POLITECNICO DI TORINO

Master's Degree in
Mathematical Engineering

Master's Degree Thesis

Implicit Neural Representation in Visual Place Recognition

Relatori
prof. Carlo Masone

Candidato
Francesco Gaza

Academic Year 2024-2025

Acknowledgements

Summary

Visual Place Recognition is the task of identifying a previously visited location from a new image using visual cues only.
In past years, the main approach was image retrieval, which consists of comparing an image's global descriptor, computed by a deep neural network, and finding the most similar images in a database. It is clear that this method has problems of robustness and dimensionality. Indeed, it requires that the query images not be too different from the database images, and as the region to be learned grows, it demands increasing memory and computation time for the search.
In recent years, some methods have been developed to overcome these limitations. This thesis proposes a new method based on the concept of Implicit Neural Regression. The idea is that since the global descriptor produced by the deep neural network can distinguish images well, we can attempt to create an implicit map in the latent space in which similar images are close and dissimilar ones are far apart and infer the position of a new, unseen image solely from its position in the latent space.
If this is possible, we could obtain a very fast and lightweight method: the position of a query image could be obtained in a single forward pass, and the required memory would be limited to the weights of the network.

Contents

Chapter 1

Introduction

1.1 Overview

This thesis will focus on the Visual Place Recognition task. This problem belongs to the field of Computer Vision.

Computer Vision is the branch of Artificial Intelligence that aims to learn how to extract all information obtainable from visual inputs such as images or videos. It is currently one of the most important areas in the scientific community because of its wide range of applications.

Computer Vision employs deep learning methods to train models. Deep Learning is a subfield of Machine Learning in which the networks involved have many layers of neurons, hence the name. This yields more powerful learning capabilities, as it permits the recognition of more complex patterns and therefore greater adaptability. In fact, it has been theoretically proven that deep neural networks are universal approximators; i.e., for every function there exists a network that can approximate it.

Thus, the Visual Place Recognition task uses deep learning models to recognize locations, often within a city or a neighbourhood. This task differs from other related tasks in Computer Vision, such as scene recognition or scene regression, in which the learned area is smaller but greater precision is required in both the camera position and its orientation. In the VPR task, the objective is for the network to recognize and consider as distinct locations those that are separated by a distance that typically is 25 m. However, this is not a simple task due to many factors that will be discussed later.

1.2 Objectives

The objective of this thesis is the study and evaluation of the feasibility and use of Implicit Neural Representations in the Visual Place Recognition task. In particular, the focus is to evaluate whether this feasibility can be achieved with a simple pipeline because this would highlight the clear relationship between positions in latent space and in real space, which is the idea underlying the method. This thesis aims to verify whether the vector representation is sufficiently expressive to define a smooth function over the latent space that permits associating every point with a set of coordinates in real space.

This would lead to a much lighter and faster method, since the position of a location could be obtained in a single forward pass through the network. It could open new application fields because the area of interest could be easily expanded, quite the opposite of the current situation, where dimensionality is a major challenge. Moreover, computation time would be greatly reduced, which could enable the application of this method to real-time tasks.

Chapter 2

Previous Work

This section provides an overview of the previous methods used in the VPR task. Historically, it has always been approached as an Image Retrieval problem, but in recent years some alternative methods have been developed.

2.1 Visual Place Recognition as Image Retrieval

2.1.1 Method

Visual Place Recognition is the task of determining whether a place has been seen before. It is commonly addressed as an image retrieval problem: given an unseen image to be localized (the query), it is matched against a database of geo-tagged images representing the known world. The top-K retrieved images, along with their geo-tags (typically GPS coordinates), provide hypotheses for the geographical location of the query.

The idea behind this formulation is to extract a vector from a deep neural network that represents the image and compare it with the vectors of all images in the database. The task is considered successful if at least one of the K retrieved images lies within a certain distance from the query location. Typically, this distance is set to 25 meters, although it is not a strict constraint and may vary depending on the specific task.

Typically, the vector is obtained through a feature extraction backbone equipped with a head that performs some form of aggregation or pooling. The backbone is usually a large pre-trained network that was not originally designed for this specific computer vision task and is then fine-tuned for it. The most common and historically popular backbone was ResNet[1], while the most widely used today is

DINOv2[2].

Various loss functions are used for training, most of them sharing the characteristic of evaluating two or more images simultaneously to help the network learn a better representation. They are called contrastive losses, and one example is the Triplet Loss:

$$\mathcal{L}(a, p, n) = \{max(dist(a, p) - dist(a, n) + margin), 0)\}$$

The objective of this type of loss functions is to help the network separate the representations of different images: they aim to have a large distance between the anchor and the negative, while maintaining a small distance between the anchor and positives depicting the same location. For this reason, the database is sometimes searched to obtain so-called informative batches, groups of images that contain particularly hard negatives and positives, in order to help the network learn a better representation.

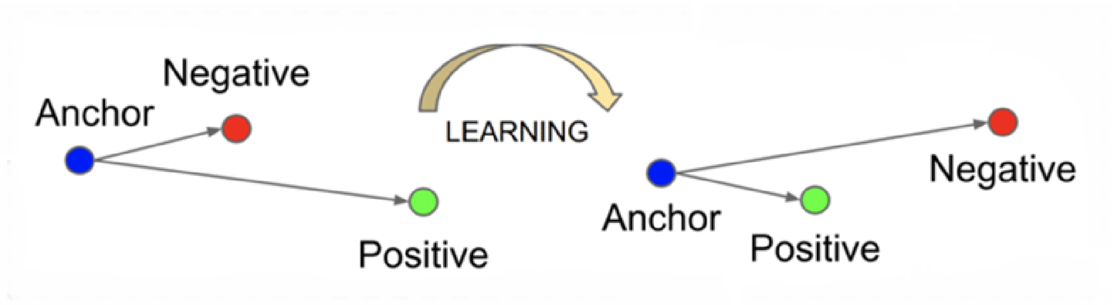


Figure 2.1: This is a visual example of the aim of a triplet loss. As we can see the vector representation is pushed close to the positive and, at the same time, is pushed away from the negative.[3]

In figure 2.2 the generic pipeline for Visual Place Recognition is shown when solved as an image retrieval problem. As we can see, the images are transformed into their vector representations by the model; a similarity search is then performed to find the most similar images in the dataset (here the top-3), and if one of those represents a location near the true one, the task is considered successful.

2.1.2 History

In this section, we provide a brief overview of the principal steps of the VPR architecture. We will not delve deeply into the past since at the beginning this task was formulated differently.

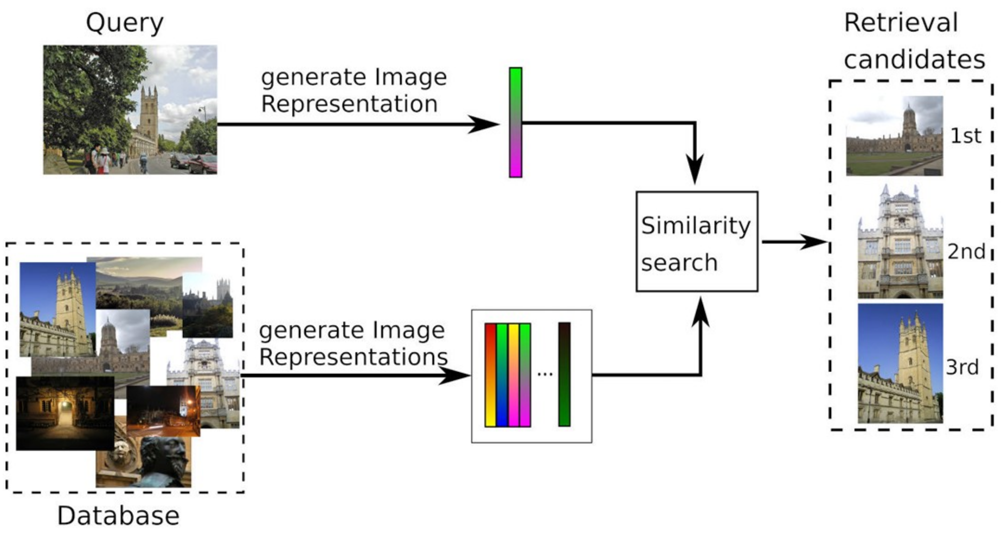


Figure 2.2: This is a generic pipeline for a simple VPR problem solved as an image retrieval.[4]

The first step toward the current formulation was the aggregation of local features to obtain a 1-vector description of the image. The idea was to use the extracted local feature vectors, map them into a higher-dimensional space, and then aggregate them to obtain a single description of the image; the first method to do this was VLAD[5]. It represented a major advance in performance since it produced a much better vector representation of the image.

The next step came with the introduction of deep learning, in particular Convolutional Neural Networks (CNNs). Their structure allows for the obtain information on both smaller and larger scales in an image. In addition, they could be used off-the-shelf, as researchers observed that they performed well even when trained on classification tasks in completely different domains. However, the results were not dramatically better, so researchers concluded that it was necessary to fine-tune the networks.

A further milestone was the combination of a CNN with an aggregation layer. This was the birth of NetVLAD[6], one of the most important methods in VPR, which was used for several years.

The most recent step came with the invention of Transformers. This architecture has profoundly changed machine learning in many fields. Transformers are important in computer vision because, due to the self-attention mechanism, they can relate every pixel or patch in an image to all others, and they can do this within a single layer. Alongside these principal features, they possess other characteristics that make the architecture highly relevant to computer vision and enable a significant improvement in performance. For this reason, the current state of the art in VPR is held by architectures that implement the Transformers in their backbones, are trained on very large and diverse datasets and then fine-tuned for the specific task with complex heads. As an example, the MiX-VPR[7] paper uses DINOv2[2] as a backbone and a complex pipeline to obtain impressive results, with recall@1 above 92% on essentially all major evaluation datasets.

2.1.3 Mathematical Formulation of Image Retrieval Method

In the image retrieval method the objective is to retrieve the exact location of where a photo was taken by comparing it with a labelled database of reference and finding the K most similar ones. In order to this, the first step is to map every photo in the latent space. So we will need a function f such that:

$$f : R^{H \cdot W \cdot 3} \rightarrow R^D$$

This function is obtained through a deep learning model. We can make explicit the dependence of the function from the learnable parameters of the model by writing $f := f_\Theta$ Actually, at the end of the net there is always a normalization layer, to help the distance evaluation. Often, this normalization projects the vector on the hypersphere of unitary radius, so:

$$f_\Theta : R^{H \cdot W \cdot 3} \rightarrow S^{D-1} := \{\boldsymbol{v} \in R^D \mid \|\boldsymbol{v}\|_2 = 1\}$$

Now that the base of the task is defined, we can look at the image retrieval step. We start from a database $\mathcal{D}$ of images called $\mathcal{I}$, in which every image I has a labeled position $P \in \mathcal{P}$:

$$\mathcal{D} := \{(I_1, P_1), (I_2, P_2), ..., (I_N, P_N)\} \subseteq \{(I, P) \mid I \in \mathcal{I}, P \in \mathcal{P}\}$$

For speed reasoning, in some tasks it is useful to evaluate all the database images only once at the beginning and store the vector representation of the images. So we would have:

$$\mathcal{D} := \{(f_\Theta(I_1), P_1), (f_\Theta(I_2), P_2), ..., (f_\Theta(I_N), P_N)\} \subseteq \{(f_\Theta(I), P) \mid f_\Theta(I) \in \mathcal{S}^{D-1}, P \in \mathcal{P}\}$$

When an image query $Q \in \mathcal{Q}$ is evaluated, we get $f_\Theta(Q)$ and perform a KNN search among the dataset to find the K most similar vector representation. With the hypothesis of normalization, the similarity search is conducted using a vector product between the query and each element in the database:

$$d_j = 1 - f_\Theta(Q)^t f_\Theta(I_j), \forall j \in [1, N]$$

When the K most similar vectors are found, they are put in order and their location represents the actual prevision for the query one.

$$\text{Hypothesis set} = \mathcal{H} = \{P_{(1)}, P_{(1)}, ..., P_{(K)}\}$$

And if one of the labels in the hypothesis set is 25 m or closer to the real position of the query image, then the prediction will count as correct.

2.1.4 Limitation

This framework has some limitations due to factors such as size, image variability, image noise, image scale, data sparsity, ambiguity in geo-localization information, and computational complexity. These factors reduce the current feasible application of the methods. Beyond dataset problems, which are not the goals of this thesis, we still have size and variability issues, the latter of which can be attributed to a lack of generalization ability.

Size

Regarding size, the crucial points are the memory and the time required. Memory is proportional to the number of images, and since several photos are needed to learn a new portion of an urban landscape, growth is very rapid and the task is often limited to learning a city or part of it.
Regarding time, the bottleneck is the KNN search. Since it scales as $\mathcal{O}(N)$, this makes real-time evaluation infeasible for large datasets and poses a severe limitation to application areas such as real-time tasks. In recent years, two main models have been proposed that try to overcome these limitations. Both attempt to extend the task to a global scale by using regression methods to scale up. There is no information about their computational time or whether they could be used in real-time tasks. However, we do not treat them here, as they are described in the next sections.

Ability of generalization

The lack of generalization is intrinsic to the framework: since the position of a new image is assessed by comparison with a database of images, the query image must be similar to a database image, or the prediction will be incorrect. Some causes are due to poor datasets, but others are intrinsic to the task.

All possible variations at a location represent a difficult challenge for the network. For example, lighting conditions are the most challenging when dealing with images taken at night. Another aspect is the presence of occlusions: the network must be capable of ignoring temporary objects, but sometimes these are too large and make correct recognition very difficult. Two aspects that might go unobserved are the weather or the season, which can make a great difference.

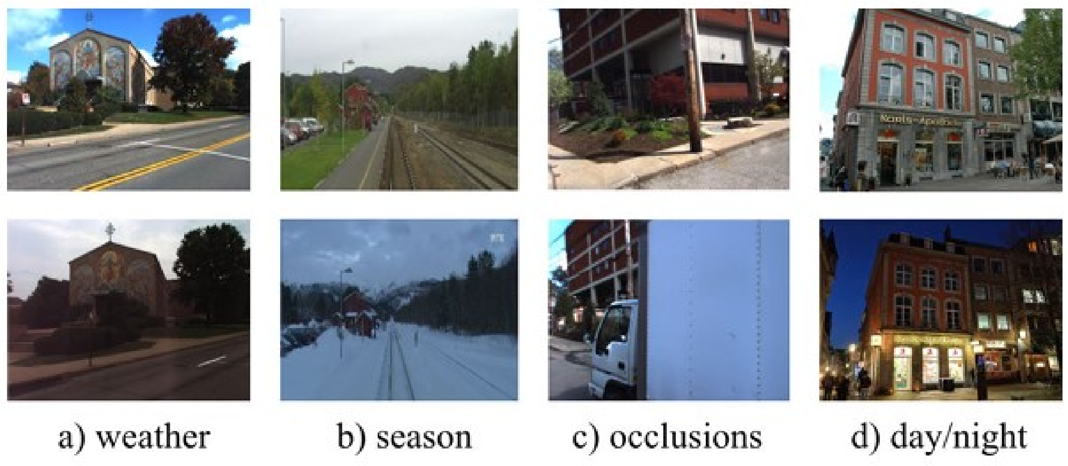


Figure 2.3: This is an example on how much can results different a photo taken in the same place but a different conditions.[4]

The point of view is also critical. If all training images are taken from Google Street View, the presence in the test database of photos taken with smartphones can be problematic. Differences in camera height and focal length can make the same building appear different.
These are only some examples of the visual challenges that illustrate how difficult the task is in practice.

Another important aspect is co-visibility. Co-visibility refers to the extent to which two different photos share the same objects. This is a further challenge because, in suburban or rural areas, two different locations (i.e., 25 m apart) may share essentially the same elements and give rise to false positives that are difficult to handle.

Over the years, increasingly complex models and pipelines have been developed to overcome these problems; with the introduction of Transformers, these advances

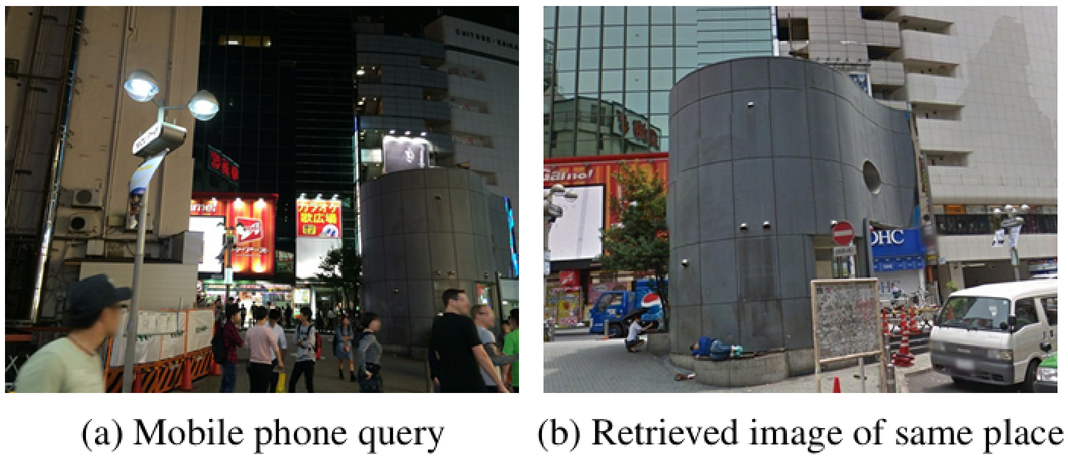


Figure 2.4: This is an example on how much can results different a photo of the same location, but taken with a mobile device.[6]

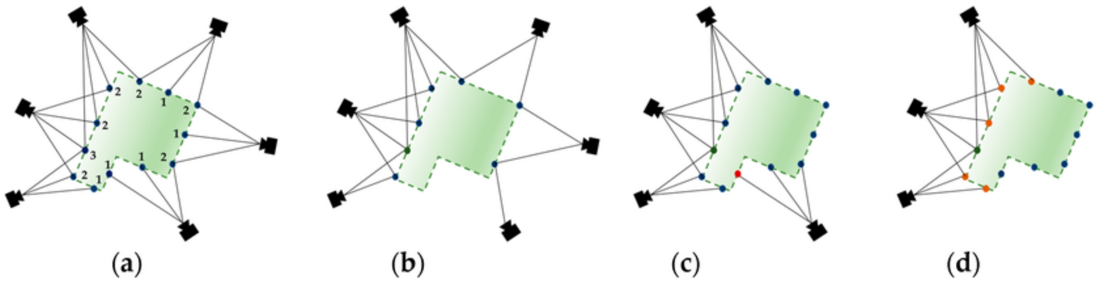


Figure 2.5: This is a visual clue on how co-visibility works. Figure (a) shows for every point how many camera can see it. Figure (b) shows only the point seen by at least two cameras. Figure (c) shows that co-visibility does not means spatial continuity, i.e. the red and the green points are close but are not co-visible. Figure (d) shows the graph after applying the co-visibility filter for the green points. The orange points depict all the co-visible points of the green points.[8]

have led to the current state of the art, which exhibits remarkable generalization ability.

Datasets Problems

The dataset problems are still worth mentioning, even though they are not treated in this thesis, because they are useful for understanding the task. The problem is that good results require good data. The presence of frequent large occlusions can worsen the learning. Even the sampling density does not have a unique effect;

in fact, it is obvious that a higher sampling density in the dataset leads to better results, but this conflicts with the size problem and a compromise must be made based on the specific task. Finally, approximate GPS labelling is very restrictive for achieving good results, since without reliable ground truth the task becomes more difficult.

2.2 Implicit Neural Representation

Implicit Neural Representation (INR) is a method that aims to learn a signal, of various types, as a continuous function through a neural network. The idea is that instead of storing every couple of coordinates and their value, we learn the function that approximates the signal. This work can be done effectively by the Multi Layer Perceptron (MLP) and has several advantages.

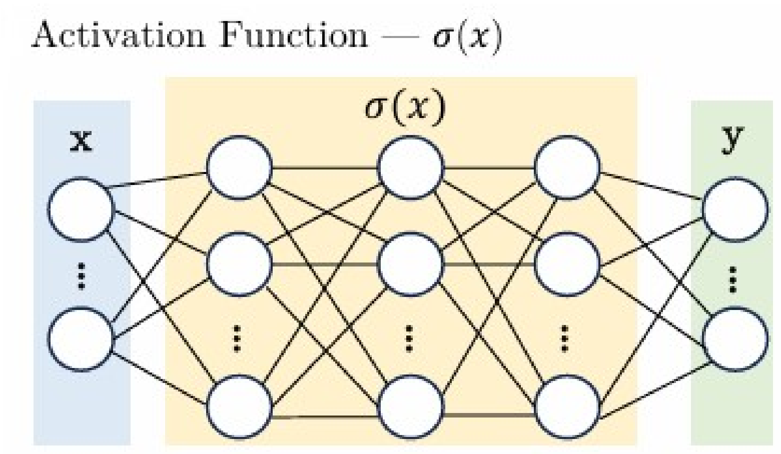


Figure 2.6: This is a generic structure of an MLP. The input is x, then there are the hidden layer linked by a non-linear function, and, at the end, there is the output y.[9]

2.2.1 Mathematical Formulation

In the INR problem, we want to find a continuous representation of the data. So we want to map every coordinate in input $x \in R^{D_{in}}$ to a specific value $f(x) \in R^{D_{out}}$. To do this, we want to learn a function that depends on x, $f(x)$ and possibly on its derivatives such that:

$$F(x, f(x), \nabla_x f(x), \nabla_x^2 f(x), ...) = 0$$

Since we use an MLP, the output of every layer would be:

$$y_i = \sigma(W_i y_{i-1} + b_i),$$

where σ is the activation function, W_i are the weights, and b_i the bias of the i-th layer.

2.2.2 Advantages

First of all, it is much lighter to store only the weight of the network, especially when the data become larger. The second key aspect is that the weight does not depend on the resolution of the signal, so there is not a greater usage of memory when increasing the resolution. Moreover, it is faster to perform a forward pass and obtain a couple of coordinates-values, instead of performing a search through the dataset. Another point is given by the complete differentiability of the model, which permits a good adaptability on the basis of the specific task.

2.2.3 Limitation

The principal limitations are given from the actual learnable functions. The method has some implicit issues in learning high-frequency functions. But to overcome this problem, different solutions have been created in the years, such as, for example, from the positional encoding to different activation function, until a complete complex structure of all whole network to obtain the best possible results. We will not go deeper into the solution since this thesis uses only the basic formulation of the model.

2.3 Implicit Neural Representation in VPR

The application of INR in the VPR task is almost a novelty. Only two methods were found that attempted to obtain a continuous function. Both focus on a worldwide scale, so they are not a useful comparison since the intrinsic difference in the problem.

GeoCLIP

The first model, GeoCLIP[10], introduces a new approach to the problem: instead of only matching images, it learns a representation of the GPS coordinates in the latent space.

To achieve this, the model starts from a backbone, in this case CLIP[11], which is kept frozen, and adds two linear layers of decreasing dimensions to fine-tune the

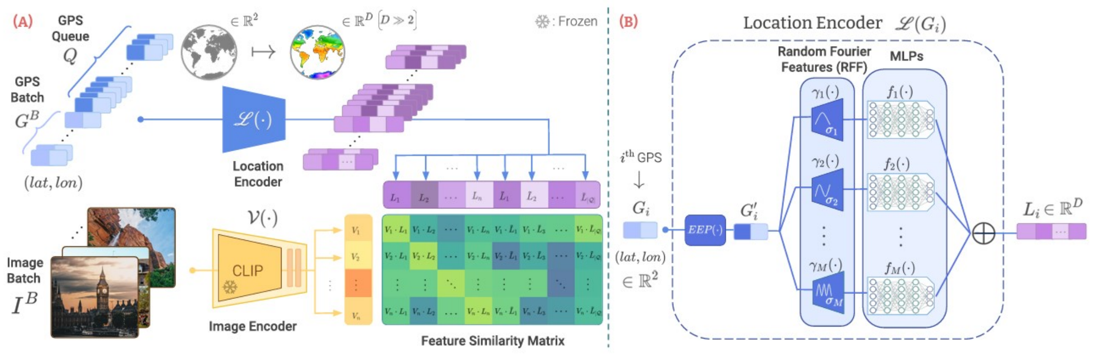


Figure 2.7: This is the pipeline for the GeoCLIP method.[10]

representation for the task.

Regarding GPS data, a sequence of operations is applied to obtain a suitable representation. First, the coordinates are transformed to avoid distortion problems (for example, near the poles). Then, three different Random Fourier Features are applied to better capture information at different scales. Each resulting vector is then processed through a separate MLP with an output dimension of 512, and finally, the outputs are summed together to form the GPS representation in the latent space.

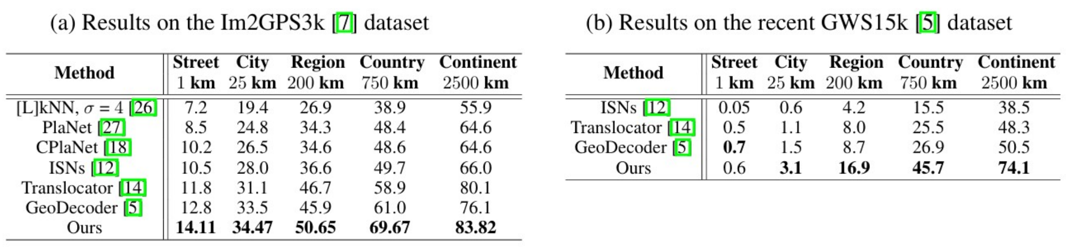
For training, the paper adopts a contrastive learning approach, comparing P different views of the same image with various augmentation transformations. Gaussian noise is also added to the coordinates to obtain a smoother function, and a fixed-length queue of negative samples is introduced, which appears to improve learning, especially at smaller scales.

The results are strong: at the time of publication, GeoCLIP[10] achieved the best performance among previous state-of-the-art models across nearly all scales and datasets.

ReGeo

ReGeo[12] is a recent paper and is more oriented toward a pure regression task. The authors did not provide a complete explanation of the model and did not release the code for ethical reasons related to potential misuse.

The model consists of a backbone, a location head, and a regression head.



(a) Results on the Im2GPS3k [7] dataset

Method	Street 1 km	City 25 km	Region 200 km	Country 750 km	Continent 2500 km
[L]kNN, $\sigma = 4$ [26]	7.2	19.4	26.9	38.9	55.9
PlaNet [27]	8.5	24.8	34.3	48.4	64.6
CPlaNet [18]	10.2	26.5	34.6	48.6	64.6
ISNs [12]	10.5	28.0	36.6	49.7	66.0
Translocator [14]	11.8	31.1	46.7	58.9	80.1
GeoDecoder [5]	12.8	33.5	45.9	61.0	76.1
Ours	**14.11**	**34.47**	**50.65**	**69.67**	**83.82**

(b) Results on the recent GWS15k [5] dataset

Method	Street 1 km	City 25 km	Region 200 km	Country 750 km	Continent 2500 km
ISNs [12]	0.05	0.6	4.2	15.5	38.5
Translocator [14]	0.5	1.1	8.0	25.5	48.3
GeoDecoder [5]	**0.7**	1.5	8.7	26.9	50.5
Ours	0.6	**3.1**	**16.9**	**45.7**	**74.1**

Figure 2.8: This is the comparison between the main old methods for Geo Localization and GeoCLIP on the main dataset at different scale.[10]

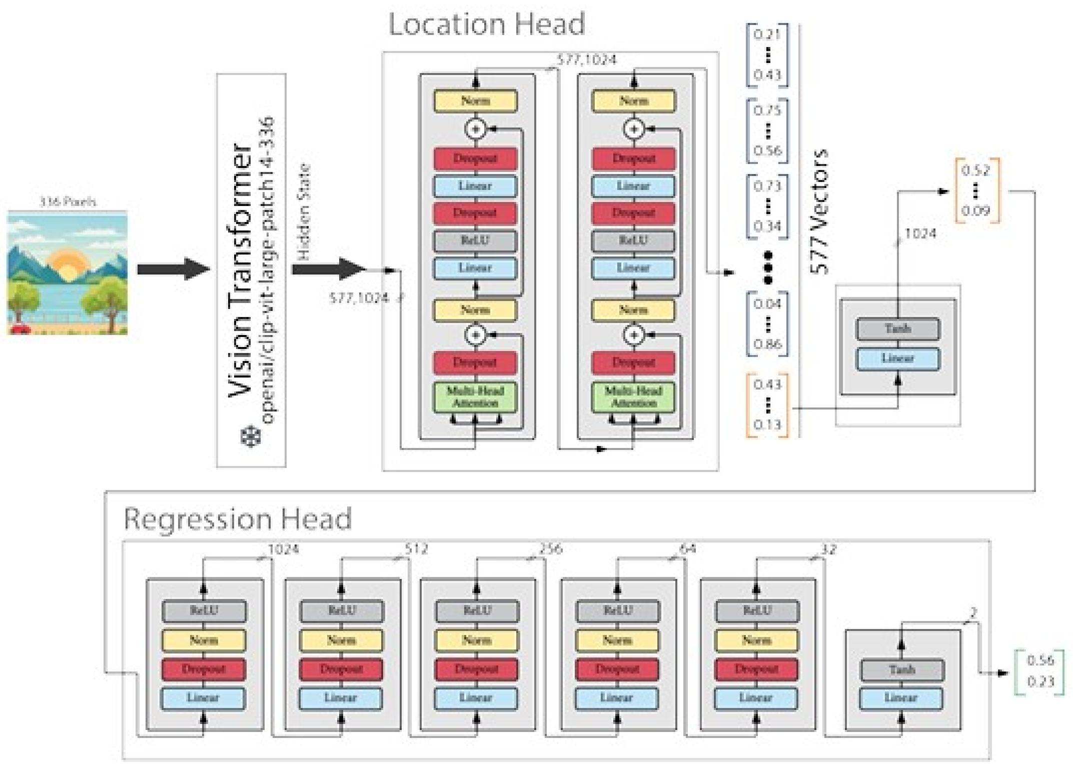


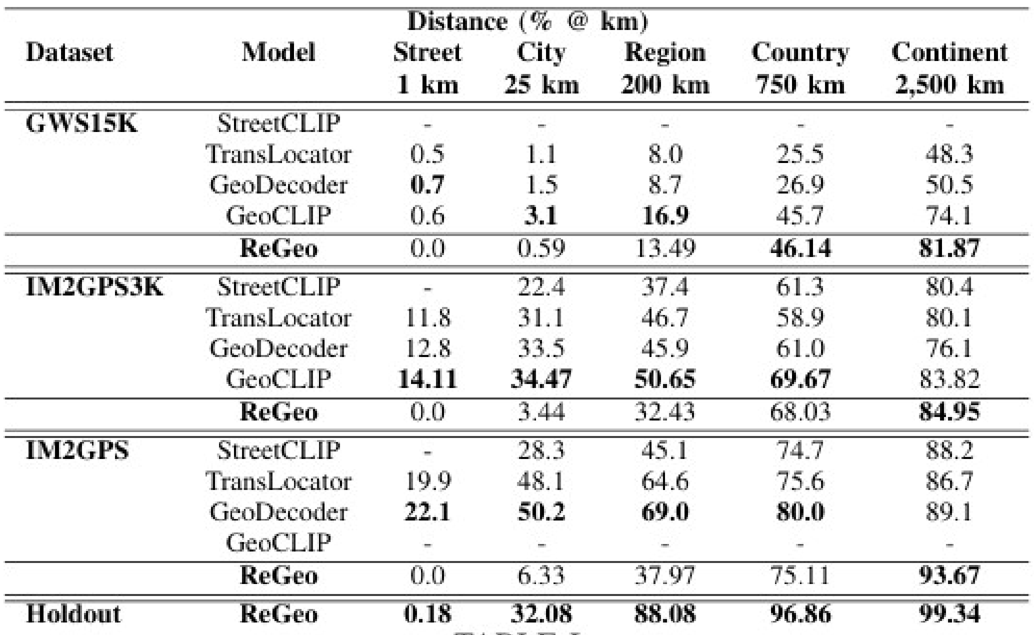
Figure 2.9: This is the pipeline for the ReGeo method.[12]

Because they use CLIP[11] as a frozen backbone, which is optimized for image–text alignment, they attach a location head. The location head comprises two additional transformer encoder layers, each containing a multi-head attention block. To complete the location head, the output passes through a linear layer with a tanh activation.

The regression head is formed by a series of identical blocks of decreasing dimension, each consisting of a linear layer, a dropout layer, a normalization layer, and an activation layer. Starting from an input dimension of 1024, the dimension is halved until it reaches 32; after this, a linear layer produces an output of dimension 2, followed by a tanh to guaranty outputs in $[-1,1]$. The two output elements are then multiplied, respectively, by 90 and 180 to obtain the correct ranges for latitude and longitude.

For training, a distance-based loss is applied, computed using the Haversine distance to obtain a more reliable estimation. To improve results, the localization head is pre-trained using a Siamese network structure. The images are also subjected to data augmentations, and Gaussian noise is added to the coordinates.

The results are strong: despite the simplicity of the method, it achieves better results on the continental scale than the other methods considered. At finer scales, it suffers from the simpler pipeline, but it is certainly a good starting point.



Dataset	Model	Distance (% @ km) Street 1 km	City 25 km	Region 200 km	Country 750 km	Continent 2,500 km
GWS15K	StreetCLIP	-	-	-	-	-
	TransLocator	0.5	1.1	8.0	25.5	48.3
	GeoDecoder	**0.7**	1.5	8.7	26.9	50.5
	GeoCLIP	0.6	**3.1**	**16.9**	45.7	74.1
	ReGeo	0.0	0.59	13.49	**46.14**	**81.87**
IM2GPS3K	StreetCLIP	-	22.4	37.4	61.3	80.4
	TransLocator	11.8	31.1	46.7	58.9	80.1
	GeoDecoder	12.8	33.5	45.9	61.0	76.1
	GeoCLIP	**14.11**	**34.47**	**50.65**	**69.67**	83.82
	ReGeo	0.0	3.44	32.43	68.03	**84.95**
IM2GPS	StreetCLIP	-	28.3	45.1	74.7	88.2
	TransLocator	19.9	48.1	64.6	75.6	86.7
	GeoDecoder	**22.1**	**50.2**	**69.0**	**80.0**	89.1
	GeoCLIP	-	-	-	-	-
	ReGeo	0.0	6.33	37.97	75.11	**93.67**
Holdout	**ReGeo**	**0.18**	**32.08**	**88.08**	**96.86**	**99.34**

TABLE I
REGEO VS EXISITING MODELS ON DIFFERENT BENCHMARKS

Figure 2.10: This is the comparison between ReGeo and other Geo Localization model across various datasets.[12]

2.3.1 Relevant Difference

Behind the scale difference there is another key aspect that regards these two models and the objective of this thesis. This is that: they want to recognize an already seen location, whether our actual objective is to infer the position of a never seen one, by implicitly, in the INR, comparing the vector representation of the query image and of the near locations. This is a small but very big difference that makes the task much more difficult.

Chapter 3

Proposed Approach

3.1 Method

The objective of this thesis is to study the feasibility of a new approach consisting of applying the concept of Implicit Neural Representation to the Visual Place Recognition task. Today, VPR methods can create very good representations of images in latent space, such that recall percentages are above 90% for most datasets. For this reason, the underlying idea is that images depicting nearby locations will also be close in latent space. This suggests that it may be possible to obtain a representative mapping of the real positions of images within the latent space. With this hypothesis, we hypothesized that a Multi-Layer Perceptron (MLP) applied after the encoding phase could capture the relationships between images in latent space and translate them into positions in real space.

This thesis tests two different methods. The first, which is the full application of the idea, is the regression method: here, we attempt to predict the coordinates and orientation of a photo from its vector description in latent space. The second method is classification, in which the area of interest is divided into cells and each photo is assigned to the appropriate cell. This second method is a simplification of the first one ad aims to provide a clearer overview of the feasibility of the problem. Then both methods are divided into two main branches in order to explore multiple solutions and select the best one.

An important point of this thesis is how the dataset is used. In VPR as an image-retrieval task, it is common to have query images and database images because it is solved by retrieving the query locations using the ones in the database. In this study, we attempt to create a map and recognize locations solely by their positions in latent space rather than by visual similarity. For this reason, the evaluation dataset in this thesis is not divided into query and database sets, but is used in its entirety to assess the performance of the model.

3.1.1 Dataset

In this thesis, only one dataset has been used: the San Francisco XL dataset[13]. This dataset is composed of Google Street View images from various years that cover the city of San Francisco. The dataset was created by cropping the 360-degree images captured by the Google car into 12 photos. Each photo is oriented at an angle that is a multiple of 30° and there is a small amount of overlap between consecutive images. Each photo is geo-tagged with latitude, longitude, and bearing. The training set contains 41 million images, while the evaluation set consists of 8,000 database images and 8,000 queries. A novelty introduced by this dataset is the overlap between the training and evaluation sets. This reflects the assumption that, in VPR, the network should be familiar with the area to be evaluated; therefore, it is preferable that the training and evaluation sets overlap rather than be merely adjacent.

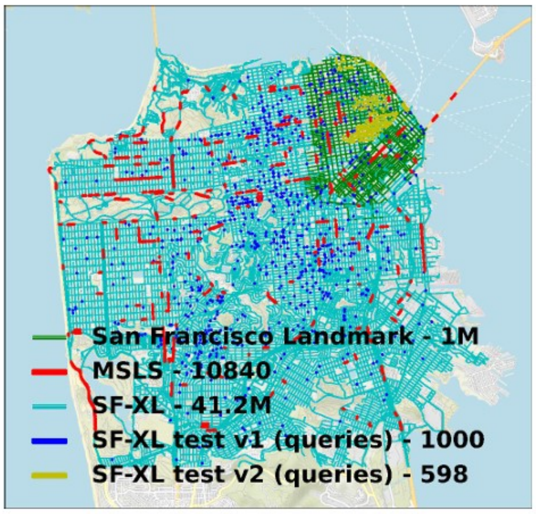
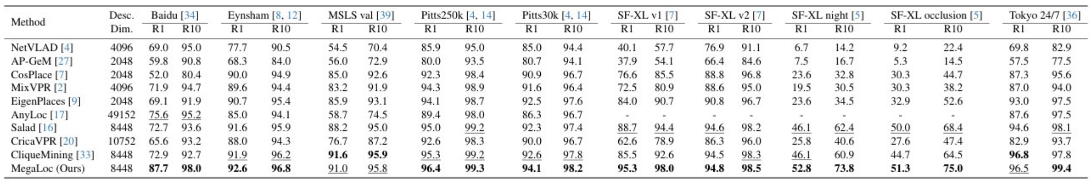

San Francisco Landmark - 1M
MSLS - 10840
SF-XL - 41.2M
SF-XL test v1 (queries) - 1000
SF-XL test v2 (queries) - 598

Figure 3.1: The image shows the coverage of the city of the SF-XL dataset.[13]

3.1.2 Backbone

For the backbone, we used two different models that have proven their ability in the VPR task, one bigger, MegaLoc[14], and the other smaller, but trained on this specific dataset, CosPlace[13]. A good backbone is essential for the task since the whole idea is founded on the ability of the network to create a good representation of the images in the latent space. The results show that the Megaloc[14] is far superior with a recall@1 above 95%, while CosPlace[13] is only around 77%. The reason for the use of CosPlace is that since it is a smaller method, it takes much less time to complete the experiments. This was an important point for the thesis, since this being a completely new method, several experiments were done.



Method	Desc.	Baidu [34]		Eynsham [8, 12]		MSLS val [39]		Pitts250k [4, 14]		Pitts30k [4, 14]		SF-XL v1 [7]		SF-XL v2 [7]		SF-XL night [5]		SF-XL occlusion [5]		Tokyo 24/7 [36]	
	Dim.	R1	R10	R1	R10	R1	R10	R1	R10	R1	R10	R1	R10	R1	R10	R1	R10	R1	R10	R1	R10
NetVLAD [4]	4096	69.0	95.0	77.7	90.5	54.5	70.4	85.9	95.0	85.0	94.4	40.1	57.7	76.9	91.1	6.7	14.2	9.2	22.4	69.8	82.9
AP-GeM [27]	2048	59.8	90.8	68.3	84.0	56.0	72.9	80.0	93.5	80.7	94.1	37.9	54.1	66.4	84.6	7.5	16.7	5.3	14.5	57.5	77.5
CosPlace [7]	2048	52.0	80.4	90.0	94.9	85.0	92.6	92.3	98.4	90.9	96.7	76.6	85.5	88.8	96.8	23.6	32.8	30.3	44.7	87.3	95.6
MixVPR [2]	4096	71.9	94.7	89.6	94.4	83.2	91.9	94.3	98.9	91.6	96.4	72.5	80.9	88.6	95.0	19.5	30.5	30.3	38.2	87.0	94.0
EigenPlaces [9]	2048	69.1	91.9	90.7	95.4	85.9	93.1	94.1	98.7	92.5	97.6	84.0	90.7	90.8	96.7	23.6	34.5	32.9	52.6	93.0	97.5
AnyLoc [17]	49152	<u>75.6</u>	<u>95.2</u>	85.0	94.1	58.7	74.5	89.4	98.0	86.3	96.7	-	-	-	-	-	-	-	-	87.6	97.5
Salad [16]	8448	72.7	93.6	91.6	95.9	88.2	95.0	95.0	<u>99.2</u>	92.3	97.4	<u>88.7</u>	<u>94.4</u>	<u>94.6</u>	98.2	<u>46.1</u>	<u>62.4</u>	<u>50.0</u>	<u>68.4</u>	94.6	<u>98.1</u>
CricaVPR [20]	10752	65.6	93.2	88.0	94.3	76.7	87.2	92.6	98.3	90.0	96.7	62.6	78.9	86.3	96.0	25.8	40.6	27.6	47.4	82.9	93.7
CliqueMining [33]	8448	72.9	92.7	<u>91.9</u>	<u>96.2</u>	**91.6**	**95.9**	<u>95.3</u>	<u>99.2</u>	<u>92.6</u>	<u>97.8</u>	85.5	92.6	94.5	<u>98.3</u>	<u>46.1</u>	60.9	44.7	64.5	**96.8**	97.8
MegaLoc (Ours)	8448	**87.7**	**98.0**	**92.6**	**96.8**	<u>91.0</u>	<u>95.8</u>	**96.4**	**99.3**	**94.1**	**98.2**	**95.3**	**98.0**	**94.8**	**98.5**	**52.8**	**73.8**	**51.3**	**75.0**	<u>96.5</u>	**99.4**

Figure 3.2: This table show the comparison of the performance between MegaLoc and CosPlace. If we focus only on the SF-XL dataset, we see how great is the difference in the recall between this two methods.[14]

MegaLoc

MegaLoc is substantially a network consisting of a DINOV2[2] backbone followed by a SALAD[15] aggregation layer. This guaranteed good performance and the peculiarity of the model is that the net was trained on different datasets to give a strong generalization ability to the model. SALAD is an improvement of the NetVLAD[6] aggregation layer in which there is less bias, introduces a dust bin to discard irrelevant features, and get a better distribution of the features.

CosPlace

CosPlace is a NetVLAD[6] network that takes inspiration for the training step from the CosFace[16] face recognition model . The model divides the dataset in 18 groups and then uses a Large Margin Cosine Loss (LMCL) on each, one per epoch, to obtain a good vector representation of the images.

3.1.3 MLP

The Multi-Layer Perceptron (MLP) is a neural network structure consisting of multiple layers of neurons, typically linked by nonlinear activation functions. Because MLPs can discover nonlinear relationships in data, they are useful tools for many tasks, including regression and classification.

The great learning capacity of the MLP requires careful determination of its size: if it is too large, it can easily lead to overfitting and thus to an incorrect

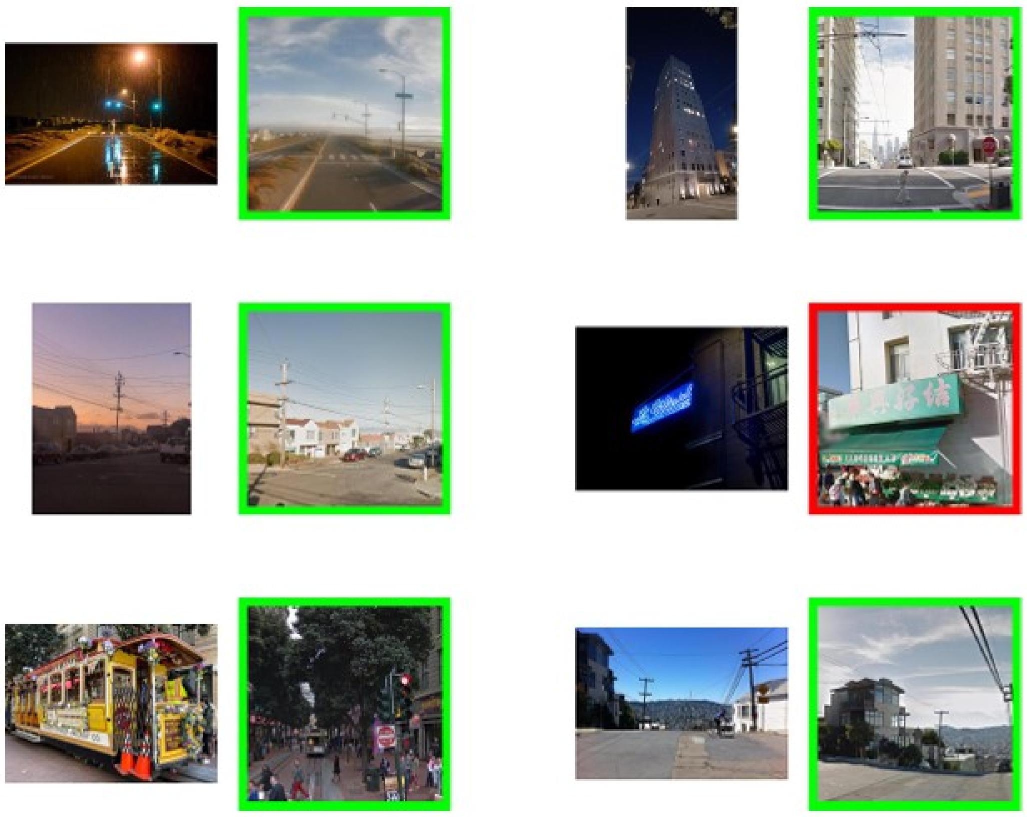


Figure 3.3: The following predictions are derived from the MegaLoc model in the SF-XL dataset, showing how good it is in the VPR task through image retrieval.[14]

model. In this thesis, the MLP is used as a regression head on top of the backbone, and its aim is to capture the relationships between positions in latent space and positions in the real world. The objective of the thesis is to evaluate whether there is a clear connection between latent-space position and real-space position; for this reason, the MLPs used here are relatively shallow and employ basic architectures and activation functions.

A well-known issue with MLPs is the low-frequency bias: it has been observed that MLPs tend to prioritize low-frequency functions, which can make learning high-frequency functions difficult. However, in this work, we adopt a simple pipeline because the hypothesis is based on a smooth function in latent space; accordingly, we verify whether a simple MLP can perform adequately for the task.

An important point to highlight is the activation function used: instead of the

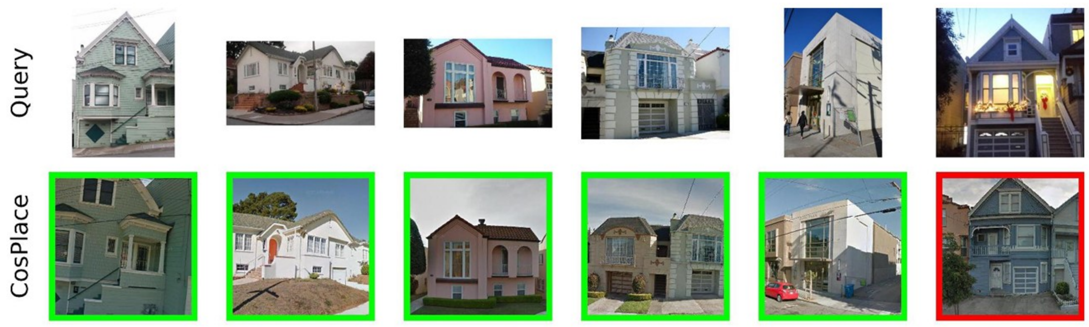


Figure 3.4: The following predictions are derived from the CosPlace model in the SF-XL dataset. This example illustrates the complexity of the task: at first glance, even a human might perceive that the last column depicts the same location.[13]

classical ReLU, performance improved when it was replaced with SiLU (also known as Swish). The main difference is that SiLU handles negative inputs as well and reduces the probability of dead neurons, which are common in deep networks that use ReLU.

Those are the definition of both the activation function:

$$\begin{aligned} relu(x) &= (x)^+ = max(0, x) \\ silu(x) &= x \cdot \sigma(x) && \text{,with } \sigma(x) \text{ the logistic sigmoid} \end{aligned}$$

3.2 Data Preparation

3.2.1 Label

Data preparation is pretty simple in this case. Every image is labeled with a string that contains all the information needed, i.e. latitude, longitude, and bearing. After extraction, there is a phase in which they are transformed to fit the MLP regression or for classification.
In the first case they are mapped to [0,1] in the direct regression method, or transformed to the new set of coordinates in the hypersphere method, which will be explained in details further ahead. It has been calculated that the transformation and anti-transformation needed in the evaluation phase does not affect the values in a significant way, so the original ones are not stored. In addiction to this in the regression case, the variables could be slightly modified with some gaussian noise in order to make smoother the map in the latent space.

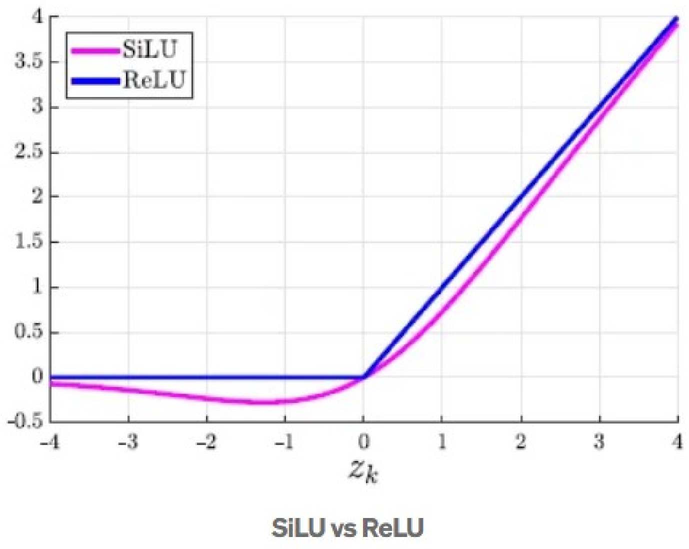


Figure 3.5: The image shows the difference between the activation function ReLU and SiLU. As we can see, the main difference is on the negative value in input for which the SiLU gives a non-zero output

For the classification case, we transform the position into labels with a mapping function. In the rigid method, this is a simple function:

$$\begin{aligned} &c = class\_lat \cdot M \cdot 12 + class\_lon \cdot 12 + class\_bear, \\ &class\_lat, class\_lon \in [0, M-1], \\ &class\_bear \in [0, 11] \end{aligned} \tag{3.1}$$

The class for lat and lon is obtained by mapping the coordinates to [0,1] and then applying the following operation:

$$\begin{aligned} &class\_i = \lfloor norm\_i \cdot M \rfloor \\ &i \in \{lat, lon\} \end{aligned}$$

while for the bearing, it is simpler due to the dataset characteristics:

$$class\_bear = \frac{bear}{30}$$

In the soft method, there is a more complex method that will be explained in detail in its own chapter.

3.2.2 Images

Regarding the images, the preparation is essentially a data augmentation with a simple pipeline. This is formed by a resize step to obtain the correct dimension needed for the DINOv2[2] backbone in MegaLoc[14]. Then there is a random choice, to give more randomness to the transformations, between a color jitter, which is the process of randomly altering an image's brightness, contrast, saturation, and hue, and a gaussian blur, which is the processing technique that smooths an image by averaging pixel values with their neighbors. This is done to help the network to generalize better and get a smoother map function.

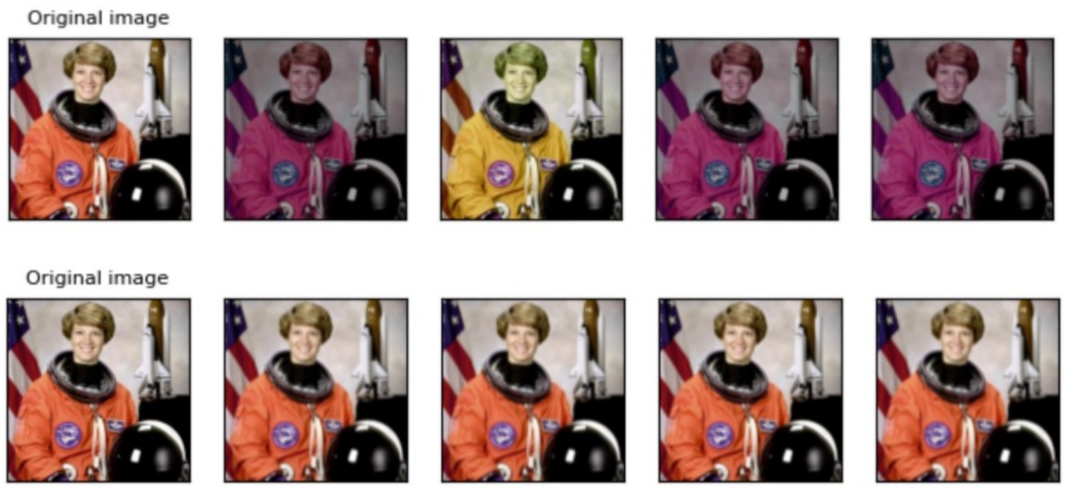


Figure 3.6: Those images show 4 random transformation of Color Jitter (above) and Gaussian Blur (below).

3.3 Regression

In the regression task our aim is to regress the coordinates and bearing of every photo from their position in the latent space. As we said before, our objective is to verify if there is a clear connection between the positions in the latent and real space, and to do this, we use a very simple pipeline. We start from our images and extract the features with two different methods: CosPlace[13] and MegaLoc[14]. After extraction, we get a vector of a certain length that we immediately pass to an MLP. In order to have better performance the output of the MLP is set to be in a $[0, 1]$ or in a $[-1, +1]$ range with a softmax or a L2 normalization depending on the sub-method.

3.3.1 Direct Regression

In the direct regression method, we try to regress the coordinates from the vector representation. To do this, after the extraction we use an MLP that has as output a 3-dimensional vector. After the forward pass through the network there is a re-mapping of the data to get in the correct range for latitude, longitude, and bearing.

3.3.2 Hypersphere Coordinates

The hypersphere method is based on the intrinsic nature of the coordinates. In the common representation of a point in space, latitude and longitude are expressed in degrees. We searched for an alternative to direct regression, which did not seem to work; at that time we thought exploring the angular nature of the coordinates as a different approach. The idea was to use latitude, longitude, and bearing to convert spherical coordinates to Cartesian coordinates.

There was a problem with this representation because it did not consider bearing. This was a mistake: the dataset images are obtained by cropping 360° photos, so there are always at least twelve images with the same latitude and longitude. If bearing is not considered, it will be impossible to obtain an implicit map of the city. This is a crucial point because in the image-retrieval method—unless one also wishes to obtain the bearing of a photo—bearing is not mandatory: it is sufficient to compare the query image with those in the database and assign the coordinates of the most similar one. Thus, if there are twelve or more images with completely different landscapes but labeled with the same geographic point, this is not a problem for retrieval.

The opposite is true for this method, in which we attempt to obtain a smooth map in latent space and then associate each point in that space with a point in real space. In other words, we are trying to define a mean vector representation for every point in space. Thus, having multiple images that depict different landscapes, and therefore have completely different vector representations, is a problem that will severely affect learning.

For this reason, to test the idea, we suggested the use of hyperspherical coordinates. The idea is to treat the bearing as a fourth dimension and convert $(latitude, longitude, bearing)$ to (w, x, y, z). The fundamental purpose was to try a different approach and verify whether it could lead to better results. The main reason was that the data might find a better distribution between the new variables. Alongside this, there were two other, more technical motivations: this change of variables could enable an L2 normalization of every prediction and, by constraining the output, could lead to improved performance. The last point is that the cartesian coordinates are affected by a minor variance along little variation in the position since:

$$\begin{aligned} x &= cos(\theta) \\ \frac{dx}{d\theta} &= -sin(\theta) \\ dx &= -sin(\theta)d\theta \end{aligned} \tag{3.2}$$

but since $|sin(\theta)| \leq 1$, the variation in the cartesian coordinates is slower and this could benefit the learning, since it might give a smoother function. After the change of coordinates there is no big difference with the direct method, here the output of the MLP will be a 4-dimensional vector instead of a 3 one.

Mathematical Formulation

We want to explore the change of coordinates from hyperspherical to cartesian and vice versa. The first point to highlight is the domain of the variables. In fact, for our application, which is limited to San Francisco city, it is very useful to exploit the restricted domain of the variables to simplify future calculation. In particular, we have $lat \simeq 37$, $lon \simeq -122$, and $bearing \in [0, 2\pi]$. So we know that the latitude will always be in the first quadrant, the longitude in the third, while the bearing can vary across all the circle.
Starting from the first, we want to map $(r, lat, lon, bear)$ to (w, x, y, z). We are operating in a unitary hypersphere, so $r = 1$ and we can forget about it. This first operation is simple, since it requires only the application of the formula:

$$\begin{aligned} w &= sin(lat) \\ x &= cos(lat)cos(lon)cos(bear) \\ y &= cos(lat)cos(lon)sin(bear) \\ z &= cos(lat)sin(lon) \end{aligned} \tag{3.3}$$

Calculations are a bit more difficult when we have to retrieve the hyperspherical coordinates starting from the cartesian ones. But the simplification due to the restricted domain is present here, since we can neglect the multiple combination that could give the same values since *lat* and *lon* are restricted to a single quadrant. So instead of having the multiple possibility for *lat* to be θ or $\pi - \theta$, and the same concept for *lon*, we can take the value in the predefined quadrant and simplify the calculations.

$$\begin{aligned}
&lat = arcsin(w)\\
&bear = arctan(\frac{y}{x})\\
&\tilde{r} = \sqrt{x^2 + y^2} = cos(lat)cos(lon)\\
&lon = arctan(\frac{z}{\tilde{r}})
\end{aligned} \tag{3.4}$$

3.4 Classification

In the classification variant, we maintain the same objective: to verify a simple relation between the position in latent space and the position in real space. This is a simplification of the regression method in which, instead of regressing the exact position, it is sufficient for the validity of the task to be within a neighborhood of it. The objective of this alternative method is to assess whether, under a simplification of the task, we can obtain good results. Indeed, if there is a real improvement in performance, we could regard that as a promising outcome indicating the direction is correct and worth pursuing.

We therefore start again from the feature extraction performed by our backbones. After that, the data are fed into the MLP, which in this case has a larger output dimension equal to the number of classes $C = M^2 \cdot 12$, with M the number of cells for latitude and longitude each and 12 the number of possible bearings. A problem that may arise in this formulation is that the number of classes grows rapidly with the precision required by the task. Although we did not have the opportunity to assess this, since we stopped at a small scale, this is certainly a crucial point for the feasibility of the task. This is especially important for larger areas, because the variable M does not define an absolute dimension for cells, but only a relative one based on the total area. Moreover, there is no softmax layer at the end of the MLP since the loss does not require it.

3.4.1 Rigid Classification

In the rigid classification case, the method is simple. Through the MLP we try to find the exact cell of every query photo. The output of the network will be the probability, not normalized, of the query photo belonging to every cell. After a normalization step, the maximal value will represent the prediction made by the net.

3.4.2 Soft Classification

Soft classification is a variant that should help the learning of the network. The idea is not to give a rigid assignment to every image, but to give a probability based on the distance from the center of the cell for the closest ones. This means that a central cell would have 8 neighbors, a side one only 5, while a corner one will have 3 neighbors.
The assignment is done by computing the distance from all the centers and store the inverse in the ground truth vector. Then for every ground truth vector, is normalized in order to obtain a summation equal to 1, i.e. representing a probability. So we obtain a diffuse probability that should help the network build this smooth map from the latent space to the real space. In other words, instead of having a rigid classification that tells the net only if the prediction is correct or not, there is a diffuse probability that should help it find the correct path.

3.5 Loss

3.5.1 Regression

For the regression method, what we used in both cases was the L2 loss. In fact, this loss is very suitable for the regression task, since it is completely differentiable. We tested other variants such as L1 loss, but L2 gave the best results. The mathematical formulation is the following.

$$L(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{N} \sum_{i=0}^{N-1} (x_i - y_i)^2$$

In addition, an alternative variant was evaluated for its efficacy in direct regression. This is a loss[17] that is calculated using two learnable parameters, alpha and beta, to weigh the position and bearing differently. The formulation is as follows:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}) = e^{-\alpha^2} \Delta_{pos} + \alpha^2 + e^{-\beta^2} \Delta_{bear} + \beta^2$$

The purpose of this loss is to assign differing weights to errors in position and bearing. This is due to the fact that they have different scales and it is more acceptable to have an uncertainty in the bearing than in the position. Unfortunately, the results indicated a slight decrease in performance, which led us to exclude it from the experimental phase

3.5.2 Classification

For the classification task, we use a Cross Entropy Loss. This loss is suitable for the classification task since it penalizes the model in a greater way when the

prediction is wrong with a small uncertainty.
This is its formulation in the rigid labelling variant:

$$L(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{N}\sum_{i=0}^{N-1} -\frac{log(exp(x_{i,y_n}))}{\sum_{c=0}^{C-1} exp(x_{i,c})}\mathbf{1}\{y_n\}$$

As we said before, it does not require the normalization into probability because, in PyTorch, it combines the Log Softmax that permits to transform the vector in probability, and then applies the Negative Log Likelihood to obtain the loss. This is done to have better numerical stability. In the smooth labelling variant, the formula is slightly different.

$$L(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{N}\sum_{i=0}^{N-1}\sum_{c=0}^{C-1} -\frac{log(exp(x_{i,c})}{\sum_{c=0}^{C-1} exp(x_{i,c})}y_{n,c}$$

In fact, we no longer have one single correct prediction for the target y_n, but since its probability is distributed among more classes, we have to consider all of them.

3.6 Evaluation

This is how we evaluated the performance of our method. It is slightly different in the two methodologies for clear reasons.

3.6.1 Regression

In the regression method, we used the geographical distance as a marker of the goodness of our model. To evaluate it, we used the Haversine distance, which is a formula to compute the minimal distance among tow points on a sphere. This is an approximation of the real distance, but the difference is negligible, especially on small distances. The formula is as follows:

$$\begin{aligned}
&\phi_1 = lat_1\\
&\phi_2 = lat_2\\
&\delta\phi = lat_2 - lat_1\\
&\delta\lambda = lon_2 - lon_1\\
&a = sin^2(\frac{\delta\phi}{2}) + cos(\phi_1)\cdot cos(\phi_2)\cdot sin^2(\frac{\delta\lambda}{2})\\
&c = 2\cdot arctan(\frac{\sqrt{a}}{\sqrt{1-a}})\\
&d = R\cdot c
\end{aligned} \tag{3.5}$$

So after the evaluation of the distance between the prediction and the ground truth, if the distance is less than 25 m, then the task is considered solved. Performance is evaluated with the fraction of correct guesses among all total prediction. In some cases, it could be interesting to evaluate the distribution of the predicted distances in order to have a better statistic of the data.

3.6.2 Classification

For what regards classification, the evaluation is simpler. We take every prediction and if it matches the target class, then it is considered correct.
So, even here, the statistics will be the number of correct guesses among the total. In this case, it is more difficult to obtain a more detailed statistic since the classes do not have a clear and simple distance, so it is not possible to evaluate how much the model is wrong, but only if it is being it or not.

Chapter 4

Experiments and Results

4.1 Overview

When we taught how to set up the experiments in this paper, we had a clear path to follow. Our idea was to verify the ability to learn a specific area, assess the ability of generalization in that area with the query images, and then proceed to increment the dimension of the surface. The original idea was to continue and then try to get some useful information such as the density of the dataset, the larger learnable area, the scalability of the MLP dimension, and so on. This is because we wanted to give a full overview of the problem, try to assess its weak points, highlight them, and try to propose some solution or path to follow. The initial idea was found to be incompatible with the results of the experiments. They gave us the hint that all our plans could not be viable, since we already had several difficulties in the first stage. So, we had to solve these problems and then try to get the most information possible about the task.

Algorithm 1 INR in VPR

```
Require: Model $Model$, Optim. $Opt$, Training Datasets $D$, Scheduler $Sched$,
    Evaluation Dataset $E$
 1: Initialize $Model$, $Opt$, $Sched$
 2: for training batch do
 3:     $B_i \leftarrow$ Load Training Batch($D_i$)
 4:     $Y_i \leftarrow Model(B_i)$
 5:     $L_i \leftarrow$ Loss($Y_i$, target$_i$)
 6:     $L_i$.backward()
 7:     $Opt$.step()
 8:     $Opt$.zero_grad()
 9: end for
10: for evaluation batch do
11:     $E_j \leftarrow$ Load Evaluation Batch($E_j$)
12:     $Z_j \leftarrow Model(E_j)$
13:     $L_j \leftarrow$ Loss($Z_j$, target$_j$)
14: end for
15: scheduler_step(eval_loss)
```

4.2 Implementation Details

For all experiments, as optimizer we choose Adam with an initial learning rate equal to 1e-4 and a weight decay of 1e-5. In our algorithm, we also used a scheduler. We choose ReduceLROnPlateau with a factor of 0.1 and patience of 5. This was introduced to have every experiment good results without having to fine-tune the hyperparameter of the optimizer or of the scheduler. In fact, ReduceLROnPlateau is a dynamic scheduler that is based on the performance of the model. It leaves unchanged the learning rate until the number of epoch in which the error on the evaluation dataset is worst, then the best in the experiment exceeds the patiente parameter, in our case 5. When this happens, it reduces the learning rate by the factor, in our case 0.1. This is useful because it lets the network learn until it gets stuck, avoiding a too large or too small learning rate at the cost of some extra iterations.

For both methods, the first objective was to be able to learn the training dataset. This was not an easy task and required a lot of tries before reaching the goal. The reason was that we were trying to obtain the result using the simplest possible pipeline. So, as mentioned before, a crucial help was given by the ReduceLROnPlateau that at the cost of few extra iterations gives us almost best possible results for every experiment. We started from a very small and shallow neural network and then increased the dimension until the goal was reached. A significant boost was given by the choice of using a different activation function with respect to the common. In fact, we implemented the SiLU function instead of the ordinary ReLU. This was done in order to reduce the probability of having dead neurons, and it improved performance.

The pipeline, as mentioned before, is composed only of the backbone and an MLP on top of it. The input dimension of the MLP depends on what there is before. In fact, Megaloc[14] has an output dimension of 8448, while CosPlace[13] has different possibilities, but we choose the one that guaranteed the best recall@1 and was 2048 with ResNet-50[1]. We saw from the experiments that a good number of hidden layers in the MLP was around 2 or 3, all of them of the input dimension, without any reduction towards the output dimension until the last one.

4.3 Regression

The idea for the experiments was to try to learn a small area and then expand it. We immediately encountered several difficulties and had to change the initial plan.

4.3.1 Direct Regression

From the very beginning, we faced problems learning the training set. Therefore, we first attempted to learn the training set of a small part of the city and then to generalize. However, even a small part of the city could not be learned. To simplify the task for the network, we restricted our training set to only one possible bearing. This significantly simplified the task and helped the network learn the training set. Here we found some surprises, since the results showed that CosPlace[13] performed better, with 92% of the training set learned, compared to only 68% for Megaloc[14]. These are still poor results, as the network could not easily learn a training dataset using only latitude and longitude. Nevertheless, we continued to experiment by changing different losses, increasing the depth or width of the MLP, using different activation functions, or removing the normalization layer at the end of the MLP. However, the best results still corresponded to poor performance.

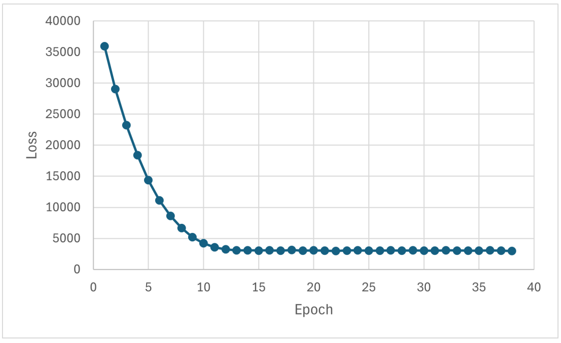


Figure 4.1: The graph represent how the loss vary across the epochs in the direct regression method. As we can see, it reach a minimum but it is not sufficient to learn successfully the dataset.

Pred Lat	Pred Lon	Target Lat	Target Lon	Distance (m)
37.75892	-122.43222	37.76174	-122.43332	327.60
37.75906	-122.4303	37.7586	-122.42734	265.163
37.76043	-122.43079	37.75982	-122.4314	86.18
37.7597	-122.43157	37.7585	-122.43036	170.62
37.76014	-122.432	37.76036	-122.42774	375.108

Table 4.1: This is an example of prediction by the network. As we can see, the relative errors on the coordinates are not big. The problem is that is required a high precision in order to have a small error on the distance in meters.

4.3.2 Hypersphere Coordinates

After the bad results with direct regression, we tried a different approach to the problem: the hypersphere method described above. Even here, the results were very poor, with 0% accuracy compared to the 0.5% obtained with direct regression. Again, we tried to identify a key issue that might enable better results. We thought that one problem with this method was that we were not using the entire hypersphere, but only a very small area of it. This happened because we were projecting onto hyperspherical coordinates that differed in the 5th decimal place and spanned only the 3rd decimal place. Thus, the idea was to distribute latitude and longitude over $[0,2\pi]$ before projecting onto the hypersphere. However, another problem arose: in a task where coordinates can vary across the entire world, periodicity is not an issue, since nearby points remain close. In our method, we would end up with situations where points close in the projection were actually very far apart in reality. To help the network learn by increasing the spatial distances between coordinates while reducing this problem, we tried projecting latitude and longitude into a single quadrant of the sphere. In this way, the coordinates would be projected into a wider area of the hypersphere while preserving their relative distances. To limit excessive distortion, we projected latitude and longitude in the range [15,45]. Even this attempt did not work, and we still obtained 0% accuracy in learning the dataset.

Pred Lat	Pred Lon	Target Lat	Target Lon	Distance (m)
38.03899	-120.81706	37.77007	-122.41783	143592.31
38.43434	-121.80215	37.77234	-122.42022	91340.84
37.44853	-122.41216	37.77157	-122.42471	35937.16
38.57946	-121.82792	37.77015	-122.42373	103974.97
38.09133	-122.32254	37.76874	-122.42084	36891.99

Table 4.2: This is a prediction using the hypersphere method. As we can see, the network is incorrect even in the unit digit in this formulation.

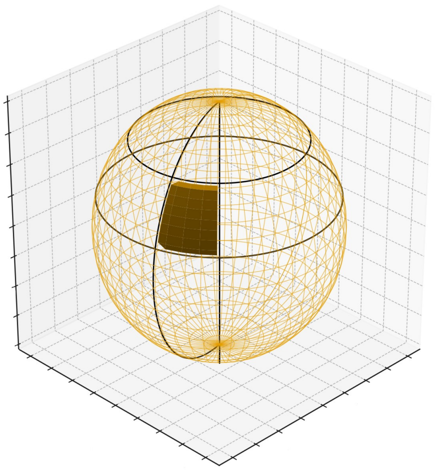


Figure 4.2: The image shows how was performed the initial projection on the sphere to obtain a better performance without occurring in distortion or periodicity problems.

4.3.3 Problems

Both regression variants failed, and since we tested many variations in an attempt to obtain results, we began to consider that the task might simply be not solvable

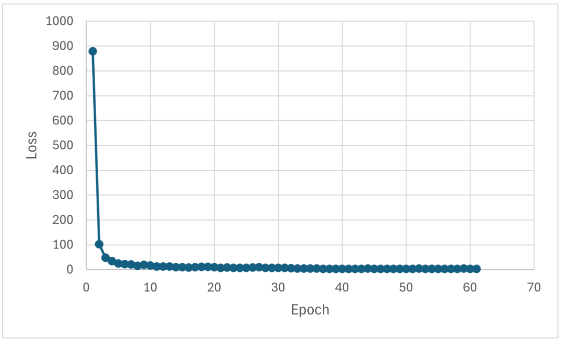


Figure 4.3: The figure shows the loss as a function of the epochs. As we can see, the network reaches a minimum after a few iterations and never improves again. Therefore, it is clear that the network in this formulation is incapable of learning.

Bear	Diff Lat	Diff Lon	Diff Bear	Distance (m)
45	9.80 e-07	1.5 e-06	0.0	0.16
135	9.80 e-07	1.5 e-06	0.0	0.16
225	9.80 e-07	2.3 e-06	5.2 e-06	0.21
315	9.80 e-07	2.9 e-06	1.0 e-06	0.26

Table 4.3: This test evaluates the accuracy of coordinate transformations and anti-transformations. Latitude and longitude are fixed in the neighbourhood of the work point. Additional information is provided by the distance, which is in the order of meters even when the error is in the 6^{th} or 7^{th} digit.

with these methods. A key point may lie in the level of precision required by the task relative to distance: in fact, we aim to obtain a representation precise to the fourth digit while considering a portion of a city. Even more precise representations have been learned in other tasks, but those approaches rely on far larger amounts of data and refer to a single, small area that does not change.

Therefore, we decided to simplify the problem by switching to a classification method. This choice was driven by time constraints, but it is valuable to evaluate the role of the dataset: for instance, it is worth investigating whether a denser dataset might yield better results, whether the backbone may be the source of the problem, or whether the task is impossible with such a simple pipeline and instead requires a more complex one to capture the relationship between positions in real space and in latent space. It is imperative that all these issues receive due consideration, as failure to do so may lead to a misjudgement of the situation.

4.4 CosPlace vs Megaloc

Since the results of the regression method were surprising, we performed one experiment to try to explain them. This experiment consists in the evaluation of the cosine similarity of 27 images over a span of 30 m. The images are ordered according to their labeled location and come from different years. After the corresponding vectors are grouped, cosine similarity is performed.

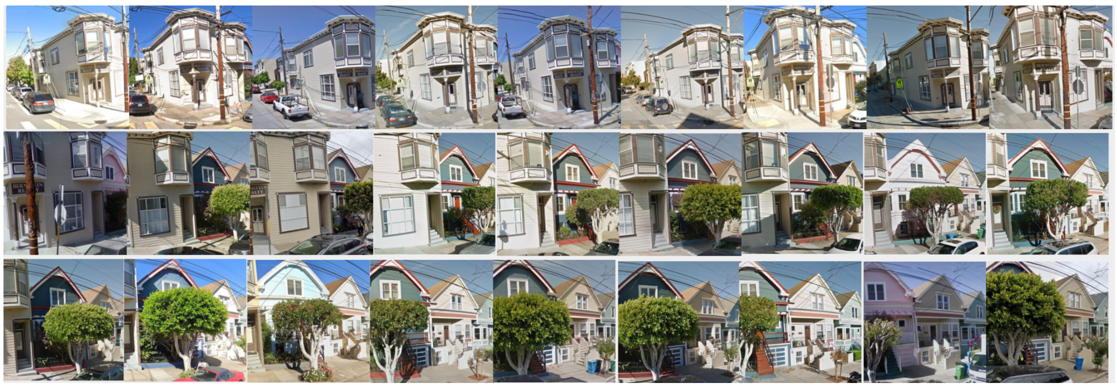
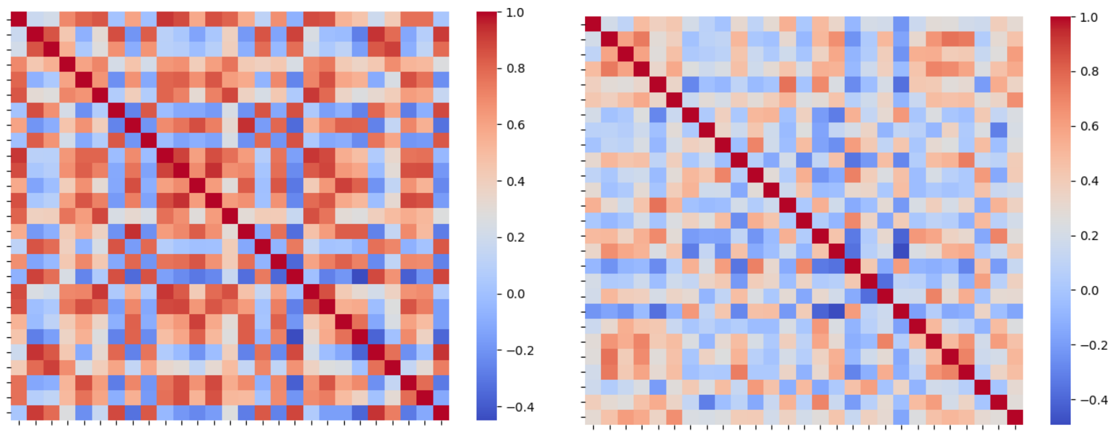


Figure 4.4: This sequence of images shows how difficult can be this task. This is images are distributed over 30 m and seems very similar to a human eye, but they still do not have a very high similarity.[13]



(a) CosPlace cosine similarity (b) MegaLoc cosine similarity

Figure 4.5: Those images show the difference in the cosine similarity between the CosPlace[13] and MegaLoc[14] on similar and close locations.

The results are remarkable, with CosPlace[13] demonstrating a substantial performance superiority over Megaloc[14]. The utilization of the term 'seem' is predicated on the recognition that the performance of the two models in the VPR task is not comparable. This observation prompts the investigation of potential explanations for this discrepancy. We hypothesized that the discrepancy arises from the dimension of the output vector. In point of fact, the function Megaloc returns a vector of length 8448, whereas the output of CosPlace can vary between 32 and 2048, with the latter being the choice in this thesis. The high vectorial dimension may explain the low similarity between the close images in the Megaloc representation. Maybe in Megaloc the decreasing in the similarity is very fast, so the absolute value seems bad, but this would explain the great performance in the task. However, this poses a new crucial point for the future studies of this field; in fact, is clear that is no more sufficient a good recall@1 to use a model as a backbone for this task but is necessarily to find a backbone that has an high similarity between close locations. This is extremely important to the success of the task, because the complete idea is based on the fact that two very close location in the real space should have very close representation in the latent space. In fact, only in this way would be possible to obtain a smooth function that links the two positions. So there is the need to use backbone that enhances this property or fine-tuning the existing in order to improve their actual representation.

4.5 Classification

The classification experiments follow the same principles of the original idea for the regression ones. We start from a very small part of the city and then try to expand it. This section contains a few experiments, since the original idea was to obtain a full regression, but due to poor results in that method, we tried to give another point of view of the task by simplifying it.

4.5.1 Rigid Classification

Ability to learn the dataset

The first point is to prove that the model is capable of learning the training dataset. To do this, we start with very few images and try to learn them. The results are more confident than the regression ones. In fact, now we are able to easily learn a small zone divided in 120k classes, while at the increasing of the dimension the performance worsened until a, still good, results of 92% correct among 100k images, always distributed among 120k classes. So, the results show that, with this method, the network can learn to associate a position in latent space with a position in real space. However, it remains to be verified whether this is due to overfitting the training data or whether it is a real mapping, and this would be tested in the next experiments.

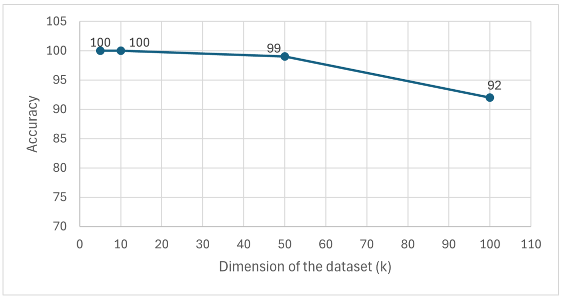


Figure 4.6: The graph shows how the accuracy of the training set varies as the dimension increase.

Ability of generalization

Although the ability to learn the dataset appeared promising, it is evident that the results of the generalization experiments did not reach the same level of success. We conducted several experiments to provide a better overview of the problem in the classification variant.

At first, we attempted to verify the ability to generalize over a very small part of the city. The size of the area is quantified by the number of images in the training dataset, and these are not round numbers. In previous experiments we could cut the dataset to a specific size and try to learn it, since we only wanted to know whether the network would be able to learn a dataset containing that quantity of information. In these experiments, however, we want to verify whether the network is able to generalize, i.e., whether it can predict the positions of unseen images using only their vectors in latent space. Because we do not know where the query images are located, we cannot cut the training images to a precise number; instead, we must choose ranges of coordinates that will contain a number of training images on the order of the size we are studying.
Another relevant point is the low number of query images. In fact, since the dimension of the area that we are restricted to study, due to poor performances, is small, we are not able to have a bigger evaluation group that will certainly lead to better statistics.

We started from around 6k images in the training set and a corresponding evaluation set composed of 41 images. Here, we obtain a 17% accuracy in the evaluation set, with 7 correct guesses. With the increase of the dimension the results do not have a clear trend; in fact, we can highlight that at a medium size of 20k training images and 96 eval images, the accuracy is 7%. While at the highest dimension of 93k training and 239 eval images, the accuracy is around 15%. These results show that instead of giving an idea of how the performance change with the dimension, maybe the crucial point is how many simple query images are in the evaluation dataset, i.e very similar images to the ones in the training set.

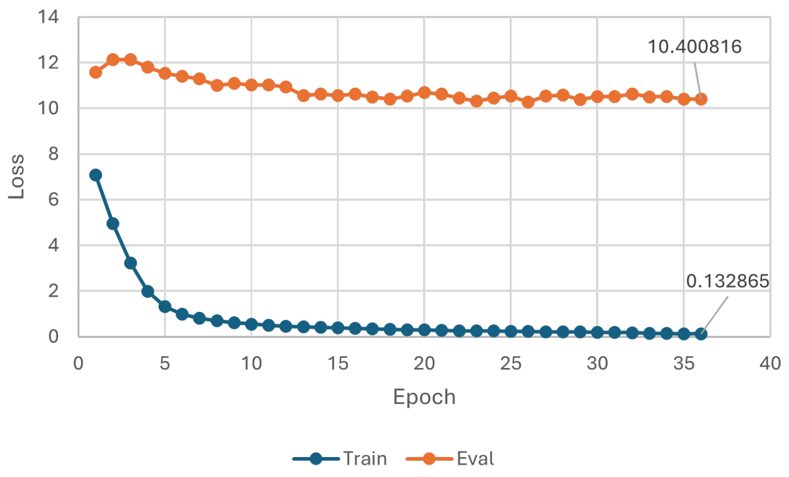


Figure 4.7: The figure shows the loss as a function of the epochs for the training and the evaluation set. The dimension of the dataset is 6k while the backbone is CosPlace[13]. As we can see, the network can learn the first, while there is no improving in the performance with unseen images. This suggest that all the images that have been correctly assigned are very similar to images in the training set, while the different ones can not assigned to correct cell.

The other point that supports this thesis is that the network does not have an improvement in the description of the evaluation dataset during the epoch. In fact, while the loss relative to the training set continues decreasing among the epoch, the one relative to the evaluation set is practically constant and, in adjoint, this is the reason for the early stopping of the training in almost all the experiment.

After evaluating the trend with respect to dimension, we performed a test to study how performance changes when the number of classes decreases. We chose to use the smallest dataset and verify the accuracy for 120k, 60k and 12k classes. Even here there is no clear trend; the reason is probably the same as in the previous experiment. The results show that the accuracies are, respectively, 17%, 20%, and 17%. The 3% increase is due to only one additional correct image. There is, however, one substantial difference that may indicate that decreasing the number of

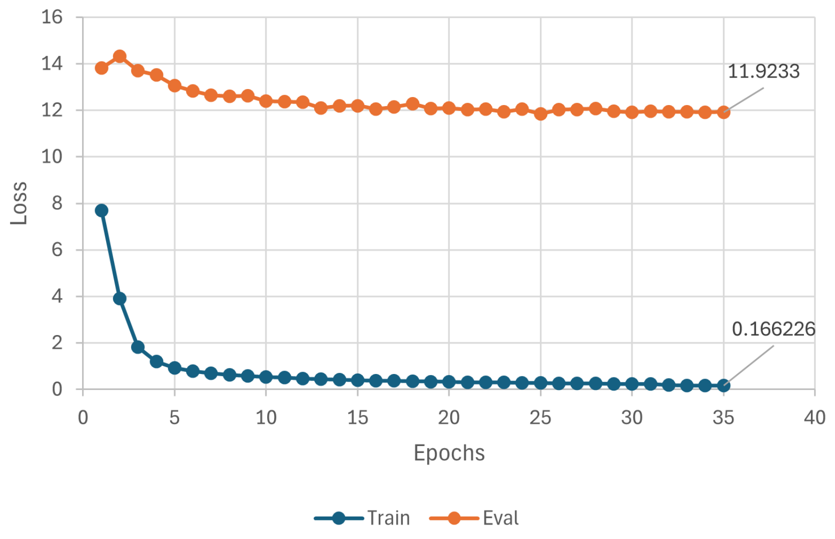


Figure 4.8: The figure shows the loss as a function of the epochs for the training and the evaluation set. In this case the dimension of the dataset is bigger, 93k, while the backbone is the same. As is evident, the trend is comparable, yielding a similar ultimate outcome.

classes too far worsens performance: the last experiment, with the fewest classes, stopped after only 10 epochs, suggesting that learning was even more difficult, maybe because images that are too different were assigned to the same cell.

Even if the results are worse than our expectations, they show that it is possible to try to associate the position in the latent space with one in the real space, and maybe this is only a matter of better representation or better pipelines.

4.5.2 Soft Classification

The soft classification experiments represent the final attempt to verify whether there has been a significant improvement in performance. These studies have been conducted on the smallest area previously studied and their results have not been favourable. Indeed, the findings reveal a great deterioration in the performance with regard to both the capacity for generalization, a phenomenon that should be facilitated by soft labelling, and, most intriguingly, the capacity to learn a dataset. This finding suggests that a significant number of experiments conducted within this configuration are unlikely to yield optimal performance outcomes. So, the study was concluded with an attempt to provide a comprehensive explanation of the results obtained and a final overview of the problem.

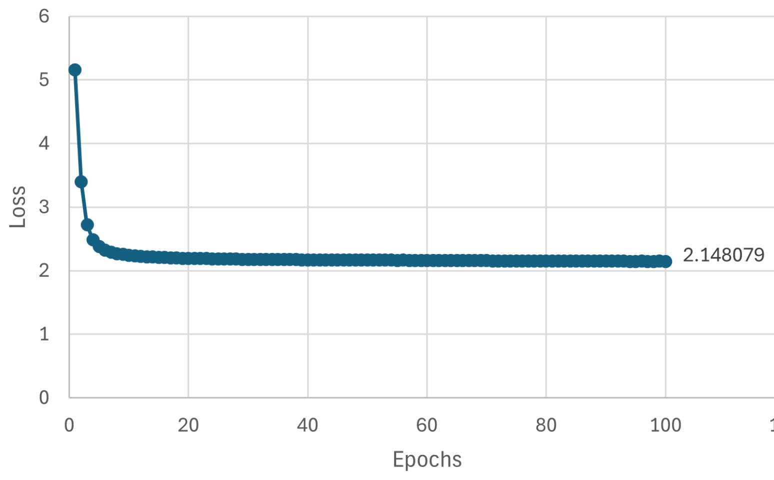


Figure 4.9: The figure shows the loss as a function of the epochs. As we can see, the final value is more than 20 times higher than the rigid labelling. Furthermore, the results are worse than those for the rigid case.

Chapter 5

Conclusion and Future Works

The overview provided by this thesis demonstrates how difficult this task is. In fact, we used one of the currently best feature extractors and still did not obtain notable results. Nevertheless, some important observations can be drawn.

First, the comparison between CosPlace[13] and MegaLoc[14] showed that recall@1 is not a sufficient statistic for choosing the backbone. It is important that the backbone produces smooth representations of the city, i.e. very close locations must have very close representations in latent space. Therefore, it may be useful to seek extractors that implement co-visibility directly in the algorithm or to fine-tune the best methods to improve this aspect. This is because, for the present task, it is not enough that two nearby locations be more similar than two distant ones: to find a function that maps a point in latent space to a point in real space, there should be no large changes in the vector representation. This issue is also connected to the MLPs. MLPs are biased toward low-frequency functions, that is our hypothesis, but this may be insufficient; perhaps the top-performing methods that yield incredible retrieval performance do not produce sufficiently smooth functions, and a deeper use of the architecture is required.

Another relevant point concerns dimensionality: it would be useful to clarify whether a high-dimensional output worsens performance or only needs to be treated differently. Related to this is the question of whether a more complex pipeline is necessary because the hypothesis of a clear relation is incorrect, or whether good results are achievable with a simple pipeline. Regarding complex pipelines, it might be useful to test whether obtaining a mean representation of each location, for example via a contrastive loss, would be beneficial.

The importance of the dataset is another key aspect: it should be determined whether a much denser dataset is mandatory for success or whether the method itself is the principal factor. This question could be addressed using a very dense dataset, such as the Oxford RobotCar dataset[18], or by testing the method on a synthetic dataset. Although a synthetic dataset would not be fully representative, it might provide clearer information about feasibility and the limits of the task and the model.

Despite the poor results for the regression method, the classification results are encouraging. We achieved successful learning of the training set and a small, but non-zero, accuracy in generalization. In addition to exploring the points above, it is important to continue pursuing the classification approach. It would be useful to determine the lower and upper limits on the number of cells to understand the attainable precision of this representation. Even if regression proves infeasible, the classification method may still yield good results and open the way to new application areas.

This is the first time this task has been formulated in this way, and poor results were to be expected; however, as this work represents an initial exploration, such outcomes are acceptable, and the positive findings are the most significant.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, arXiv:1512.03385 [cs.CV], 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[2] M. Oquab et al., *Dinov2: Learning robust visual features without supervision*, arXiv:2304.07193 [cs.CV], 2024. [Online]. Available: https://arxiv.org/abs/2304.07193

[3] Y. Huang, S. Zhang, H. Hu, D. Chen, and T. Su, “Resetting-label network based on fast group loss for person re-identification”, *IEEE Access*, vol. PP, pp. 1–1, 2019. DOI: 10.1109/ACCESS.2019.2932073

[4] C. Masone and B. Caputo, “A survey on deep visual place recognition”, *IEEE Access*, vol. 9, pp. 19 516–19 547, 2021. DOI: 10.1109/ACCESS.2021.3054937

[5] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation”, in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3304–3311. DOI: 10.1109/CVPR.2010.5540039

[6] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, *Netvlad: Cnn architecture for weakly supervised place recognition*, arXiv:1511.07247 [cs.CV], 2016. [Online]. Available: https://arxiv.org/abs/1511.07247

[7] J. Hu et al., “Enhancing visual place recognition with hybrid attention mechanisms in mixvpr”, *IEEE Access*, vol. 12, pp. 159 847–159 859, 2024. DOI: 10.1109/ACCESS.2024.3487171

[8] Y. Zhou, X. Zheng, R. Chen, X. Hanjiang, and S. Guo, “Image-based localization aided indoor pedestrian trajectory estimation using smartphones”, *Sensors*, vol. 18, p. 258, 2018. DOI: 10.3390/s18010258

[9] A. Essakine et al., *Where do we stand with implicit neural representations? a technical and performance survey*, arXiv:2411.03688 [cs.CV], 2025. [Online]. Available: https://arxiv.org/abs/2411.03688

[10] V. V. Cepeda, G. K. Nayak, and M. Shah, *Geoclip: Clip-inspired alignment between locations and images for effective worldwide geo-localization*, arXiv:2309.16020 [cs.CV], 2023. [Online]. Available: https://arxiv.org/abs/2309.16020

[11] A. Radford et al., *Learning transferable visual models from natural language supervision*, arXiv:2103.00020 [cs.CV], 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[12] T. Rothlin and M. Purandare, "Regeo: A direct regression approach for global image geo-localization", in *2025 IEEE Swiss Conference on Data Science (SDS)*, 2025, pp. 178–181. DOI: 10.1109/SDS66131.2025.00035

[13] G. Berton, C. Masone, and B. Caputo, *Rethinking visual geo-localization for large-scale applications*, 2022. arXiv: 2204.02287 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2204.02287

[14] G. Berton and C. Masone, *Megaloc: One retrieval to place them all*, arXiv:2502.17237 [cs.CV], 2025. [Online]. Available: https://arxiv.org/abs/2502.17237

[15] S. Izquierdo and J. Civera, *Optimal transport aggregation for visual place recognition*, arXiv:2311.15937 [cs.CV], 2024. [Online]. Available: https://arxiv.org/abs/2311.15937

[16] H. Wang et al., *Cosface: Large margin cosine loss for deep face recognition*, arXiv:1801.09414 [cs.CV], 2018. [Online]. Available: https://arxiv.org/abs/1801.09414

[17] A. Kendall and R. Cipolla, *Geometric loss functions for camera pose regression with deep learning*, arXiv:1704.00390 [cs.CV], 2017. [Online]. Available: https://arxiv.org/abs/1704.00390

[18] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset", *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2016. DOI: 10.1177/0278364916679498

[19] E. Brachmann, T. Cavallari, and V. A. Prisacariu, *Accelerated coordinate encoding: Learning to relocalize in minutes using rgb and poses*, arXiv:2305.14059 [cs.CV], 2023. [Online]. Available: https://arxiv.org/abs/2305.14059

[20] P.-E. Sarlin et al., *Back to the feature: Learning robust camera localization from pixels to pose*, arXiv:2103.09213 [cs.CV], 2021. [Online]. Available: https://arxiv.org/abs/2103.09213

[21] Y. Shavit and Y. Keller, *Camera pose auto-encoders for improving pose regression*, 2022. arXiv: 2207.05530 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2207.05530

[22] Y. Shavit, R. Ferens, and Y. Keller, *Coarse-to-fine multi-scene pose regression with transformers*, arXiv:2308.11783 [cs.CV], 2023. [Online]. Available: https://arxiv.org/abs/2308.11783

[23] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle, *Coordinet: Uncertainty-aware pose regressor for reliable vehicle localization*, 2021. arXiv: 2103.10796 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2103.10796

[24] G. Trivigno, G. Berton, J. Aragon, B. Caputo, and C. Masone, *Divide&classify: Fine-grained classification for city-wide visual place recognition*, arXiv:2307.08417 [cs.CV], 2023. [Online]. Available: https://arxiv.org/abs/2307.08417

[25] F. Qi, M. Dai, Z. Zheng, and C. Wang, *Geodecoder: Empowering multimodal map understanding*, arXiv:2401.15118 [cs.CV], 2024. [Online]. Available: https://arxiv.org/abs/2401.15118

[26] A. Moreau, T. Gilles, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. de La Fortelle, "Imposing: Implicit pose encoding for efficient visual localization", in *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023, pp. 2891–2901. DOI: 10.1109/WACV56688.2023.00291

[27] Y. Shavit, R. Ferens, and Y. Keller, *Learning multi-scene absolute pose regression with transformers*, arXiv:2103.11468 [cs.CV], 2021. [Online]. Available: https://arxiv.org/abs/2103.11468

[28] M. A. Musallam, V. Gaudilliere, M. O. del Castillo, K. A. Ismaeil, and D. Aouada, *Leveraging equivariant features for absolute pose regression*, arXiv:2204.02163 [cs.CV], 2022. [Online]. Available: https://arxiv.org/abs/2204.02163

[29] S. Chen, T. Cavallari, V. A. Prisacariu, and E. Brachmann, *Map-relative pose regression for visual re-localization*, arXiv:2404.09884 [cs.CV], 2024. [Online]. Available: https://arxiv.org/abs/2404.09884

[30] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, *Nerf: Representing scenes as neural radiance fields for view synthesis*, 2020. arXiv: 2003.08934 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2003.08934

[31] C. Qiao, Z. Xiang, and X. Wang, *Objects matter: Learning object relation graph for robust camera relocalization*, arXiv:2205.13280 [cs.CV], 2022. [Online]. Available: https://arxiv.org/abs/2205.13280

[32] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, *Implicit neural representations with periodic activation functions*, arXiv:2006.09661 [cs.CV], 2020. [Online]. Available: https://arxiv.org/abs/2006.09661

[33] A. Kendall, M. Grimes, and R. Cipolla, *Posenet: A convolutional network for real-time 6-dof camera relocalization*, arXiv:1505.07427 [cs.CV], 2016. [Online]. Available: https://arxiv.org/abs/1505.07427

[34] M. Leyva-Vallina, N. Strisciuglio, and N. Petkov, *Regressing transformers for data-efficient visual place recognition*, arXiv:2401.16304 [cs.CV], 2024. [Online]. Available: https://arxiv.org/abs/2401.16304

[35] S. Wang, Q. Kang, R. She, W. P. Tay, A. Hartmannsgruber, and D. N. Navarro, *Robustloc: Robust camera pose regression in challenging driving environments*, arXiv:2211.11238 [cs.CV], 2023. [Online]. Available: https://arxiv.org/abs/2211.11238

[36] X. Jiang, F. Wang, S. Galliani, C. Vogel, and M. Pollefeys, *R-score: Revisiting scene coordinate regression for robust large-scale visual localization*, arXiv:2501.01421 [cs.CV], 2025. [Online]. Available: https://arxiv.org/abs/2501.01421

[37] S. Sinha, J. Y. Zhang, A. Tagliasacchi, I. Gilitschenski, and D. B. Lindell, *Sparsepose: Sparse-view camera pose regression and refinement*, arXiv:2211.16991 [cs.CV], 2022. [Online]. Available: https://arxiv.org/abs/2211.16991

[38] F. Wang, X. Jiang, S. Galliani, C. Vogel, and M. Pollefeys, *Glace: Global local accelerated coordinate encoding*, arXiv:2406.04340 [cs.CV], 2024. [Online]. Available: https://arxiv.org/abs/2406.04340

List of Figures

List of Tables

List of Algorithms