Image Retrieval Visual Geolocalization

*Final Project, Machine Learning and Deep Learning

Alberto Martín Garre
s293972
s293972@studenti.polito.it

Luis Arias Gonzales
s293581
s293581@studenti.polito.it

Khudayar Farmanli
s276310
s276310@studenti.polito.it

***Abstract*—Visual Place Recognition (VPR) is the task to accurately recognize the location of a given query photograph. In addition, some of the most important applications behind VPR are robotics and autonomous systems, 6-DoF localization, visual SLAM and structure-from-motion pipelines. Some of the most challenging goals in this area are building a domain robust deep network or combine some methodologies to improve the performance of an specific architecture. In this paper, we provide different methodologies such as smart data augmentation, multi-scale testing, ensembling methods for different models, and experiments with several optimizers and schedulers to improve the performance of the model in terms of robustness achieving the state-of-the-art in some specific datasets. The data augmentation strategies we have applied are focused on improving the results of the model against night images and occlusions. Multi-scale testing considers different resolutions of a single image in testing obtaining considerable improvements in the performance of the model. Ensembling methods combine descriptors from different models trying to improve the original model's performance. And, finally, we have made several experiments with new optimizers (ADAMW and ASGD) and schedulers analyzing the behavior of the model with these changes. Evaluation on 4 viewpoint-varying and viewpoint-consistent benchmark datasets confirms that some of the extensions proposed, lead to state-of-the-art. Recall@N performance for global descriptor-based retrieval. Source code is publicly available at https://github.com/albertomg098/MLDL_Final_Project.**

I. Introduction

Visual Place Recognition (VPR) can be defined as the ability of recognizing the same place despite significant changes in appearance and viewpoint. Many studies refer to VPR as image retrieval problem. How is this problem faced? At the beginning, it is assumed that the world is represented by a set of geotagged images. Then, it is designed an image representation extractor function, which provides a mapping from the image to the image representation space. And finally, given a query image, we extract its representation, find the closest database representation, and transfer its location.

VPR is challenging due to uncontrollable environmental factors such as illumination, viewpoint, and seasonal transitions [2], [3], [10]. Therefore, learning a generic but robust place representation has become an active area of research [1].

In this paper, we use as base the model provided by [11], in which it is used a backbone architecture such as Resnet18 with finetuning and it is added an aggregation layer. The advantage of this base model is that it is structured in a way that allows you to select different backbones and different aggregation methods. Nevertheless, for the sake of simplicity, we will use as backbone the one set as default (Resnet18 with the last 4 layers set for fine-tuning).

In this paper we propose different methods for improving the performance of the base model. Our main contributions are:

- *Train experiments*. We have perform several extensions related to the way the model is train. First, in order to improve the model's results against datasets with night images we have implemented a smart data augmentation, and to increase robustness to occlusions we have included some data augmentation for changing the perspective of the images or erasing some parts of an image. And second, we have experimented with different new optimizers such as ADAMW or ASGD, and we have combined them with some schedulers such as Plateau or Cosine Annealing.
- *Test experiments*. In a parallel way, we have develop extensions for testing. First, following [1] we have implemented in the test process a multi-scale pyramid in order to obtain better descriptors. And second, we have done some experiments ensembling descriptors from different models.

The reminder of this paper is organized as follows: Section II reviews the related work; Section III describe the methods implemented; Section IV discusses the experiments performed comparing results with original base model; and Section V presents conclusions with future work directions.

II. Related Work

A. Deep Visual Geo-localization Benchmark

This paper, written by some professors of 'Politecnico di Torino', is the base or backbone of this project. They provide an open-source benchmarking framework that allows to build, train and test a huge range of different architectures, aggregations methods, and data augmentation techniques for the task of VPR [11]. However, we will not use all the methods they provide, specifically, we will use NetVLAD and GeM as aggregation layers, and ResNet18 as backbone architecture with finetuning set to the last 4 layers. These two components will form the main structure of our 'vanilla' model with which we will perform all the comparisons of the new implementations.

We should analyze deeper the base model. Once we have defined the structure, we should be familiar with the training strategy followed. They use triplet loss for the model's training. Maybe the most significant part of the vanilla model is the **triplet mining**. The distinctive aspect of triplet loss is that it requires a triplet of images for computing the loss: the query, a positive, and a negative. It is easy to find positive images: based on coordinates, it is possible to find images within a radius of 10 meters from the query. But the challenge is in finding a negative image. At first, it will be easy to pick a very different image that guarantees to be a negative of the query. Nevertheless, the purpose of the model is to be as robust as possible. Therefore it is recommended to pick negatives the nearest possible to the query. Following this idea, and taking into account that **triplets are composed of one query, one positive, and ten negatives**; given a query, triplet mining is based on two procedures.

Positive mining is done by finding the best positive images by a Nearest Neighbor algorithm applied to the UTM coordinates within a radius of 10 meters, and then the best positive one is chosen by applying again Nearest Neighbors to these candidates, but this time in the feature space.

Negative mining follows a similar procedure than positive mining. First, soft positives are computed (images within a radius of 25 meters from the query). This way, it is possible to perform the first division, as all the images that are not soft positive images are negative candidates. Doing some sampling in order to reduce computation time, a Nearest Neighbor in the feature space is performed in these non-soft-positive images trying to find the ten nearest negatives to the query.

Finally, related to the aggregation layer, we will use mainly NetVLAD, but also GeM. NetVLAD is an extension of VLAD ('Vector of Locally Aggregated Descriptors') whose aim is to obtain a more compact representation of the descriptors. It performs clustering in the feature space and computes the distance between each feature and the centroid of each cluster, achieving a vector representation of descriptors. The main advantage of NetVLAD is that it is readily pluggable into any CNN architecture and amenable to training via backpropagation [12].

B. MultiRes-NetVLAD: Augmenting Place Recognition Training with Low-Resolution Imagery

This research has been used as guideline to build the multi-scale test extension. In this paper they have achieved to compute better descriptors of an image (richer place representations) by training the model with a multi-low resolution pyramid. It allows to extract more information from a single image based on considering lower resolutions of an image. They have implemented this method in a full VPR framework: a backbone architecture, such as ResNet, and a NetVLAD aggregation layer. This means that their proposed model is trainable end-to-end.

C. Random Erasing Data Augmentation

Random Erasing is a data augmentation technique that **randomly** selects a rectangle region in an image and erases its pixels with random values [13].

Some of the parameters that are involved in this method are the probability that the random erasing operation will be performed, the proportion of erased area against input image, the range of aspect ratio of erased area or the erasing value (it is related with the random value assigned to each pixel in the erased area; and it can be a single value or a tuple of length 3 depending on how we want to erase the pixels; but generally it is done randomly).

III. Methodology

For this section we have decided to divide it in two parts: training extensions and testing extensions. We will explain briefly the maths and the methods behind each one of the implementations.

A. Train Extensions

- **Smart Data Augmentation**. In order to improve the robustness of the model related to night queries, we have implemented this technique, following two different Pytorch tools and comparing them with respect to the original architecture. *ColorJitter* is a technique based on the random change of the brightness, contrast, saturation and hue of an image. It iterates between a range of values, so the obtained new queries can be updated in a lower or upper way (for example, about brightness we can obtain both a brighter and a darker image). *Functional adjust* is another transformation, but in this case it is passed a parameter factor that provides the information about how much to adjust the parameter (for example, brightness factor of 0 gives a black image, 1 gives the original image while 2 increases the brightness by a factor of 2). Both of them are implemented with many different values of these four parameters in order to get the better performance of the model.
- **Perspective and Oclussions Data Augmentation**. This improvement is done in a combined way by composing two distinct transformations. *RandomPerspective* performs a random perspective transformation of the given image with a given probability. Some of the parameters that influence in this method are the probability of performing, the degree of distortion (the higher, the more perspective change) and the fill (the higher, the whither the empty pixels will be filled). *RandomErasing* is a transformation that randomly selects a rectangle region in a torch Tensor image and erases its pixels. To perform this transformation, it is assumed that the ratio $\frac{erasingregion(S_e)}{inputimageregion(S)}$ is in the range (0.02-0.33) and the aspect ratio of erased area r_e to get $H_e = \sqrt{S_e r_e}$ and $W_e = \sqrt{\frac{S_e}{r_e}}$ is in the range (0.3-3.3) and no erasing value (related to pixels).

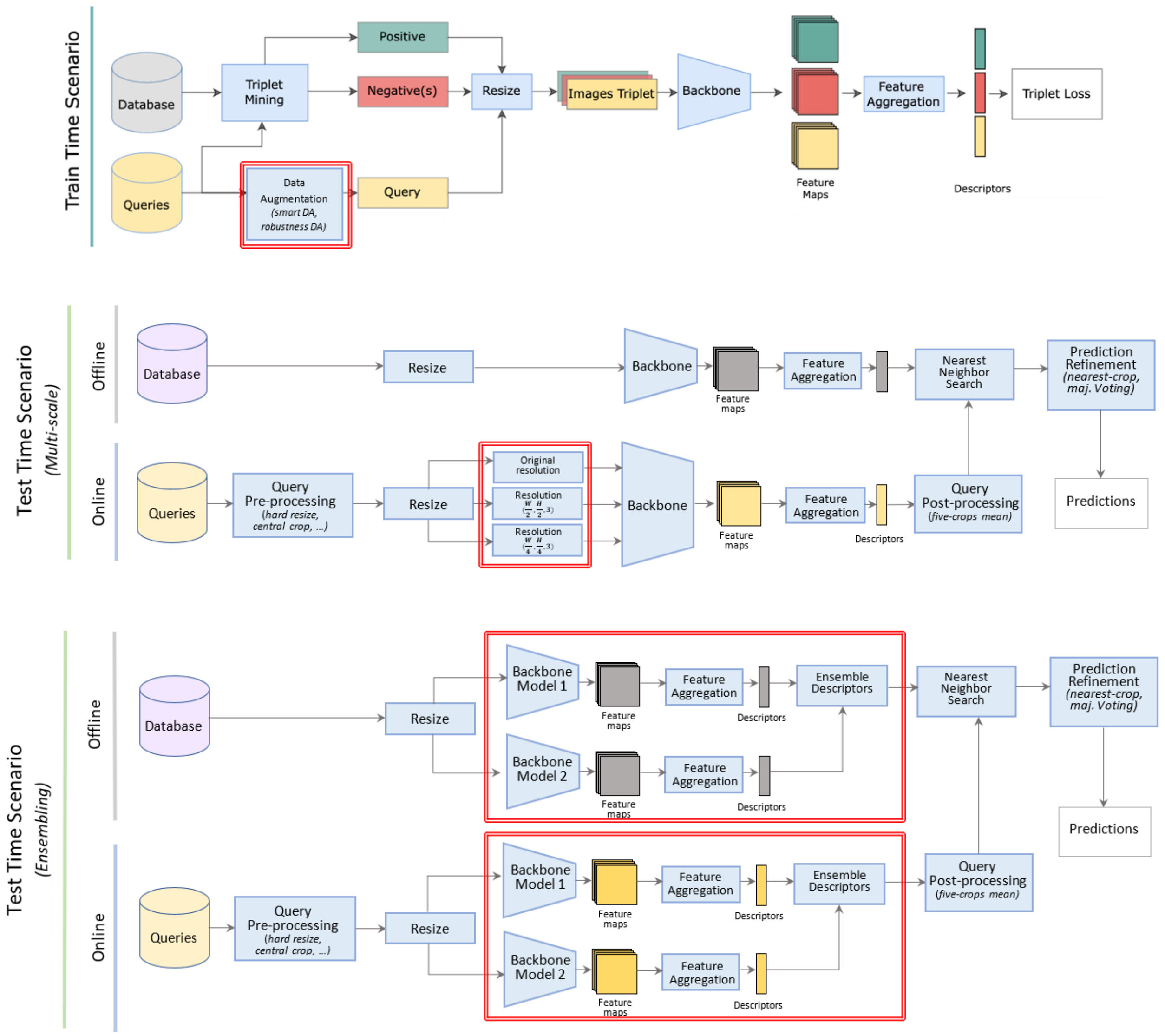

Fig. 1: Extension methods

- **Optimizers & Schedulers**. The optimizers used to improve the performance of the architecture are Average Stochastic Gradient Descent (**ASGD**) and Adaptive Moment Estimation with Weight Decay (**AdamW**). With ASGD, the basic idea is to do regular stochastic gradient descent $w_{t+1} = w_t - \eta_t \nabla Q(w_t)$, but then take the mean $\bar{w} = \frac{1}{N}\sum_{t=1}^{N} w_t$ as final solution in order to reduce the effect of noise. Doing AdamW, L2 regularization is used to adjust the weight decay term to influence in the gradient update. After finding the best optimizers, these are combined with **Plateau** and **CosineAnnealing** schedulers. Plateau scheduler will update the value of the learning rate when a metric has stopped improving, while CosineAnnealing will always update the value of the learning rate following this equation:

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right)$$

B. *Test Extensions*

- **Multi-scale testing**. Following the paper [1] as baseline, we have implemented a small pyramid for testing. This pyramid computes low resolutions versions of an image following equation 1. Once they are computed, the images are passed through the backbone architecture (ResNet18) and the features are concatenated. Then, the concatenated features are passed through the aggregation layer to obtain a vector representation. For this extension, we have performed experiments with 2 low resolutions of an image and using NetVLAD an GeM as aggregation methods. Finally, we have compared these results against vanilla recalls. A graphical representation of this extension is showed in figure 1.

$$I^l = (3, \frac{H}{l}, \frac{W}{l}) \tag{1}$$

- **Ensembling**. In this extension we have tried to exploit different models by combining them. Specifically, we have consider two models: GeM and NetVLAD. First, we compute the descriptors for each model and, before preforming the Nearest Neighbor algorithm, we ensemble the descriptors from each model. The first approach is to perform a simple concatenation of both descriptors. Nevertheless, GeM's dimension compared to NetVLAD's is $\frac{256}{16384}$, which is around 1.6%. Therefore, doing a simple concatenation of the model's descriptors will result in neglecting GeM's contribution. In order to solve this problem, we have applied PCA to NetVLAD descriptors reducing dimensions to 2048. This number is still higher than GeM's descriptors dimension, however this value makes sense since NetVLAD performance is much higher than GeM's. So, it is reasonable to assign a bigger weight to NetVLAD. This extension is graphically represented in figure 1.

IV. Experiments

This is the most important part of the paper. In this section, we will show all the results obtained from applying the extensions that we have discussed in the previous section. As well as it has been done in the methodology we will divide the experiments into training and testing.

A. Training Extensions

- **Smart data augmentation**. This technique will be implemented by assigning different values to {brightness, contrast, saturation, hue} = {b, c, s, h} with two different tools: *ColorJitter* and *Functional Adjust*.
  There are three possible stages with *ColorJitter*:
  - M1: {b=1, c=1, s=2, h=0}
  - M2: {b=rand(0.2-3), c=rand(0.2-3), s=rand(0.2-3), h=rand(0-0.5)}
  - M3: {b=rand(0-1), c=rand(0-1), s=rand(0-1), h=rand(0-0.5)}

  There are three possible stages with *FunctionalAdjust*:
  - M4: brightness =0.25
  - M5: brightness = 0.5
  - M6: {b=rand(0-1), c=rand(0-1), s=rand(0-1), h=rand(0-0.5)}

  All experiment results are in table VI. The best performance is given by M2 stage, improving the vanilla model. From this, the information that can be inferred is that data augmentation for night query robustness provides better results when working with a wide range of parameter values.
- **Perspective and occlusions data augmentation**. This improvement is applied by combining *RandomPerspective* and *RandomErasing*. Afterall, this technique is implemented by changing the following four parameters: probability of applying Random Perspective technique (pRP), distortion in Random Perspective (dRP), fill in Random Perspective (fRP) and probability of applying Random Erasing technique (pRE).
  Following this idea, there will be these stages:
  - M1: {pRP=0.33, dRP=0.4, fRP=0, pRE=0.33}
  - M2: {pRP=0.33, dRP=0.4, fRP=150, pRE=0.33}
  - M3: {pRP=0.33, dRP=0.4, fRP=300, pRE=0.33}
  - M4: {pRP=0.5, dRR=0.2, fRP=0, pRE=0.2}
  - M5: {pRP=0.5, dRP=0.2, fRP=150, pRE=0.2}
  - M6: {pRP=0.5, dRP=0.2, fRP=300, pRE=0.2}

  All experiment results are in table VII. The best performance is given by M2 stage, improving the vanilla model. From this, the information that can be inferred is that aspects such as providing an important weight to both RP and RE (p = $\frac{1}{3}$), and applying an appropiate fill and distortion operations can influence in the final results.
- **Optimizer and schedulers**. First, we have performed a grid search for finding the best learning rate (*lr*) and weight decay (*wd*) with each optimizer (ASGD and ADAMW), with one epoch models; second, we have trained more complex models (5 epochs) with the hyper-parameters found; and we have evaluated both models (1 and 5 epochs) in the rest of the datasets. The grid search is shown in table II resulting in the parameters in table III. For the models trained with one epoch, we have decided to plot evaluation on all datasets with Recall@5. This analysis is in figure 2 and we can observe how the new optimizers improve performance in all datasets.In the case of models trained for 5 epochs, there is a strange behavior because for both ADAMW and ASGD there seems to be overfitting after the second or third epoch as validation recalls do not improve. Having said this, models trained for 5 epochs provide huge improvements on sf_xs dataset as shown in figure 3.

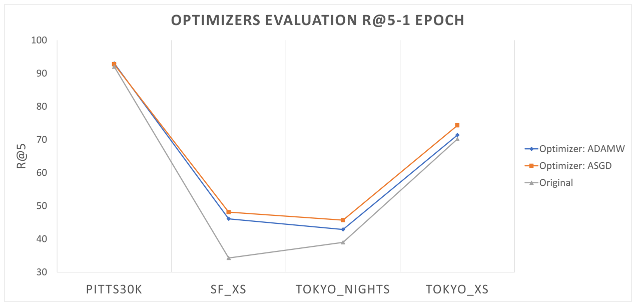

Fig. 2: Recall@5 Evaluation with different optimizers using models trained for 1 epoch

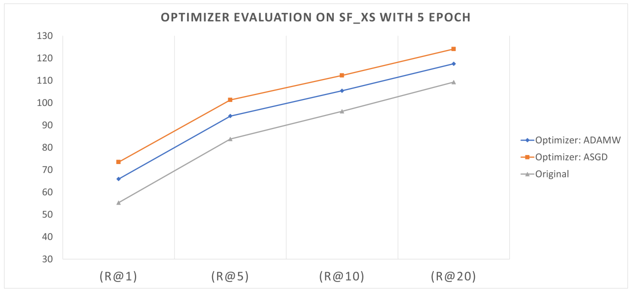

Fig. 3: Recalls on sf_xs with different optimizers using models trained for 5 epochs

TABLE I: Multi-Scale Extension Results

Dataset Used	Aggregation Method	R@1		R@5		R@10		R@20	
		Multiscale	*Vanilla*	*Multiscale*	*Vanilla*	*Multiscale*	*Vanilla*	*Multiscale*	*Vanilla*
pitts30k	*Gem*	76,6	75,8	88,8	88,6	92	91,7	94,3	94,3
	NetVLAD	86,1	85,7	93	92,8	95	94,9	96,8	96,4
sf_xs	*Gem*	14,1	13,5	26,3	25,2	33,2	32,1	39,3	39
	NetVLAD	36,6	34,1	51,8	49,5	57,5	55,2	62,9	61,7
tokyo_nights	*Gem*	10,5	9,5	23,8	26,7	32,4	31,4	37,1	38,1
	NetVLAD	29,5	26,7	47,5	46,7	54,3	55,2	63,8	64,8
tokyo_xs	*Gem*	34	35,9	47,6	50,5	56,5	57,1	65,1	64,4
	NetVLAD	61,6	59,7	74	74	78,7	79	84,4	83,5

TABLE II: Grid Search for Learning Rate and Weight Decay

Method	lr	wd	R@1	R@5	R@10	R@20
Vanilla	-	-	83.9	92	94.4	96.1
Optimizer: ASGD	*0.00001*	*0*	70.9	84.9	89.6	93.2
	0.0001	*0*	71.6	85.5	89.9	93.4
	0.001	*0*	76,3	88,1	91,8	94,5
	0.01	*0*	82,4	91,4	93,8	96
	0.1	*0*	**85,4**	92,6	94,6	96,1
		0.0001	**85,4**	92,6	94,5	**96,3**
		0.001	85,2	**92,7**	**94,7**	96,1
		0.01	81,3	90,7	93,5	95,7
		0.1	4,4	15,7	24,4	36,7
Optimizer: ADAMW	*0.00001*	*0.001*	84	92,1	94,4	96
	0.0001	*0*	84,8	92,1	94,3	95,9
		0.0001	84,8	92,1	94,3	95,9
		0.001	**85,1**	**93**	**95**	**96,4**
		0.01	84,9	92,8	94,6	96,2
		0.1	84,5	92,2	94,4	96
	0.001	*0.001*	79,4	89	92	94,3
	0.01	*0.001*	67,5	83,3	87,9	92
	0.1	*0.001*	11	26,2	36,9	50,2

TABLE III: Results from grid search

Method	lr	wd	R@1	R@5	R@10	R@20
Vanilla	-	-	83.9	92	94.4	96.1
ADAMW	***0.0001***	***0.001***	85.1	**93**	**95**	**96.4**
ASGD	***0.1***	***0.001***	**85.2**	92.7	94.7	96.1

Once we have found the best optimizers (with their hyperparameters), they will be combined with Plateau and CosineAnnealing schedulers. In fig 4, it can be shown that the combination ADAMW optimizer - Plateau scheduler gives better perfomance than only applying the scheduler. Similarly, in fig 5, it is demonstrated that model works better if we add CosineAnnealing scheduler to ASGD optimizer.

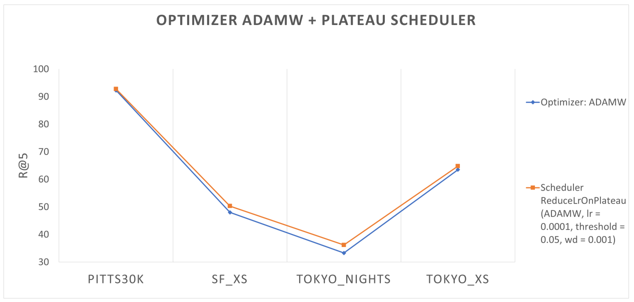

Fig. 4: Recall@5 Evaluation ADAMW optimizer - Plateau scheduler

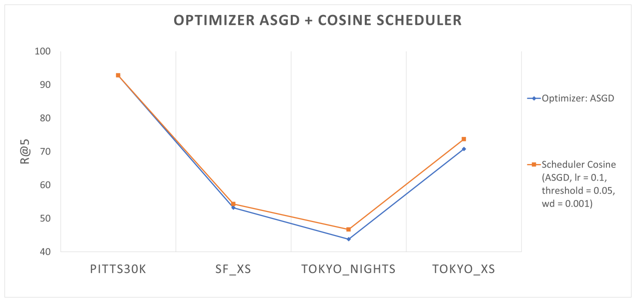

Fig. 5: Recall@5 Evaluation ASGD optimizer - CosineAnnealing scheduler

B. Testing Examples

- **Multi-scale testing**. For analyzing the performance of this extension we have evaluated the model with and without the extension on the four datasets, with a common backbone model trained for 5 epochs. Results are shown in table I. Multi-scale testing extension provides relevant improvements in both pitts30k and sf_xs datasets, establishing the new state-of-the-art performance. This extension is not focused on improvements for night images and, therefore, results in tokyo_nights and tokyo_xs datasets are not significant.
- **Ensembling**. As explained above in the methodology section, we have done two experiments: simple concatenation of descriptors obtained from different models (GeM and NetVLAD), and concatenation after computing PCA on NetVLAD's descriptors. Results for the experiments are shown in tables IV and V, respectively. The performance of the new method improves in all datasets with respect to GeM vanilla method, but it provides worse results compared to NetVLAD vanilla method. However, it seems reasonable because, since GeM's performance is worse than NetVLAD's, the ensembling method acts as if it was averaging both models. In order to balance the weight of each model, we have decided to apply PCA to NetVLAD's descriptors. Nevertheless, even though we avoid the problem of neglecting GeM descriptors, the performance decreases since so much information is lost because of PCA, and NetVLAD's descriptors are less representative.

TABLE IV: Concatenation Ensembling Extension Results

Dataset Used	**Aggregation Method**	**R@1**		**R@5**		**R@10**		**R@20**	
		Vanilla	*Concat. Ensemble*	*Vanilla*	*Concat. Ensemble*	*Vanilla*	*Concat. Ensemble*	*Vanilla*	*Concat. Ensemble*
pitts30k	*Gem*	75.8	85.4	88,6	92,6	91.7	94,8	94.3	96.4
	NetVLAD	85.7		92.8		94.9		96.4	
sf_xs	*Gem*	13.5	32.4	25.2	48.2	32.1	54.1	39	60.9
	NetVLAD	34.1		49.5		55.2		61.7	
tokyo_nights	*Gem*	9.5	22.9	26.7	44.8	31.4	48.6	38.1	60
	NetVLAD	26.7		46.7		55.2			
tokyo_xs	*Gem*	35.9	58.1	50.5	72.4	57.1	76.2	64.4	82.2
	NetVLAD	59.7		74		79		83.5	

TABLE V: PCA Concatenation Ensembling Extension Results

Dataset Used	**Aggregation Method**	**R@1**		**R@5**		**R@10**		**R@20**	
		Vanilla	*PCA Ensemble*	*Vanilla*	*PCA Ensemble*	*Vanilla*	*PCA Ensemble*	*Vanilla*	*PCA Ensemble*
pitts30k	*Gem*	75.8	84.2	88,6	92,3	91.7	94,7	94.3	96.2
	NetVLAD	85.7		92.8		94.9		96.4	
sf_xs	*Gem*	13.5	29.8	25.2	45.6	32.1	51.5	39	59.3
	NetVLAD	34.1		49.5		55.2		61.7	
tokyo_nights	*Gem*	9.5	21	26.7	38.1	31.4	44.8	38.1	51.4
	NetVLAD	26.7		46.7		55.2			
tokyo_xs	*Gem*	35.9	50.8	50.5	66	57.1	72.7	64.4	78.1
	NetVLAD	59.7		74		79		83.5	

TABLE VI: Data Augmentation Night Robustness Extension Results

Method Used	**tokyo_nights**				**tokyo_xs**			
	R@1	*R@5*	*R@10*	*R@20*	*R@1*	*R@5*	*R@10*	*R@20*
M1	28.6	48.6	***59***	***76.2***	57.8	74.3	80	***87.9***
M2	***34.3***	49.5	58.1	70.5	***61***	***75.6***	***80.6***	86
M3	32.4	***52.4***	58.1	74.3	58.7	73.7	80	87
M4	29.6	45.7	51.4	59	54.9	74.2	77.8	82.2
M5	26.7	39	47.6	55.2	54.6	70.8	76.5	81.3
M6	21.9	37.1	44.8	57.1	51.4	70.2	74.6	81.3
Vanilla	20	39	46.7	57.1	54.6	70.2	75.2	80.3

TABLE VII: Data Augmentation Random Perspective Extension Results

Method Used	**tokyo_nights**				**tokyo_xs**			
	R@1	*R@5*	*R@10*	*R@20*	*R@1*	*R@5*	*R@10*	*R@20*
M1	23.8	41.9	***52.4***	***62.9***	51.4	69.5	76.5	82.5
M2	25.7	***42.9***	***52.4***	61	52.4	***72.4***	***77.5***	82.5
M3	25.7	40	***52.4***	59	53.7	70.5	***77.5***	82.2
M4	23.8	***42.9***	51.4	60	50.5	70.5	75.9	82.2
M5	***28.6***	41	50.5	60	53	72.1	76.8	82.5
M6	24.8	39	***52.4***	***62.9***	52.4	69.2	***77.5***	***83.5***
Vanilla	20	39	46.7	57.1	***54.6***	70.2	75.2	80.3

V. Conclusion

In this paper, we have proposed several extensions to Deep Visual Geo-localization Benchmark with Resnet18 as backbone and GeM and NetVLAD as aggregation layers. For the training framework, we have proposed two different data augmentation strategies: one for improving the behavior of the model against night images, and another one for improving the performance on datasets with occlusions and different perspectives. We evaluate these extensions on 4 different datasets proving significant improvements in the performance on specific datasets. We have also trained the model with new optimizers, ADAMW and ASGD, performing a grid search for finding the best learning rate and weight decay for each optimizer. Setting the parameters to these values, we have performed several experiments with these new optimizers obtaining relevant improvements on the metrics Recall@N. Trying to increase even more the performance, we have applied two different schedulers for controlling the learning rate, providing the best results. For the test framework, we have implemented a multi-scale testing pyramid in order to build richer descriptors of an image, achieving great improvements in the performance. Finally, we have implemented a way of combining two models by ensembling them. First, we have just concatenated both model descriptors, and afterward, we have tried to weight each model in a balanced way by applying PCA to equalize the descriptor's dimensions. Experiments have been performed using GeM and NetVLAD as the two models to compute the combination. Results show that the model provides something similar to an average of the models.

References

[1] Khaliq, A., Milford, M., and Garg, S. (2022, April). MultiRes-NetVLAD: Augmenting Place Recognition Training With Low-Resolution Imagery. IEEE Robotics and Automation Letters, 3882–3889. https://doi.org/10.1109/lra.2022.3147257.

[2] S. Lowry, N. Sunderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," IEEE Transactions on Robotics, vol. 32, no. 1, pp. 1–19, 2015.

[3] S. Garg, T. Fischer, and M. Milford, "Where is your place, visual place recognition?" Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, Aug 2021. [Online]. Available: http://dx.doi.org/10.24963/ijcai.2021/603.

[4] C. Masone and B. Caputo, "A survey on deep visual place recognition,"IEEE Access, vol. 9, pp. 19 516–19 547, 2021.

[5] C. Cadena, L. Carlone et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1309–1332, 2016.

[6] P.-E. Sarlin, A. Unagar, M. Larsson, H. Germain, C. Toft, V. Larsson, M. Pollefeys, V. Lepetit, L. Hammarstrand, F. Kahl, and T. Sattler, "Back to the feature: Learning robust camera localization from pixels to pose," 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3246–3256, 2021..

[7] J. L. Schonberger and J.-M. Frahm, "Structure-from-motion revisited," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 4104–4113.

[8] X. Zhang, L. Wang, and Y. Su, "Visual place recognition: A survey from deep learning perspective," Pattern Recognition, vol. 113, p. 107760, 2021.

[9] G. Trivigno, "Deep learning for sequence-based visual geolocalization," Torino TO, 2021.

[10] 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, 2017. [Online]. Available: https://doi.org/10.1177/0278364916679498.

[11] Berton, G., Mereu, R., Trivigno, G., Masone, C., Csurka, G., Sattler, T., & Caputo, B. (2022, April). Deep Visual Geo-localization Benchmark. https://doi.org/10.48550/arXiv.2204.03444

[12] Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., & Sivic, J. (2018, June). NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1437–1451. https://doi.org/10.1109/tpami.2017.2711011

[13] Zhong, Z. (2017, August 16). Random Erasing Data Augmentation. arXiv.Org. https://arxiv.org/abs/1708.04896