Image Retrieval for Visual Geolocalization: Methods and Experiments

Salvatore Junior Curello
Politecnico Di Torino
s268066@studenti.polito.it

Antonio Cirigliano
Politecnico Di Torino
s275053@studenti.polito.it

Simone Rogato
Politecnico Di Torino
s289863@studenti.polito.it

Abstract

Visual Place Recognition (VPR) is the task of identifying, with a certain grade of accuracy, the exact location where a particular photo was taken. This activity is conducted as an image retrieval task in which query images are compared to a dataset of geo-tagged images, and the hypothesis of the location is computed according to the visual similarity of pair-wise embeddings. In this paper, we employed a pretrained ResNet-18 in the VPR pipeline to understand how performance changes with respect to different state-of-the-art design choices, such as aggregators, miners, function losses, processing methods, and combinations of the previous. We conducted testing on different datasets (SF-XS and Tokyo-XS) to assess how the model settings are heavily dependent on the domain and contextual information of the images for achieving good performance. Our code is available at `https://github.com/salvatorecurello/Simple_VPR_codebase`

1. Introduction

Visual Geo-localization (VG) encompasses the challenge of precisely determining the location where a given photograph was captured. Conventionally, this task has been approached as an image retrieval problem comprising two primary stages:

- Initially, from a collection of geotagged images, a concise single-vector depiction is generated for each image and adaptively stored in memory.
- Subsequently, upon receipt of a query image, a comparative analysis is conducted against all stored image representations. This entails employing a similarity function, relying on a distance metric, to discern the most fitting matches. Through additional refinement procedures, the location corresponding to the singular best match is ultimately derived.

The single-vector image representation can be implemented using either a pooling method such as Average-Pooling or an aggregation method based on classical encodings such as GeM [14], applied to local descriptors. In the latest years, CNNs have proved to be powerful feature extractors and they are now widely used to build image place recognition models.

The primary obstacles within the image retrieval challenge predominantly have origins from the ever-changing nature of the world, encompassing a wide spectrum of natural conditions. Furthermore, images of places often include extraneous elements, such as pedestrians, vehicles, or obstructing objects, rather than featuring discernible subjects in the foreground, which have limited relevance to the task at hand. Moreover, different point-of-views of the same location can drastically alter perceptual features of the images, sometimes leading different places to unexpectedly match because of akin visual patterns.

Commencing with the present state-of-the-art foundational models tailored for the task of place geolocalization, we provide a concise exploration of potential modifications that can be integrated to address some of their inherent limitations. This endeavor draws inspiration from the extensive body of literature pertinent to this particular task.

2. Related works

In the pursuit of addressing the challenges posed by Visual Geo-localization, researchers have explored various techniques and methodologies to enhance the efficiency and effectiveness of this task.

NetVLAD [3], a derivative of VLAD with differentiable properties, has emerged as an important player in this field. It exhibits impressive capabilities by training end-to-end with the underlying CNN backbone, directly catering to place recognition. This discovery has served as the bedrock for subsequent research endeavors. However, a notable hurdle posed by NetVLAD is its production of high-dimensional descriptors, thereby amplifying memory demands within VG systems.

This challenge has driven the research into the creation of more compact descriptors. This is achieved through techniques such as dimensionality reduction or the replacement of NetVLAD with lighter pooling layers such as GeM [14], MixVPR [2], CosPlace [4] and Conv-AP [1]. The architectural structure on which the VG is based for learning im-

age embeddings is trained using distance metric learning (DML) loss functions such as Triplet loss, Contrastive loss and Multi-similarity loss functions.

3. Method

In this section we briefly describe the algorithms and methods used to build our models.

3.1. Aggregators

Aggregators play a pivotal role in creating compact and discriminative representations of features extracted from neural networks. The primary goal of aggregators is to enhance the robustness of neural networks to variations in feature position, scale, and orientation, making them a fundamental asset in the success of place recognition applications.

GeM (Generalized Mean Pooling)

Conventional pooling operations like max pooling and average pooling compute the maximum and average of the values within a pooling window respectively. On the other hand, GeM pooling generalizes this concept and computes a power mean of the values within the pooling window, where the power can be any real number greater than or equal to 0. The formula for GeM pooling is given by:

$$\text{GeM}(X) = \left(\frac{1}{|x_k|} \sum x^{p_k} \right)^{\frac{1}{p_k}} \tag{1}$$

where X is the input tensor and p are a learnable hyperparameter. The default value of p is 3, which gives good results in most cases. By allowing the value of p to be learned, GeM pooling provides a more flexible pooling operation compared to the max and average pooling, which can adapt to different image characteristics [14].

Conv-AP

The convolutional aggregation layer (Conv-AP [1]) operates in two main steps:
Channel-wise pooling: This performs a 1x1 convolution on the input feature maps $\mathbf{F}$ to reduce the channel dimension from c to d:

$$\mathbf{F}' = \mathbf{W} * \mathbf{F} \tag{2}$$

Where $\mathbf{F}'$ is the output feature maps of size $h \times w \times d$, $\mathbf{W}$ is a learnable 1x1 convolution kernel of size $d \times c \times 1 \times 1$, and $*$ denotes convolution.
This essentially compresses each $h \times w$ spatial descriptor in $\mathbf{F}$ from c-dim to d-dim.
Spatial pooling: This applies adaptive average pooling on $\mathbf{F}'$ to reduce the spatial dimensions from $h \times w$ to $s_1 \times s_2$:

$$\mathbf{z} = \text{AAP}(\mathbf{F}')$$

Where AAP splits $\mathbf{F}'$ into $s_1 \times s_2$ spatial regions and averages the descriptors in each region.
This results in the final descriptor $\mathbf{z}$ of size $(s_1 \times s_2 \times d)$, which is then flattened and L_2 normalized.
In summary, Conv-AP performs channel-wise pooling to reduce channel dimension, followed by spatial pooling to reduce spatial dimensions. The parameters d and $s_1 \times s_2$ control the size of the final descriptor.
The key advantage is that Conv-AP operates fully convolutionally on the feature maps, without discarding spatial information like global pooling. The spatial pooling keeps some coarse spatial layout making the descriptor more discriminative than methods that simply flatten or globally pool the feature maps.

CosPlace

CosPlace is a highly-robust method that allows training on large-scale amounts of data. Contrarily to other traditional approaches it needs much less memory to build a good model that generalizes extremely well to other domains.
The underlying idea of CosPlace is to divide a large image dataset into square geographical cells and then splitting them into multiple classes according to position and heading of the images. Training is then computed on CosPlace Groups, which are computed considering disjoint set of classes related to a grid of non-adjacent cells as a method to lower misclassification. Finally, this portion of the dataset is fed into a CosFace module [16] which at training time gives highly meaningful descriptors. Cosplace method consists in the iteration of this procedure on different Cosplace group one at a time [4].

MixVPR

MixVPR [2] is a new technique for aggregating global context from CNN activation maps for robust visual place recognition. It flattens the maps into vector representations $\mathbf{X}_i$ that encompass information about the entire input image, rather than treating them as local features.
These flattened maps $\mathbf{F} \in \mathbb{R}^{c \times n}$ are fed into cascaded Feature Mixer blocks, each of which incorporates relationships between features using multilayer perceptrons (MLPs):

$$\mathbf{X}_i \leftarrow \mathbf{W}_2(\sigma(\mathbf{W}_1\mathbf{X}_i)) + \mathbf{X}_i, \quad i = 1, \ldots, c \tag{3}$$

Where W_1 and W_2 are the weights of two fully connected layers that compose the MLP and σ is a non-linearity. The input X i of the MLP is then added back to the resulting projection in a skip connection.
Stacking the Feature Mixer blocks recursively enriches the global context as information is mixed across blocks.
After L blocks, the enriched features $\mathbf{Z} \in \mathbb{R}^{c \times n}$ are projected into a compact descriptor space through channel-wise

and row-wise projections:

$$\mathbf{Z}' = \mathbf{W}_d(\text{Transpose}(\mathbf{Z})) \quad \text{where} \quad \mathbf{Z}' \in \mathbb{R}^{d \times n} \tag{4}$$

$$\mathbf{O} = \mathbf{W}_r(\text{Transpose}(\mathbf{Z}')) \quad \text{where} \quad \mathbf{O} \in \mathbb{R}^{d \times r} \tag{5}$$

The final compact descriptor $\mathbf{O}$ is flattened and L_2 normalized for place recognition.
Ultimately, MixVPR provides an efficient and robust approach for blending global informations into feature maps relaxing the traditional local approach. The design choices result in state-of-the-art performance on multiple place recognition benchmarks.

3.2. Loss

Contrative loss

This function aims at maximizing the similarity between positive pairs and minimizing it for negative pairs as follows:

$$L_{\text{contrast}} = (1 - I_{ij})[S_{ij} - m]_+ - I_{ij}S_{ij} \tag{6}$$

where I_{ij} specifies whether the two given data points are similar ($I_{ij} = 1$) or dissimilar ($I_{ij} = 0$). The margin m avoids optimizing indefinitely the negative pairs that are already dissimilar enough [1] [6].
This is the loss we used in our baseline model.

Triplet loss

The triplet margin loss function is one of the most widely used ones in image retrieval tasks. It works by considering positive and negative examples in relation to an anchor (ground truth) image. It aims at minimizing the distance between matching pairs (anchor-positive) while maximizing over a margin the one between anchor and negative pairs. In this context, examples are "positive" if they represent the same place as the anchor, and "negative" otherwise. The general loss formula is given by:

$$L(a, p, n) = max(d(a, p) - d(a, n) + m, 0) \tag{7}$$

where a, p, n is a triplet made of the anchor, a positive example and a negative one, $d(\cdot)$ is a similarity function and m is the margin (typically 0.1).

Multi-Similarity loss

Wang et al. [17] recently introduced this loss function which incorporates advanced pair weighting schemes and can be calculated as follows:

$$L_{MS} = \frac{1}{N}\sum_{i=1}^{N}\left\{\frac{1}{\alpha}\log\left[1 + \sum_{j \in P_i} e^{-\alpha(S_{ij}-m)}\right] + \frac{1}{\beta}\log\left[1 + \sum_{k \in N_i} e^{-\beta(S_{ik}-m)}\right]\right\} \tag{8}$$

where for each instance z_i in the batch, P_i represents the set of indices j that form a positive pair with z_i and N_i the set of indices k that form a negative pair with z_i. Instead α, β and m represent constants (hyperparameters) that control the weighting scheme.

3.3. Miners

In conjunction with the loss function, how to select informative pairs to construct the loss function has become fundamental for more robust feature learning. Informative pairs, comprising positive and negative instances, yield substantial loss values that steer the training process [7]. The challenging use of miners relies in striking the right balance of pairs' informativeness and an overall faster convergence. Two prominent approaches dominate pair selection: offline mining, entailing high computational costs, and online mining, adaptable to the model's performance during training. In this study, we focused on online mining, with particular attention to MultiSimilarityMiner and PairMarginMiner. Our aim is to assess their impact on enhancing machine learning performance.

- The **MultiSimilarityMiner** is an approach of pair mining that aims to balance the selection of positive and negative instance pairs. By leveraging multiple similarity metrics, it identifies informative pairs within data batches. This method aims to provide dynamic and efficient pair selection, adapting during the training process.
- The **PairMarginMiner** approach involves selecting specific pairs within the data, including both positive and negative pairs. Positive pairs consist of similar instances, while negative pairs involve dissimilar instances. This method provides more direct control over pair composition but may require additional efforts for offline pair extraction.

3.4. Optimizer and Scheduler

Optimization is a critical focal point in the realm of computational mathematics and holds significant importance. In the context of deep learning, choosing the right optimization is a key aspect of model design. Remarkably, even when the dataset and model architecture remain identical, the use of different optimization algorithms can lead to disparate training outcomes.
Gradient descent stands as one of the most commonly employed optimization techniques in the realm of neural networks. Gradient descent means that, given the model parameters to be optimized $\theta \in R^d$ and the objective function $J(\theta)$, the algorithm minimizes $J(\theta)$ by back propagation updating θ of the gradient $\nabla_\theta J(\theta)$. The learning rate determines the update step size at each moment.
As for the baseline, selects **SGD** as the default optimizer algorithm with a learning rate 1×10^{-3}, and the parameter

update formula is:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla J(\theta_t) \tag{9}$$

We will introduce other three different optimizers for our experiments and fine-tune them to obtain the best learning rate and weigh decay.

- **Adam**
  Adam (Adaptive Moment Estimation) [8] computes adaptive learning rates for each parameter by combining first-order gradient moments and second-order gradient moments. It updates parameters using the formula:

$$\theta_t = \theta_{t-1} - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon} \tag{10}$$

  where θ_t is the parameter at time step t, α is the learning rate, m_t and v_t are the first and second moment estimates of the gradient, and ϵ is a small constant to prevent division by zero.
- **AdamW**
  AdamW [12] is a variant of Adam optimizer that modifies the typical implementation of weight decay by decoupling weight decay from gradient update.
- **RAdam**
  RAdam [9], short for Rectified Adam, builds upon the previous Adam optimizer by introducing a rectification step that helps stabilize training, particularly in the early stages. RAdam has gained attention for its ability to offer faster convergence and improved performance in certain scenarios.

Furthermore, we also implemented a scheduler to add a dynamic learning rate during the training process. We use a single scheduler:

- **CosineAnnealingLR**
  The CosineAnnealingLR [11] scheduler is a learning rate adjustment technique. It cyclically adjusts the learning rate using a cosine annealing schedule, gradually reducing it to a minimum value and then increasing it again in a smooth manner during training. This helps improve convergence and find better model solutions.
  Mathematically, the learning rate is given as

$$\eta_t = \eta_{\text{min}} + \frac{1}{2}(\eta_{\text{max}} - \eta_{\text{min}}) \left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\text{max}}}\pi\right)\right) \tag{11}$$

  where η_t is the current learning rate, η_{max} and η_{min} are the maximum and the minimum learning rates respectively, T_{cur} is the current number of accumulated epochs and T_{max} is the total number of iterations.

4. Experiments

In this section we firstly describe the datasets used (section 4.1) and evaluation methodology (section 4.2), and give quantitative (section 4.3) and qualitative (section 4.4) results to validate our approach. All the methodologies have been implemented with PyTorch and performed using GPU support provided by Google Colab.

4.1. Datasets

We employed the usage of the GSV-Cities dataset introduced by *Ali-Bey et al.* [1] which is a large-scale dataset suitable for the task of VPR, with a wide variety of perceptual changes over a 14-year period. It contains 529506 images, with more than 62k different places, spread across multiple cities around the globe where each place is depicted by at least 4 images (up to 20 images), as shown in Figure 1. Moreover, all places are physically distant at least 100 meters between any pair of places. The distribution of images across the 23 cities is not uniform: London, for instance, includes the highest number of images at 58,672, while Medellín has the fewest, with 6,024 images. In order to decrease the computational cost of the training process, we used a reduced version of the GSV-Cities dataset in which the resolution of each image is 224x224 with respect to 400x300 of the original one.

For both validation and testing we used the San Francisco eXtra Small (SF-XS) which is a subset of the San Francisco eXtra Large dataset [4]. It is composed of images collected by a car-mounted camera and it provides a dense coverage of the city. The database contains 27191 while the queries are 1000.

Finally, we used only for testing the Tokyo eXtra Small (Tokyo-XS) which is a subset of the Tokyo 24/7 dataset [15]. It is composed of 315 queries and 12771 database images (instead of 76k images of eXtra Large version). This dataset is very challenging since the queries were taken, using mobile phone cameras, at daytime, sunset and night, while the database images were only taken at daytime as they originate from Google Street View.

4.2. Evaluation methodology

As in [15], [5], [4], [3], [10], [5] we follow the standard place recognition evaluation technique. The query image is deemed correctly localized if at least one of the top N retrieved database images is within a certain distance of the query location. Following the common practice in the literature we used 25 meters as distance threshold from the ground truth position of the query. The percentage of correctly recognized queries (Recall) is then computed for different values of N, in our work we focus on R@1 and R@5 in order to compare models.

As starting point of our experiments, we considered a model based on ResNet-18 trained over 20 epochs using Contrastive Loss, SGD as optimizer and learning rate and weight decay set to 0.001. This set the baseline for our experiments.

Figure 1. A simple map illustrating the coordinates of the city of Barcelona, emphasizing that all locations are physically distant from each other, with a minimum separation of 100 meters between any pair of places. Furthermore, each location is represented using a number of images between 4 and 20. In this location there are five images.

	San Francisco		Tokyo	
Aggregator	R@1	R@5	R@1	R@5
Baseline (Avg)	30.60	46.80	47.61	61.58
Gem	31.00	47.30	46.03	68.89
CosPlace	30.10	45.80	45.57	66.98
ConvAP	32.80	48.80	55.87	71.11

Table 1. Result obtained considering different aggregators in combination with Contrastive Loss.

	San Francisco		Tokyo	
Removed Layers	R@1	R@5	R@1	R@5
3, 4	30.80	43.50	51.11	67.62
4	**41.10**	**55.10**	**60.63**	**74.60**
None	33.00	48.30	55.87	73.65

Table 2. MixVPR aggregator evaluated with various ResNet18 configurations.

4.3. Quantitative results

In order to search for the best possible performances, we first conducted our analysis by trying to quantitatively express the behavior of different aggregators with several metric learning loss functions. Consequentially, we conducted further experiments with miners and optimizers in order to find improvements on components that had performed well in previous couplings. All experiments have been conducted with 10 epochs, halving the training time of baseline. Firstly, we evaluated different aggregators already introduced in section 3.1. Considering the impressive performance on the task of VPR that outperformed GeM and NetVLAD and achieved state-of-the-art results on multiple benchmarks, we decided to analyze the usage of the Cosplace aggregator introduced by *Berton et al.* [4]. Then, we also evaluate the Conv-AP aggregator that in some scenario outperformed CosPlace [1]. The results are shown in Table 1.

We found that in combination with the Contrastive loss, Conv-AP performed better than other aggregators. At Tokyo-XS, it achieved a 8.26% and 15.5% increase from the baseline on the R@1 and R@5 respectively. Due to its average pooling strategy, Conv-AP is able to better leverage the spatial order present in the feature maps achieving higher performance. Consequently, referring to the results achieved by *Ali-bey at al.* [2] we have examined the MixVPR aggregator since its different mechanism is able to weight features in a holistic way. Using ResNet18 we tried different crop to the backbone in order to find the best model configuration in terms of performance and at the same time accelerate computation and reduce memory usage. As depicted in Table 2 by just removing the last residual block we obtained the best results that outperformed the previous aggregators in both datasets. As in [2], we fix L = 4 the number of stacked Feature-Mixer blocks.

Furthermore, we conducted the analysis of these aggregators with other loss functions such as Multisimilarity and Triplet loss. The combined use of MixVPR and Multisimilarity loss achieves the best result in both test datasets, as shown in Table 3 and Table 4. This can be explained by the fact that in the Multisimilarity loss, a positive pair interacts with all positive pairs and all negative pairs of the mini-batch and this mechanism well-matches with the series of Feature Mixer blocks that enhance the global relationships of the descriptors.

Despite the incredible effectiveness of the loss functions and aggregator layers applied, still the model suffers from prohibitive sample complexity when to construct pairs or triplets over all examples for the training. To tackle this problem, pairs are constructed within a mini-batch [13] in which the process is further boosted by two pair mining methods: PairMargin and Multisimilarity miner. In Table

Aggregator	San Francisco		Tokyo	
	R@1	R@5	R@1	R@5
Gem	26.90	43.60	44.44	65.08
CosPlace	25.00	39.70	41.59	64.44
ConvAP	28.00	42.80	46.35	68.89
MixVPR	**38.30**	**54.20**	**62.20**	**75.55**

Table 3. Result obtained considering different aggregators in combination with Multisimilarity Loss.

Aggregator	San Francisco		Tokyo	
	R@1	R@5	R@1	R@5
Gem	20.40	33.70	29.84	55.87
CosPlace	18.60	32.20	34.60	55.87
ConvAP	23.70	37.30	38.73	59.36
MixVPR	26.50	39.60	45.08	61.59

Table 4. Result obtained considering different aggregators in combination with Triplet Loss.

Loss	Miner	San Francisco		Tokyo	
		R@1	R@5	R@1	R@5
Contrastive	No miner	32.80	48.80	55.87	71.11
Contrastive	Multisimilarity	33.20	49.00	49.52	69.21
Contrastive	PairMargin	32.20	48.70	52.06	69.24
Multisimilarity	No miner	28.00	42.80	46.35	68.89
Multisimilarity	Multisimilarity	29.60	44.50	48.57	65.39
Multisimilarity	PairMargin	28.70	43.40	47.93	66.98

Table 5. Results obtained considering different combinations of Losses and Miners using the ConvAP aggregator.

Loss	Miner	San Francisco		Tokyo	
		R@1	R@5	R@1	R@5
Contrastive	No miner	41.10	55.10	60.63	74.60
Contrastive	Multisimilarity	38.10	53.40	58.73	74.28
Contrastive	PairMargin	39.50	52.20	60.95	75.24
Multisimilarity	No miner	38.30	54.20	62.20	75.55
Multisimilarity	Multisimilarity	36.50	51.70	53.33	72.06
Multisimilarity	PairMargin	30.30	44.90	51.68	68.57

Table 6. Results obtained considering different combinations of Losses and Miners using the MixVPR aggregator.

5 and 6 it is shown that using the adopted miners does not effectively improve performances if not for some weak increases. It can be explained by the fact that the selected pairs are not meaningful enough and do not helps to mitigate the computational and storage burden in order to reach faster convergence. As a further step in seeking better performance, we identified that smaller learning rates usually coupled with the weight decay regularization factor could bring higher results. For this aspect we have applied some optimizers such as Adam, AdamW and RAdam, as shown in Table 7.

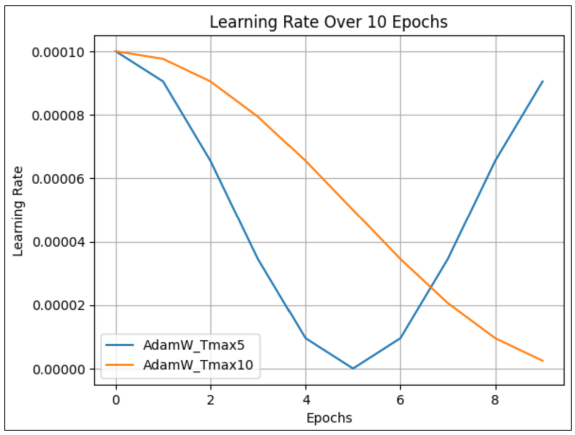

Figure 2. The changing of the learning rate over 10 epochs when we applied ConsineAnnealingLR with AdamW optimizer. At epochs T_max the learning rate is equal to zero. Similar behavior is obtained using RAdam.

Finally, in order to improve the training process we applied the CosineAnnealingLR scheduler. Its task is to adjusting the learning rate over time in a cyclical and smooth manner. Since the learning rate determines how quickly a model learns and how well it converges to an optimal solution, a fixed learning rate may not be ideal during training. CosineAnnealingLR provides a smooth and gradual transition of the learning rate, helping to prevent the model from overshooting optimal solutions and getting stuck in local minima. We applied this method with AdamW and RAdam obtaining a slight improvement, as shown in Figure 2.
The best overall performance has been given from the combination of MixVPR, Multisimilarity loss, AdamW and CosineAnnealingLR achieving an increase of 21.05% R@1 (22.86%) and 18.30% R@5 (22.23%) in the SF-XS (Tokyo-XS) dataset w.r.t. the baseline.

4.4. Qualitative results

Although the two test datasets have about the same image collection method, a difference in urban context and variation of lighting conditions could have led to different results. The best performance, as previously analyzed, is achieved by the combination of MixVPR aggregator and Multisimilarity loss inasmuch their capability to grasp contextual information at a general level is able to depict marginal differences of the embeddings. Since Tokyo-XS dataset is mostly composed of places captured at three different times of a day, usually the point-of-view of the shoot is preserved and since the model can handle well the challenge of different light conditions, predictions were more trivial on Tokyo-XS compared to San Francisco dataset which did not benefit of this property. As shown in Figure 3, we can visually appreciate the difference between of Conv-AP and MixVPR in retrieving the right reference of

Loss	Miner	Optimizer	Learning Rate	Weight Decay	San Francisco		Tokyo	
					R@1	R@5	R@1	R@5
Contrastive	No miner	Adam	0.0001	0.001	36.80	49.90	52.38	68.89
Contrastive	No miner	AdamW	0.0001	0.001	43.30	56.30	60.63	73.65
Contrastive	PairMargin	AdamW	0.0001	0.001	42.80	55.50	60.95	76.51
Contrastive	PairMargin	SGD	0.0001	0.001	21.90	33.80	42.85	59.68
Multisimilarity	No miner	AdamW	0.0001	0.001	51.70	63.90	65.40	82.22
Multisimilarity	No miner	AdamW	0.0001	0.0001	46.60	63.10	64.62	81.27
Multisimilarity	No miner	AdamW	0.0001	0.01	50.70	64.60	66.98	79.68
Multisimilarity	No miner	AdamW	0.0001 (*10)	0.001	49.80	64.90	70.16	83.17
Multisimilarity	No miner	AdamW	0.0001 (*5)	0.001	**51.72**	**65.10**	**70.47**	**83.81**
Multisimilarity	No miner	RAdam	0.001	0	43.00	57.70	54.28	66.67
Multisimilarity	No miner	RAdam	0.0001	0	51.30	64.31	68.25	79.68
Multisimilarity	No miner	RAdam	0.0001 (*5)	0	49.30	65.70	68.89	82.54

Table 7. Results obtained evaluating some of the best configurations according different combinations of Optimizers, Learning rate and Weight decay using the MixVPR aggregator. Note that (*N) indicates that CosineAnnealingLR has been applied with T_max=N.

Figure 3. A comparison of the prediction of two queries extracted by Tokyo-XS in which it is predicted the same place during the day (on the right) and night (on the left). MixVPR aggregator is capable to give correct predictions even if the query is taken during the night while Conv-AP cannot.

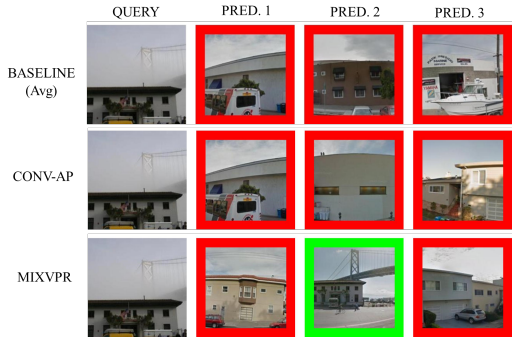

Figure 4. A query extracted from SF-XS which shows that in case of noise or weird weather condition in the image, MixVPR performs slightly better than Conv-AP.

the same query image taken at daylight and night. Moreover, San Francisco dataset is characterized by buildings with very repetitive and undistinguished geometric structures which favors the process of producing false positive couplings. Additionally, we saw that in case of nuisances mostly related to weather conditions, the model does not perform well, as shown in 4.

5. Conclusion

In this paper we present various techniques to address some issues regarding the VPR task. The choice of the aggregation module significantly impacts performance. MixVPR offers the best results and its combination with the MultiSimilarity loss function provides further performance gains. The fine-tuning of the optimizers regarding learning rates is crucial for achieving better results. Smaller learn-

ing rates with AdamW and RAdam optimizers produced the highest scores. CosineAnnealingLR scheduling further improved convergence. Our main contribution has been an increase in performance averagely of 21% w.r.t. the initial baseline at half training time. As future research could be interesting to repeat our experiments with a bigger training set since we used a reduced version of the GSV-cities dataset to fit our computational resources. This could possibly increase the performance of the CosPlace aggregator since, as mentioned by *Berton et al.* in [4], it is not suited for training on small datasets. It can be useful to analyze how the performances change by varying the distance threshold (25 meters) since a smaller value requires more precise matching. This analysis could help tuning visual place recognition systems for different applications and environments. In conclusion, our experiments revealed that the optimal combination of aggregation, loss function, mining and optimization techniques can pave the way for further research in the field of visual place recognition.

References

[1] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. GSV-cities: Toward appropriate supervised visual place recognition. *Neurocomputing*, 513:194–203, nov 2022.

[2] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. Mixvpr: Feature mixing for visual place recognition, 2023.

[3] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition, 2016.

[4] Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications, 2022.

[5] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. Deep visual geo-localization benchmark, 2023.

[6] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742, 2006.

[7] Mahmut KAYA and Hasan Şakir BİLGE. Deep metric learning: A survey. *Symmetry*, 11(9), 2019.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[9] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, 2021.

[10] Liu Liu, Hongdong Li, and Yuchao Dai. Stochastic attraction-repulsion embedding for large scale image localization, 2019.

[11] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[13] Qi Qi, Yan Yan, Xiaoyu Wang, and Tianbao Yang. A simple and effective framework for pairwise deep metric learning, 2020.

[14] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation, 2018.

[15] Akihiko Torii, Relja Arandjelović, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):257–271, 2018.

[16] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition, 2018.

[17] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning, 2020.