Visual Geolocalization: mapping images to GPS

Davide Aiello
s303296

Gabriele Greco
s303435

Abstract

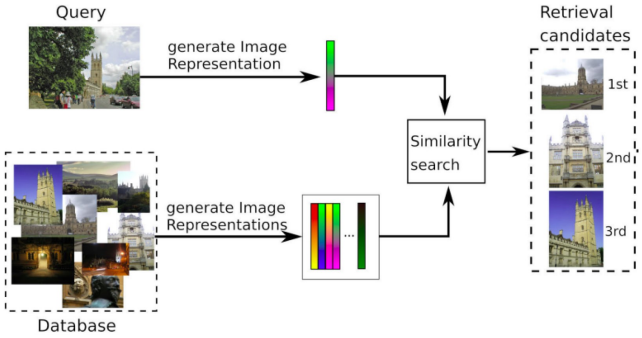
Visual Geo-localization (VG) is the task of estimating the position where a given photo was taken by comparing it with a large database of images of known locations. Building upon the innovative approach called CosPlace, which was recently published at the CVPR 2022 conference, we aim to further enhance this method by exploring different implementations. In our experiments, we explored the use of Compact Transformers as a new backbone architecture, we implemented a re-ranking technique known as Geowarp, which seamlessly blends into existing pipelines and additionally, we applied DELG, an avant-garde technique specifically crafted to efficiently derive both global and local features, as a further enhancement. We provide a detailed analysis of CosPlace, with a specific focus on the Cosface loss function used, as well as two similar losses: Arcface and Sphereface. Furthermore, we observed promising results by integrating Geowarp and DELG, which significantly improved the accuracy of the training technique. Dataset, code and trained models are available, in separate branches for each implementation, at https://github.com/GabriG23/AG.

1. Introduction

Visual Geo-localization also know as visual place recognition is, widely speaking, the task of recognizing the place depicted in an image (or a sequence of images). This task is commonly addressed as an image retrieval problem. In this formulation, the prior knowledge of the places of interest for the task is represented as a collection of images (database). Each image in the database is tagged with an identifier of its location, e.g., the name of a landmark or a GPS coordinate. When a new picture needs to be localized (query), the place recognition system searches through the database for images that are similar to it.

If similar pictures are found, their tagged locations are used to infer the location of the query. This retrieval process is typically implemented as a three-stages pipeline:

1) An encoding procedure extracts from each image a vector representation of its content (image representation).
2) A similarity search performs a pairwise comparison between the representations of the query and of every image in the database according to a scoring function (e.g., Euclidean distance or cosine similarity), and returns the best matches.
3) A post-processing stage refines the results produced by the similarity search.



The starting point for all our implementations is CosPlace [5], a novel method for visual geolocalization (VG) that merges the advantages of retrieval systems (which tolerate a few meters of error) and classification approaches (which boast high scalability). Distinct from many solutions that adopt contrastive learning, which necessitates the definition of positive and negative samples, CosPlace circumvents this laborious process by using a classification task as a proxy during training. This unique approach enables the utilization of vast datasets, thereby overcoming the scalability issues tied to contrastive learning. Furthermore, CosPlace functions similarly to a traditional retrieval system, where the trained model is used to extract image descriptors and perform image retrieval. Although there are parallels with previous classification-based methods, CosPlace differentiates itself through its innovative partitioning strategy. Traditional methods group images within large geographical cells (extending up to hundreds of kilometers), presupposing that nearby scenes have similar semantics.

In contrast, CosPlace's partitioning approach leverages the availability of densely collected data and ensures that if two images belong to the same class, they represent the same scene.

The implementations we have explored aim to enhance robustness to perspective changes or occlusions, which is achieved through the use of the DELG [3] (DEep Local and Global) descriptors. Additionally, we have employed reranking methods, such as DELG and Geowarp [8], to further improve the performance of the system. Furthermore, we have incorporated transformer networks [1], which offer a different architectural approach, in order to investigate their potential benefits in the context of visual geolocalization.

2. Related Work

Architecture. Convolutional Neural Networks (CNNs) [22] have traditionally formed the foundation of computer vision technology, and they serve as the underlying basis for Cosplace [5]. ResNet [15] showed that convolutions are effective in vision-based problems due to their invariance to spatial translations. CNNs leverage three important concepts [12]: sparse interaction, weight sharing, and equivariant representations. On the other hand, Transformers have started to gain relevance in modern machine learning research, with [13] being the first major demonstration of a pure transformer backbone applied to computer vision tasks. A new paradigm has been presented by [1], introducing ViT-Lite, a smaller version of ViT that incorporates Sequence Pooling for a Compact Vision Transformer (CVT), and adds an additional convolutional block to the tokenization step, creating a Compact Convolutional Transformer (CCT). Cosplace has already tried ViT and CCT implementations, so we are going to focus on CVT in order to explore an unseen environment. The concepts of ViT explained in [13] and [1] are based on a few components:

- **Image tokenization:** It involves dividing an image into non-overlapping square patches and converting them into sequential vectors (tokens) that can be processed by a transform model.
- **Positional embedding:** This technique incorporates spatial information into a sequence of tokens.
- **Transformer encoder:** It consists of a series of stacked encoding layers, each comprising two sublayers: Multi-Headed Self-Attention (MHSA) and Multi-layer Perceptron (MLP) head.
- **Classification:** This step adds a learnable $[class]$ token to the sequence of embedded image patches. This token represents the overall class of the image and its state after being processed by the transformer encoder.

CNNs still remain the go-to models for smaller datasets due to their computational and memory efficiency compared to transformers. While both Transformers and CNNs possess highly desirable qualities for statistical inference and prediction, they also come with their own trade-offs.

Global Features. Usually, descriptors are extracted by convolutional architectures, where images are passed through a convolutional encoder to extract features. These features are then fed into aggregation or pooling layers such as NetVlad [18] or GeM [6]. Global descriptors are used to encapsulate the overall visual content of an image. Nevertheless, they can exhibit vulnerabilities when dealing with obstructions and clutter, and may struggle to identify similarities between two images that share minimal overlap. A key drawback of relying solely on global features is the omission of spatial configuration details of the visual elements within the image.

Local Features. Typically, local descriptors have been always extracted by using Hand-crafted techniques such as SIFT [16] while the global ones are obtained by aggregating these in a more compact form. Today, most high-performing feature extractions are based on deep convolutional neural networks, especially for global features, while the local descriptors could, for instance, be extracted and compared by using the Bag-of-Words approach [20] or by using an attention module as has been done in DELF [17], which is strictly related to the DELG [3] implementation. Recent studies have yielded encouraging results through the use of deep local features, followed by spatial verification, frequently employing RANSAC [7]. This approach is pursued to tackle challenges associated with viewpoint variations. The integration of local features, which include descriptors and geometric details of distinct image areas, helps to mitigate this issue, particularly through the execution of a spatial verification process

Reranking. These techniques are commonly used in image retrieval to reassess the retrieved predictions, trading computational time for a boost in accuracy. Geowarp [8] and DELG [3] are two methods both based on local descriptors. The key point of DELG is to unify global and local features into a single deep model, enabling accurate retrieval with efficient feature extraction and avoiding the complexity of doing the two phases separately. Indeed, DELG stands for Deep Local and Global features. DELG is a model which proposes a way to exploit the combination of GeM [6] for global features and the attentive selection for local features in a joint and end-to-end manner. Furthermore, for this reason, in order to reduce the dimensionality of the local features without requiring post-processing stages that make training more complex (such as PCA), an autoencoder [11] is used. The latter can be jointly and efficiently learned with the rest of the network at the cost of

adding an extra loss function. The usage of both local and global descriptors allows for efficient inference by extracting an image's global feature, detected keypoints and local descriptors within a single model. This requires carefully controlling the gradient flow between the global and local network heads during backpropagation, to avoid disrupting the desired representations. In a recent study [8], a new dense matching method called GeoWarp was introduced, that exhibits some invariance to geometric transformations. It performs a neighbor search using state-of-the-art (SOTA) descriptors and applies a novel dense matching technique centered around a lightweight warping regression module. This module can be efficiently trained in a self-supervised fashion, enabling training on unlabeled data. Since CosPlace and GeoWarp share the same global feature extractor, integrating these methods together would be straightforward. Following that, we will construct a local feature extractor specifically designed for the warping operation.

3. Method

3.1. Loss Ablation Study

CosPlace utilizes the CosFace [9] loss, also known as Margin Cosine Loss (LCML). It is based on the Normalize Softmax Loss (NSL), where a margin term is introduced in the cosine similarity between the features and the corresponding class center. This margin term pushes the features of the same class further apart, enhancing the inter-class separation and improving the discriminative power of the features. Compared to NSL, LCML (CosFace) is more robust, as it is less likely to make incorrect decisions when there are small perturbations around the decision boundary. The cosine margin is consistently applied to all samples, regardless of the angles of their weight vectors.

In Figure 1, the geometrical interpretation of LCML and its difference from NSL is illustrated. Here, θ_i represents the angle between the feature and weight vector of class i, and m (where $m \geq 0$) is a fixed parameter introduced to control the magnitude of the cosine margin. It is worth noting that CosFace shares some similarities with other two losses, Arcface [14] and Sphereface [21]. These three losses are specific cases of the general angular margin penalty-based loss, as shown in the following formula:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(cos(m_1\theta_{y_i}+m_2)-m_3)}}{e^{s(cos(m_1\theta_{y_i}+m_2)-m_3)}+\sum_{j=1,j\neq y_i}^{n}e^{scos\theta_j}} \quad (1)$$

Where $m1$ is a multiplicative angular margin used by SphereFace, $m2$ is an additive angular margin used by Arcface and $m3$ is an additive cosine margin used by Cosface. All of them enforce the intra-class compactness and inter-class diversity by penalizing the target logit. s is the value of the embedding feature $||x||$ after a L_2 normalization. This normalization step on features and weights makes the predictions only depend on the angle between the feature and the weight. The learned embedding features are thus distributed on a hypersphere with a radius of s. On figure 2 we can see the different decision margin applied to each loss. Sphereface is similar to Cosface, but has a different formulation, it uses an angular margin to optimise the features on a hypersphere, which enforces a larger angular margin for the correct class and a smaller angular margin for the incorrect classes and provides a more natural geometric interpretation of the features on the hypersphere, at the cost of computational complexity. Meanwhile Arcface introduce an arc-like margin term in the cosine similarity space, performing well in handling large variations in pose, illumination, but again at the cost of computational complexity.

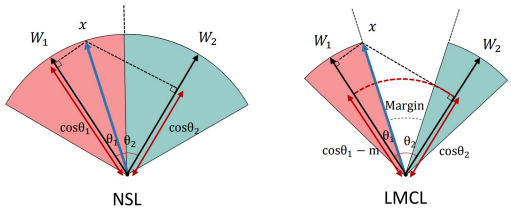


Figure 1. A geometrical interpretation of LMCL from feature perspective. Different color areas represent feature space from distinct classes. LMCL has a relatively compact feature region compared with NSL.

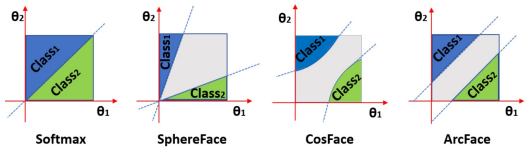


Figure 2. Decision margins of different loss functions under binary classification case. The dashed line represents the decision boundary, and the grey areas are the decision margins.

3.2. DELG

Model. Our implementation of the DELG paper tries to be more similar to the original one as possible. Indeed, the latter has been translated from TensorFlow to Pytorch framework. The approach to extract the global features remained unchanged compared to our starting point, Cosplace [5], while an attention module and an autoencoder have been added in the network following the original implementation. In order to control the gradient flow of the network, one of the key point of DELG [3], we detach the feature map at the end of block 3 during the forward pass.

Reranking. As far as the reranking phase is concerned, our whole pipeline differs from DELF [17], which is mainly

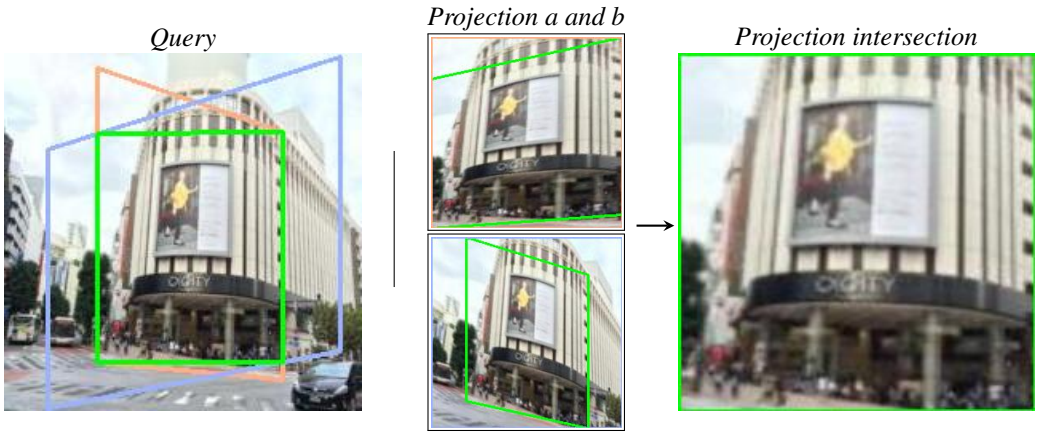


Figure 3. **GEOWARP.** Example of warping operation in day image of tokyo-xs.

responsible for the retrieval and testing of the local features, and so from DELG as a consequence, but it remains quite similar to them. First of all, no image pyramids have been used for this work. We take into account just one scale for both local and global descriptors. The reasons for this choice are the Google Colaboratory limitations, since the idea was to use a 3 scale pyramid of $\{0.5, 1.0, 2.0\}$ computed by rescaling the database images during the extraction from the dataloader. This approach exceeds the RAM memory capacity. Furthermore, this obviously also leads to spend more time on matching for each image-query pair, since the number of descriptors for each database image is tripled. This is something that could easily exceed the time limits on Google Colaboratory, even if this last computation is done on CPU. Once the global descriptors and the local descriptors have been extracted from both query and database images, the step related to the retrieval of the nearest neighbors exploiting the global descriptors remains the same as in Cosplace [5]. At this point, only the local descriptors and attentions scores related to the images considered as positive, for a certain query and by the global descriptors matching phase, are allowed to pass through several other steps in order to perform the reranking based on the local features. As the first thing, only K keypoints with the highest attention scores are taken from each image (query included) with relative coordinates and descriptors (these latter ones taken from the reduced feature maps). The K descriptors are L_2 normalized. The previous step is very different from DELG implementation because we decide not to compute the receptive field [2] since it was a very complex task. Furthermore, we were not very sure if the pre-computed values of the receptive fields were able to fit our architectures since they were computed on Tensorflow models. So we directly compute the coordinates of the keypoints by translating the coordinates of the k-th box in the real coordinates of the image and then by taking the center of the box. At the next step, we use a combination of KD-tree [4] , in order to find putative matches between the descriptors of query image and the those of the database image, and RANSAC [7], in order to verify and eventually validate these matches by filtering only the matching coordinates of descriptors that are defined as "inliers" by the affine model built by RANSAC on the coordinates themselves. This last step is the Spatial Verification and it's computed for each query-image pair. According to the inliers found by RANSAC and the distance values of the global descriptors provided by the nearest neighbor used at that step, the order of the images is changed and the recalls are computed on this new order.

3.3. Geowarp

The first task of this method is to perform a similarity search over global descriptors, which produces a set of predictions P, given an unseeen image I_q given a database of geotagged images G. Then we use the matching method to sort the top predictions in P based on a similarity measure with a query computed from dense local descriptors. The global descriptors are gathered using the same CNN pipeline used in Cosplace [5] followed by a pooling layer (GeM).

Re-ranking with dense local descriptors. For the local descriptors [8] propose a warping regression module that takes a query I_q and a prediction I_p and estimates a homographic transformation for each of the two images. The goal is to better align two different views of a same scene, even when they have limited overlap. We can define the mapping as $W(I_q, I_p) = [t_q, t_p]$ where t_q and t_p are the estimated parameters for the transformations. Following [10] a eight-points transformations is performed, bringing two overlapping views of the same location to a closer perspective $\widehat{I_q}$ = proj(I_q, t_q) and $\widehat{I_p}$ = proj(I_p, t_p). From $\widehat{I_q}$ and $\widehat{I_p}$ we extract dense local features $\widehat{f_q}$ and $\widehat{f_p}$ used to compute a similarity

score. This procedure is repeated for all the predictions in P.

The warping regression module W aim to describe the relationship between representations of co-planar points since planar surfaces are abundant in VG, and consists of 3 steps:

- Extracting features from the two images f_q and f_p
- The features are fed to a matching layer M that computes the correlation map c_{qp} between each pair of local feature descriptors from f_q and f_p
- c_{qp} is given to a network R designed to estimate the transformations of the two images

To train the module W, [8] uses 3 different losses L_{ss} self-supervised loss, L_fw features wise loss and L_{cons} consistency loss. L_{fw} is designed to ensure that the features extracted from the training pair after warping are as close as possible, and L_{cons} aims to self-generate pseudo labels as ground truths to further improve the robustness. We are not going to use L_{fw} and L_{cons} since they require a prediction dataset from train and we lack this kind of dataset. We are going to use just the L_{ss} that guides the network to learn to estimate the points describing where two input images intersect, and our results show that this one is enough. When training W, the encoder E is kept frozen, this way we can rely on features that are optmized for the geolocalization task.

3.4. CVT-CCT

As mentioned in section 2, the encoder consists of transformer blocks, each including MHSA layer and MLP block. From the architecture of ViT, [1] introduce a Sequence Pooling on CCT and CVT, that is an attention-based method which pools over the output sequence of tokens. This operation consists of mapping the output sequence using the transformation $T : R^{b\times n\times d} : R^{b\times d}$. SeqPool allows the network to weigh the sequential embedding of the latent space produced by the transformer encoder and correlate data across the input data. So by replacing the conventional class token in ViT, CVT is created. In order to introduce an inductive bias into the model, we replace patch and embedding, and a simple convolutional block has been introduced. This block follows a conventional design, which consists of a single convolution, ReLU activation, and max pool. Given an image x_0 = MaxPool(ReLU(Conv2d(x))), where the Conv2d operations have d filters, same number as the embedding dimension of the transformer backbone. The convolutional tokenizer, along with SeqPool and the transformer encoder create Compact Convolution Transformers. We are going to focus mainly on CVT since [5] have already tried out CCT and ViT, thus we can explore an unseen environment. In figure 4 we can see the architecture of CVT and CCT.

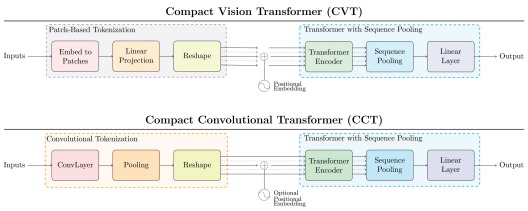


Figure 4. Comparing CVT (top) to CCT (middle). CVT may be preferable with more limited compute, as the patch-based tokenization is faster.

4. Experiments

4.1. Implementation Details

Dataset: We will use the San Francisco small dataset, sf-xs, as our training, validation, and test dataset. The geolocalized images in this dataset have dimensions of 224 $\times$ 224, which is smaller compared to the larger San Francisco dataset with dimensions of 512 $\times$ 512. Both sf-xs and tokyo-xs datasets contain domain shifts, occlusions, and perspective changes, indicating that the queries are significantly different from the database images. The sf-xs dataset consists of 1000 queries, while the tokyo-xs dataset has 300 queries. Additionally, we created a new dataset from tokyo-xs called tokyo-night, which contains 100 queries. Tokyo-night will help us gain a better understanding of how the methods perform on night images.
Cosplace parameters: For this task, we followed the implementation described in [5] and used only 1 group. The distance for KNN retrieval was set to d = 25m. As for the hyperparameters, we set M = 10 meters, α = 30°, N = 5, and L = 2. The minimum number of images per class was set to 10. An epoch consists of 10,000 iterations over a group. Validation is performed after each epoch, and once the training is completed, testing is conducted using the model that achieved the best performance on the validation set.

4.2. Loss functions

We wanted to investigate how s and m influence the discriminative power of our loss. As shown in Table 1, we trained Cosplace multiple times using all three losses, while tuning the hyperparameters s and m to find the best function and optimal values. Initially, we focused on exploring the angular margin settings. For Cosplace, we obtained good results with m = 0.4, for Arcface with m = 0.5, and for Sphereface with m = 2. Increasing m leads to a more discriminative feature distribution on the sphere and larger angular margin, as expected. However, since Cosface does not directly optimize the decision boundary, it may not perform

Loss Function	sf-xs				tokyo-xs				tokyo-night			
	$R@1$	$R@5$	$R@10$	$R@20$	$R@1$	$R@5$	$R@10$	$R@20$	$R@1$	$R@5$	$R@10$	$R@20$
Cosface s = 30 m = 0.4	33.5	48.8	56.0	62.0	47.9	66.0	73.3	81.0	31.4	43.8	54.3	**65.7**
Cosface s = 30 m = 0.3	32.6	47.6	54.7	61.0	47.6	67.0	73.0	79.7	30.5	45.7	56.2	**65.7**
Cosface s = 30 m = 0.5	33.5	48.7	56.3	61.9	48.9	66.7	74.0	81.0	32.4	46.7	56.2	63.8
Cosface s = 50 m = 0.4	**35.7**	**50.0**	56.8	**63.3**	48.9	65.1	71.7	78.7	31.4	45.7	50.5	61.9
Cosface s = 64 m = 0.4	34.2	49.4	56.7	62.0	45.7	66.0	70.8	78.7	28.6	46.7	51.4	61.9
Arcface s = 64 m = 0.5	33.9	50.8	**57.0**	62.5	47.0	66.7	70.8	78.7	28.6	**49.5**	52.4	62.9
Arcface s = 30 m = 0.3	31.7	47.2	53.4	59.7	47.6	65.1	72.1	79.7	30.5	43.8	56.2	**65.7**
Arcface s = 30 m = 0.4	33.2	48.5	54.6	61.5	47.0	67.0	73.3	79.0	31.4	44.8	55.2	**65.7**
Sphereface s = 30 m = 1.5	33.0	48.5	54.6	61.5	47.3	**67.3**	72.7	79.0	31.4	44.8	56.2	**65.7**
Sphereface s = 30 m = 2	33.4	48.5	56.1	61.4	**49.2**	66.3	**74.6**	**81.3**	**33.3**	45.7	**57.1**	64.8
Sphereface s = 30 m = 3	33.5	48.1	56.2	61.6	**49.2**	66.3	74.0	**81.3**	31.4	45.7	56.2	64.8

Table 1. **Loss Functions Results.** All the threes losses obtain similar recalls with some little differences.

well when faced with significant variations in pose, illumination, and expression. This can be observed in the results of the Tokyo-night dataset, which are lower compared to the other two losses. Additionally, we experimented with different values for the feature re-scale parameter, s. Smaller values of s result in a small hypersphere, compacting the feature vectors towards the center of the hypersphere and causing overlap in the feature space, thus reducing performance. On the other hand, larger values of s lead to a larger hypersphere, allowing the feature vectors to spread out more and creating a more discriminative feature space. However, it's important to avoid exaggeration to prevent overfitting or convergence issues.

4.3. DELG

Model backbone and implementation. Our implementation presents several differences with respect to the original one [3]. The architecture used are ResNet-18 and ResNet-50. The shallower feature map $\mathcal{S}$ is obtained at the end of the block 3 of the network, while the deeper feature map $\mathcal{D}$ is obtained at the end of the block 4. The number of channels in $\mathcal{D}$ is $\mathcal{C}_\mathcal{D}$ = 512 and $\mathcal{C}_\mathcal{D}$ = 2048 for ResNet-18 and ResNet-50 respectively, and the GeM pooling [6] is applied with parameter p = 3 which is learnable as a difference with respect to DELG. At the output of the GeM pooling, each global descriptor has dimension 512 as for the Cosplace [5] implementation. The number of channels in $\mathcal{S}$ is $\mathcal{C}_\mathcal{S}$ = 256 and $\mathcal{C}_\mathcal{S}$ = 1024 for ResNet-18 and ResNet-50 respectively. The autoencoder module learns a reduced dimensionality for this feature map with $\mathcal{C}_\mathcal{T}$ = 128. The attention network M follows the setup from [17], with 2 convolutional layers, without stride, using kernel sizes of 1; as activation functions, the first layer uses ReLU and the second uses Softplus.

Training details. The augmentation techniques and the dataset are the same as in the other implementations; the dimensions of each image are 224 $\times$ 224 resolution which lead us to have 196 regions of dimensions 14 $\times$ 14 pixels for each local descriptor. The batch size is 32 and the standard values for optimization (Adam as optimizer for all the losses) of Cosplace [5] have been used, so no scheduling on the learning rate (lr = 0.01 for the global descriptors classification and lr = 0.00001 for the other network parameters). For the global descriptors classifier we use the CosFace loss setting margin m = 0.40 and hypersphere radius s = 30. No weights on the losses have been used. Models are initialized from pre-trained ImageNet weights. We trained the model with the ResNet-18 architecture jointly for 3 epochs and then we train only the local features for other 21 epochs. For the ResNet-50 architecture, we trained jointly the model for 3 epochs and then only the local features for other 9 epochs. We stop the global descriptors training part (thus the joint learning) at 3 epochs to make the model comparable with the other implementations and with the starting configuration [5]. Each epoch counts 10000 iterations. Other configurations have been used, but they will not appear here since the parameters for the test part were not finely tuned (a crucial aspect for this type of task), in part due to the technical limitations of Google Colaboratory.

4.4. Geowarp

We conducted dense matching tests using ResNet18 and ResNet50, followed by a GeM aggregation layer. The dimension of the global descriptors is set to 512. To ensure that the method works with images of different sizes, the feature maps produced by the encoder are resized to 15 $\times$ 15 $\times$ C, where C represents the number of channels in the last convolutional layer. The correlation map, which is the output of the matching layer, is then passed to the homography regression. This regression is implemented through a sequence of six convolutional layers and a fully connected layer with an output dimensionality of 16. The weights of this final layer are initialized to 0, and the biases are set such that the initial estimated points correspond to the four corners of the input images. For the Cosface parameters, we keep the values fixed at m = 0.4 and s = 30. We use

Figure 5. **DELG.** Comparison of database images before and after re-ranking for a given query image.

batch sizes of 16 and 32 for the generated image pairs in the L_{ss}. The warping regression module is trained for 10,000 iterations and 3 epochs. Based on the ablation study conducted in [8], we assign k = 0.6, which is used in the self-supervised training method. Higher values of k correspond to more aggressive warping. Finally, we set λ_{ss} to either 1 or 10, considering the limited number of epochs trained.

4.5. CVT-CCT

We have replaced Cosplace backbone with the CVT/CCT, removing GeM, and using a 224-D SeqPool aggregation output of the backbone as descriptors. We removed the last linear layer for classification as it was already implemented in cosface function. We followed [19] implementation for both CVT and CCT. For CVT we setted the values of parameters as: layers = 12, heads = 12, and for CCT layers = 6 and head = 4. As explained we used convolutional embedding for CCT and sequential embedding for CVT. For both encoder's transformers the patch-size = 16, batch-size = 32, embedding-dim = 224, scale-dim = 4. We trained CVT model for 5 epoch 10k iterations each.

4.6. Results

DELG. Both models, which we will refer to as Res18-DELG and Res50-DELG going forward, were trained as outlined in the previous sections. The fine-tuning of all the necessary parameters for testing was carried out on the validation set. The parameters to tune are: K = 85 (keypoints with the highest attention scores), *descriptor matching threshold* = 0.65 (the distance upper bound of the KD-Tree [4] search), the *ransac residual threshold* = 15.0 (the threshold to establish if a point is an outlier or not), *num ransac trials* = 100 (number of iterations of the ransac algorithm), *min ransac samples* = 3 (minimum number of needed samples used to compute the model at each iteration). 3 is the lower bound for an affine transformation. These parameters lead to the best results for both models. Lastly, the *random state* has been set to 4 to make the computation reproducible.
As we can see in 2, although we have not followed the image pyramid approach with DELG, we obtain a considerable increase in performance with respect to the Cosplace implementation which uses only global descriptor. The best results are observed on Tokyo-night. Here, the Res50-DELG model shows a performance increase of 20% relative to the same architecture that does not use local descriptors, despite the fact that the queries involve night scenes. This demonstrates the invariance of local descriptors to these types of scenario. The performance of the Res18-DELG model is also noteworthy. Despite the greater complexity of the Res50, the Res18-DELG often delivers comparable, or even superior, results especially with respect to R@1. This suggests that the reranking process is working as intended.

Geowarp. In 2 we compared the results obtained with DELG. Geowarp archives remarkable improvement from the standard result of Cosplace as we can see, and some small gains in respect to the other reranking method. DELG obtain better results using Resnet18 on the first recalls, but Geowarp outperforme it when used in combination with Resnet50.

CVT-CCT. In 5 we reported the test results for CVT. Even if we we reached good results after every epoch on the validation set, the model di not obtain good results at test

Model	sf-xs				tokyo-xs				tokyo-night			
	$R@1$	$R@5$	$R@10$	$R@20$	$R@1$	$R@5$	$R@10$	$R@20$	$R@1$	$R@5$	$R@10$	$R@20$
CVT-224	1.4	4.3	7.0	10.9	4.0	10.0	14.2	21.0	9.2	18.7	24.8	32.4
Res18	34.5	49.8	55.7	61.6	48.9	66.0	70.5	77.5	31.4	43.8	49.5	60.0
Res18-DELG	47.5	**57.3**	59.4	61.6	**63.8**	**71.7**	73.7	77.5	**46.7**	**53.3**	55.2	60.0
Res18-GEOWARP	**49.1**	56.5	**60.0**	61.6	57.5	70.5	**76.2**	77.5	41.0	51.4	**59.0**	64.8
Res50	44.5	59.0	64.1	69.5	54.0	70.2	80.3	85.4	34.3	48.6	61.0	71.4
Res50-DELG	55.0	64.0	**67.5**	69.5	**68.9**	79.0	82.2	85.4	**54.3**	61.9	66.7	71.4
Res50-GEOWARP	**57.6**	**65.0**	67.1	69.5	66.7	**79.4**	**82.9**	85.4	51.4	**63.8**	**68.6**	72.4

Table 2. **Comparison of architectures and methods.** In this table we compare the architectures trained only for 3 epochs, whose results are obtained solely from the global features, and the architectures (trained as described in the previuos section) which perform reranking based on local features according to DELG [3] and GEOWARP [8]

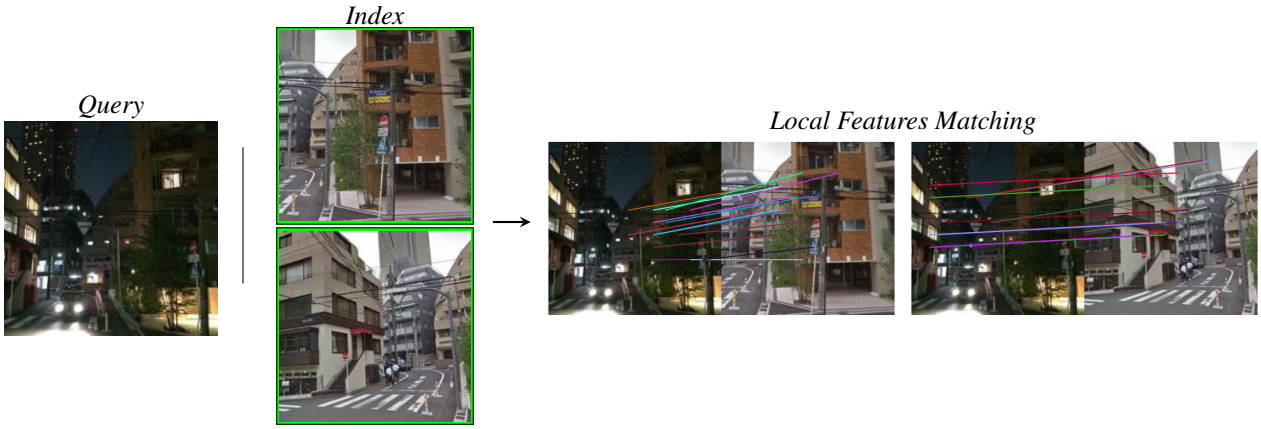


Figure 6. **DELG.** Examples of correct local feature matches, for image pairs depicting the same object/scene.

time. This could be caused by an oversimplied architecture, training on a few epoch, or for the nature of datasets. Furthermore, due to Google Colaboratory computational limitations we could not train the CCT backbone, as it exceeded GPU limit.

DELG Latency and memory. The matching latency for each query-image pair is approximately 80 ms, with a considerable amount of RAM required to store all the descriptors for reranking, despite the low image resolution. The cost of reranking using DELG is very high since each descriptor for each image has to be compared with each descriptor of each query image, and then RANSAC has to be applied. Even though all the local descriptors and attention weights were downloaded and the test was performed offline on a more powerful CPU compared to that of Google Colaboratory, some fine-tunings of the reranking phase, performed on the validation set, took up to four and a half hours to complete, especially due to the large number of queries in that set.

5. Conclusion

Starting from CosPlace, we have pursued several approaches to both improve the results and to explore additional alternatives. Our investigations began by trying to strengthen the system's resistance to perspective changes. Following this, we moved on to examine re-ranking methods and considered altering the existing network structure. As predicted, enhancements were observed when employing GeoWarp and DELG, even though these improvements came with an increased computational cost, particularly during the testing phase of DELG. As for the backbone architecture, Convolutional Neural Networks (CNNs) remain the go-to models for smaller datasets because they are more efficient in terms of computational cost and memory usage compared to transformers.

References

[1] N. Shah A. Abuduweili J. Li A. Hassani, S. Walton and H. Shi. Escaping the big data paradigm with compact transformers. In *preprint*, 2021. 2, 5

[2] Norris W. Sim J Araujo, A. Computing receptive fields of convolutional neural networks. *Distill*, 2019. https://distill.pub/2019/computing-receptive-fields. 4

[3] J. Sim B. Cao, A. Araujo. Unifying deep local and global features for image search. In *ECCV*, 2020. 2, 3, 6, 8

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. 1975. 4, 7

[5] G. Berton, C. Masone, and B. Caputo. Rethinking visual geo-localization for large-scale applications. In *CVPR*, 2022. 1, 2, 3, 4, 5, 6

[6] G. Tolias F. Radenovic and O. Chum. Fine-tuning cnn image retrieval with no human annotation. In *TPAMI*, 2018. 2, 6

[7] Bolles R. Fischler, M. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. 1981. 2, 4

[8] V. Paolicelli B. Caputo G. Berton, C. Masone. Viewpoint invariant dense matching for visual geolocalization. In *ICCV*, 2021. 2, 3, 4, 5, 7, 8

[9] Z. Zhou X. Ji D. Gong J. Z. Z. Li W. Liu H. Wang, Y. Wang. Cosface: Large margin cosine loss for deep face recognition. In *CVPR*, 2018. 3

[10] Richard I. Hartley. In defense of the eight point algorithm. 1997. 4

[11] G. Hinton. Connectionist learning procedures. 1989. 2

[12] Aaron Courville Ian Goodfellow, Yoshua Bengio and Yoshua Bengio. Deep learning. 2016. 2

[13] An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. Alexey dosovitskiy, lucas beyer, alexander kolesnikov, dirk weissenborn, xiaohua zhai, thomas unterthiner, mostafa dehghani, matthias minderer, georg heigold, sylvain gelly, jakob uszkoreit, neil houlsby. In *preprint*, 2021. 2

[14] J. Yang N. Xue I. Kotsia S. Zafeiriou J. Deng, J. Guo. Arcface: Additive angular margin loss for deep face recognition. In *TPAMI*, 2022. 3

[15] Shaoqing Ren Kaiming He, Xiangyu Zhang and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[16] D. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV*, 2017. 2

[17] Araujo A. Sim J. Weyand T. Han B. Noh, H. Large-scale image retrieval with attentive deep local features. In *ICCV*, 2017. 2, 3, 6

[18] A. Torii T. Pajdla R. Arandjelovic, P. Gronat and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *TPAMI*, 2018. 2

[19] rishikksh20. Compact transformers. 2021. https://github.com/rishikksh20/compact-convolution-transformer. 7

[20] Zisserman A. Sivic, J. Video google: A text retrieval approach to object matching in videos. In *ICCV*, 2003. 2

[21] Z. Yu M. Li B. Raj L. Song W. Liu, Y. Wen. Sphereface: Deep hypersphere embedding for face recognition. In *CVPR*, 2017. 3

[22] John S Denker Donnie Henderson Richard E Howard Wayne Hubbard Yann LeCun, Bernhard Boser and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. In *Neural computation*, 1989. 2