Visual Geo-localization: mapping images to GPS

Alessio Carachino
s296138
Politecnico di Torino
alessio.carachino@studenti.polito.it

Francesco Di Gangi
s301793
Politecnico di Torino
francesco.digangi@studenti.polito.it

Abstract

This work aims to implement additional features to the recent paper "Rethinking Visual Geo-localization for Large-Scale Applications" of CVPR(2022) [1]. Visual Geo-localization consists in estimating the position where a given photo, called query, was taken by comparing it with a large database of images of known locations. Briefly, this paper shows how SOTA techniques of Domain Adaptation, Re-ranking and more, can be exploited to significantly improve the recalls by working on the most crucial problems in Visual Geo-localization such as Domain shift, due to different weather or lighting conditions, and obstruction, due to the presence of obstacles in the dataset. The code can be found on the following GitHub repository: https://github.com/CarachinoAlessio/VisualGeolocalization.

1. Introduction

The Visual Geo-localization task is commonly addressed as an Image Retrieval problem: given an unseen image to be localized (query), it is matched against a database of geo-tagged images that represent the known world. The N-top retrieved images, with their geo-tag (typically GPS coordinates), provide the hypothesis for the query's geographical location. These architectures are trained via *contrastive learning*, usually with a **triplet loss** and leveraging the geo-tag of the images database as a form of weak supervision [1].
This project focuses in an innovative approach that extends the so-called CosPlace method, described in [1]. In CosPlace, it is exploited a loss function called CosFace. Besides CosFace, that is the loss used in CosPlace, in this work are shown other loss functions, which are:

- *ArcFace*, described in [4]
- *SphereFace*, described in [7]

All the work has been done by conducting the training and the experiments on three different datasets:

- *San Francisco eXtra Small (sf-xs)*: contains a train, validation and test set, and it is used to train and test the models.
- *Tokyo eXtra Small (tokyo-xs)*: only contains a test set: it is used to test the models.
- *Tokyo Night (tokyo-night)*: subset of tokyo-xs with only photos taken at night: it is used to test the models.

Tokyo-xs and *sf-xs* contain domain shifts, occlusions and perspective changes, meaning that the queries are very different from the database images.

1.1. CosPlace with ArcFace

ArcFace [4] is a face recognition algorithm and is not directly related to visual geolocalization, to integrate ArcFace with CosPlace it is only considered the loss function that ArcFace utilizes.
ArcFace focuses on improving the accuracy of face recognition by designing a loss function that considers the inter-class variations and intra-class compactness of the features extracted from face images. It is applied the same principle that is applied to the CosPlace loss function to train the images.
The common factor is that both losses (and, as it is shown in the next subsection, also SphereFace) are based on the cosine loss.

1.2. CosPlace with SphereFace

SphereFace [7] is a deep learning-based face recognition algorithm, so it is not directly related to visual geolocalization. Also in this part, the paper wants to show how the loss function may be integrated with CosPlace.
SphereFace aims to improve the robustness and accuracy of face recognition by using a deep neural network architecture that maps the feature representations of face images into a hypersphere with a specific angular margin. This margin helps to separate the feature representations of different faces and improve the discriminative ability of the model.

1.3. Implementation and results

Since CosPlace is developed for *Image Retrieval and Visual Geo-Localization*, meanwhile ArcFace and SphereFace are developed for *Face Recognition*, the models are quite different.
CosPlace uses the Large Cosine similarity loss [5], SphereFace and ArcFace use a cosine based loss. As shown in [4], the three losses are specific cases of the general angular margin penalty-based loss:

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

where m_1 is the parameter used by SphereFace, m_2 is the parameter used by ArcFace and m_3 is the parameter used by CosFace.

Loss function	sf-xs (test)	Tokyo-xs	Tokyo-night
CosFace	**R1: 52.3**	**R1: 69.5**	**R1: 49.5**
	R5: 66.3	R5: 84.8	**R5: 73.3**
SphereFace	R1: 50.5	R1: 68.9	R1: 46.7
	R5: 65.0	**R5: 84.1**	R5: 72.4
ArcFace	R1: 51.9	R1: 67.3	R1: 47.6
	R5: 66.6	R5: 80.6	R5: 70.5

Table 1. Results obtained with different loss functions

Due to limited computational resources, here are shown the results of the models trained on sf-xs for 3 epochs and 10.000 iterations per epoch. As shown in *Table 1*, only *Recall@1* and *Recall@5* are considered. From the results, it is clear that CosFace leads to better R@1, where SphereFace and ArcFace provided slightly worse results. This might be due to the low number of epochs or mis-calibration of the parameters of ArcFace and SphereFace.

2. Extensions

As mentioned, the main focus of this paper is to develop and test new extensions. From now on, **the model trained on sf-xs, with 3 epochs, 10.000 iterations per epoch and CosFace loss is considered as the *baseline model***. The main faced challenges concern robustness of the model to domain shifts and generalization. The main shown extensions are:

- *Gradient Reversal Layer* to improve robustness to domain shifts, including night domains.
- *Data augmentation* to improve robustness to in night domains.
- *Re-ranking method with GeoWarp* to further improve the recalls.
- *Testing different backbones* to perform a wider set of experiments.
- *Ensembles techniques* to boost the recalls.
- *Multi-Scale techniques* to improve generalization.

3. Domain shift with Gradient Reversal Layer

Gradient Reversal Layer (GRL), as described in "Unsupervised Domain Adaptation by Backpropagation" [11], consists in placing a Gradient Reversal Layer between the feature extractor and the domain classifier.

Intuitively, Gradient Reversal Layer acts as an identity function during forward propagation and, during backpropagation, the output is multiplied by a certain factor that generally is -1.

The pipeline that involves Gradient Reversal Layer is shown below:

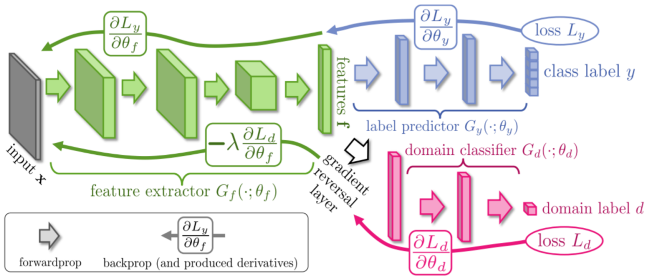
$\frac{\partial L_y}{\partial \theta_f}$
$\frac{\partial L_y}{\partial \theta_y}$
loss L_y
class label y
features $\mathbf{f}$
label predictor $G_y(\cdot;\theta_y)$
$-\lambda\frac{\partial L_d}{\partial \theta_f}$
input $\mathbf{x}$
feature extractor $G_f(\cdot;\theta_f)$
gradient reversal layer
domain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

Figure 1. Domain Adaptation with GRL

In other words, during backpropagation, the output of Gradient Reversal Layer is basically leading to the opposite of Gradient descent that is performing gradient ascent on the feature extractor with respect to the classification loss of Domain predictor.

In this case, GRL is exploited to adapt the domain to a target domain made only by few night images of *Tokyo-night*. This aims to improve the robustness of the model when the query is a night image.
It is interesting to see how, without any other changes in the original baseline, GRL alone leads to a significant improvement in terms of recalls in all cases, at a cost of an higher training time (1.5x the time required by the baseline).
A table with the results is shown as follows:

	baseline		with GRL	
Dataset	R1	R5	R1	R5
sf-xs(test)	52.3	66.3	**54.7**	**68.1**
tokyo-xs	69.5	84.8	**73.0**	**87.3**
tokyo-night	49.5	73.3	**58.1**	**78.1**

Table 2. baseline and GRL improvements

	Baseline		GRL	
	R1	R5	R1	R5
None	49.5	73.3	58.1	78.1
First type, factor=0.1	52.4	70.5	60.0	78.1
First type, factor=0.15	51.4	71.4	**60.0**	80.0
First type, factor=0.2	**55.2**	74.3	58.1	**81.0**
First type, factor=0.3	52.4	74.3	57.1	80.0
First type, factor=0.4	51.4	**75.2**	60.0	80.0

Table 3. Results on tokyo-night with different brightness factors

4. Data Augmentation

Generally, Data augmentation is used to increase the size and diversity of the dataset by applying transformations to existing data, in order to avoid over-fitting and improve the model's generalization performance.
Specifically, in this case, data augmentation is employed to improve the robustness in night domain and therefore improves the performance on Tokyo-night.

The technique is described in the following paragraph.

4.1. Data Augmentation: brightness filter

When the training dataset is mostly characterized by daylight images, it is challenging for the model to work with night images. In this section, it is proposed a very naive change: adjust the brightness of the image with a certain brightness factor in order to have darker images respect to the originals. Here are some examples:

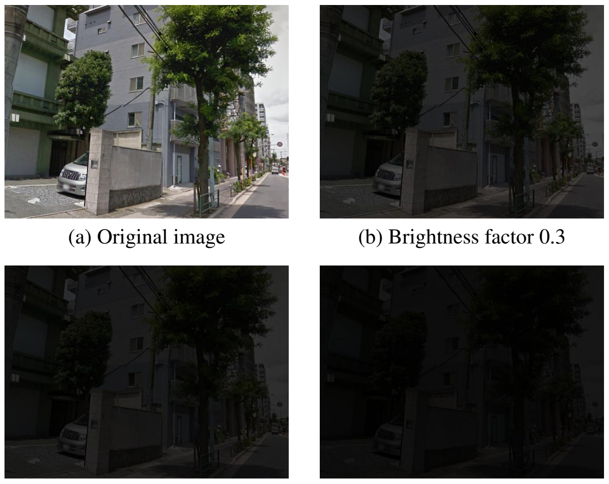
(a) Original image (b) Brightness factor 0.3
(c) Brightness factor 0.2 (d) Brightness factor 0.1

Figure 2. Impact of different brightness factor

From the results, it appears clear that a brightness filter with a factor between 0.1 and 0.2 improves the results. On the other hand, when the brightness factor is too low, the feature extractor will not be able to extract properly the features. For brightness factor greater than 0.4, the extracted features will not be much different from the case where the filter is disabled.

From now on, if not specified, when the filter is involved, the brightness factor is fixed to 0.15. The changes are performed at test time on the whole dataset and this allows to get slightly better performance on night domain images with almost zero additional computational costs.
In particular, since Tokyo-night uses Tokyo-xs dataset, this first type of data augmentation is performed only on Tokyo-xs dataset (at test time).

5. Re-ranking method with GeoWarp

Visual Geo-localization is the task of recognizing where a given photo was taken, this photo is usually called *query*. Given a large dataset of images, the goal is to find the most similar to the query. The problem is approached using both dense local features and global features, **GeoWarp** [2] works with dense local features and GeM that extracts also the global features.

At test time, the goal is to find a number of predictions using a CNN with global features, in this paper it is used *GeM pooling*, then re-rank these predictions using GeoWarp. In order to do this, the couple query-prediction is passed to an *Homography Regression Module* which estimates the core points in each image.
With the *Homographic Projection* the images are then warped.
Finally, given the warped images, with an encoder it is possible to extract the dense local features which are used to compute the similarity between the warped images.

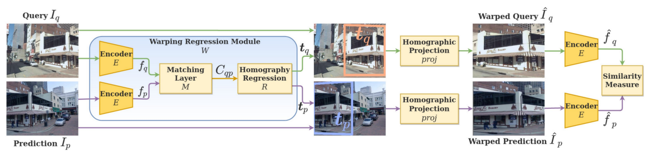

Figure 3. GeoWarp architecture. The warping regression module estimates two quadrilaterals from the query and the prediction images. The images are then warped, and their similarity is computed on their deep local features.

Dense local features are highly informative but do not resist to POV changes, that is the reason why a *Homographic Projection* is applied.

5.1. Train of the Warping Regression Module

The training is executed with a *self-supervised* way, since there are no large dataset with homography labels.

1. An image, from the training set, is taken and two intersecting trapezoids are drawn.
2. An homographic projection is applied on both to extract two images.
3. It is now possible to have two images with known intersection, that we can use to train the CNN.

It is significant to enline that *self-supervised training* data helps train the network but it **does not** represent real world distributions, this is because query-predictions pairs at test-time have temporal variation. To overcome this issue, it is possible to mine query-positive pairs in a *weakly supervised manner*, then use them to better train the network, introducing two losses.

- **Feature wise loss**: compute the pairwise warping, their dense local features and minimize the distance between local features.

$$L_{fw} = \sum_{i=0}^{w_f-1} \sum_{j=0}^{h_f-1} (\hat{f}_q(i,j)^T \hat{f}_g(i,j))^2$$

- **Consistency loss**: after the application of random augmentations to the pair of images, the second step is to estimate homographies for augmented pairs and use them to estimates pseudo-labels, and minimize the distance among them.

$$L_{cons} = \frac{1}{2N}(\tau_i^{-1}(\tau_i(I_q), \tau_i(I_g))) - [t_q^*, t_g^*])^2 +$$

$$(\tau_i^{-1}(\tau_i(I_g), \tau_i(I_q))) - [t_g^*, t_q^*])^2$$

5.2. Experiments conducted

The experiments are conducted using *ResNet18* as backbone and *GeM pooling* [9] for feature extraction, that is a technique that calculates a generalized mean of values within a pooling region (instead of using max pooling, for example), it is more flexible because of a parameter that controls the type of mean. The training is performed with three epochs of 10.000 iterations each, and the dataset used to train is *sf-xs*.

	sf-xs(test)		tokyo-xs		tokyo-night	
Model	R1	R5	R1	R5	R1	R5
Baseline	52.3	**66.3**	69.5	**84.8**	49.5	**73.3**
Baseline + GeoWarp	**60.2**	66.1	**76.2**	83.2	**60.0**	70.5

Table 4. Results with GeoWarp respect to baseline

It is possible to see that GeoWarp significantly improves the R@1 in all the three datasets, especially with *sf-xs* and *Tokyo-night* datasets but it slightly worsens R@5 in all three datasets.
This suggests the fact that GeoWarp can achieve even better results using a different backbone network. A descriptor aggregator such as *GeM* can prove that GeoWarp is also viable for large-scale problems.

Finally, it must be pointed out that GeoWarp necessarily increase the training time due to the training of the final network, but it is employed without adding a significant computational cost at testing time.

6. Testing different backbones

All the experiments conducted until now are performed with *ResNet18* as backbone and *GeM pooling* as descriptor aggregator.
In the following section, it is possible to find tests with a different backbone.

6.1. EfficientNetV2S

EfficientNetV2 [8] are a family of image classification models, which achieve better parameter efficiency and faster training speed than previous models. Built upon EfficientNetV1, EfficientNetV2 models use neural architecture search (NAS) to jointly optimize model size and training speed, and are scaled up in a way for faster training and inference speed.

EfficientNetV2-S is a pre-trained network with ImageNet1K and it is a new version of the previous ones, with an higher complexity.
In the following table, it can be seen that the results have been generated not only for a vanilla EfficientNetV2s but

also with EfficientNetV2s and the extensions discussed above. For what concerns training time, Effv2s requires about 1.5x more time, considering the same setup.

	sf-xs(test)		tokyo-xs		tokyo-night	
Model	R1	R5	R1	R5	R1	R5
Baseline	52.3	66.3	69.5	84.8	49.5	73.3
ResNet18 + GeoWarp	60.2	66.1	76.2	83.2	60.0	70.5
ResNet18 + GRL	54.7	68.1	73.0	87.3	58.1	78.1
EffV2s	63.9	74.8	83.8	92.4	73.3	**87.6**
EffV2s + Geowarp	**68.6**	74.9	**85.4**	92.1	**75.2**	83.8
EffV2s + GRL	63.4	**76.4**	83.5	**93.0**	73.3	84.8

Table 5. Results of different models and comparison between ResNet18 and EfficientNetV2s.

For Tokyo-night, the previously described data augmentation technique is employed with $brightness\_factor$ equal to 0.15.
It is possible to see that EfficientNetV2s (EffV2s) outperforms the model trained with ResNet18 (at least for the first three epochs). However, the best model is the one that combines **GeoWarp and EffV2s** for the Recall@1; meanwhile **EffV2s and GRL** model performs better in sf-xs and Tokyo-xs on the Recall@5. Regarding, *Tokyo-night* the best Recall@5 is achieved using only **EffV2s**, but the dataset is made of just 105 queries so this result is not very reliable. Of course, using these combinations, will imply in a slightly higher computational cost price at inference time.

7. Ensembling techniques

An ensemble refers to a combination or collection of multiple individual models in order to produce a better predictive model.

7.1. Model soups

Model Soup [10] is an approach that instead of selecting the individual fine-tuned model which achieves the highest accuracy on the validation set, averages the weights of models fine-tuned independently, and refer to the result as a model soup.

7.1.1 Usage of the model soup

To use model soup, different processes are needed:

- **Uniform Soup**: uniformly average all of the model weights.
- **Greedy Soup**: only using models that make an improvement.

The authors of the original paper [10] conducted a large random parameter search for fine-tuning a CLIP model on ImageNet.

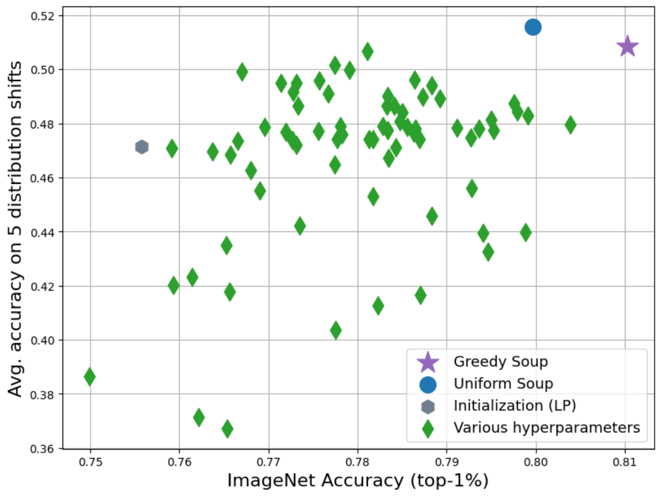

Figure 4. Tuning of the parameter.

The Greedy Soup outperforms every single fine-tuned model in terms of accuracy. *Uniform Soup*, is also good since it achieves the highest accuracy on five distribution shifts.

7.1.2 Effectiveness of the approach

Similar-initialized models with different seed or parameters might wind up on opposite sides of the optimum point of loss and error landscape, yet when compared to separate models interpolation the weights of the two solutions, can improve the accuracy.
The original research presents a novel approach to fine-tuning models that is far less heavy in terms of time and performances. Model Soups might be a terrific idea for time-saving methods.

7.1.3 Experiments conducted

The experiments have been conducted with *ResNet18* as backbone, *GeM* as descriptor, three epochs and 10.000 iterations for epoch. The training dataset is *sf-xs*:

- CosPlace
- CosPlace with ArcFace
- CosPlace with SphereFace

Model Soups	*sf-xs*	*Tokyo-xs*	*Tokyo-night*
CosFace	52.3	69.5	49.5
ArcFace	51.9	67.3	47.6
SphereFace	50.5	68.9	46.7
Model Soup	52.0	68.2	47.7

Table 6. Results of the model soup with CosFace + ArcFace + SphereFace.

Results are not very encouraging, considering that only default values are applied due to limited resources. For this reason, it is expected that the performance improves with different parameters and possibly a better trained model, not possible due to the setup limitations. Furthermore, testing the model soup with a different backbone would require to train the models with the new backbone and ArcFace and SphereFace.

7.2. Ensemble

7.2.1 Ensemble of GeoWarp and GRL

This technique consists in a combination of:

- GRL (3)
- GeoWarp (5)

Both models have been trained with *EfficientNetV2s* as backbone and *GeM* as descriptor aggregator.

These are the two best models so far, and, more importantly, their architecture is quite different, **which is a key characteristic to build an effective ensemble**. For this reason, it is expected that a such ensemble leads to an even better solution.

In order to obtain a solution that merges somehow the two models, two different methods have been developed for this purpose, and are described as follows:

- **Nearest Indices First (NIF)**: With this technique, the predictions and the distances arrays are computed for both models. The final combined predictions array consists in the combination of the two original arrays and sorted by distance. In this way, the closer the image is to the query, the more likely the image is considered "positive" for the query.
- **Shared Indices First (SIF)**: With this technique, only the predictions arrays are computed. Since GeoWarp performs slightly better than the GRL model, the final predictions array is the result of the combination of the images that both appears in the original predictions arrays and the images that appears only in the GeoWarp predictions.

	sf-xs(test)		tokyo-xs		tokyo-night	
Model	R1	R5	R1	R5	R1	R5
Geowarp	**68.6**	74.9	85.4	92.1	75.2	83.8
GRL	63.4	**76.4**	83.5	**93.0**	73.3	84.8
Geowarp + GRL (NIF)	67.4	76.0	83.8	**93.0**	**76.2**	**86.7**
Geowarp + GRL (SIF)	**68.6**	75.0	**86.3**	92.7	74.3	84.8

Table 7. Results of the two techniques and comparison.

For what concerns *Tokyo-night*, *NIF* effectively improves the R@1 because in night domains, the more distant elements are hidden in the dark so it is reasonable to give more importance to what is closer.
On the other hand, for *sf-xs* and *Tokyo-xs*, the best results (in terms of R@1) are obtained with *SIF* because the datasets are much larger, suggesting that it is better to give more importance to photos were the two models agree on.

8. Multi-scale

Multi-scale is commonly employed in image retrieval tasks and consists in exploiting somehow descriptors associated to the same image at different resolutions. As shown in Patch-NetVLAD: Multi-Scale Fusion of Locally-Global Descriptors for Place Recognition [6] and Deep Visual Geo-localization Benchmark [3], multi-scale can be effective in this case, too.
Here it is shown a basic version of multi-scale implementation that consists in generating for the same query, different descriptors associated to the query at different resolutions (the resolution multipliers are $[0.526, 0.588, 1, 1.7, 1.9]$), and aggregates them in a unique descriptor array through average, sum, min and max's logic.
Following, the tables with test results and a brief discussion on the results:

Multi-scale testing on ResNet18 + GRL

	sf-xs		Tokyo-xs		Tokyo-night	
Method	R@1	R@5	R@1	R@5	R@1	R@5
/	**54.7**	68.1	73.0	**87.3**	58.1	**78.1**
avg	54.3	**68.1**	**74.3**	86.3	**59.0**	77.1
sum	50.5	65.0	72.1	77.8	57.1	73.3
max	50.1	64.0	73.0	85.4	59.0	75.2
min	50.5	63.7	71.4	84.4	59.0	76.2

Table 8. Results of the multi-scale testing for GRL + ResNet18.

Multi-scale testing on ResNet18 + GeoWarp						
	sf-xs		Tokyo-xs		Tokyo-night	
Method	R@1	R@5	R@1	R@5	R@1	R@5
/	60.2	66.1	76.2	83.2	60.0	70.5
avg	**60.9**	**66.4**	75.1	**85.2**	**64.8**	**74.3**
sum	59.1	64.6	75.6	84.1	57.1	65.7
max	60.2	65.3	**77.8**	84.8	60.0	72.4
min	59.9	64.5	74.0	82.9	60.0	70.5

Table 9. Results of the multi-scale testing for GeoWarp + ResNet18.

Multi-scale testing on EfficientNetV2s + GRL						
	sf-xs		Tokyo-xs		Tokyo-night	
Method	R@1	R@5	R@1	R@5	R@1	R@5
/	**63.4**	**76.4**	**83.5**	**93.0**	**73.3**	84.8
avg	62.9	74.3	81.9	92.7	68.6	**86.7**
sum	60.6	74.2	78.4	91.1	67.6	82.9
max	60.6	72.1	78.4	92.1	69.5	84.8
min	60.3	72.0	79.0	91.7	66.7	85.7

Table 10. Results of the multi-scale testing for GRL + EfficientNetV2s.

Multi-scale testing on EfficientNetV2s + GeoWarp						
	sf-xs		Tokyo-xs		Tokyo-night	
Method	R@1	R@5	R@1	R@5	R@1	R@5
/	68.6	74.9	85.4	92.1	75.2	83.8
avg	**68.6**	**74.9**	**86.0**	**92.7**	**79.0**	**86.7**
sum	67.6	73.0	81.3	89.8	73.3	81.9
max	66.5	71.9	85.4	92.4	75.2	83.3
min	66.9	72.2	83.2	89.5	75.2	82.9

Table 11. Results of the multi-scale testing for GeoWarp + EfficientNetV2s.

As expected, **the best logic is the one that averages** the descriptors because, in this case, it leads the model to generalize better, considering the different resolutions.
On the other hand, sum's logic performs poorly as the final descriptors array has a different scale compared to the descriptors that the model expects, suggesting that a new trained model, with the proper resources and with multi-scale enabled, may perform better.

8.1. Multi-scale testing with SIF and NIF

Results collected by the combination of NIF and SIF with multi-scale are shown as follows:

	sf-xs		Tokyo-xs		Tokyo-night	
Method	R@1	R@5	R@1	R@5	R@1	R@5
/	**68.6**	**75.0**	**86.3**	92.7	74.3	84.8
avg	67.4	72.9	86.0	92.7	**77.1**	**87.6**
sum	67.1	72.1	81.3	90.2	71.4	81.9
max	66.4	72.2	85.4	**93.3**	75.2	83.8
min	66.8	72.4	83.5	90.8	75.2	84.8

Table 12. Results obtained combining **SIF** and different multi-scale approaches (EffV2s as backbone)

	sf-xs		Tokyo-xs		Tokyo-night	
Method	R@1	R@5	R@1	R@5	R@1	R@5
/	**68.6**	**75.0**	**86.3**	92.7	**74.3**	84.8
avg	62.2	73.6	81.6	93.0	69.5	**86.7**
sum	61.1	73.9	74.9	89.8	67.6	81.9
max	60.7	72.5	80.3	**93.0**	69.4	83.8
min	61.2	73.0	78.7	90.8	68.6	83.8

Table 13. Results obtained combining **NIF** and different multi-scale approaches (EffV2s as backbone)

For these results, the same final considerations of the section (8) are valid.

9. Discussion and Conclusion

At this point, it is clear that in Visual Geolocalization tasks, the model performance are affected by a lot of different factors. For this reason, the extensions that have to-be included in the final model are very application-dependant. In fact, the majority of the extensions bring a significant increase at training and/or inference time.

However, there are some points for possible future works and improvements, such as the ensemble (7). An ensemble of more trained models might lead to better results, since during this work there were some limitations due to the setup.

To conclude, the best results, for each dataset, are collected in the following list. All due considerations can be found in the related paragraphs.

- ***sf-xs***: about the R@1 it is possible to see that the best model reaches the **68.6%** of accuracy, with **EfficientNetV2s + GeoWarp** (same result obtained with *avg* method for multiscale and SIF). On the other hand, for the R@5 the best model reaches the **76.4%** with the **same model**.

- ***Tokyo-xs***: about the R@1 it is possible to see that the best model reaches the **86.3%**, with **SIF**. On the other hand, about the R@5 the best model reaches the **93.3% SIF** using the *max* method with *Multi-Scale.*
- ***Tokyo-night***: about the R@1 it is possible to see that the best model reaches the **79.0%** of accuracy, with **EfficientNetV2s and GeoWarp** applying the *avg* Multi-Scale. On the other hand, about the R@5 the best model reaches the **87.6%** with only **EfficientNetV2s** or applying the *avg* method on **SIF**.

References

[1] G. Berton, C. Masone, and B. Caputo. Rethinking visual geo-localization for large-scale applications., 2022. CVRP. 1

[2] Gabriele Berton, Carlo Masone, Valerio Paolicelli, and Barbara Caputo. Viewpoint invariant dense matching for visual geolocalization, October 2021. 3

[3] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. Deep visual geo-localization benchmark, 2022. 6

[4] J. Deng, J. Guo, J. Yang, N. Xue, I Kotsia, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition., 2022. TPAMI. 1, 2

[5] 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., 2018. CVRP. 2

[6] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition. *CoRR*, abs/2103.01486, 2021. 6

[7] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition., 2017. CVRP. 1

[8] Quoc V. Le Mingxing Tan. Efficientnetv2: Smaller models and faster training, June 2021. 4

[9] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. 4

[10] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time, 17–23 Jul 2022. 5

[11] G. Yaroslav and V. Lempitsky. Unsupervised domain adaptation by backpropagation, 2015. 2