Image Retrieval for Visual Geolocalization: Extensions and Experiments

Francesco Gaza
s315489

Giovanni Monco
s315001

Erika Spada
s318375

Abstract

In this paper, some additional features to a framework made for Visual Geo-localization (VG) are proposed to build, train, and test a wide range of commonly used architectures, with the flexibility to change individual components of a geo-localization pipeline. The research aims to explore new methods and combinations of techniques to improve results and performances. In detail, some experiments are carried out to test various optimizers in combination with different schedulers, with the aim of choosing the optimal learning rate. Then, multiple retrieval losses are analysed and various miners are tested to select more informative batches, thus improving the learning process. An implementation and testing of a naïve proxy miner is also included in the project. The code is available at https://github.com/Monco315001/VPR$_{project}$.

1. Introduction

Visual Geo-localization (VG), also called Visual Place Recognition (VPR), is the task of determining the geographical position where a given photograph was taken. This 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 representing the known world. The top-N retrieved images, with their geotags (typically GPS coordinates), provide the hypothesis for the query's geographical location.
VG has become increasingly important in various fields, yet it remains challenging due to factors such as image variability, image noise, image scale, data sparsity, ambiguity in geo-location information and computational complexity. To tackle these challenges, researchers have developed advanced techniques that leverage image features, geospatial information, and machine learning.
At first, a baseline is provided using the Average pooling descriptor aggregator and later, from GeM pooling [9], the research aims to explore new methods and combinations of techniques to improve results and performances. In detail, the study is focused on:

- testing proposed optimizers like AdamW [7] and ASGD [8], with different schedulers such as ReduceLrOnPlateau and CosineAnnealingLR to achieve the optimal learning rate and results during training [4], [6];
- experimenting with retrieval losses such as CosFace [12], ArcFace [5] and MultiSimilarity [14];
- exploring the use of several miners to obtain more informative batches to enhance the learning process. In addition, our naive implementation of the proxy miner [2] is exploited and tested.

2. Related Work

NetVLAD has proven to be a highly effective method for visual place recognition, offering a differentiable implementation of the traditional VLAD technique [3]. It is trained end-to-end with a CNN backbone, making it a popular starting point for subsequent research in this field. It has been particularly useful when applied to large-scale datasets and images over different time periods, such as the one derived from Google Street View (GSV) cities [1]. This dataset provides the widest geographic coverage to date with highly accurate ground truth, covering more than 40 cities across all continents over a 14-year period.
Despite its effectiveness, NetVLAD has a notable drawback: the high-dimensional descriptors it generates, which result in significant memory requirements for visual place recognition systems. To address this issue, researchers have explored various alternatives, including the use of lighter pooling layers such as Generalized Mean (GeM) pooling, which provides a more compact and efficient representation while maintaining high performances [9].
Recent research has also incorporated advanced mining strategies, including proxy-based miners, alongside the usual miners [2]. These proxy-based approaches have shown to further improve performance by effectively selecting informative samples during training. The integration of these methods has resulted in more robust visual place recognition systems, demonstrating superior accuracy.

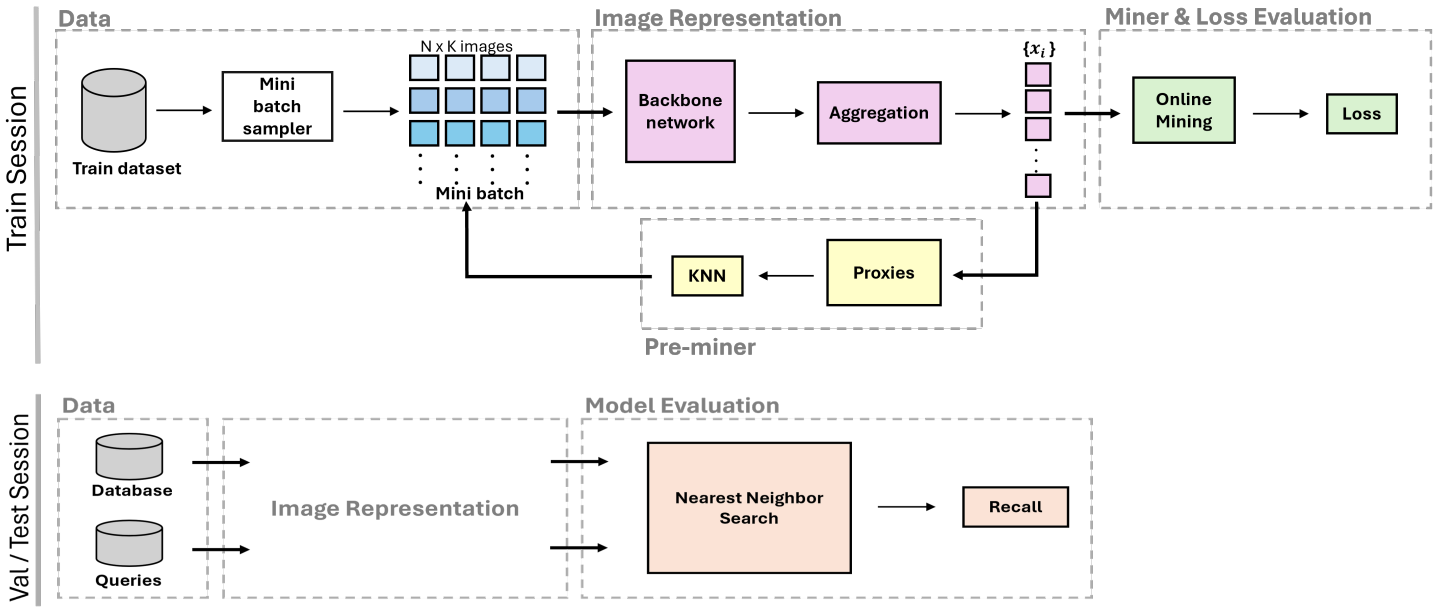

Figure 1. Visual Geolocalization pipeline.

3. Methodology

In this section, it is provided a brief overview of the configuration used in our experiments, along with a general description of the network and datasets employed for training, validation and testing.

3.1. Visual Geolocalization pipeline for training

The general pipeline for the training process of visual geolocalization using image retrieval is summarized in Fig.1. The Pre-miner step will be discussed later in Sez. 4.4.
First, the process starts with a complete dataset of geolocalized images, from which a mini-batch sampler selects a batch of images. This batch typically contains a fixed number of NxK pictures, where N denotes the number of different places and K the total amount of photos for the same place. Next, the sampled images are processed through a feature extraction network, called backbone network. The backbone extracts the relevant features from the images and converts them into a more interpretable form.
The extracted features then undergo an aggregation process that merges them into compact representations, crucials for effective comparison and matching of images.
Following the aggregation, based on similarity scores, the system may perform a technique known as online mining. This step involves selecting challenging examples, specifically sets of images called triplets. Each triplet consists of an anchor image, a set of positives (one or more images which are similar to the anchor), and a set of negatives (one or more images which are dissimilar to the anchor). Finally, these triplets are used to compute a loss value.

3.2. Visual Geolocalization pipeline for testing

The test and validation pipeline for visual geolocalization with image retrieval is designed to measure the effectiveness of a model in recognising and locating query images within a reference database. The process is divided into several steps shown in Fig.1.
Two distinct sets of images are employed: the database and the query. The database contains the reference images that are used as the basis for comparison, while the query represents the instances that the system needs to geolocalize. Both sets of images undergo the image representation step, in which through a neural network, they are processed as described in subsection 3.1.
Once the descriptors for the database and the query photos are obtained, a nearest neighbors search is performed to find the most similar images in the database. For each query image, the system returns a list of database images sorted by similarity, where the most similar images are those with the shortest distance to the query. This list is then compared with the true positives, i.e. the images in the database that are within a 25-metres radius of the query real location. Then recall@N is calculated. It measures the system's ability to correctly retrieve matching images: for each query, it is checked whether any of the first N images returned from the network's positive list contains at least one of the predefined true positive images. In other words, recall assesses whether the system is able to find correct matches within the first results returned, thus providing a measure of the accuracy of the model.

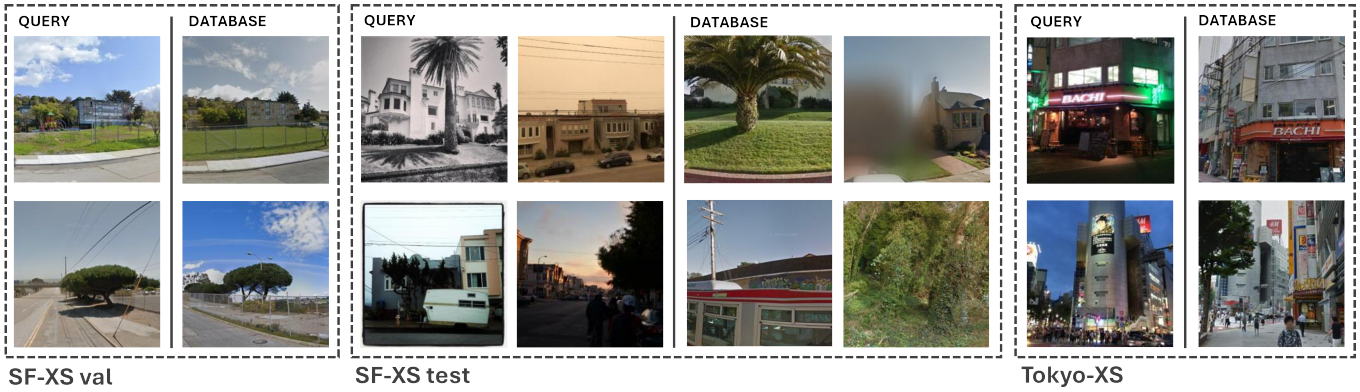

Figure 2. A collection of query and database images from datasets used in validation and testing steps. For SF-XS val and Tokyo-XS, the database and query images correspond to the same location. As regards SF-XS test instead, only the first query has a match in the database. The others have been chosen because they show the greatest noise and poor quality.

3.3. Datasets

Three highly heterogeneous datasets which together cover a variety of real-world scenarios are used. The datasets are critical for training, validation, and testing the visual geo-localization system, ensuring robust evaluation across different urban environments and conditions. Here are the details of each dataset:

- **GSV-XS dataset**: a small version of the GSV-cities dataset [1] has been used specifically for training the model. It contains a set of images from various cities, allowing the model to learn a wide range of urban features and characteristics.
- **San Francisco-small (sf-xs)**: consists of a large database of images collected by a car-mounted camera, supplemented by a significantly smaller number of query images taken by phone. It is divided into three subsets: sf-xs-train (not used in this research), sf-xs-val for validation session and sf-xs-test for testing the final performances.
- **Tokyo-xs**: a subset of the Tokyo 24/7 dataset [11]. It presents a relatively large database of images, along with a smaller number of query images. The queries are taken using mobile phone cameras at daytime, sunset and night, while the database images are only taken at daytime as they are from Google Street View.
  This dataset is used exclusively for testing.

3.4. Network details

Within the project framework, ResNet-18 truncated at Conv3 is utilized as the backbone architecture. ResNet is a deep learning architecture designed to tackle the issue of vanishing gradients in very deep neural networks. It introduces residual connections, which allow information to skip over multiple layers and flow directly from earlier layers to later ones. These connections effectively solve the vanishing gradients problem, enabling the training of much deeper networks with improved effectiveness.
As regards the aggregator instead, two types have been used:

- **Adaptive Average Pooling**: unlike traditional pooling layers, dynamically adjusts the pooling window based on input and desired output sizes, then performs average pooling within this window.
- **Generalized Mean pooling**: computes a power mean of the values within the pooling window, where the power p_k can be any real number greater than or equal to 0.
  The exact formula is:

$$f_k = \left(\frac{1}{|X_k|} \sum_{x \in X_k} x^{p_k} \right)^{\frac{1}{p_k}}$$

  where X_k is the input tensor.

3.5. Baseline assessment

To asses the baseline scores, the paper of GSVCities [1] is used as a starting point, but then is revised and tested using the technical specifications summarised in Table 1 below.

From the table, it can be seen that the first experiment utilizes **Stochastic Gradient Descent with momentum (SGD)** and **Contrastive loss**.
SGD is an iterative method for optimizing a differentiable objective function. It updates the weights and bias terms by computing the gradient of the loss function over mini-batches of data. In this context, SGD is used with momentum, a term that enhances the speed and stability of convergence. The momentum term accumulates a fraction of the

Component	Details
Backbone	ResNet18 truncated at conv3
Pretrained model	ImageNet365
Descriptor aggregator	Avg/GeM
Loss	Contrastive Loss
Miner	None
Optimizer	SGD with momentum
Scheduler	None
Learning rate	0.001
Weight decay	0.001
Momentum	0.9

Table 1. Configuration of the model for the baselines

past gradients, reducing oscillations and accelerating movement along more stable directions. At each iteration, the momentum v and the weights w are updated according to these formulas:

$$v \leftarrow \beta v + \eta \nabla L(w)$$

$$w \leftarrow w - v$$

where β is the momentum factor and η the learning rate.

As regard the loss, it works on pairs of inputs categorized into positive and negative pairs. Positive pairs $(y_{ij} = 1)$ aim to minimize the distance between similar items in the embedding space, while negative pairs $(y_{ij} = 0)$ aim to maximize the distance between dissimilar items. The loss function penalizes based on the distance between embeddings of these pairs:

$$L(x_i, x_j, y_{ij}) = 0.5(y_{ij} d^2 + (1 - y_{ij})\max(m - d, 0)^2)$$

By minimizing the distance d for positive pairs and ensuring a margin m for negative pairs, the model learns to encode meaningful representations where similar objects are clustered together.

The baselines for both Average pooling and Gem pooling are depicted in the Table 2.

	SF-XS val	SF-XS test	Tokyo-XS
Average pooling			
R@1	61.58	24.00	35.87
R@5	74.88	38.60	54.29
Gem pooling			
R@1	63.18	29.10	43.81
R@5	75.40	45.80	61.27

Table 2. Baseline scores

These results could be expected, as the SF-XS val dataset has similar queries and database images, taken during the day and with uniform quality. This consistency makes the database easier to manage and produces better results. By contrast, the Tokyo-XS dataset introduces the complexity of having query images taken with mobile phones at day and night. Although the images are similar to those in the database, the variability of light and the lower quality of queries result in worse performances. As regards SF-XS test instead, as it can be seen from the picture 2, query images are often very different from those in the database, and this problem is intensified by the presence of noise and variety in terms of colour, quality, light and subjects represented. These factors together make the geolocalization task significantly more complex and challenging for this dataset.

4. Experiments and results

In this section, further extensions and their corresponding results applied to the model are discussed. As a standard procedure, the validation part is performed using SF-XS val, while the final performances are tested on SF-XS test and Tokyo-XS. Due to the limitation of computational power, these additions are tested exclusively using the network with Gem pooling, which emerged as the optimal choice based on the baseline results. We have used Kaggle, so the setup for our tests is composed by 2-core Xeon @2.00GHz, 32GB RAM and Nvidia Tesla T4x2.

4.1. Optimizer and Scheduler

In this section, various optimization algorithms are applied during model training wiht their default parameters:

- **ASGD**
  Averaged Stochastic Gradient Descent [8]. It uses the same parameter update formula as SGD, but averages the weights that are calculated in every iteration.
- **Adam**
  Adaptive Moment Estimation. It estimates the first and second moments of the gradient to update the weights efficiently, according to the following formula:
  $$w_{t+1} = w_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$
  where $\hat{m}_t$ is the first moment estimation, $\hat{v}_t$ the second moment estimation and ϵ a correction to avoid division by zero.
- **AdamW**
  It's very similar to Adam, but differs in the way how the weight decay is implemented [7]. A correction λ that penalizes weights during the update is introduced, as shown below:
  $$w_{t+1} = w_t - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} - \lambda \eta w_t \,.$$

- **RMSProp**
  Root Mean Square Propagation, unlike Adam, does not use first-order moments and instead focuses on maintaining an exponential moving average of squared gradients [10].

The results obtained for SF-XS val are summarized in Table 3.

Optimizer	R@1	R@5
SGD (Baseline)	63.18	75.40
ASGD	43.97	60.22
Adam	**68.15**	**79.91**
AdamW	68.00	79.61
RMSProp	61.91	76.14

Table 3. Results for different optimizer using lr= 0.001 and other default parameters.

Additionally, for the best optimizer Adam, learning rates lr = [1e-05, 0.0001, 0.001] and weight decay wd = [1e-05, 0.0001, 0.001, 0.01] are tested, in order to find the most suitable values. Also, two different schedulers are used to implement dynamic learning rates:

- **ReduceLROnPlateau**
  It is a scheduling technique that monitors a metric and reduces the learning rate when the metric fails to improve. Improvement is assessed based on whether the metric increases or decreases by a predefined minimum threshold [4].

- **CosineAnnealingLR** [6]
  It is a learning rate scheduling technique that cyclically reduces the learning rate according to the cosine function:

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

In Table 4 only simulations with the most promising results are shown. The best combination found is scheduler = CosineAnnealingLR, lr = 0.0001 and wd = 0.01, values that will be used for all other simulations from now on.
As regard the results on the test datasets, they are summarized in Table 5

4.2. Losses

Loss functions play a fundamental role in training deep learning models, directly influencing the model's ability to learn useful representations and improve performance on specific tasks. Apart from the Contrastive loss, descripted in Sez.3.5, there are several other loss functions that are considered:

Lr	Wd	Scheduler	R@1	R@5
Baseline			63.18	75.40
0.0001	1e-05	None	70.5	81.5
		ReduceLROnPlateau	70.55	81.61
		CosineAnnealingLR	70.94	81.73
	0.0001	None	71.14	81.96
		ReduceLROnPlateau	70.62	81.91
		CosineAnnealingLR	71.05	81.97
	0.001	None	69.86	81.27
		ReduceLROnPlateau	69.00	80.23
		CosineAnnealingLR	71.25	82.08
	0.01	None	67.07	78.78
		ReduceLROnPlateau	66.86	79.06
		CosineAnnealingLR	**71.44**	**82.16**
0.001	1e-05	None	67.47	79.37
		ReduceLROnPlateau	69.75	81.29
		CosineAnnealingLR	70.14	81.58
	0.0001	None	65.14	77.87
		ReduceLROnPlateau	68.86	80.63
		CosineAnnealingLR	70.25	80.99
	0.001	None	60.69	74.15
		ReduceLROnPlateau	65.58	78.44
		CosineAnnealingLR	66.41	78.71
	0.01	None	48.03	64.54
		ReduceLROnPlateau	56.8	71.64
		CosineAnnealingLR	58.01	72.79

Table 4. Results for different learning rates, weight decays and schedulers.

	R@1	R@5
SF-XS test	32.00	46.80
Tokyo-XS	43.17	59.68

Table 5. Scores on test datasets with the best parameters for scheduler and optimizer obtained using the validation step.

- **CosFaceLoss**
  CosFace Loss is a variant of Softmax Loss, designed to add an angular margin to class discrimination [12]. This angular margin, denoted by γ, is crucial because it helps to increase the angle between the image feature and the correct target class vector. In essence, the goal is to push the image features closer to the desired target class vector, while separating them from the vectors of the other classes. It is formulated as follows:

$$L_c = \frac{1}{N} - \sum_i \log \frac{e^{s(\cos(\theta_{y_i})-\gamma)}}{e^{s(\cos(\theta_{y_i})-\gamma)} + \sum_{j\neq y_i} e^{s\cos(\theta_{j,i})}}$$

  with s a scaling parameter.

- **ArcFaceLoss**
  It is built on the same reasoning as CosFaceLoss, but in this case an additive margin is used on the angle itself, so instead of being $\cos(\theta_{y_i}) - \gamma$ in the formula, it will be $\cos(\theta_{y_i} + \gamma)$ [5].

- **TripletMarginLoss**
  It is based on the use of triplets of examples that include an Anchor reference example (A), an example of the same class as the anchor called Positive (P) and an example of a different class from the anchor called Negative (N). The loss aims to pull anchor and positive together, and push anchor and negative away by a margin m, according to the mathematical expression:

$$L = \frac{1}{N}\sum_i [\|f(A_i) - f(P_i)\|^2 - \|f(A_i) - f(N_i)\|^2 + m]_+$$

  where $f(\cdot)$ represents the function that maps an image into its feature space.

- **MultiSimilarityLoss**
  It aims to optimize the feature representations by leveraging multiple perspectives of similarity. The core idea is to improve the discrimination power of the embeddings by focusing on informative pairs and weighting them appropriately. This is done using the following formula:

$$L_{MS} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{\alpha}\log[1 + \sum_{k\in P_i} e^{-\alpha(S_{ik}-\lambda)}] +$$

$$+ \frac{1}{\beta}\log[1 + \sum_{k\in N_i} e^{\beta(S_{ik}-\lambda)}]$$

  where $S_{i,j}$ is the cosine similarity between embedding i and embedding j, α and β are scaling parameters that regulate the relative importance of positive and negative pairs and λ is a hyperparameter that separates positive similarities from negative ones [14].

The results obtained for SF-XS val are summarized in Table 6.

Loss	R@1	R@5
Contrastive (Baseline)	71.44	82.16
CosFaceLoss	24.74	37.83
ArcFaceLoss	23.62	37.27
TripletMarginLoss	62.85	76.90
MultiSimilarityLoss	**76.73**	**86.01**

Table 6. Results for different losses with the best parameters for scheduler and optimizer from previous step.

MultiSimilarityLoss achieves the highest accuracy, while low scores for CosFace and ArcFaceLoss may indicate the need for parameter tuning to optimize their performances.
As regard the results on the test datasets, they are shown in Table 7 below.

	R@1	R@5
SF-XS test	39.30	54.80
Tokyo-XS	47.30	60.95

Table 7. Scores on test datasets with the best parameters for scheduler, optimizer and loss obtained using the validation step.

4.3. Miners

Miners play a crucial role by facilitating the extraction of meaningful and discriminative features from visual data. They aim to identify hard examples and obtain informative batches. By focusing on these challenging and informative instances, miners enhance the learning process, enabling them to distinguish between similar-looking places and improve overall accuracy.

- **MultiSimilarityMiner** [14]
  It selects negative pairs if their similarity is greater than that of the hardest positive pair minus a margin (epsilon):

$$S_{ij}^- > \min_{\boldsymbol{y}_k = \boldsymbol{y}_i} S_{ik} - \epsilon\,.$$

  Instead, positive pairs are selected if their similarity is less than that of the hardest negative pair plus a margin (epsilon), as shown below:

$$S_{ij}^+ < \max_{\boldsymbol{y}_k \neq \boldsymbol{y}_i} S_{ik} + \epsilon.$$

  The default similarity metric used is cosine similarity.

- **TripletMarginMiner**
  It improves batch informativeness by using the TripletMarginLoss. In particular, for each anchor A, selects a positive sample P and a negative sample N such that:

$$\|f(A_i) - f(P_i)\|^2 \leq \|f(A_i) - f(N_i)\|^2 + m$$

- **AngularMiner**
  Angular miner selects the informative triplet using the angular loss [13] defined as:

$$L_{ang} = \frac{1}{N}\sum_{x_a\in B}\left\{\log\left[1 + \sum_{\substack{x_n\in B \\ y_n\neq y_a, y_p}} e^{f_{a,p,n}}\right]\right\}$$

  It filters set of images that have a loss greater than a given threshold.

To evaluate the performances of the miners, not just the best loss was considered, but all of them, and the results are shown in Table 8.

Miner	R@1	R@5
ContrastiveLoss		
MultiSimilarityMiner	70.19	82.55
TripletMarginMiner	67.42	80.27
AngularMiner	71.51	83.06
TripletMarginLoss		
MultiSimilarityMiner	63.49	77.30
TripletMarginMiner	63.67	77.88
AngularMiner	65.04	77.95
MultiSimilarityLoss		
MultiSimilarityMiner	71.98	82.63
TripletMarginMiner	73.55	83.64
AngularMiner	**74.63**	**83.76**

Table 8. Results for different losses and miners using best parameters for scheduler and optimizer from previous step.

As regard the results on the test datasets, they are shown in Table 9 below.

	R@1	R@5
SF-XS test	39.10	52.10
Tokyo-XS	48.89	61.90

Table 9. Scores on test datasets with the best parameters for scheduler, optimizer, loss and miner obtained using the validation step.

4.4. Proxy-Miner

In this section a naïve implementation of the proxy miner, based on previous work [2], is implemented and described.
This miner may actually be considered as a 'pre-miner', since it takes place before the training step and prepares the data so the training itself is more efficient and effective, as shown in Fig.1. The algorithm focuses on creating informative batches using proxies, meaningful representations of the input data. Each proxy stands for the barycentre of the representation associated with a location and is obtained by averaging the descriptors of the four photos of that place. Initially, during the first iteration, the proxy vectors are calculated with the aim of representing each place in a compact manner. Starting with the second iteration, these vectors, together with the corresponding class labels, are used to form informative batches by means of a KNN search. Each batch consists of a proxy vector and the K most similar to it.

The results obtained for all datasets, using Proxy-Miner with and without the use of AngularMiner to assess possible variations in the performance of the model, are summarized in Table 10.

	No Miner		Miner	
Dataset	**R@1**	**R@5**	**R@1**	**R@5**
SF-XS val	72.66	81.71	73.33	82.51
SF-XS test	38.40	55.10	39.90	54.60
Tokyo-XS	47.94	60.95	43.17	58.41

Table 10. Scores with the best parameters for scheduler, optimizer, loss and miner obtained from previous steps.

To further explore the extension, as performed by the paper [2], the Proxy-Miner is tested with different batch sizes and the results are summarised in Fig. 3.

5. Discussions and findings

This work tries to examine different approaches to enhance the performance of a resnet18-based model for image retrieval for visual geo-localization tasks. These experiments show how specific changes can affect, to a greater or lesser extent, the performances.
Optimizer and Scheduler Adam and AdamW have proven to be the most effective optimizers, improving performance over the benchmark without the need for parameter tuning. Furthermore, by tuning the learning rate and weight decay, Adam is able to increase performance even more. This underlines the importance of the optimal combination of learning rate and weight decay.
Therefore, a possible extension to obtain a complete and accurate framework is to explore different values of learning rate and weight decay. This could reveal optimal configurations even for optimizers that initially did not seem promising, such as ASGD and RMSProp.
Losses It emerges that ArcFaceLoss and CosFaceLoss show significantly lower performances than other functions, such as MultiSimilarityLoss, which is the best. This discrepancy could be due to the fact that ArcFaceLoss and CosFaceLoss are mainly designed to improve angle discrimination in face recognition problems. In addition, they require a precise configuration of margin and scale, and a failure to optimize these parameters may lead to a significant decrease in scores. For this reason, an in-depth tuning of the specific parameters of ArcFaceLoss and CosFaceLoss could be useful as a possible extension.
Finally, it can be seen that the loss shift has a significant impact on SF-XS test dataset, but not on Tokyo-XS.
Miners It can be seen that the use of miners on these datasets does not significantly change performances. In de-

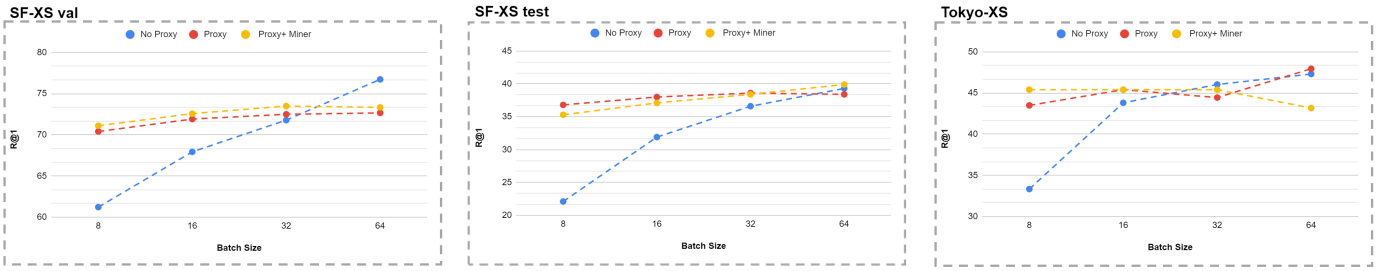

Figure 3. Recall@1 behaviour analysis on the SF-XS val dataset (left), SF-XS test (centre), Tokyo-XS (right) according to batch size, Proxy-Miner and AngularMiner.

tail, among all those tested, the AngularMiner seems the most promising and indeed, with the use of TripletMarginLoss and ContrastiveLoss, improves results slightly, while it worsens them with MultiSimilarityLoss. On the other hand, the use of AngularMiner in combination with MultiSimilarityLoss slightly betters performances on Tokyo-XS, but deteriorates it on SF-XS test set.
A possible extension could be to try and test further miners with different parameters to get a more comprehensive overview of the problem.
Proxy-Miner The Fig.3 suggests that the application of the Proxy-Miner is most effective for small batch sizes, whereas above a certain threshold, performances deteriorate. In particular, on the SF-XS val and SF-XS test datasets, the critical threshold is batch size = 32, while for the Tokyo dataset, the only value that worsens the results with the use of the Proxy-Miner is 32. In general, there is a significant increase in the scores of R@1, about +10% for all datasets, with the smallest batch size.
Combining Proxy-Miner with AngularMiner in some cases results in a slight improvement in performance, but in others it leads to a deterioration. This suggests that, even in terms of computational costs, the use of the miner is not decisive. A possible extension of the work could be the use of different training datasets to evaluate the behaviour of the preminer in a more comprehensive way.

In conclusion, the research that has been conducted provides valuable results, which, with the right computational resources, may be further improved by considering a higher number of epochs and more powerful neural networks.

References

[1] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. Gsv-cities: Toward appropriate supervised visual place recognition. 2022. 1, 3

[2] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. Global proxy-based hard mining for visual place recognition. 2023. 1, 7

[3] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. 2018. 1

[4] Francois Chollet. *Keras: Deep Learning library for Theano and TensorFlow*. Manning, 2015. 1, 5

[5] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. 2022. 1, 6

[6] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. 2017. 1, 5

[7] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. 2019. 1, 4

[8] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. 1992. 1, 4

[9] Filip Radenovic, Giorgos Tolias, and Ondrej Chum. Fine-tuning cnn image retrieval with no human annotation. 2018. 1

[10] T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop: Divide the gradient by a running average of its recent magnitude. 2012. 5

[11] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. 06 2015. 3

[12] 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. 1, 5

[13] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep metric learning with angular loss. 2017. 6

[14] Xun Wang, Xintong Han, Weilin Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. 2020. 1, 6