most places typically appear in multiple images. This allows the construction of a dense dataset, which is a desirable quality in Visual Place Recognition.

Later some example of Street View datasets will be provided.
From the VPR point of view, the main issue with Google Street view is the fact that all images are took in the same exact manner, from the same type of viewpoint and from the same type of cameras. Moreover, they are all taken during the daytime and do not generally contain significant amount of obstructions. This is not the case in most of the queries that the model will see at test time, the quality of which is less consistent by nature.

In this chapter the work regarding the first experimental scenario is presented. Firstly, the problem will be further introduced, highlighting the main challenges and difficulties. Secondly the discussion will focus on the dataset used and their most relevant characteristic. Then the proposed approach is presented and compared to the ones in literature. Finally the numerical performances will be shown and analyzed, focusing on the optimal values of the hyperparameters and discussing the insights that emerged from the results.

4.1 The Scenario

In the first experimental scenario the goal is to determine whether or not it is possible to overcome the domain shift in the deployment of a large scale Visual Place Recognition model by integrating a Domain Adaptation technique in its training. This is done through two different setting, each one set up to examine a different kind of domain shift:

1. In the first one the focus will be on the shift of viewpoint. As already mentioned, the training data is taken from Google Street View, where the viewpoint is generally the same. If we test the model on photos taken by non professional with commercial cameras, this is not the case anymore. An example is represented in Figure 4.1;

2. The second experiment adds another type of gap by not only considering the shift in viewpoint, but also switching to a large city of another country during test. This adds a significant challenge, as the environment and general aspect of building is not coherent with respect to the training. An example is represented in Figure 4.2.

In order to build a fair evaluation environment in both settings while simultaneously keeping the computation burden low, it is important to choose the proper datasets that reflect the aforementioned issues and a large scale model that works well with limited resources. The former choice is discussed in Section 4.2 and the latter in Section 4.3.

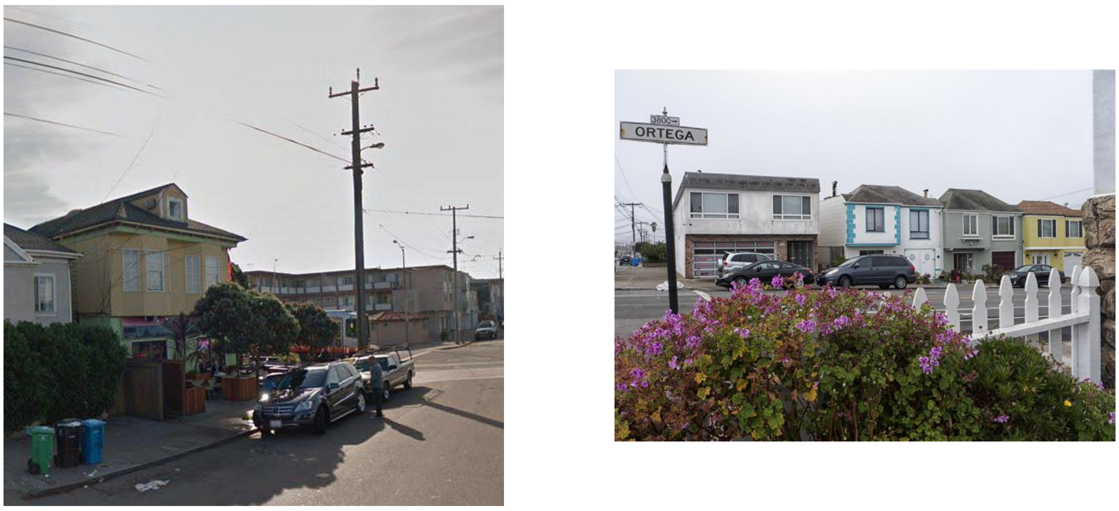


Figure 4.1: The picture on the left is a database image, taken from Google Street View. The one on the left is a user taken photo. It is clear to see that the viewpoint on the left could not be replicated by any image from Google, as it is taken from inside a private property, thus creating a significant shift. Both are from San Francisco XL dataset Berton et al. [2022a]

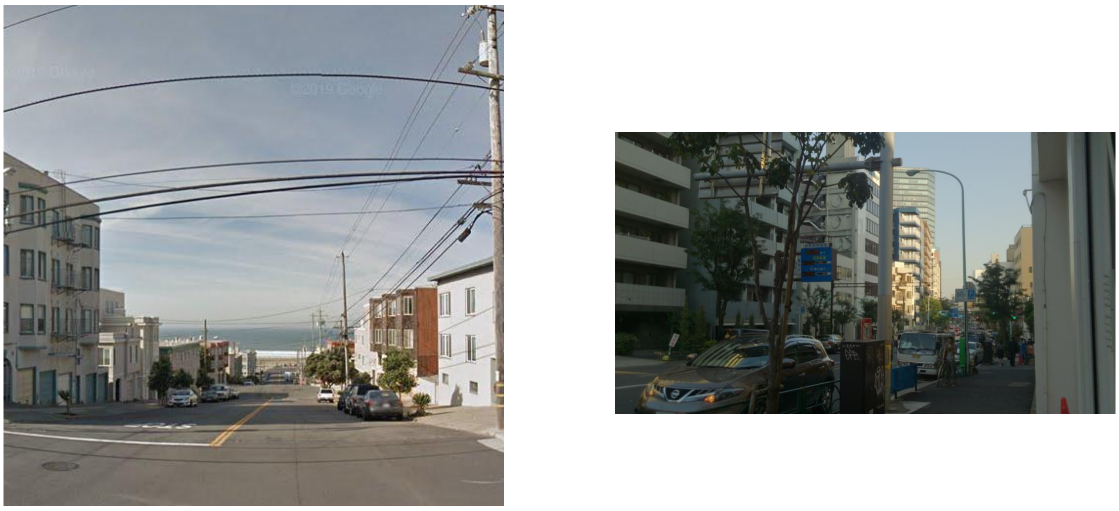


Figure 4.2: The picture on the left is a database image from San Francisco XL Berton et al. [2022a], while the one on the right is a query from Tokyo 24/7 Torii et al. [2015]. There is a significant shift both in viewpoint, as the latter one is taken from a pavement, and in scene, as the city appearance is different.

4.2 The Datasets

Choosing the proper datasets is a crucial step in building a setting that reflects the desired properties. Over the years the number of datasets for Visual Place Recognition has increased significantly. However, not all of them fits the requirements. For example, *Pitts250K* is a popular dataset introduced in Torii et al. [2013] that has been consistently used over the years for both training and testing (Arandjelović et al. [2016], Berton et al. [2022a], Keetha et al. [2023]). However, its the query set is generated from the same source of the database (Google Street View) and therefore does not contain significant shifts in viewpoint. For this reason it will not be considered for both settings. Many other datasets share this limitation, including MSLS (Warburg et al. [2020]), SVOX (Moreno Berton et al. [2021]) and St. Lucia (Warren et al. [2010]), as they are made only of frontal view taken from moving cars and do not contain the desired kind of viewpoint change.
The chosen datasets are:

1. **San Francisco XL** (Berton et al. [2022a]): a large scale dateset that covers the whole city of San Francisco with images taken from Google Street View between 2009 and 2021. To be more specific, the authors collected 3.41 millions 360° panoramas and extracted 12 horizontal crops for each. The total number of images available for training amount to 41.2 millions, all paired with GPS and heading labels. Since extracting the features for all of them would take too much time on a single GPU, the test split only contains 2.8 millions images of the same year but still covers the whole city. As for the queries, they are 1000 hand-picked photos from Flickr whose

GPS coordinates have been manually verified. These have been chosen to cover a wide range of viewpoints and some illumination changes.

2. **Tokyo 24/7** (Torii et al. [2015]): a smaller dataset composed of Street View images of a Tokyo neighborhood (Shibuya) that covers around 1.6 km^2. The total number of database images is around 76000, considerably smaller than the one for SF-XL. The queries are 315 and do not come from the web, but rather were taken by the authors during the creation of the dataset using commercial smartphones. Additional 810 queries are available but are not generally used for evaluation purposes.

In particular, San Francisco XL training images will be used in both settings as the *source* distribution. Its test split will be used as the *target* distribution in the first one, as the domain gap between these two is mainly due to the shift in viewpoint of the images. The test split of Tokyo 24/7 will be instead used as the target distribution in the second setting, as this effectively simulates the scenario in which a model is trained in one city and deployed in a somewhat different one. A clear visualization of both test sets can be found in Figure 4.3.

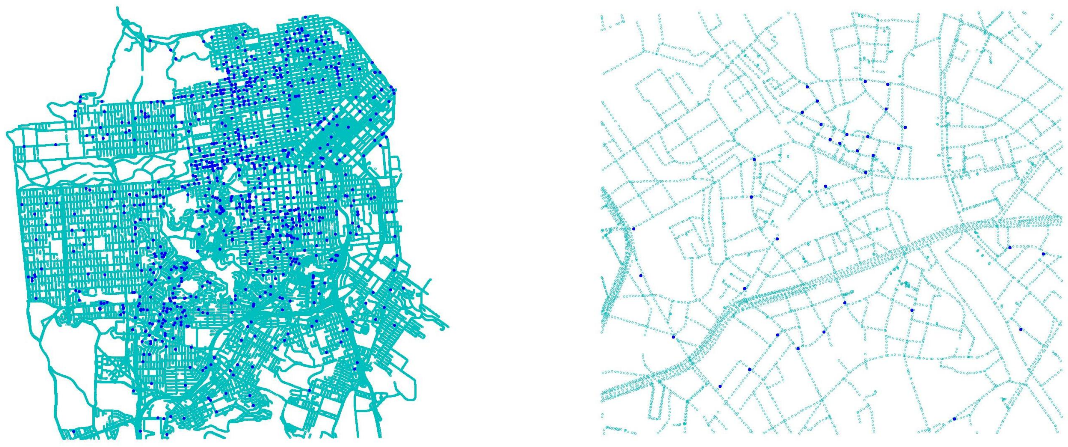


Figure 4.3: Blue points are queries, cyan points are database images.
The picture on the left is visual representation of the test split of San Francisco, while on the right there is the Tokyo 24/7 counterpart. As one can see, the density of SF-XL is very high and it covers the whole city, while Tokyo 24/7 is much smaller and sparser. Please note that the representation of Tokyo 24/7 may be misleading, as each blue dot actually represents 9 distinct queries, as for each of the 125 position 9 images were taken (the combinations of three different viewing directions and three times of the day).

4.3 The Model

Now that the settings and the dataset have been presented, it is necessary to pick a state-of-the-art Visual Place Recognition approach and a Domain Adaptation technique to integrate. Out of the many possibilities, Cosplace (Berton et al. [2022a]) was selected as the most suitable candidate. Its architecture is composed of:

- A convolutional backbone that acts as a feature extractor;
- An aggregation block composed of, among others, a GeM Pooling layer Radenović et al. [2018] and a fully connected layer;
- Multiple classifiers paired with a Large Margin Cosine Loss (LMCL) (Wang et al. [2018]), also known as *CosFace*, which is generally used for face recognition.

At test time the classifiers are discarded and the remaining parts are used to extract features and perform image retrieval.

The training procedure for Cosplace first divides the area spanned by the dataset into cells of equal area in a grid-like fashion, then further splits each of these into classes according to the orientation of the images inside them. To be clearer, each cell is $M \times M$ m^2 and there are $\frac{360}{\alpha}$ classes within each cell, all of which cover $\alpha°$ of orientation (e.g. $0°$-$\alpha°$). An image is assigned to a class if it belongs in the same cell and its orientation falls into the class range. This process generally done though UTM coordinates and heading information.
At each epoch, only a subset of non-adjacent classes is considered, a so-called *CosPlace Group*. Each group is matched with a unique classifier. The classes in a group have to satisfy two requirements:

- If two classes are in the same cell, then there must be exactly $(L - 1) * \alpha$ degrees of orientation between their ranges;
- If they belong in different cells, then there must be exactly a distance of $(N - 1) * M$ meters between them.

Moreover, a class can only belong in a single group. Please note that α, M, N and L are all tunable hyperparameters. This is done in order to avoid the possibility of ambiguous classification labels, as the grid partition is done arbitrarily and could therefore put images of the same place in adjacent classes. A more comprehensive explanation can be found in the original paper.

As for domain adaptation, the Gradient Reversal Layer technique was chosen. Please refer to Section 3.2 for an in-depth presentation and discussion of this approach and its theoretical foundations.

These two choices were made because these models and techniques have proven to be extremely effective in their respective tasks, but also because they seamlessly fit together.

The CosFace classifier can be interpreted as the label classifier in the GRL setting, while a domain classifier can be simply added as a detaching branch before or after the aggregation layer. During training the labeled training images are used as data from the source domain, while the unlabeled queries from the test splits are used as data from the target domain. By doing this, the backbone learns simultaneously to correctly assign each source image to its class and to generate features that are invariant with respect to the domain gap between source and target distributions.

4.4 The Numerical Results

it's Although there are some differences between the first and second setting, the general configuration was mostly the same and will now be fully described:

- *the CosPlace hyperparameter*: CosPlace hyperparameters were kept at the same values specified in Berton et al. [2022a]. This includes the ones related to the creation of the classes, the ones related to the data augmentation techniques applied and the learning rates of the backbone and the classifiers. A full rundown of the values can be found in Table 4.1

- *The hardware*: all experiments were run in the GPU cluster of the VANDAL group, composed of 16 NVIDIA GeForce GTX 1080 GPUs. A small amount of preliminary tests were run on a different machine equipped with a NVIDIA GeForce GTX 1070 and a TITAN RTX;

- *the software*: All experiments were run using Python 3.10, PyTorch 2.0.1 and torchvision 0.15.2.

Hyperparameter	Role	Value
M	The lenght of the edges of all cells	10
α	The range of orientations that a singel class covers	30
N	see Section 4.3	5
L	see Section 4.3	2
groups_num	Number of Cosplace groups created	8
brightness	-	0.7
contrast	Values passed to Pytorch's	0.7
hue	ColorJittering function	0.5
saturation	-	0.7
random_resized_crop	The minimum possible area for the random crop	0.5
lr	the learning rate of the backbone	10^{-5}
classifiers_lr	the learning rate of all classifiers	10^{-2}
batch_size	Batch size of images from the *source* domain	32

Table 4.1: The values of the parameters that have been unchanged during all experiments presented in this section. The horizontal lines divide them into semantic groups

As for the hyperparameters related to the Gradient Reversal Layer architecture, the experiments focused on four different ones:

- *Learning Rate* μ: the learning rate of the domain classifier;
- *Weight of the domain classifier loss* λ: the domain classifier is paired with a cross entropy loss. Thus, the total loss is:

$$Total\_Loss = CosFace\_Loss + \lambda * CrossEntropy\_Loss \tag{4.1}$$

  where λ represents the weight assigned to the domain classifier loss. This is arguably the most important hyperparameter as it needs to balance the two learning objective of the adversarial framework;

- *GRL backpropagation (exponential) coefficient* γ: in Section 3.2 it was mentioned that the GRL layer changes the sign of the gradients during backpropagation. In practice, it actually multiplies them by a negative coefficient during backpropagation. This value changes during the training, following the formula presented in Ganin and Lempitsky [2015]:

$$\text{GRL\_coeff} = \frac{2}{1 + \exp(-\gamma p)} - 1 \tag{4.2}$$

  where p represents the training progress linearly changing from 0 to 1. Instead, γ represents the speed at which the coefficient converges to 1. This is done to ignore the noisy gradient updates that come from the first few epochs, when the domain classifier is not yet well trained. The faster the convergence, the less epochs are ignored;

- *Target Domain Batch Size*: The number of images from the target domain for each batch fed to the domain classifier. To be more specific, this architecture requires to create two separate batches. The first one is composed of 32 images from the source domain, as illustrated in Table 4.1, and is fed to the backbone and all classifiers. The second one is composed of images from the target domain and is fed to the backbone and the domain classifier, as no labels are available. During preliminary tests it was found that a target batch size of 8 was a solid trade-off between performances and additional computational time. To achieve this the source batch was modified so that all 32 images were fed to the backbone and the CosPlace classifiers, but only 8 were fed to the domain classifier. By doing this no class re-weighting was necessary.

4.4.1 The First Setting

In order to determine the best combination of GRL-related hyperparameters values, three candidates were selected for each of them. Then a grid search was performed by training a model for each configuration. The candidates were:

- $\mu \in \{10^{-4}, 10^{-3}, 10^{-2}\}$.
  In the original paper that introduced the GRL technique there is not a specific discussion regarding the learning rate of the domain classifier, so the values picked span a wide range;

- $\lambda \in \{10^{-1}, 1, 10\}$.
  As for the previous hyperparameter, no indication were given in the paper, so the values are chosen to cover a wide range. In particular, this choices would be able to prove whether or not the secondary loss should be enhanced or reduced with respect to the default value. This is useful as it may indicate the direction that a future, denser grid search may follow regarding λ;

- $\gamma \in \{7, 10, 15\}$.
  The proposed value is 10, but the authors mentioned that no further tests were made to find the optimal value. Similarly to λ, the choices are made to prove whether or not to increase or decrease the number of initial epochs to consider as noisy. In Figure 4.4 it is possible to visualize how these values actually affect the coefficient of the GRL layer.

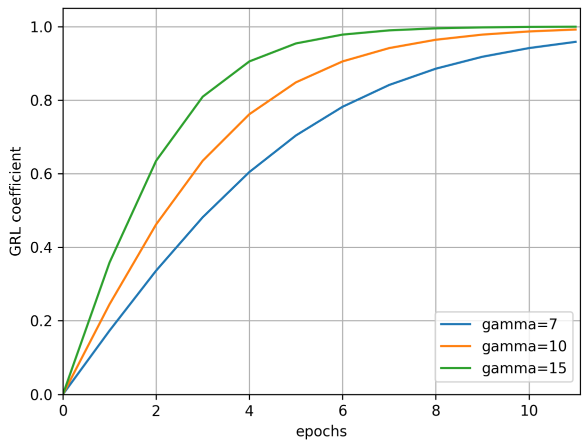


Figure 4.4: Visual representation of how different values of γ affect the GRL coefficient. For example the 0.9 threshold is reached after 9, 6 and 4 epochs for γ = 7, 10 and 15 respectively.

Now that the hyperparameters choices have been presented, the focus will now shift on the architectural and training choices. Since the goal of these experiments is not to find the best performing model, but rather to investigate whether a DA technique can help a VPR model, the choices were made to lighten the computational burden as much as possible while keeping the results coherent and insightful.

- The backbone architecture is the ResNet18 He et al. [2015], which is the smallest of the family, and the weights were loaded from the checkpoints available on PyTorch Hub. This allows for a significant reduction in the number of epochs of training;

- The descriptors dimension, which is the dimension of the output vector of the aggregation block, is set to 512. This value was chosen as good trade-off between increased performance and additional memory usage and computations;

- The iterations per epoch and number of epochs and is set to 10000 and 20 respectively. The former is the default value of CosPlace, while the latter was determined through the preliminary experiments as a good threshold;
- Since using the full San Francisco XL test split would require a significant amount of time, and since it is fair to expect that a good part of the experiments will result in failure, the San Francisco XS test split will be used to evaluate each model. This is a subset of San Francisco XL with the same queries but a limited database, so that the number of features to extract is lower. All the experiments that were conducted indicate that the performances on this subset are fine indicators for the performances on the XL counterpart, but this choice allows for faster discarding of the failing configurations. The successful models were then also tested on the full version.

In table Table 4.2 the performances of the configurations generated through the grid search are shown. All results are expressed in terms of Recall@1 using the standard threshold distance of 25m. Please note that the baseline used for these experiments is the PyTorch Hub CosPlace model (with the same backbone and descriptors dimension), but a configuration that beats the baseline is not automatically considered successful. This is due to the general trend that this training procedure follows: because of the definition of the of the GRL coefficient (Equation (4.2)), during the first epoch its value is always zero, so no adversarial training actually happens. However the model is able to adjust its batch normalization statistics to better fit the target images, thus already bringing an improvement to the performances. In the test set however, this improvement is minimal. When the coefficient becomes non null, there is generally a dip in performances that lasts for some epochs. The model is considered successful if it's able to "recover" from this dip and perform better than in the first epoch, i.e. better than 69%[1].
The first insight that can be extracted is that the exponential coefficient γ does not clearly impact the performances. Secondly, the best configurations of the other parameters are $(\mu = 10^{-3}, \lambda = 10^{-1})$ and especially $(\mu = 10^{-2}, \lambda = 10)$. In order to see how this model generalizes, those successful configuration have been tested on SF-XL test split. Those results are presented in Table 4.3. All the chosen model outperforms the baseline.

4.4.2 The Second Setting

The procedure for the second is partially similar to the first one, however some newly-found insight are exploited to focus the hyperparameter search on range of values that have been seen to work well. Out the the two configuration that worked the best on San Francisco XL, $(\mu = 10^{-2}, \lambda = 10)$ is chosen as the fixed one for all experiments, while the search range for the optimal γ is expanded to the range $[5, 16]$. All architectural and training configurations were unchanged from the first setting with the obvious exception of the test set, which is the Tokyo24/7 test split. The validation set is the Tokyo-XS test

[1]All experiments were conducted using the same randomness seed. Thus the R@1 reached in the first epoch is always 69%.

Learning Rate μ	Weight λ	Coefficient γ		
		7	10	15
10^{-4}	10^{-1}	69.6	69.6	69.2
10^{-4}	1	69	69	69
10^{-4}	10	69	69	69
10^{-3}	10^{-1}	**70.6**	**69.9**	69.2
10^{-3}	1	69	69	69
10^{-3}	10	69	69	69
10^{-2}	10^{-1}	69	69	69
10^{-2}	1	69	69	69
10^{-2}	10	**70.3**	**69.7**	**70.5**
baseline		67.7		

Table 4.2: Results on San Francisco XS of Cosplace trained for 20 epochs on SF-XL using GRL for domain adaptation on the queries of the test split of SF-XL. For compactness, the values of γ have been moved along the columns. The highlighted value represents the configurations that have been considered successful.

Learning Rate μ	Weight λ	Coefficient γ	R@1	R@5
10^{-3}	10^{-1}	7	68.5	77
10^{-3}	10^{-1}	10	67.3	75.6
10^{-2}	10	7	69.2	76,8
10^{-2}	10	10	66.7	75.1
10^{-2}	10	15	67.8	76.2
baseline			65.5	74.1

Table 4.3: Results on San Francisco XL of the successful models highlighted in Table 4.2. The highlighted value represents the best configuration.

split.

The results are shown in Table 4.4. The baseline, as in the previous setting, is the PyTorch Hub CosPlace model with the same backbone and descriptors dimension. The best performing values for γ seems to be 10, 11 and 12, but it appears that the model is pretty robust with respect to that hyperparameter.

Additional test have been carried out to answer the following questions:

1. Does increasing the number of epochs significantly increase performances?
2. Does having additional queries (exclusively for training) lead to better generalization and thus better performances?

The first answer was found by extending the training of three of the best performing model (identified by $\gamma = [10, 12]$) for another 20 epochs. Out of the three, only one showed

Learning Rate μ	Weight λ	Coefficient γ	R@1	R@5
10^{-2}	10	5	80.3	90.5
		6	84.8	92.1
		7	83.2	91.7
		8	83.2	93.7
		9	82.9	92.1
		10	**85.4**	**94**
		11	**85.7**	**93**
		12	**86**	**94.3**
		13	83.8	93.7
		14	84.8	93.3
		15	83.5	94
		16	**85.4**	**93.7**
baseline			81.9	90.8

Table 4.4: Results on Tokyo 24/7 of Cosplace trained for 20 epochs on SF-XL using GRL for domain adaptation on the queries of the test split of Tokyo 24/7. The best performing configurations are highlighted.

light improvements, thus it is possible to conclude that 20 epochs are enough to reach a plateau in performances and no further training is required. Full results are showed in Table 4.5.
The second answer was instead found by adding 810 extra queries to the training process. These are available with the Tokyo 24/7 dataset but are not generally used for evaluation. Full results are shown in Table 4.6. It is possible to see that, on average, a configuration gains a few % points. This indicates that an increased number of queries can be useful to improve the performances of the models trained used the GRL technique. Further tests were carried out to verify that adding training epochs does not significantly impact the results: although most configurations' recalls increase, the average improvement is below 0.5%. This cannot justify the additional time and computations required by the additional epochs.

Learning Rate μ	Weight λ	Coefficient γ	R@1	R@5
10^{-2}	10	10	**86**	**94**
		11	85.7	93
		12	86	94.3
baseline			81.9	90.8

Table 4.5: Results on Tokyo 24/7 of three top performing Cosplace model from Table 4.4 trained for another 20 epochs on SF-XL using GRL for domain adaptation on the queries of the test split of Tokyo 24/7. The only configuration that improved is highlighted.

This chapter is concluded with a brief analysis of the training of the domain classifier.

Learning Rate μ	Weight λ	Coefficient γ	R@1	R@5
10^{-2}	10	5	85.4	93.7
		6	85.1	93.7
		7	83.5	94
		8	84.1	92.1
		9	82.5	92.1
		10	**86.7**	**95.9**
		11	83.8	94.6
		12	83.5	92.4
		13	**85.4**	**94.9**
		14	**85.1**	**93.3**
		15	**85.1**	**94.9**
		16	84.8	93.7
baseline			81.9	90.8

Table 4.6: Results on Tokyo 24/7 of Cosplace trained for 20 epochs on SF-XL using GRL for domain adaptation on **all** the 1125 queries of the test split of Tokyo 24/7. The best performing configurations are highlighted.

Since the goal for the backbone is to effectively fool it by extracting domain invariant features, it is interesting to visualize its accuracy during the training and compare it to the final performance of the model. Generally speaking, during a successful training the accuracy of the domain classifier is expected to oscillate around the 50% mark, as this represents the fact the backbone is completely fooling it. This happens in Figure 4.5, where after some noisy first epochs the accuracy becomes considerably stable around 50%. In most of the unsuccessful trainings, such as in Figure 4.6, the reason for the failure is that the updates to the weights that come from the domain loss are not significant enough for the backbone to trick the domain classifier, thus resulting in poor performances. This issue is generally found in configurations where either μ or λ are too low, as the general architecture ends up being not balanced at all.

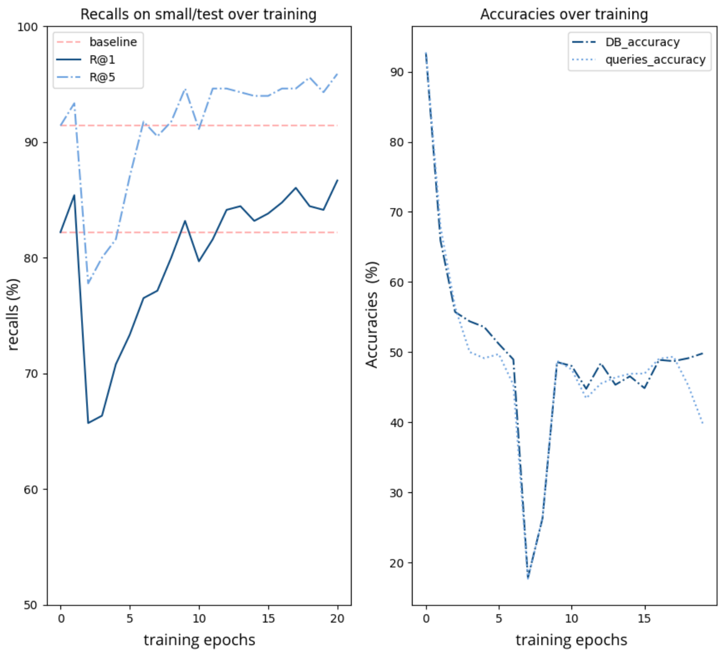


Figure 4.5: Details of the successful training procedure of the configuration with $\gamma = 12$ in Table 4.6

.

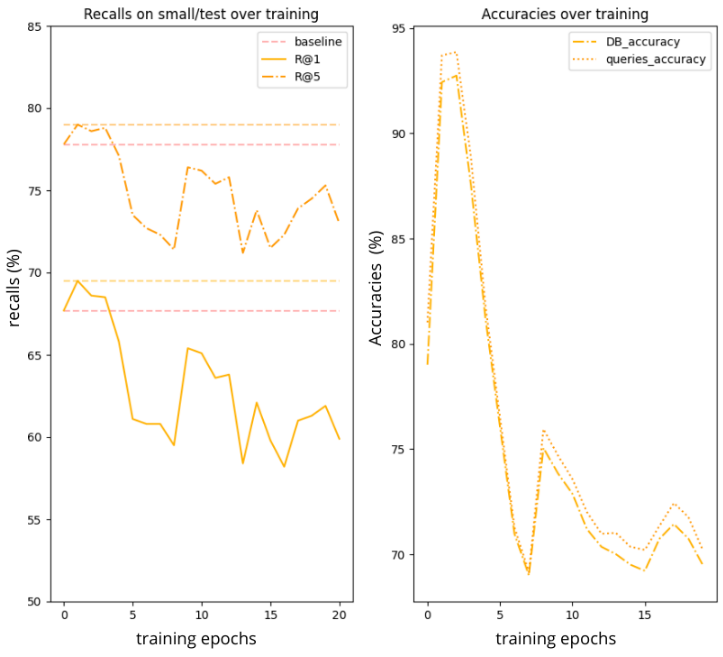


Figure 4.6: Details of the unsuccessful training procedure.

Chapter 5

Second Scenario: Fine tuning of Visual Place Recognition Models for Indoor Retrieval

As discussed in Section 2.2, image retrieval is a key step in the global relocalization pipeline as it allows to select a restricted number of candidates to compute 2D-2D correspondences between them and the query. The optimal Visual Place Recognition model should be able to find at least one appropriate candidate within the first k predictions, with k being as small as possible. This would significantly increase the speed of the pose estimation algorithm, as in most cases the matching step is the bottleneck of the whole procedure. Additionally, the restriction to the k candidates has another advantage: since the matching relies on local descriptors, it is exposed to the risk of noisy matches that may come from similar images taken in different places. This is especially true in indoor environment, where repetitive and non-informative structures are frequent (see Figure 2.7). By limiting the matching to the candidates, we reduce the probability to find noisy matches.

As of the writing of this thesis, Image Retrieval in indoor environment is not a popular topic in deep learning literature. Out of the state-of-the-art methods that are going to be used in this scenario, the majority were designed and trained to work in urban environment (Arandjelović et al. [2016], Revaud et al. [2019a], Ge et al. [2020], Berton et al. [2022a], Ali-bey et al. [2022], Berton et al. [2023] Ali-bey et al. [2023] and Izquierdo and Civera [2023]), while only Keetha et al. [2023] attempts to build a model that works well in a general setting. However there is a benchmark, presented in Humenberger et al. [2022], that offers an in-depth analysis of the role of Image Retrieval in the larger context of pose estimation, with a particular attention to indoor environments. The main limitation of this work is that only a small amount of VPR approaches are tested, namely NetVlad, DenseVlad (Torii et al. [2015]), AP-GeM and DELG (Cao et al. [2020]). Moreover, the methods are used without any attempt at fine-tuning them to the specific domain.

In this chapter the work regarding the second scenario is presented. Firstly, a thorough introduction will further present the problem and provide a complete rundown of the datasets and methods that will be tested. Secondly, a wide benchmark on indoor environment will be described and its result presented. From this, some of the most effective models are selected to be fine-tuned. The performances achieved by them will offer great insight into what makes a VPR work effectively in the indoor domain.

5.1 The Datasets

The main issue regarding indoor Visual Place Recognition is the lack of large scale datasets. For the very nature of the environment, it is daunting task to build a dataset that contains a large number of images and, at the same time, covers a large and varied area. Since there are no easily accessible sources such as Google Street View, all images have to be manually taken. This is usually done with some sort of mobile vehicle that can hold multiple cameras and instruments, often of different natures. Indeed, since the underlying task is generally pose estimation rather than simple retrieval, it is common to use laser scanner to build 3D models. This requires significant additional work and thus reduce the possible extension of the datasets.
For these reasons, many existing datasets were not considered for this work, as they present some relevant limitations. For example, *7-scenes* (Glocker et al. [2013]) contains more than 40k images, but it covers the area of only 7 small rooms. A similar issue is present in *17 places* (Sahdev and Tsotsos [2016]. In Taira et al. [2018] the InLoc dataset is introduced and its size, both in terms of number of images and area covered, should warrant effectiveness for the purpose of this work. However, its sparsity makes it so that VPR performs poorly on it and cannot really be used under strict evaluation thresholds.

The chosen datasets for this scenario are:

- Gangnam Station B1-B2 (Lee et al. [2021]): this dataset was created in one of the busiest metro station in Seoul, which was *not* closed while the images were taken. Thus it contains a large amount of human obstruction in most images, and was created with the idea of a robustness benchmark for VPR models. Each section covers an entire floor of the station: B1 contains scenes of a mall and some turnstiles, while B2 contains images of the metro station platform. Some image samples are shown in Figure 5.1;

- Hyundai Department Store B1-1F-4F (Lee et al. [2021]): this dataset was instead created in a large South Korea department store, so there is large presence of commercial activities. Once again, each section covers an entire floor: B1 contains many cafes, restaurants and supermarkets, and its images are captured under low-light conditions, while 1F and 4F both contains mainly fashion-related shops. In particular, the fourth floor was still partially under construction when the images were taken, so a lot of temporary and texture-less walls are included. Some image samples are shown in Figure 5.2;

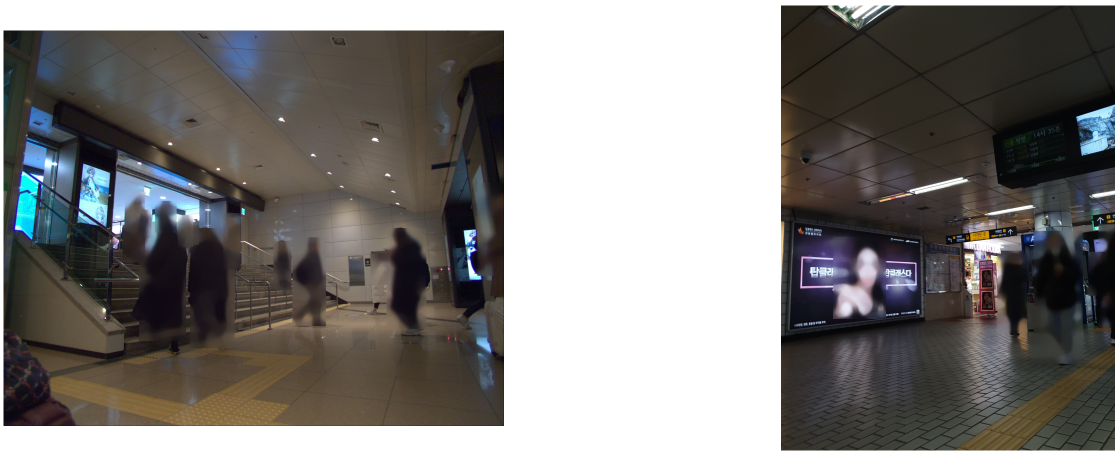


Figure 5.1: An example of a database image, on the left, and a query, on the right, from Gangnam Station B1.

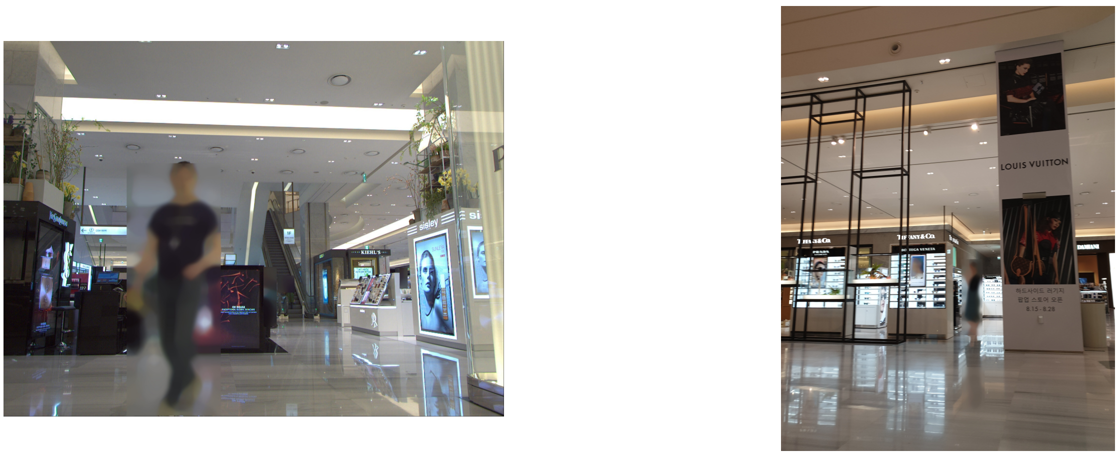


Figure 5.2: An example of a database image, on the left, and a query, on the right, from Hyundai Department Store 1F.

- Baidu Mall (Sun et al. [2017]): This dataset, similarly to Hyundai Department Store, was created in the ground floor a large mall in China. Even though one may expect them to be similar, they are actually very different for three reasons:
  - The general environment in Baidu Mall is more open and less cluttered than in Hyundai Dept., thus the general appearance of the images are different;
  - In Baidu Mall, the queries are not taken from the same moving structure as the database images, but rather by different users using commercial phones. This is not the case in Hyundai Dept.;
  - Baidu Mall is significantly sparser, as one can see in the bottom right of Figure 5.5.

Some image samples are shown in Figure 5.3;

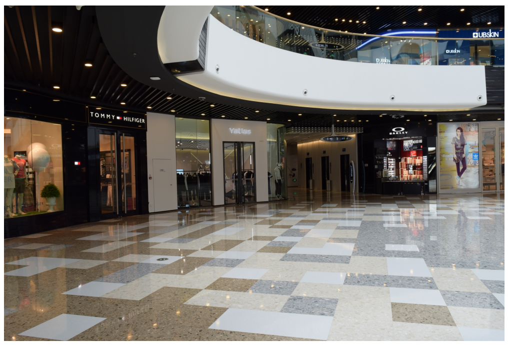
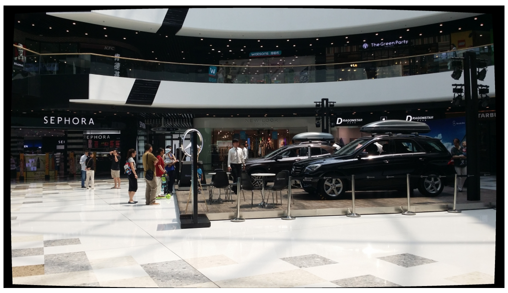
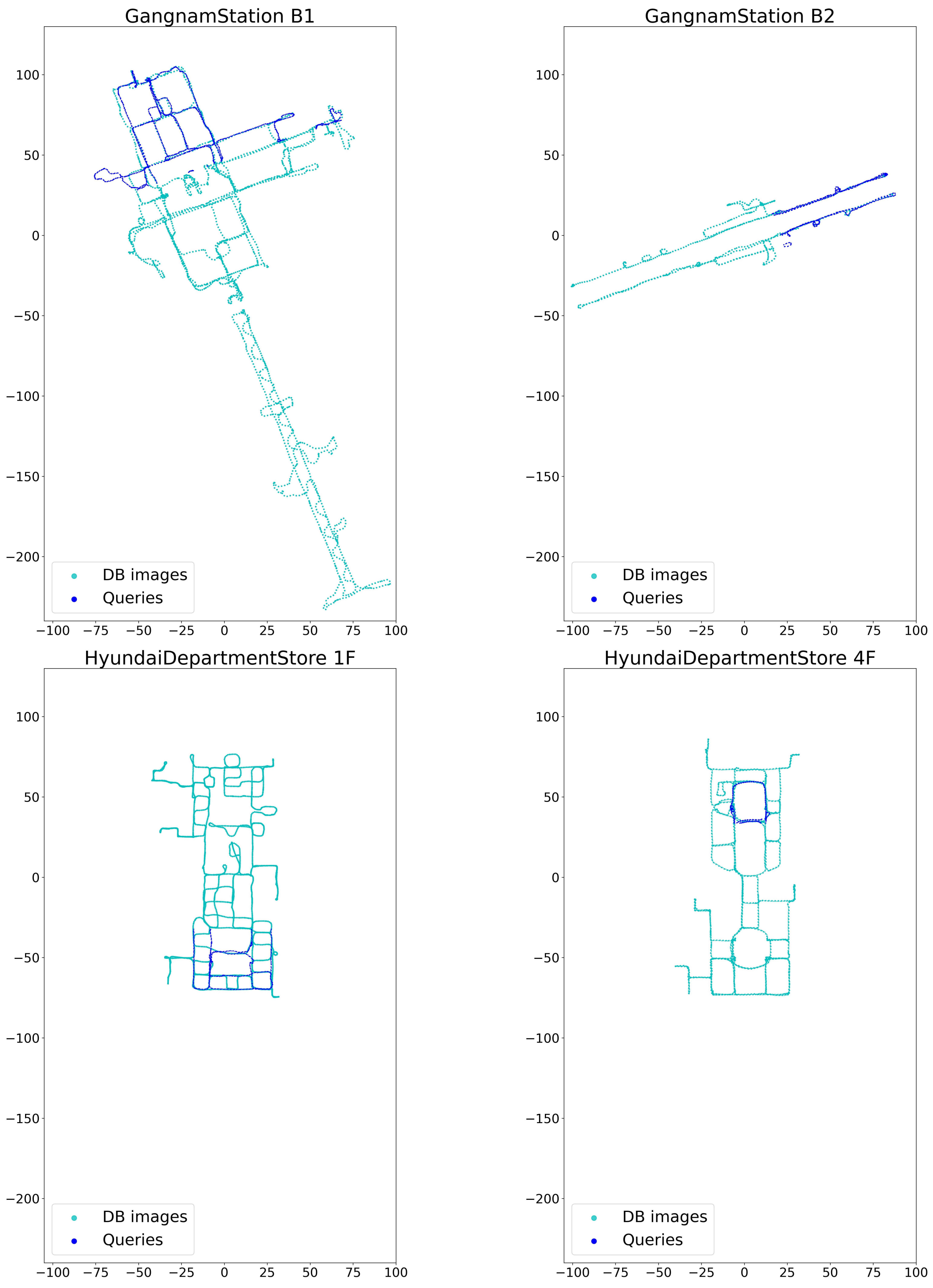



Figure 5.3: An example of a database image, on the left, and a query, on the right, from Baidu Mall.

One noteworthy limitation of both the Gangnam and Hyundai datasets is that the label of the test set are not available and the evaluation on those splits can only be done through a specific website and with respect to extremely strict thresholds. Thus, the test queries cannot be considered for this work. The labels of the validation set are indeed available, so those will be used instead.s Additional information about the chosen datasets can be found in Table 5.1, while it is possible to visualize them in Figure 5.5.

Dataset Name	Scene	N° of DB Image	N° of queries (val)	Area (m^2)
Gangnam Station	B1	16,536	2,620	20,900
	B2	4,518	916	5,250
Hyundai Dept. Store	B1	20,579	1,751	8,500
	1F	16,222	734	10,000
	4F	7,482	584	8,350
Baidu Mall	-	689	2,292	9,200

Table 5.1: A recap of the most relevant properties of the chosen datasets.


GangnamStation B1
GangnamStation B2
HyundaiDepartmentStore 1F
HyundaiDepartmentStore 4F
DB images
Queries
100
50
0
−50
−100
−150
−200
−100
−75
−50
−25
0
25
50
75
100

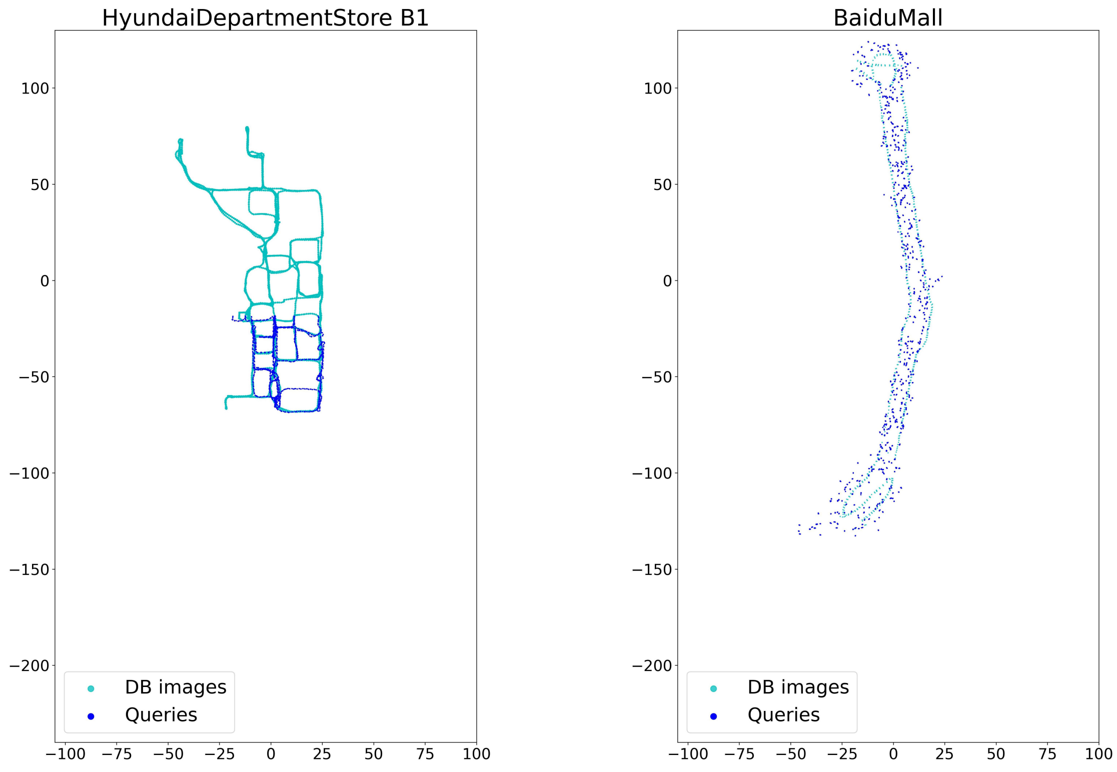


Figure 5.5: A visual representation of the chosen indoor datasets. Since each one has its own reference system, the position of all images have been normalized, thus each image is in scale. As one may expect by looking at Table 5.1, the Gangnam datasets cover a wider area than the others and Baidu is the most sparse. The blue points only partially cover the area because the labels of the test split queries are not available.

5.2 The Models

As said in the introduction to this chapter, the only benchmark for indoor Visual Place Recognition, presented in Humenberger et al. [2022], includes just four methods. The goal of the first part of the experimentation is to significantly broaden that work by including all current state-of-the-art methods. For this reason this section is devoted to their presentation and a brief discussion about what can be expected in terms of performances on the indoor datasets.

- **EigenPlaces** (Berton et al. [2023]): this method was designed with the specific intention of tackling the issue of viewpoint shift i.e. building a model that could reliably recognize a place from all points of view. In order to do so, the authors propose a clever subdivision of the training set. Firstly, the area covered is divided into 15m × 15m cells. For each of them, the UTM coordinates of the images that they contain are used to compute the first two principal components, which are in turn exploited to find two focal points: a lateral and a frontal one. The idea behind this comes from the fact that most urban environment dataset is composed of images taken from cars, thus it is likely that the first principal component of each cell will follow the direction of the road, and that the second one (which has to be perpendicular) will always face the building on the side. The classes are defined as the set of images within a cell that face a specific focal point. The training is very similar to that of CosPlace, as the Large Margin Cosine Loss is employed and at each epoch only 1 in every N cells is utilized (both in the longitudinal and latitudinal directions). A visual representation of the classes is shown in Figure 5.6;

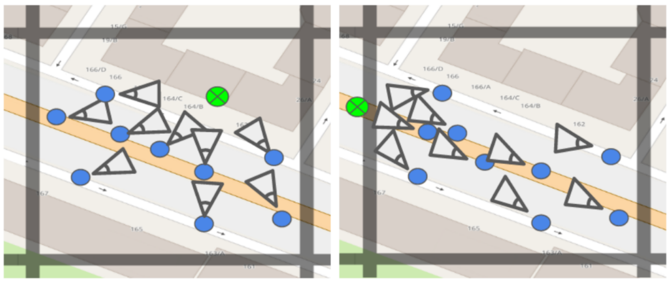


Figure 5.6: A visual representation of the classes within a cell (lateral on the left, frontal on the right). Each blue dot is an image paired with its heading. The green dots are the focal points. Image from Berton et al. [2023]

- **MixVPR** Ali-bey et al. [2023]: MixVPR exploits the ability of fully-connected layers to automatically aggregate features in a holistic way. The architecture is shown in Figure 5.7. Given a single image, the backbone extracts a feature map of dimension $c \times h \times w$, where c is the number of channels. For the purpose of this method, it can be seen as c separate activation maps of size $h \times w$, each of which get separately flattened and then re-joined together as a 2 dimensional array of size $c \times (h * w)$.

This is then fed to the Feature Mixture i.e. a sequence of isotropic MLP blocks that iteratively incorporate relationships between spatial features in each individual activation map. The output is the projected depth and row-wise to generate he final descriptor array. The computations are mainly matrix multiplications which are generally efficiently computed, especially when comparing with the self attention layers of the visual transformers;

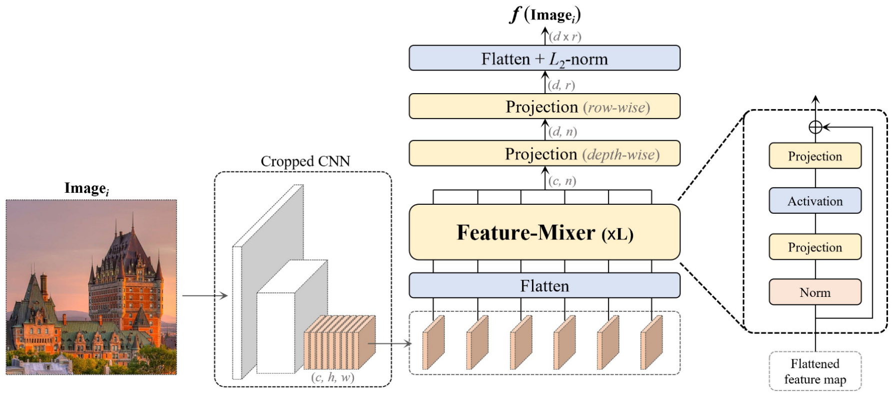


Figure 5.7: The architecture of MixVPR. Image from Ali-bey et al. [2023].

- **Anyloc** Keetha et al. [2023]: as the author said: *"Anyloc is the first concrete attempt at building a VPR model that works anywhere, anytime and across anyview"*. This approach exploits the incredible representation power of the *Dinov2* backbone (Oquab et al. [2023]) to build a model that is incredibly robust and competitive across a wide range of domains. The main drawback is that, out of the state-of-art methods, it is by far the most computationally and memory intensive. On a single GPU, the benchmark evalutations took 10× more than the ones of the second slowest method (that also employs Dinov2);
- **Salad** Izquierdo and Civera [2023]: similarly to Anyloc, Salad also utilizes DinoV2. However it builds a training pipeline that includes a few block of the backbone and proposes a revisited VLAD layer for cluster assignments and aggregation of the local features. The former is seen as an optimal transport problem where the unitary mass of the feature vector must be effectively distributed among the clusters, which includes a new so-called "dustbin", which should contain non-informative features that will eventually be dropped. In each cluster the features are added up, then the resulting vectors are concatenated with all the ones from the other clusters and simple global token. This procedure allows for the removal of VLAD priors, i.e. the starting cluster centroids, and shows excellent performances in Visual Place Recognition.
- **NetVlad** (Arandjelović et al. [2016]): this method is a staple of VPR and is based on a differentiable version of VLAD. A lengthier discussion can be found in Section 2.1;
- **SFRS** Ge et al. [2020]: this approach proposes a change in the generic loss framework. First of all, it add a secondary, self-supervised, loss function that is based on

the similarity scores between the queries and the fine-grained regions of the first k gallery images. Secondly, in the main loss the negative samples are replaced with their "worst" region. This allows the training procedure to ignore false positives and focus on actually informative database images;

- **CosPlace** (Berton et al. [2022a]): as this was the chosen approach in the first scenario, a complete description can be found in Section 4.3;
- **Conv-AP** (Ali-bey et al. [2022]): this approach simply introduces a novel aggregation layer composed of two steps. The first one is a channel-wise weighted pooling i.e. a 1×1 convolution, while the second is an adaptive average pooling;
- **AP-GeM** (Revaud et al. [2019a]): AP-GeM utilizes directly the average precision as the loss function. The reasons are explained in Section 2.1.

Since none of these methods is trained specifically for indoor Visual Place Recognition, is it hard to predict which one could perform the best. The only exceptions are Anyloc and Salad, which employ Dinov2 as a backbone and thus are consistently competitive across domains. As for the others, it will be interesting to see which training procedures, losses and aggregation would perform better.

5.3 The Benchmark

The benchmark that will be presented in this section aims at being the most comprehensive one yet. For this reason, all the available configurations for each of the state-of-the-art methods presented in the previous section have been tested across the 5 indoor datasets (Gangnam Station B1-B2 and Hyundai Department Store B1-1F-4F). Moreover, each evaluation was done using six different threshold: three solely distance-based (1m, 10m, 25m) and three that also consider angular distance (1m 5°, 10m 30°, 25m 60°). Out of the 54 configurations tested, the the top ten performing ones are shown in Tables 5.2 to 5.6. They are ranked based on the average performance across all datasets and thresholds.

From those results we can gather that, in terms of complexity, the Gangnam datasets are significantly harder benchmarks for the models than the Hyundai ones. Under any threshold of 10 meters and 30° or looser, in the latter the average performance is around 90% for 1F and 4F and around 85% for B1. This result is confirmed by the upper bounds, which are only significant for the two lowest thresholds, and are generally higher for the Hyundai datasets. This matches the expectations based on their composition. A busy metro station environment is rich of obstructions, repetitive or non informative structures and noisy elements. While those problems would also arise in a department store, their frequency is generally lower.
The threshold were arbitrarily selected by looking at the most common values in the Visual Localization literature. Given these results, it is fair to conclude that the combination of 1m and 5° is too strict for a pipeline that includes only image retrieval. Moreover, for higher positional distances the addition of an angular bound does not have an high impact

on the results.
As for the methods, the gathered data offer precious insights:

- SFRS is the best overall method: This is true particularly for the 1m threshold, where it is consistently on top of the scoreboard by a quite consistent margin. On Gangnam Station B2, which by construction contains the most amount of noise, it beats all competitors across a wide range of thresholds. From this behaviour it can be deduced that the constrastive training that SFRS models undergo is helpful for indoor deployment, especially for scenarios with strict thresholds, probably because the region subdivision allows them to more effectively ignore noise and focus on the relevant aspects of the images;

- Eigenplaces is the most robust across configurations: out of all the methods that have multiple configurations (which is all except sfrs, anyloc and salad), eigenplaces is the one with the highest consistency. Out the total 54, seven out of eleven are in the top 20 and none are in the bottom 15. This indicates that the training procedure is able to fully take advantage of the representation power of all backbones and lead to good performances. In Table 5.7 some eigenplaces configurations are compared to the top models on Gangnam B2. One interesting surprise is that the ResNet50 version outperform the larger ResNet101 one (with the same descriptors dimension) by quite a relevant margin;

- Dinov2 is effective: both salad and anyloc consistently perform the best, even though salad is better on average. One peculiar issue that is found is that both severely struggle under the 1 m threshold. In particular, in four out of five cases salad is the best choice under the 1m 5° bound, but it trails severely ($\sim$ -12%) under the 1m one. This is not simple to explain, as it is even outperformed by netvlad, on which it is based. A possible reason is that the Dinov2 backbone generates descriptors so robust that, under that strict threshold, they become ineffective. There is no doubt that under looser threshold these approaches thrive and outclass all competitors;

- MixVPR and NetVLAD are solid alternatives but not much more: both methods are constantly among the top then, but rarely are the best ones. The only instance in which that happens is in Hyundai Department Store 4F, where mixVPR outperforms all competitors by a narrow margin;

- Cosplace, AP-GeM and Conv-AP are not competitive: as one can see in Table 5.8, the best configurations of such methods are not nearly as effective as the top ones. The only small exception is AP-GeM on Gangnam Station B2, which is somewhat closer. Note that AP-GeM was the best performing method for the Visual Localization benchmark in Humenberger et al. [2022].

Finally, when weighting the performances with the computational burden of each method, its clear that the improvements brought by salad and especially anyloc over, for example, SFRS and eigenplaces are considerably expensive. However, they still represent the state of the art in a significant portion of this benchmark. For this reasons, **salad** and **eigenplaces** models are chosen as starting points for the fine-tuning on indoor environment.

Method	Backbone	Desc. Dim.	Gangnam Station B1 (Recall@1)					
			1m 5°	10m 30°	25m 60°	1m	10m	25m
sfrs	VGG16	4096	3.6	46.4	54.2	**26**	55.7	60.4
mixvpr	ResNet50	4096	3.6	50.7	56.6	24.4	56.6	60.7
eigenplaces	ResNet50	2048	4	46.6	53.4	25.5	54	60
salad	DINOv2	8448	**5.2**	**59.4**	**61.5**	21.1	62	67.9
eigenplaces	ResNet101	2048	3.8	44.3	50.8	25.4	51.8	57.9
eigenplaces	ResNet101	512	3.7	45.2	51.7	22.4	53.1	59.3
eigenplaces	ResNet50	512	3.5	41.8	48.4	23.1	50	55.8
netvlad	VGG16	4096	2.6	38	46.6	20.3	49.9	55.7
anyloc	DINOv2	49152	2.8	44.4	54.5	17.9	**63.9**	**70.9**
netvlad	VGG16	32768	2.6	38	46.9	20.2	51.4	57.2
Upper Bounds			10.4	97.1	100	81	97.2	100

Table 5.2: Benchmark results of the top ten best performing models on Gangnam Station B1, ranked based on the average performance across all datasets and thresholds.

Method	Backbone	Desc. Dim.	Gangnam Station B2 (Recall@1)					
			1m 5°	10m 30°	25m 60°	1m	10m	25m
sfrs	VGG16	4096	3.9	**50.8**	**63.8**	**40.8**	**60**	70
mixvpr	ResNet50	4096	4.3	44.5	56.9	31.2	52.6	63.1
eigenplaces	ResNet50	2048	4.5	45.4	57.2	30.8	54.3	65.5
salad	DINOv2	8448	4.6	40.2	48.4	22.2	45.5	57.6
eigenplaces	ResNet101	2048	4.3	41.7	52.5	28.3	49.8	58.1
eigenplaces	ResNet101	512	3.9	39	50.7	26	48.6	58.8
eigenplaces	ResNet50	512	3.4	46	55.1	29.5	52.8	64.2
netvlad	VGG16	4096	3.3	44.8	55.7	36.7	54.4	62.6
anyloc	DINOv2	49152	**4.7**	40.9	53.9	26.9	55.2	**70.1**
netvlad	VGG16	32768	2.7	43.7	55.3	34.6	53.7	64.5
Upper Bounds			8.8	100	100	90.9	100	100

Table 5.3: Benchmark results of the top ten best performing models on Gangnam Station B2, ranked based on the average performances across all datasets and thresholds.

Method	Backbone	Desc. Dim.	Hyundai Dept. Store 1F (Recall@1)					
			1m 5°	10m 30°	25m 60°	1m	10m	25m
sfrs	VGG16	4096	9.5	84.6	92.6	**38.7**	**93.5**	96.2
mixvpr	ResNet50	4096	7.6	87.3	93.2	26.6	92.9	96.3
eigenplaces	ResNet50	2048	7.9	81.9	88.6	26.6	89.4	94.7
salad	DINOv2	8448	**11.9**	**91.4**	**94**	22.6	92.6	96.6
eigenplaces	ResNet101	2048	8.7	83.8	88.8	26.8	90.1	95
eigenplaces	ResNet101	512	8.6	81.7	86.6	25.2	88.1	94.1
eigenplaces	ResNet50	512	8	76.4	83	26.4	84.9	92.4
netvlad	VGG16	4096	8.3	74.9	86.8	35.4	87.9	95.2
anyloc	DINOv2	49152	8.4	75.9	83	25.3	89	**98**
netvlad	VGG16	32768	8	74.1	84.1	32.6	86.2	92.8
Upper Bounds			18	100	100	84.7	100	100

Table 5.4: Benchmark results of the top ten best performing models on Hyundai Department Store 1F, ranked based on the average performances across all datasets and thresholds.

Method	Backbone	Desc. Dim.	Hyundai Dept. Store 4F (Recall@1)					
			1m 5°	10m 30°	25m 60°	1m	10m	25m
sfrs	VGG16	4096	14.7	91.4	96.4	**61**	97.4	99.3
mixvpr	ResNet50	4096	17.1	**96.1**	**98.1**	56.3	**99**	**99.8**
eigenplaces	ResNet50	2048	12.5	92.6	95.9	52.1	96.4	99.1
salad	DINOv2	8448	**18.2**	95.5	97.3	42.5	97.1	99
eigenplaces	ResNet101	2048	13.5	91.6	96.4	51.9	95.5	99
eigenplaces	ResNet101	512	11.6	91.6	96.2	46.6	93.8	98.5
eigenplaces	ResNet50	512	13.9	91.3	93.8	49.8	94.7	97.4
netvlad	VGG16	4096	9.9	89.9	96.9	56.3	97.9	99.3
anyloc	DINOv2	49152	12.5	88.4	94	39	96.4	99.1
netvlad	VGG16	32768	9.4	87.8	95.5	53.8	96.9	99.3
Upper Bounds			40	100	100	98.4	100	100

Table 5.5: Benchmark results of the top ten best performing models on Hyundai Department Store 4F, ranked based on the average performances across all datasets and thresholds.

Method	Backbone	Desc. Dim.	Hyundai Dept. Store B1 (Recall@1)					
			1m 5°	10m 30°	25m 60°	1m	10m	25m
sfrs	VGG16	4096	9.4	83.6	90.1	**33.6**	89.6	96.5
mixvpr	ResNet50	4096	8.7	84.1	87.8	26.6	88.1	93.4
eigenplaces	ResNet50	2048	9.1	82.1	88.3	29.1	87.8	95.7
salad	DINOv2	8448	**12**	**92.3**	**93.8**	27.6	**93.9**	**98.2**
eigenplaces	ResNet101	2048	9.4	85	89.6	28.8	90.3	96.9
eigenplaces	ResNet101	512	8.9	81.7	87.2	28.4	87.5	95.7
eigenplaces	ResNet50	512	8.2	79.6	85.7	27.8	85.8	95.7
netvlad	VGG16	4096	8.7	74.2	82.9	31.5	83.6	94
anyloc	DINOv2	49152	8.8	77.3	83.7	24.4	86.9	97.9
netvlad	VGG16	32768	8.3	72.7	80.5	30.3	81.8	93.9
Upper Bounds			21.6	100	100	90.4	100	100

Table 5.6: Benchmark results of the top ten best performing models on Hyundai Department Store B1, ranked based on the average performances across all datasets and thresholds.

Method	Backbone	Desc. Dim.	Gangnam Station B2 (Recall@1)					
			1m 5°	10m 30°	25m 60°	1m	10m	25m
sfrs	VGG16	4096	3.9	**50.8**	**63.8**	**40.8**	**60**	70
mixvpr	ResNet50	4096	4.3	44.5	56.9	31.2	52.6	63.1
anyloc	DINOv2	49152	**4.7**	40.9	53.9	26.9	55.2	**70.1**
eigenplaces	ResNet50	2048	4	46.6	53.4	25.5	54	60
	ResNet101	2048	3.8	44.3	50.8	25.4	51.8	57.9
	ResNet101	512	3.7	45.2	51.7	22.4	53.1	59.3
	ResNet50	512	3.5	41.8	48.4	23.1	50	55.8
	VGG16	512	2.7	40	46.5	18.5	49	54.3
	ResNet50	256	3.3	39.2	45.2	19.9	46.7	53.2
	ResNet101	256	3.1	38.7	45.1	19.8	46.1	52
Upper Bounds			8.8	100	100	90.9	100	100

Table 5.7: Benchmark results of a few of the best performing models and the top seven eigenplaces configurations on Gangnam Station B2. The separate groups are ranked based on the average performance across all datasets and thresholds.

Method	Backbone	Desc. Dim.	Gangnam Station B1 (Recall@1)					
			1m 5°	10m 30°	25m 60°	1m	10m	25m
sfrs	VGG16	4096	3.6	46.4	54.2	**26**	55.7	60.4
mixvpr	ResNet50	4096	3.6	50.7	56.6	24.4	56.6	60.7
salad	DINOv2	8448	**5.2**	**59.4**	**61.5**	21.1	62	67.9
cosplace	VGG16	512	3.3	41.4	46.8	18.1	48.2	53.8
apgem	ResNet101	2048	2.1	35	46.1	19.6	50.7	56.5
convap	ResNet50	4096	3.4	36.1	41.1	14.9	41.5	47.7
Upper Bounds			10.4	97.1	100	81	97.2	100

Method	Backbone	Desc. Dim.	Gangnam Station B2 (Recall@1)					
			1m 5°	10m 30°	25m 60°	1m	10m	25m
sfrs	VGG16	4096	3.9	**50.8**	**63.8**	**40.8**	**60**	70
mixvpr	ResNet50	4096	4.3	44.5	56.9	31.2	52.6	63.1
anyloc	DINOv2	49152	**4.7**	40.9	53.9	26.9	55.2	**70.1**
cosplace	VGG16	512	3.9	33.5	43	20.6	39.8	53.9
apgem	ResNet101	2048	3.5	41.3	57.1	30.6	55.5	69.2
convap	ResNet50	4096	4	34.1	43.8	18.4	40.1	53.6
Upper Bounds			8.8	100	100	90.9	100	100

Table 5.8: Benchmark results of a few of the best performing models and the best configurations (on average) of Cosplace, AP-GeM and Conv-AP on Gangnam Station B1 and B2. As one can see, the former are much more effective than the latter on both datasets.

5.4 Finetuning Models for Indoor VPR

In the previous section it was established that, out of the most effective deep Visual Place Recognition methods, none are designed to work in indoor environment. Moreover, the size of available indoor datasets is generally too small to achieve a satisfactory training of a deep model. For this reason, the only way to improve performances is to construct a fine-tuning procedure that exploits the limited amount of labeled data available. This is done in the contrastive learning framework proposed by Berton et al. [2022b]: in each epoch a number of queries are randomly selected and their positives and negatives are mined from the database to form triplets. Then the model extracts their features and feed them to a Triplet Loss. This aims at pushing the queries representations closer to the positives and far from the negatives ones in the latent space. A visualization of this process is shown in Figure 5.8. Although the idea is quite simple and standard, it offers a great deal of new information and insight into what direction researchers could move to tackle the task of Indoor Visual Localization.

Firstly, it is necessary to determine which of the methods tested in the benchmark to pick for the fine-tuning. Since none of them is globally optimal, this choice is not trivial. Based on their overall performance, robustness across configurations and ease of

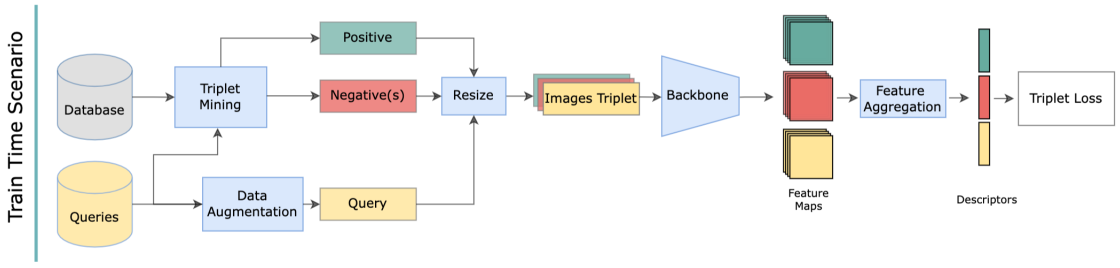


Figure 5.8: Visual represetation of the fine-tuning procedure on indoor environments. Image taken from Berton et al. [2022b]

technical implementation, EigenPlaces and Salad are selected. Secondly, but arguably more importantly, there is the need of defining a train-validation-test split across the available datasets. Since each of them is already quite small (see Table 5.1), the split will not actually divide any of them but simply assign them to either the training, validation or test group. The number of possible combinations is too high to perform a full search, so some general rules are set:

- Gangnam Station B2 is only ever used for testing, as it is the second smallest and is the most noisy;
- Both B1 sections are either used for training or validation. This is because they are the two largest ones, and thus offer the largest amounts of queries for training, and because they are challenging benchmarks for the validation;
- Hyundai Department Store 1F and 4F are only used for testing, as they are too small for training and not challenging enough for validation. In a few experiments this rule has to be broken as both the B1s are used in training. It has been determined experimentally that, in those situations, 4F generates better performances;
- During the first phase of experiments Baidu Mall is only used for testing, but is added at a later time to the training group so that its impact can be rigorously analyzed;

Then a mining method must be defined. Since all queries have available labels, no expensive mining is actually required. The procedure follows these steps:

1. The queries are randomly sampled from the training group and their hard positives are computed. A hard positives is a database image whose position is closer to the query than a threshold (referred to as *hard positives threshold*). Those who have none are ignored. The number of queries sampled depends on the training dataset and will be discussed later;
2. A number of database images are sampled, then both their and the queries features are extracted;
3. For each query, the best positive and hardest negatives are selected. This is the hard positive whose feature vector is the closest to the query's one in the latent space;

4. Finally, the hardest negatives are chosen for each query. These are obtained by first selecting all sampled database images whose distance to the query is larger than a second threshold, called *soft positives threshold* and always greater the soft counterpart, and then choosing the ten whose representations are the closest the query's ones. The "triplets" are therefore formed by twelve images: the query, the best positives and ten hardest negatives. The number of negatives picked could be object for optimization, but it is out of the scope of this work.

Starting from these rules and through a long series of preliminary tests, the candidate groups for the training-validation and testing split are presented in Table 5.9.
Finally, the test thresholds for Recall@1 are slightly adjusted from the benchmark: the

Training	Validation	Test
Gangnam Station B1	Hyundai Dept. Store B1	Gangnam Station B2 Hyundai Dept. Store 1F-4F Baidu Mall
Hyundai Dept. Store B1	Gangnam Station B1	Gangnam Station B2 Hyundai Dept. Store 1F-4F Baidu Mall
Gangnam Station B1 Hyundai Dept. Store B1	Hyundai Dept. Store 4F	Gangnam Station B2 Hyundai Dept. Store 1F Baidu Mall

Table 5.9: Selected train-validation-test splits for the fine-tuning.

values are 1m, 5m, 10m and 25m. Thresholds based on angular distances have been dropped because of the very role of image retrieval within visual localization, which is to find relevant images in the database rather than estimate the image pose.

Now that the architecture and the procedures have been presented, it is now the time to present the actual experiments, their goals and their results. Please note that, from now on, all tables will contain a short abbreviation in place of the full names of the datasets. They should be easy to understand, but a dictionary can be found in Table 5.10.

Full Name	Abbreviation
Gangnam Station B1	GB1
Gangnam Station B2	GB2
Hyundai Department Store 1F	H1F
Hyundai Department Store 4F	H4F
Hyundai Department Store B1	HB1
Baidu Mall	BAI

Table 5.10: Brief dictionary of the abbreviations that will be used from now on.

First Batch: analysis of optimal thresholds and the training sets composition

The first batch of experiments has a twofold objective: firstly it studies how different hard and soft positives thresholds affect the performances of the fine tuning. This is done by setting them to three different values, chosen arbitrarily, which spans the domain of conceivable values. To be precise, the chosen option are (2,5), (5,10) and (10, 25). Secondly, it aims at finding what the best way is to make use of the training datasets. Since in most cases the training group is composed of multiple dataset, it is not clear whether or not they should be used separately or as a single larger one. The training could either:

- Use a single dataset in each epoch, rotating through them until the training stops. This choice would limit the sampling to the database in which the queries belong and not allow for cross-dataset hard negatives;
- Combine all datasets in the training group into a larger one, allowing the choosing of a hard negative from a different dataset. This is done by arbitrarily shifting one of the coordinates of all images by $k * 10^5$, where k is a small positive integer that identify the original dataset. The hard positives remains obviously unchanged.

The second option could potentially generate harder negatives, thus improving the quality of the learned feature, but at the same time the quality of the database sampling is diluted. Since this approach has never been discussed in the literature, there is not a standard way go. The experiments are therefore designed to evaluate the two possibilities and determine which one works best. Please note that, in this first batch, this difference is only relevant when using the third split in Table 5.9 i.e. the only one that has multiple datasets in the training group. In this batch all experiments fine-tune the same EigenPlaces model, with ResNet50 as the backbone and a descriptors dimension of 2048.

Since a single huge table would be hard to read and understand, the results will be split into smaller tables each designed to show a relevant insight. The general rule for all tables is that cursive values indicates that the performances is better than all baselines, while bolded ones indicates that it is the best overall. The baseline performances of the off-the-shelf methods is reported in Tables 5.11 and 5.12. The first experiments results are presented in Tables 5.13 and 5.14. They show the results of all the experiments that employ separated training datasets grouped by the training-validation-testing split. The numbers show that there are not globally optimal choices for neither the threshold nor the splits. Generally speaking, the (5,10) thresholds seems to lead to the overall best performances on Baidu but significantly worse on the other benchmarks. This is probably due to the fact that those distances better fit the properties of Baidu, which is significantly sparser. Both (10,25) and (2,5) performs better overall, with the former working slightly better on Baidu and worse on the others. Again, this is probably due to its sparsity. In terms of the training-validation-testing splits, both GB1 and GB1-HB1 generates the best results, while HB1 trails in all testsets.
Overall, the fine-tuned models outperform the off-the-shelf counterparts by a significant margin, especially under the stricter threshold. This is not true for Baidu, in which the

results are almost always unchanged or slightly worse. Notably, no single methods outperforms the best baselines. This is probably due to the that the Gangnam and Hyundai are created in a very similar way, while Baidu is not, thus there is a significant divergence in the general way the images look, even if they are all indoor datasets. By not using Baidu in the training, the models struggle on it as they seem to overfit on the appearances of the others.

In Table 5.15 the results of the training with aggregated datasets using both B1 sections. For this instances, the results seems to get worse on average. This indicates that, on average, for this configuration of datasets the additional hard negatives do not compensate for the dilution in the sampling.

Baseline Methods	GB2 (R@1)				H1F			
	1m	5m	10m	25m	1m	5m	10m	25m
sfrs	***40.8***	*59.1*	*60*	70	***38.7***	*89.1*	*93.5*	96.2
mixvpr	31.2	51.4	52.6	63.1	26.6	84.7	92.9	96.3
eigenplaces	30.8	53.1	54.3	65.5	26.6	82.8	89.4	94.7
salad	22.2	43.9	45.5	57.6	22.6	84.7	92.6	96.6
netvlad	36.7	53.4	54.4	62.6	35.4	82.8	87.9	95.2
anyloc	26.9	50.9	55.2	*70.1*	25.3	80	89	*98*

Table 5.11: Performances of the baseline methods on Gangnam Station B2 and Hyundai Department Store 1F. The configurations are the best performing ones in the benchmark.

Baseline Methods	H4F (R@1)				BAI			
	1m	5m	10m	25m	1m	5m	10m	25m
sfrs	***61***	94.9	97.4	99.3	***5.1***	38	61	71.2
mixvpr	56.3	*96.1*	*99*	***99.8***	3.3	39.5	67.9	77.3
eigenplaces	52.1	93.8	96.4	99.1	4.1	39.5	65.5	75.3
salad	42.5	93.7	97.1	99	3.4	38.4	72.1	82.3
netvlad	56.3	*96.1*	97.9	99.3	4.6	40.3	64.4	75.7
anyloc	39	92.8	96.4	99.1	3.3	***40.7***	***76.1***	***88.6***

Table 5.12: Performances of the baseline methods on Hyundai Department Store 4F and Baidu Mall. The configurations are the best performing ones in the benchmark.

Train	Val	Thr.	GB2 (R@1)				H1F			
			1m	5m	10m	25m	1m	5m	10m	25m
GB1	HB1	10,25	39.1	*61.1*	*62.3*	*71.7*	34.7	*89.2*	93.2	96.3
		5,10	36.4	*60*	*61.5*	69.8	36.2	***89.8***	*93.9*	96.7
		2,5	38.6	*61.1*	*61.8*	69.4	33.2	88.8	*94.6*	96.6
HB1	GB1	10,25	34.4	55.8	56.7	68.4	32	87.6	92.6	96.3
		5,10	34.1	55.1	56.2	68.8	31.1	85.8	92.2	95.9
		2,5	35.5	56.9	58	68.6	31.5	88	92.8	96.6
GB1-HB1	H4F	10,25	37.6	*61*	*62.6*	*71.2*	34.9	*89.2*	93.1	96.2
		5,10	35.2	*60.2*	*61.2*	70.1	33.8	88.6	92.9	95.6
		2,5	38.8	*62.3*	*63.4*	*72.5*	35	88.1	93.1	96.3

Table 5.13: Performances of fine-tuned versions of EigenPlaces on Gangnam Station B2 and Hyundai Department Store 1F. For the last configuration the training has been done with separated datasets.

Train	Val	Thr.	H4F (R@1)				BAI			
			1m	5m	10m	25m	1m	5m	10m	25m
GB1	HB1	10,25	57.4	*97.1*	98.6	99.3	4.2	39.1	64	73.2
		5,10	56.3	*97.4*	99	99.5	3.7	41.4	66.7	75.7
		2,5	59.1	***97.6***	99	99.5	3.6	39.4	63.4	72.9
HB1	GB1	10,25	48.6	*96.4*	97.8	98.6	3.7	38.7	62.3	71.2
		5,10	48.1	95.4	98.5	99	3.8	37.6	62.2	71.6
		2,5	50.9	94.7	97.3	99	4.1	36.2	59.7	68.7
GB1-HB1	H4F	10,25					4.1	39.6	64	72.8
		5,10		-			3.8	38.7	64	72.3
		2,5					3.7	38.5	62.4	71.6

Table 5.14: Performances of fine-tuned versions of EigenPlaces on Hyundai Department Store 4F and Baidu Mall. For the last configuration the training has been done with separated datasets.

Thr.	GB2 (R@1)				H1F				BAI			
	1m	5m	10m	25m	1m	5m	10m	25m	1m	5m	10m	25m
10,25	36.7	*61.2*	*62.3*	70	37.1	88.1	93.3	97.1	3.8	39.4	63.9	73.3
5,10	35.4	57.4	58.3	67.7	32.4	86.2	91.6	95.8	3.6	39.4	62.6	71.8
2,5	38.3	*60.8*	*61.6*	69.9	33.8	89	*93.9*	97	3.8	39	63.7	72.8

Table 5.15: Performances of fine-tuned versions of EigenPlaces trained on GB1-HB1. The training has been done with aggregated datasets.

Second Batch: including Baidu Mall into the training

The second batch of experiments adds the Baidu Mall dateset into the training. The configurations are kept the same to ensure that results can be compared. The only difference is that the difference between separated and aggregated dataset during training is now always relevant. The results of the models trained with separated datasets are shown in Tables 5.16 and 5.17, while those of the ones trained with aggregated datasets are shown in Tables 5.18 and 5.19. First of all, there is a clear indication that this do not work well under the (2,5) thresholds. This confirms our finding in the first batch, i.e. these thresholds are too low for Baidu, and this is true regardless of the separation or aggregation of the training datasets. Under (10,25) the models seem to perform the best, both with GB1-HB1-BAI and GB1-BAI as training group. Similarly to the first batch, HB1-BAI is not effective. There does not seem to be a clear improvement or decline when aggregating rather then separating the training datasets.

Train	Val	Thr.	GB2 (R@1)				H1F			
			1m	5m	10m	25m	1m	5m	10m	25m
GB1-BAI	HB1	10,25	37.8	***63.8***	***65.1***	***73.7***	36	89	93.5	96.3
		5,10	36.2	*60.9*	*61.9*	*71.2*	33.7	87.2	92.2	95.8
		2,5	35.4	57.6	59.1	69.1	33.7	88.4	92.5	96
HB1-BAI	GB1	10,25	34.8	57.3	59.1	68.8	32.3	87.7	92	96.2
		5,10	29.7	55.6	57.3	68.2	33.4	85.4	90.3	94.3
		2,5	31.7	53.2	54.7	66.3	31.9	86.8	91.1	94.6
GB1-HB1-BAI	H4F	10,25	35.2	*61.7*	*63.3*	*71.5*	36.1	87.2	91.8	94.6
		5,10	38.2	*61*	*62.3*	70.1	34.5	87.1	91.7	95.6
		2,5	36.7	*59.4*	*60.8*	68.2	33.5	88.8	*93.7*	96.6

Table 5.16: Performances of fine-tuned versions of EigenPlaces on Gangnam Station B2 and Hyundai Department Store 1F. For all configurations the training has been done with separated datasets.

Train	Val	Thresholds	H4F (R@1)			
			1m	5m	10m	25m
GB1-BAI	HB1	10,25	53.9	*97.3*	***99.1***	99.7
		5,10	56.2	*96.4*	97.9	99
		2,5	55.5	95.9	97.8	99.3
HB1-BAI	GB1	10,25	50.2	94.9	97.3	98.8
		5,10	46.4	*96.9*	98.5	99.5
		2,5	48.3	94.9	97.3	99

Table 5.17: Performances of fine-tuned versions of EigenPlaces on Hyundai Department Store 4F. For all configurations the training has been done with separated datasets.

Train	Val	Thr.	GB2 (R@1)				H1F			
			1m	5m	10m	25m	1m	5m	10m	25m
		10,25	36.7	58.4	59.8	*70.5*	33.9	88	93.3	96.7
GB1-BAI	HB1	5,10	35.6	*59.7*	*61.2*	*70.6*	33.4	***89.8***	***95***	97.4
		2,5	37.2	58.4	59.5	70	34.9	88.6	*94.6*	97.3
		10,25	33.6	56.4	58.1	69.7	34.3	85.4	91.6	96.2
HB1-BAI	GB1	5,10	36.6	*59.5*	*60.9*	*71.5*	30.8	88.4	*93.9*	97
		2,5	34.2	54.7	56.2	66.9	31.7	87.3	92.5	95.6
		10,25	35.4	58.7	59.6	69.2	35.1	*89.5*	*93.6*	96.3
GB1-HB1-BAI	H4F	5,10	34.2	54.7	55.8	65.6	32.2	88.6	*93.6*	97.3
		2,5	36.8	58.1	59.1	68	33.4	89	93.7	*97.4*

Table 5.18: Performances of fine-tuned versions of EigenPlaces on Gangnam Station B2 and Hyundai Department Store 1F. For all configurations the training has been done with aggregated datasets.

Train	Val	Threshold	H4F (R@1)			
			1m	5m	10m	25m
		10,25	53.8	*97.1*	97.9	99
GB1-BAI	HB1	5,10	54.3	*96.6*	98.5	99.5
		2,5	52.4	*96.7*	98.8	99.7
		10,25	48.8	95	97.1	99
HB1-BAI	GB1	5,10	49.7	94.3	96.9	96.6
		2,5	45.4	95	97.8	99

Table 5.19: Performances of fine-tuned versions of EigenPlaces on Hyundai Department Store 4F. For all configurations the training has been done with aggregated datasets.

Third Batch: Using Salad as the starting model

The third and final batch of experiments aims at studying the performing of the fine-tuning process on a different model, in particular one that employs a different backbone and a completely different training regimen: Salad. Dinov2 is significantly more demanding to fine-tune than the ResNet50 in terms of GPU memory usage and time required, thus the number of experiments has been reduced by only focusing on training using aggregated datasets. Moreover, the size of the images had to be reduced. The numerical results are shown in Tables 5.20 and 5.21

The numbers clearly indicates that the overfitting issue mentioned in the comments to the first batch of experiments happens regardless of the method chosen. There are multiple instances in which Salad beats all competitors in Gangnam B2 and Hyundai 1F-4F while getting significantly worse in Baidu. As for the thresholds and the splits, once again training on only HB1 or using (5,10) does not lead to the best results. At the same time, using (2,5) seems to emphasize the overfitting issue. In most cases, however, there is a significant improvement over the off-the-shelf Salad model.

The addition of Baidu to all training groups have been thoroughly tested, similarly to the second batch, but since the performances are all significantly worse and no interesting insights was found, the numerical results will not be shown.

Train	Val	Thr.	GB2 (R@1)				H1F			
			1m	5m	10m	25m	1m	5m	10m	25m
GB1	HB1	10,25	29.6	*60,5*	*62,8*	*71,6*	26,8	*89,9*	*95,9*	***98,6***
		5,10	35.4	*61,8*	*62.7*	*70,6*	28,2	***91,7***	*95*	97.8
		2,5	33.7	*59,5*	*61*	*70,2*	30.2	*92*	*95.6*	97.5
HB1	GB1	10,25	24,6	49	50.7	59,2	24,3	88,4	92.9	96.7
		5,10	26,3	50	51,4	61,2	27,1	*89.2*	*94.1*	97,3
		2,5	33.1	52.4	52,9	64.1	33.4	*91.8*	*94.6*	96.9
GB1-HB1	H4F	10,25	27.7	53.4	54.9	65.4	28.9	*90.9*	*95.1*	97.5
		5,10	28.2	52.5	53.1	64.1	27.9	88	*94*	97.7
		2,5	38.6	***65.5***	***66.2***	***74.9***	31.7	*93.1*	*96.5*	*98.2*

Table 5.20: Performances of fine-tuned versions of Salad on Gangnam Station B2 and Hyundai Department Store 1F. For the last configuration the training has been done with separated datasets.

Train	Val	Thr.	H4F (R@1)				BAI			
			1m	5m	10m	25m	1m	5m	10m	25m
GB1	HB1	10,25	52,9	*97.1*	98.3	99.3	2,9	38,7	73,1	86,3
		5,10	52,6	*98.6*	*99,1*	99.5	3.3	37,4	68.5	79.9
		2,5	51.7	***97.6***	98,5	99.1	2.7	39.2	71,8	83,3
HB1	GB1	10,25	42	*96.4*	97.9	99.1	3	37.2	66.3	77.5
		5,10	45	*95.7*	97.4	98.8	3.6	37.9	65.4	76.7
		2,5	55.3	*97.9*	98.5	99	3.1	34.7	55.5	66.1
GB1-HB1	H4F	10,25					3	35.3	62.3	75.3
		5,10	-				3.4	38.7	72.8	85
		2,5					3.4	39.4	68.4	79.2

Table 5.21: Performances of fine-tuned versions of Salad on Hyundai Department Store 4F and Baidu Mall. For the last configuration the training has been done with separated datasets.

Chapter 6

Conclusions

This chapter will contain the final remarks regarding this thesis, including a discussion on the results, limitations and insights for future works of both experimental scenarios.
In the first one the goal was to determine whether or not a domain adaptation technique could help a large scale Visual Place Recognition model deal with the domain gap between the training images and the queries. This divergence exists because the former generally come from a single source, Google Street View, while the latter may be taken by anyone under any circumstance. Moreover, it would be very helpful to pick a model that's trained on a city and deploy it into another one without having to fetch another large dataset to repeat the training on, as this process would be very long and costly. This issues are reproduced in two different settings:

- Using the training and the test splits of the San Francisco XL datasets, it is possible to isolate a domain gap concerning viewpoint shift, because the queries are taken by consumers through commercial cameras;
- Using the training split from SF-XL and the test split from the Tokyo 24/7 dataset an additional layer of complexity is added, since not only are the queries of the same nature as those in the other setting, but they also come from a different city;

In both cases the idea is to train a CosPlace model, which is currently one of the best performing approaches to VPR, and inserting it into a specific adversarial training regimen which employ a simple yet effective Gradient Reversal Layer. The objective is to create a feature extractor that is domain invariant with respect to the database images and queries. This method has been choosen because of his proven effectiveness and strong theoretical foundations.
In the first setting, the experiments prove that this procedure brings actual improvement over the baseline. The gap is however quite slim ($\sim 4.5\%$), thus proving the fact that the original model is already quite robust to this type of domain shift.
In the second setting the results are similar in terms of increase in performances, however they prove not only that the adversarial approach can work well in bridging the domain gap between cities, but also that, by increasing the number of sample queries from the target domain (i.e. Tokyo) the whole process becomes more robust to its hyperparameters

and thus more reliable for deployment. In both cases it was establish that convergence could be reached within a limited number of epochs.

In the second scenario the goal is to study in what way could the performances of state-of-the-art be improved for indoor environment. This is extremely useful in the context of Visual Localization, which is a deep learning task in which, given a picture, the 6 degrees of freedom that define the pose of the camera are estimated. This is generally done through local descriptors matching between the queries and the candidates from the database. The quality and speed of the process is directly proportional to the quality and number of candidates that a VPR model can retrieve.
The first step is to create the wider possible benchmark using most of the available datasets created in different indoor environments. Generally speaking, it was found that all approaches that utilizes a Dinov2 backbone work egregiously on larger evaluation thresholds, such as 10 and 25 meters, while the constrastive-based SFRS is consistently the best under the strictest threshold (1m). Overall, EigenPlaces, Salad, SFRS and Anyloc are proven to be some of the most effective approaches, and the first two are therefore selected for a fine-tuning procedure that exploits the few available labeled queries. The models are put through a few epochs of constrastive learning in order to study what is the best way to boost their performances in this particular domain. Results show that using loose thresholds to define hard and soft positives (i.e. 10 and 25 meters) leads to the overall best performances, while using stricter ones (2 and 5 meters) generates some overfitting over a specific type of dataset. Moreover, the dataset Gangnam Station B1 seems to be able to offer good training quality just by itself or in combination with Hyundai Department Store B1. The experiments also investigates into the addition of the Baidu Mall dataset into the training, revealing that it does not improve the performances. This is probably due to its different appearance with respect to the other datasets. This means that what the model learns from it is not useful during testing. Finally, Salad was also tested in the same configurations and the results confirms all the previous findings.
The main limitation that seriously hinders the potential work on this topic is the serious lack of large scale indoor datasets for Visual Place Recognition. Currently there are very few options available and are generally limited in number of images and area covered. This is because such they are created specifically for Visual Localization and thus require an accurate 3D mapping, that can be done only through more complex, costly and slow equipment. The extent of the future work on this topic will heavily depend on the quantity and quality of newly-proposed datasets.

Bibliography

Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. GSV-Cities: Toward appropriate supervised visual place recognition. *Neurocomputing*, 513:194–203, November 2022. ISSN 09252312. doi: 10.1016/j.neucom.2022.09.127. URL https://linkinghub.elsevier.com/retrieve/pii/S0925231222012188.

Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. MixVPR: Feature Mixing for Visual Place Recognition, March 2023. URL http://arxiv.org/abs/2303.02190. arXiv:2303.02190 [cs].

Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition, May 2016. URL http://arxiv.org/abs/1511.07247. arXiv:1511.07247 [cs].

Artem Babenko, Anton Slesarev, Alexandr Chigorin, and Victor Lempitsky. Neural Codes for Image Retrieval. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 584–599, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10590-1. doi: 10.1007/978-3-319-10590-1_38.

Mahsa Baktashmotlagh, Mehrtash Harandi, Mathieu Salzmann, and Gabriela Csurka. *Learning Domain Invariant Embeddings by Matching Distributions*, pages 95–114. Springer International Publishing, Cham, 2017. ISBN 978-3-319-58347-1. doi: 10.1007/978-3-319-58347-1_5. URL https://doi.org/10.1007/978-3-319-58347-1_5.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, June 2008. ISSN 10773142. doi: 10.1016/j.cviu.2007.09.014. URL https://linkinghub.elsevier.com/retrieve/pii/S1077314207001555.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine Learning*, 79(1):151–175, May 2010. ISSN 1573-0565. doi: 10.1007/s10994-009-5152-4. URL https://doi.org/10.1007/s10994-009-5152-4.

Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking Visual Geo-localization for Large-Scale Applications, April 2022a. URL http://arxiv.org/abs/2204.02287. arXiv:2204.02287 [cs].

Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. Deep Visual Geo-Localization Benchmark. pages 5396–5407, 2022b. URL https://openaccess.thecvf.com/content/CVPR2022/html/Berton_Deep_Visual_Geo-Localization_Benchmark_CVPR_2022_paper.html.

Gabriele Berton, Gabriele Trivigno, Barbara Caputo, and Carlo Masone. EigenPlaces: Training Viewpoint Robust Models for Visual Place Recognition. pages 11080–11090, 2023. URL https://openaccess.thecvf.com/content/ICCV2023/html/Berton_EigenPlaces_Training_Viewpoint_Robust_Models_for_Visual_Place_Recognition_ICCV_2023_paper.html.

Karsten M. Borgwardt, Arthur Gretton, Malte J. Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J. Smola. Integrating structured biological data by Kernel Maximum Mean Discrepancy. *Bioinformatics*, 22(14):e49–e57, July 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl242. URL https://doi.org/10.1093/bioinformatics/btl242.

Bingyi Cao, Andre Araujo, and Jack Sim. Unifying Deep Local and Global Features for Image Search, September 2020. URL http://arxiv.org/abs/2001.05027. arXiv:2001.05027 [cs].

Changhao Chen, Bing Wang, Chris Xiaoxuan Lu, Niki Trigoni, and Andrew Markham. Deep Learning for Visual Localization and Mapping: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2023. ISSN 2162-2388. doi: 10.1109/TNNLS.2023.3309809. URL https://ieeexplore.ieee.org/abstract/document/10260323. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

Gabriela Csurka, Timothy M. Hospedales, Mathieu Salzmann, and Tatiana Tommasi. *Visual Domain Adaptation in the Deep Learning Era*. Synthesis Lectures on Computer Vision. Springer International Publishing, Cham, 2022. ISBN 978-3-031-79170-3 978-3-031-79175-8. doi: 10.1007/978-3-031-79175-8. URL https://link.springer.com/10.1007/978-3-031-79175-8.

N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893, San Diego, CA, USA, 2005. IEEE. ISBN 978-0-7695-2372-9. doi: 10.1109/CVPR.2005.177. URL http://ieeexplore.ieee.org/document/1467360/.

Bharath Bhushan Damodaran, Benjamin Kellenberger, Rémi Flamary, Devis Tuia, and Nicolas Courty. DeepJDOT: Deep Joint Distribution Optimal Transport for Unsupervised Domain Adaptation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, Lecture Notes in Computer Science, pages 467–483, Cham, 2018. Springer International Publishing. ISBN 978-3-030-01225-0. doi: 10.1007/978-3-030-01225-0_28.

Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):5962–5979, October 2022. ISSN 0162-8828, 2160-9292, 1939-3539. doi: 10.1109/TPAMI.2021.3087709. URL http://arxiv.org/abs/1801.07698. arXiv:1801.07698 [cs].

Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL https://dl.acm.org/doi/10.1145/358669.358692.

Yaroslav Ganin and Victor Lempitsky. Unsupervised Domain Adaptation by Backpropagation, February 2015. URL http://arxiv.org/abs/1409.7495. arXiv:1409.7495 [cs, stat].

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. ISSN 1533-7928. URL http://jmlr.org/papers/v17/15-239.html.

Yixiao Ge, Haibo Wang, Feng Zhu, Rui Zhao, and Hongsheng Li. Self-supervising Fine-grained Region Similarities for Large-scale Image Localization, July 2020. URL http://arxiv.org/abs/2006.03926. arXiv:2006.03926 [cs].

Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time RGB-D camera relocalization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 173–179, October 2013. doi: 10.1109/ISMAR.2013.6671777. URL https://ieeexplore.ieee.org/abstract/document/6671777?casa_token=SSgbMBckDgQAAAAA:H_lpCjy4M9gdNfpwEd1GVx4TWt65ikaAYwe4kGVSLBDYfTej5dYqZh2peag-5A5MKhkLDuCc6w.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper_files/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html.

Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-end Learning of Deep Visual Representations for Image Retrieval, May 2017. URL http://arxiv.org/abs/1610.07940. arXiv:1610.07940 [cs].

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. URL http://arxiv.org/abs/1512.03385. arXiv:1512.03385 [cs].

Martin Humenberger, Yohann Cabon, Noé Pion, Philippe Weinzaepfel, Donghwan Lee, Nicolas Guérin, Torsten Sattler, and Gabriela Csurka. Investigating the Role of Image Retrieval for Visual Localization. *International Journal of Computer Vision*, 130(7): 1811–1836, July 2022. ISSN 1573-1405. doi: 10.1007/s11263-022-01615-7. URL https://doi.org/10.1007/s11263-022-01615-7.

Sergio Izquierdo and Javier Civera. Optimal Transport Aggregation for Visual Place Recognition, November 2023. URL http://arxiv.org/abs/2311.15937. arXiv:2311.15937 [cs].

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*, pages 3304–3311, June 2010. doi: 10.1109/CVPR.2010.5540039. URL https://ieeexplore.ieee.org/abstract/document/5540039?casa_token=HOmO7qklAGkAAAAA:a6i4hRHKF5qWtcDDOh1qqqn6ytJdLBAgYMQgPxiBvDC5dOZPmRxNAZGyaTWABmHvNjk5Gc5hjw. ISSN: 1063-6919.

Nikhil Keetha, Avneesh Mishra, Jay Karhade, Krishna Murthy Jatavallabhula, Sebastian Scherer, Madhava Krishna, and Sourav Garg. AnyLoc: Towards Universal Visual Place Recognition. *IEEE Robotics and Automation Letters*, 9(2):1286–1293, February 2023. ISSN 2377-3766. doi: 10.1109/LRA.2023.3343602. URL https://ieeexplore.ieee.org/abstract/document/10361537. Conference Name: IEEE Robotics and Automation Letters.

Wouter M. Kouw and Marco Loog. A Review of Domain Adaptation without Target Labels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(3):766–785, March 2021. ISSN 1939-3539. doi: 10.1109/TPAMI.2019.2945942. URL https://ieeexplore.ieee.org/document/8861136. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

Donghwan Lee, Soohyun Ryu, Suyong Yeon, Yonghan Lee, Deokhwa Kim, Cheolho Han, Yohann Cabon, Philippe Weinzaepfel, Nicolas Guerin, Gabriela Csurka, and Martin Humenberger. Large-Scale Localization Datasets in Crowded Indoor Spaces. pages 3227–3236, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Lee_Large-Scale_Localization_Datasets_in_Crowded_Indoor_Spaces_CVPR_2021_paper.html.

Shuang Li, Chi Liu, Qiuxia Lin, Binhui Xie, Zhengming Ding, Gao Huang, and Jian Tang. Domain Conditioned Adaptation Network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07):11386–11393, April 2020. ISSN 2374-3468. doi: 10.1609/aaai.v34i07.6801. URL https://ojs.aaai.org/index.php/AAAI/article/view/6801. Number: 07.

Tie-Yan Liu. Learning to Rank for Information Retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, June 2009. ISSN 1554-0669, 1554-0677. doi: 10.1561/1500000016. URL https://www.nowpublishers.com/article/Details/INR-016. Publisher: Now Publishers, Inc.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning Transferable Features with Deep Adaptation Networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 97–105. PMLR, June 2015. URL https://proceedings.mlr.press/v37/long15.html. ISSN: 1938-7228.

D.G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157 vol.2, Kerkyra, Greece, 1999. IEEE. ISBN 978-0-7695-0164-2. doi: 10.1109/ICCV.1999.790410. URL http://ieeexplore.ieee.org/document/790410/.

Stephanie Lowry, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. Visual Place Recognition: A Survey. *IEEE Transactions on Robotics*, 32(1):1–19, February 2016. ISSN 1941-0468. doi: 10.1109/TRO.2015.2496823. URL https://ieeexplore.ieee.org/abstract/document/7339473. Conference Name: IEEE Transactions on Robotics.

Carlo Masone and Barbara Caputo. A Survey on Deep Visual Place Recognition. *IEEE Access*, 9:19516–19547, 2021. ISSN 2169-3536. doi: 10.1109/ACCESS.2021.3054937. URL https://ieeexplore.ieee.org/abstract/document/9336674. Conference Name: IEEE Access.

Gabriele Moreno Berton, Valerio Paolicelli, Carlo Masone, and Barbara Caputo. Adaptive-Attentive Geolocalization from few queries: a hybrid approach. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 2917–2926, January 2021. doi: 10.1109/WACV48630.2021.00296. URL https://ieeexplore.ieee.org/document/9423382. ISSN: 2642-9381.

Arsalan Mousavian and Jana Kosecka. Deep Convolutional Features for Image Based Retrieval and Scene Categorization, September 2015. URL http://arxiv.org/abs/1509.06033. arXiv:1509.06033 [cs].

Aude Oliva and Antonio Torralba. Building the gist of a scene: the role of global image features in recognition. In S. Martinez-Conde, S. L. Macknik, L. M. Martinez, J. M. Alonso, and P. U. Tse, editors, *Progress in Brain Research*, volume 155 of *Visual Perception*, pages 23–36. Elsevier, January 2006. doi: 10.1016/S0079-6123(06)55002-2. URL https://www.sciencedirect.com/science/article/pii/S0079612306550022.

Eng-Jon Ong, Sameed Husain, and Miroslaw Bober. Siamese Network of Deep Fisher-Vector Descriptors for Image Retrieval, February 2017. URL http://arxiv.org/abs/1702.00338. arXiv:1702.00338 [cs].

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, February 2023. URL http://arxiv.org/abs/2304.07193. arXiv:2304.07193 [cs].

Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010. ISSN 1558-2191. doi: 10.1109/TKDE.2009.191. URL https://ieeexplore.ieee.org/document/5288526. Conference Name: IEEE Transactions on Knowledge and Data Engineering.

Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples, September 2016. URL http://arxiv.org/abs/1604.02426. arXiv:1604.02426 [cs].

Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning CNN Image Retrieval with No Human Annotation, July 2018. URL http://arxiv.org/abs/1711.02512. arXiv:1711.02512 [cs].

Jerome Revaud, Jon Almazan, Rafael Rezende, and Cesar De Souza. Learning With Average Precision: Training Image Retrieval With a Listwise Loss. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5106–5115, Seoul, Korea (South), October 2019a. IEEE. ISBN 978-1-72814-803-8. doi: 10.1109/ICCV.2019.00521. URL https://ieeexplore.ieee.org/document/9010047/.

Jerome Revaud, Philippe Weinzaepfel, César De Souza, Noe Pion, Gabriela Csurka, Yohann Cabon, and Martin Humenberger. R2D2: Repeatable and Reliable Detector and Descriptor, June 2019b. URL http://arxiv.org/abs/1906.06195. arXiv:1906.06195 [cs].

Raghavender Sahdev and John Tsotsos. Indoor Place Recognition System for Localization of Mobile Robots. pages 53–60, June 2016. doi: 10.1109/CRV.2016.38.

Sivic and Zisserman. Video Google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2, Nice, France, 2003. IEEE. ISBN 978-0-7695-1950-0. doi: 10.1109/ICCV.2003.1238663. URL http://ieeexplore.ieee.org/document/1238663/.

Baochen Sun and Kate Saenko. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. In Gang Hua and Hervé Jégou, editors, *Computer Vision – ECCV 2016 Workshops*, Lecture Notes in Computer Science, pages 443–450, Cham, 2016. Springer International Publishing. ISBN 978-3-319-49409-8. doi: 10.1007/978-3-319-49409-8_35.

Baochen Sun, Jiashi Feng, and Kate Saenko. Return of Frustratingly Easy Domain Adaptation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1), March 2016. ISSN 2374-3468. doi: 10.1609/aaai.v30i1.10306. URL https://ojs.aaai.org/index.php/AAAI/article/view/10306. Number: 1.

Xun Sun, Yuanfan Xie, Pei Luo, and Liang Wang. A Dataset for Benchmarking Image-Based Localization. pages 7436–7444, 2017. URL https://openaccess.thecvf.com/content_cvpr_2017/html/Sun_A_Dataset_for_CVPR_2017_paper.html.

Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor Visual Localization with Dense Matching and View Synthesis, April 2018. URL http://arxiv.org/abs/1803.10368. arXiv:1803.10368 [cs].

Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations, February 2016. URL http://arxiv.org/abs/1511.05879. arXiv:1511.05879 [cs].

Akihiko Torii, Josef Sivic, Toma Pajdla, and Masatoshi Okutomi. Visual Place Recognition with Repetitive Structures. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 883–890, Portland, OR, USA, June 2013. IEEE. ISBN 978-0-7695-4989-7. doi: 10.1109/CVPR.2013.119. URL http://ieeexplore.ieee.org/document/6618963/.

Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 Place Recognition by View Synthesis. 2015.

Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, 2000. ISBN 978-1-4419-3160-3 978-1-4757-3264-1. doi: 10.1007/978-1-4757-3264-1. URL http://link.springer.com/10.1007/978-1-4757-3264-1.

Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. CosFace: Large Margin Cosine Loss for Deep Face Recognition. pages 5265–5274, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Wang_CosFace_Large_Margin_CVPR_2018_paper.html.

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 *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2623–2632, Seattle, WA, USA, June 2020. IEEE. ISBN 978-1-72817-168-5. doi: 10.1109/CVPR42600.2020.00270. URL https://ieeexplore.ieee.org/document/9156836/.

Michael Warren, D. McKinnon, H. He, and Ben Upcroft. Unaided stereo vision based pose estimation. In Gordon Wyeth and Ben Upcroft, editors, *Australasian Conference on Robotics and Automation*, Brisbane, 2010. Australian Robotics and Automation Association. URL http://eprints.qut.edu.au/39881/.

Shuhei Yokoo, Kohei Ozaki, Edgar Simo-Serra, and Satoshi Iizuka. Two-stage Discriminative Re-ranking for Large-scale Landmark Retrieval, March 2020. URL http://arxiv.org/abs/2003.11211. arXiv:2003.11211 [cs].

Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. Supervised Representation Learning: Transfer Learning with Deep Autoencoders. *In AAAI International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.