# Image Geo-localization Through Retrieval

Mirhamidreza Sajjadi
Politecnico di Torino
s308129@studenti.polito.it

Arash Daneshvar
Politecnico di Torino
s314415@studenti.polito.it

Mohamad Samaei
Politecnico di Torino
s314577@studenti.polito.it

## Abstract

*Visual geo-localization (VG) or Visual Place Recognition (VPR) determines the geographic location of a photograph. We trained the model on a subset of GSV-cities dataset and tested on San Francisco extra small (SF-XS) and Tokyo extra small (Tokyo-XS) datasets, which present challenges due to cultural, lighting, and perspective variations, affecting recall@N metric performance. In this paper, we compare different types of optimizers, including classic optimizers with different learning rates, momentum based optimizers and adaptive optimizers. Afterwards, the effect of schedulers was investigated on the performance. Additionally, we analyze the impact of different commonly used loss functions, such as TripletLoss, ContrastiveLoss, ArcFace, and CircleLoss, on VG performance. Our goal is to achieve a favourable result in terms of the recall@N metric. Dataset, code and trained models are available for research purposes at this link.*

## 1. Introduction

The primary objective of image geo-localization is to accurately determine the location of an exterior photo. This is achieved through the process of image retrieval, where a query image is compared against a database of images to identify visually similar places. Convolutional neural networks (CNNs) are widely employed for feature extraction in image retrieval. However, several challenges arise in this context, including variations in viewpoints, lighting conditions, the presence of moving and temporary objects, and even permanent changes in the environment over time [1]. Evaluation of VPR (Visual Place Recognition) methods often relies on datasets. However, current datasets may have certain limitations in terms of their geographical coverage, temporal coverage, sparsity, and representativeness [2]. To ensure the applicability of these methods in real-world scenarios, it is crucial to have access to large-scale and dense datasets that better capture the diversity of real-world environments. Additionally, scalability becomes a significant concern when dealing with massive amounts of data. While Contrastive learning is a commonly employed technique for VPR training, it can be computationally expensive. Hence, there is a need for alternative strategies that effectively leverage the available data resources. Overall, the field of image geo-localization and VPR research strives to overcome these challenges by enhancing the accuracy and robustness of location determination, improving dataset quality and representativeness, and developing efficient training strategies that optimize computational resources.

## 2. Related Work

### 2.1. Visual Geo-localization

Visual geo-localization is commonly regarded as an image retrieval problem, where the retrieval of images within a predefined range from the query's ground truth position is the goal [3]. Various methods utilize learned embeddings generated by feature extraction backbones like NetVLAD [4], which are trained using contrastive learning with triplet or other loss functions [3]. Mining techniques are employed to identify negative examples [5]. However, scalability issues are encountered during the training and testing phases of these methods due to expensive mining techniques and large descriptor sizes [3]. The availability of large geo-localized image datasets and the necessity for accurate localization systems in applications such as 3D reconstruction, augmented reality, and navigation have led to the growth of research on visual place recognition [6]. Visual place recognition commonly employs image retrieval techniques, which involve feature extraction, aggregation, similarity research, and candidate re-ranking. Some approaches explore 3D-based methods or transform the localization task into classification. Cross-domain and cross-view VPR are also investigated as alternative solutions [7].VG, visual localization (VL), and landmark retrieval (LR) are tasks related to computer vision. VG aims to discover the geographic location of a query image, VL focuses on estimating camera pose, and LR identifies instances of a landmark in a database [8]. VG often utilizes methods initially proposed for LR or general image retrieval (IR). Traditionally, image representations for IR are obtained through feature extraction and aggregation, with widespread usage of CNNs [7]. NetVLAD is a popular approach for VG, but it encounters challenges due to high-dimensional descriptors and memory requirements [3]. Compact descriptors, attention modules, and contextual re-weighting have been explored in VG [9]. Existing benchmarks for VG/VPR, such as VPR-Bench, either compare off-the-shelf models or prioritize practical performance [8]. In this research, a modular framework is proposed to evaluate the impact of algorithmic changes on VG systems under identical conditions [8]. The benchmark takes into account factors such as hardware-agnostic statistics (FLOPs, model size), training complexity, and storage requirements [8]. Overall, the field of visual geo-localization and VPR research aims to address scalability challenges, enhance retrieval accuracy, and optimize the utilization of available data resources through improved methodologies, compact descriptors, and comprehensive evaluation frameworks.

## 2.2. Image retrieval

Image retrieval is a task within computer vision that involves searching for images in a large database that exhibits similarity to a given query image. The primary objective of image retrieval is to identify images that align with specific interests, whether based on visual similarity or other criteria. Content-based image retrieval (CBIR) systems prioritize the extraction of features that are directly related to the content of the images. These features encompass visual characteristics such as color, texture, or shape [10]. In contrast, spatial-based image retrieval focuses on the extraction of features that capture spatial relationships or geometric properties within the images. These features take into account factors such as the spatial layout, positioning, or distribution of objects within the image. The ultimate goal is to represent images in a manner that facilitates efficient searching based on spatial information, encompassing spatial relationships, relative positions, or geometric structures [11].

## 3. Datasets

Deep neural networks require substantial amounts of data, and in our research, we utilize four datasets for this purpose. Further explanations regarding each dataset will be provided. It is worth noting that each dataset is available in both a complete version and an extra small version. For our work, we specifically utilize the extra small version of each dataset. The GSV eXtra Small (GSV-XS) dataset is employed for training, while the San Francisco eXtra Small validation (SF-XS val) dataset is utilized for validation. For testing, we use either the San Francisco eXtra Small test (SF-XS test) dataset or the Tokyo eXtra Small (Tokyo-XS) dataset. In the task at hand, our trained model generates multiple predictions for each test image, and these predictions are ranked based on their probability of matching the test image. When a prediction is accurate, it is indicated by a green line surrounding the corresponding image. On the other hand, if a prediction is incorrect, a red line is displayed. This aspect can be observed in the figures Fig. 1 and Fig. 2.

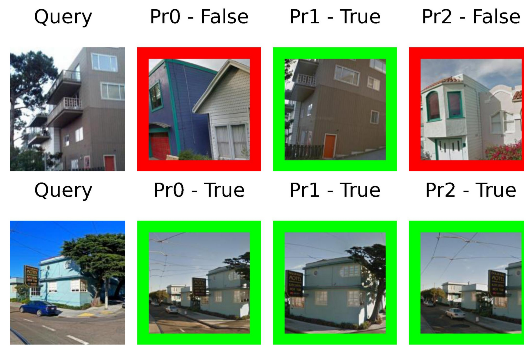

Figure 1. San Francisco dataset. The green surrounding line indicates the correct prediction and the red surrounding line is a false prediction.

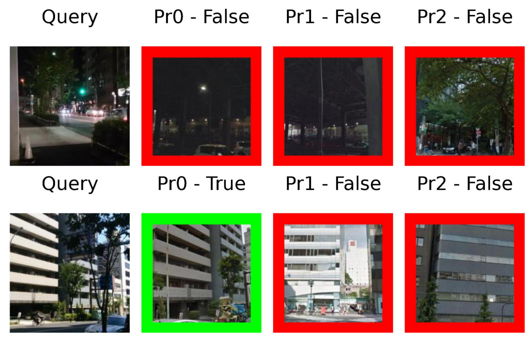

Figure 2. Tokyo dataset. The green surrounding line indicates the correct prediction and the red surrounding line is a false prediction.

## 3.1. GSV

The dataset contains a total of 560k images, all of which are associated with accurate geographical coordinates. These images are distributed across 67k different locations. Each location is represented by a collection of

| Method | R@1 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Epoch0** | **Epoch1** | **Epoch2** | **Epoch3** | **Epoch4** | **Epoch5** | **Epoch6** | **Epoch7** | **Epoch8** | **Epoch9** | **Test** |
| Tokyo without GeM | 39.7 | 46.0 | 49.4 | 51.9 | 53.8 | 54.5 | 56.0 | 56.8 | 58.0 | 59.0 | 40.0 |
| Tokyo with GeM | 39.3 | 46.6 | 50.9 | 53.8 | 55.1 | 57.0 | 58.9 | 58.8 | 59.4 | 60.0 | 43.2 |
| SF without GeM | 39.5 | 46.3 | 50.2 | 52.0 | 54.3 | 54.7 | 55.6 | 57.2 | 57.1 | 57.8 | 27.2 |
| SF with GeM | 40.4 | 47.5 | 51.0 | 53.9 | 55.7 | 57.2 | 58.9 | 59.7 | 60.2 | 61.3 | 28.3 |

Table 1. Comparison with and without GeM layer, R@1

images, ranging from 4 to 20, capturing various situations and angles. These images have been sourced from 23 different cities, covering a time period spanning from 2007 to 2021. The collection of images has been made possible through the utilization of the Google Street View Time Machine service [12].

### 3.2. SF

The dataset has been compiled by leveraging Google Street View data from 2009 to 2021. It encompasses a rich collection of images captured from various locations, providing multiple perspectives and angles. The initial collection phase involved gathering a substantial number of panoramic images, totalling around three million. These panoramic images were further processed by cropping them into 12 distinct pieces, resulting in an average of approximately 41 images per location. The dataset comprises images depicting diverse location types, including vegetation-rich areas, residential neighborhoods, and outskirts. Notably, all the images in the dataset were captured during daylight hours, ensuring consistent lighting conditions [3].

### 3.3. Tokyo

Distinguished from the other two datasets, the present dataset has been obtained through the utilization of a mobile camera. Specifically, the Tokyo dataset serves as a test set, encompassing a total of 1125 query images. These images have been sourced from 125 distinct locations situated in Tokyo, Japan. What sets this dataset apart is the fact that each location is depicted from three different viewpoints, providing multiple perspectives, and further captures have been taken at different times of the day.

## 4. Methodology

### 4.1. Base model

In this project, the ResNet-18 network is utilized as the base model for training the datasets. ResNet-18 is a widely utilized convolutional neural network (CNN) architecture in visual recognition tasks, including object detection and image classification. Its effectiveness in addressing the vanishing gradient problem lies in the introduction of residual connections, enabling the network to learn residual mappings. These connections facilitate the propagation of gradients throughout the network, simplifying the training of deep models. ResNet-18 comprises 18 layers and achieves impressive performance while maintaining a lightweight architecture compared to deeper variants. Its popularity can be attributed to its ability to handle optimization challenges and produce competitive results in visual recognition tasks. Regarding the base configuration, the SGD optimizer with a learning rate of 0.01, weight decay of 0.001, and momentum of 0.9 is employed. The loss function employed is ContrastiveLoss, introduced in PyTorch metric learning, with the parameters pos-margin set to 0 and neg-margin set to 1.

Following that, we will explain our work on the baseline. Firstly, we incorporate the GeM Layer into the network. Furthermore, we aim to compare and analyze the variations resulting from different loss functions and optimizers. To achieve this, we modify the configuration accordingly and examine the resulting outcomes.

### 4.2. GeM layer

In convolutional neural networks (CNNs), feature maps are generated by convolving filters with the input data, and the bias term also plays a role in this process. After applying filters, the resulting feature maps contain representations of different features present in the input. The pooling operation, such as max pooling or average pooling, is then applied to these feature maps in a non-overlapping manner. This pooling process involves downsampling the feature maps by taking either the maximum or average value within each pooling region. The pooled feature maps are subsequently fed into subsequent layers of the neural network for further processing and learning. By default, the project is using ResNet-18 and as a result, takes advantage of average pooling in the pooling layer. The generalized mean is a mathematical concept that extends the notion of average by adding a power parameter. Implementation of this parameter enables the pooling layer to treat every pixel in the feature map in a non-linear manner. This means that weighting schemes are applied to allow us to capture specific patterns or features. The difference between GeM pooling and average pooling is that for a specific pooling region, the generalized mean is computed as follows: Each value within the pooling region is raised to the power of ”p”.

| **Optimizer** | **R@1** | | | | | | | | | | **Test** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **00** | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | |
| Adam | 18.3 | 26.0 | 23.9 | 30.6 | 20.6 | 34.8 | 34.6 | 29.3 | 28.4 | 31.9 | 9.5 |
| AdamW | 29.8 | 38.0 | 42.6 | 46.3 | 49.2 | 51.0 | 52.2 | 53.9 | 49.9 | 55.5 | 21.9 |
| ASGD | 26.9 | 32.8 | 35.6 | 37.7 | 40.0 | 41.5 | 43.0 | 44.3 | 46.2 | 47.1 | 33.0 |
| SGD | 49.2 | 54.3 | 57.5 | 59.6 | 60.8 | 61.5 | 62.2 | 62.7 | 63.4 | 63.6 | **48.3** |

Table 2. Comparison of R@1 in different optimizers

| **Scheduler** | **R@1** | | | | | | | | | | **Test** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **00** | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | |
| CosineAnnealingLR | 48.7 | 54.3 | 56.8 | 58.4 | 59.7 | 61.1 | 61.3 | 61.9 | 62.4 | 62.5 | 44.4 |
| ReduceLrOnPlateau | 55.3 | 58.0 | 60.6 | 62.1 | 62.4 | 62.6 | 63.0 | 62.9 | 63.7 | 63.7 | 43.8 |
| None | 49.2 | 54.3 | 57.5 | 59.6 | 60.8 | 61.5 | 62.2 | 62.7 | 63.4 | 63.6 | **48.3** |

Table 3. Comparison of R@1 in different schedulers

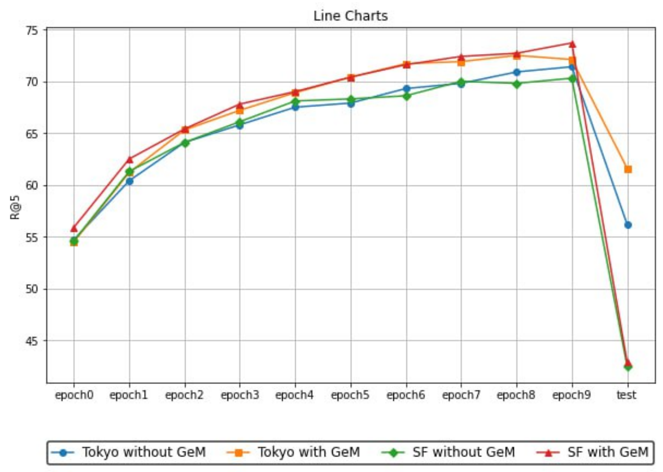

Figure 3. Comparison with and without GeM layer R@1

The raised values are averaged within the pooling region. Finally, the result is raised to the power of 1/p to obtain the generalized mean. Given an input feature map X of size (C, H, W), where C is the number of channels, H is the height, and W is the width and a power parameter p, Xij represents the value at spatial location (i, j) in the feature map X, the GeM pooling operation can be represented as

The results and comparisons are presented in Table 1 and Fig. 3, showcasing the impact of incorporating the GeM layer for the SF dataset and Tokyo dataset.

### 4.3. Optimizers and schedulers

Optimizers and schedulers play crucial roles in the training process of neural networks. Optimizers, such as Adam and SGD, determine the step size for updating model parameters during training, with SGD updating parameters faster using the current gradient and Adam incorporating adaptive learning rates and momentum. Adaptive learning rates automatically adjust the learning rate for each parameter based on gradient behavior, improving efficiency and effectiveness. AdamW extends Adam by including weight decay regularization to enhance stability and convergence. Schedulers, like CosineAnnealingLR and ReduceLROnPlateau, dynamically adjust the learning rate during training. CosineAnnealingLR reduces the learning rate using a cosine function, gradually decreasing it to near-zero before increasing it again. ReduceLROnPlateau decreases the learning rate when a specified metric stops improving for a defined patience period. Schedulers provide better control over the learning process, allowing for improved convergence and avoiding stagnation. They are particularly useful when dealing with uncertain model behavior or when fine-tuning the learning rate is essential for optimal training.

### 4.4. Loss function

A loss function measures the error between the predicted output and the target output, guiding parameter updates during training. Contrastive Loss encourages similar examples to be closer together and dissimilar examples to be further apart in the learned representation space. ArcFace introduces an angular margin to enhance discrimination by modifying the angles between feature vectors. TripletMarginLoss aims to optimize the embedding space by minimizing the distances between similar samples and increasing the distances between dissimilar samples. Circleloss encourages intra-class compactness and inter-class separability by defining circles around samples and dynamically adjusting their radii based on similarities. These loss functions contribute to learning discriminative features for tasks like face recognition.

| Learning Rate | R@1 | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | Test |
| 0.001 | 40.4 | 47.5 | 51.0 | 53.9 | 55.7 | 57.2 | 58.9 | 59.7 | 60.2 | 61.3 | 28.3 |
| 0.0025 | 49.2 | 54.3 | 57.5 | 59.6 | 60.8 | 61.5 | 62.2 | 62.7 | 63.4 | 63.6 | 48.3 |
| 0.004 | 51.7 | 57.5 | 58.9 | 60.5 | 61.4 | 61.9 | 63.0 | 62.6 | 63.3 | 62.3 | 46.3 |
| 0.00085 | 21.3 | 25.2 | 28.1 | 30.9 | 32.8 | 34.2 | 35.6 | 36.8 | 37.4 | 38.5 | 29.5 |

Table 4. Comparison of different learning rates

| Optimizer | Learning Rate | Weight Decay | Momentum |
|---|---|---|---|
| SGD | 0.0025 | 0.001 | 0.9 |
| ASGD | 0.0025 | 0.001 | 0.9 |
| Adam | 0.0025 | - | - |
| AdamW | 0.0025 | 0.001 | - |

Table 5. Hyperparameters in optimizer

## 5. Experiments and results

In the results section, we conduct a comprehensive evaluation of the model's performance by examining various combinations of optimizers and loss functions. Initially, we delve into the specifics of the optimizer adjustments, thoroughly discussing the alterations made and their corresponding impact on the model's performance. Subsequently, we provide a detailed analysis of the employed loss functions, elucidating their characteristics and their influence on the model's overall effectiveness.

### 5.1. Implementation details

In this optimizer implementation section, we discuss the utilization of four different optimizers: SGD, Adam, AdamW, and ASGD. For the SGD optimizer, as we talked about in the learning rate section, we set the learning rate to 0.0025, and the other hyperparameter is the default value, weight decay of 0.001and momentume = 0.9. Then we employ ASGD as an extension of SGD, with LR set to 0.0025 and weight decay to 0.001. After that we test our model with an adaptive optimizer and use the Adam optimizer, we employ an LR of 0.0025. Additionally, we utilize the AdamW optimizer with the LR of 0.0025 and weight decay of 0.001. Incorporating these diverse optimizers allows us to explore various strategies for optimizing our model and enhancing its overall performance.

In our report, we explored the use of different schedulers in our visual geo-localization (VG) system. We tested three situations: 'none' (no scheduler), CosineAnnealingLR, and ReduceLrOnPlateau. The default setting was 'none', followed by testing CosineAnnealingLR and ReduceLrOnPlateau. These schedulers dynamically adjust the learning rate during training to potentially enhance the system's convergence and performance. The effects of these schedulers on our VG system are discussed in detail in the subsequent sections.

In the loss implementation section, we describe the implementation of five different loss functions: ContrastiveLoss, Triplet, ArcFace and CircleLoss using Adam and SGD optimizer. In this study, SGD with LR = 0.025 has the best result in optimizer, so we fixed this element when we change the loss function. The result of these various loss functions are shown in Table 8 and Figure Fig. 6. We see that CircleLoss the function is the best performance according to the other one so we try this model with the Adam optimizer to get a better performance. But the result of this combination is not as good as the combination of CircleLoss loss function and SGD optimizer.

All the hyperparamerts are summarised in the next section.

### 5.2. Hyperparameters

All the hyperparameters considered in this project are summarized in this section in Sec. 4.4, Sec. 4.4, Tab. 7.

### 5.3. Comparison of models with different methods

When comparing the four optimization algorithms (Adam, SGD, AdamW, and ASGD), clear differences in their performance can be observed. SGD consistently outperforms the other optimizers, achieving higher recall values across all epochs, including the "Test" epoch. ASGD also demonstrates competitive performance, consistently achieving higher recalls than Adam and AdamW. While AdamW performs better than Adam in the early epochs, Adam eventually catches up and surpasses it in the later epochs. Overall, SGD and ASGD prove to be more reliable choices for optimizing the model, providing superior recall values throughout the training process. The results of these various optimizers are displayed in Table 6 and Fig. 4.

| Loss Function | Pos-margin | Neg-margin | Margin | num_classes | embedding_size | margin_scale | M | Gamma |
|---|---|---|---|---|---|---|---|---|
| ContrastiveLoss | 0 | 1 | - | - | - | - | - | - |
| Triplet | - | - | 0.3 | - | - | - | - | - |
| ArcFace | - | - | - | 62514 | 514 | 28.6 | 64 | - |
| CircleLoss with SGD | - | - | - | - | - | - | 0.25 | 64 |
| CircleLoss with Adam | - | - | - | - | - | - | 0.25 | 64 |

Table 6. Hyperparameters in the loss functions

| Schedulers | optimizer | T_max | eta_min | mode | factor | patience |
|---|---|---|---|---|---|---|
| CosineAnnealingLR | SGD | 10 | 0 | - | - | - |
| ReduceLROnPlateau | SGD | - | - | Max | 0.1 | 5 |

Table 7. Hyperparameters in the schedulers

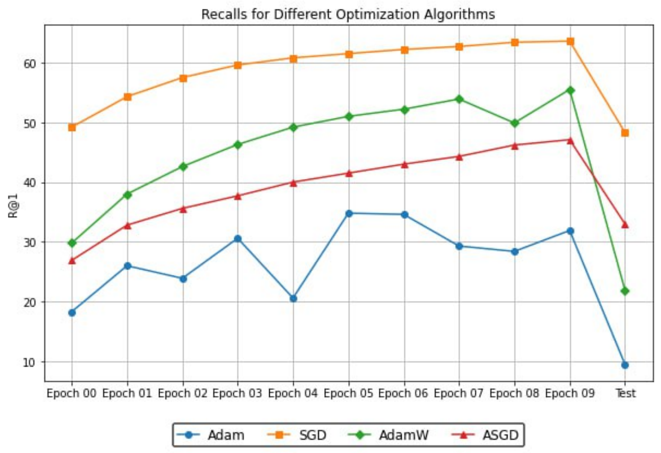

Figure 4. R@1 for different optimization algorithms

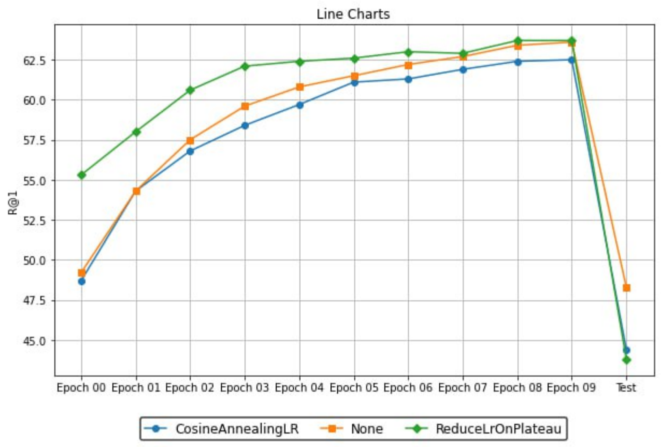

Figure 5. R@1 for different schedulers

The table displays R@1 for different schedulers across multiple epochs. Three schedulers, namely CosineAnnealingLR, None, and ReduceLrOnPlateau are compared based on their performance. CosineAnnealingLR starts with a recall of 48.7 in Epoch 01 and gradually increases to 62.5 in the final epoch. None shows a similar trend, starting at 49.2 in Epoch 01 and reaching 63.6 in the last epoch. The ReduceLrOnPlateau scheduler performs the best among the three, starting at a higher recall of 55.3 in Epoch 01 and consistently improving to achieve recalls of 63.7 in both Epoch 09 and the Test epoch. Overall, ReduceLrOnPlateau exhibits superior performance compared to CosineAnnealingLR and None, providing higher recall values throughout the epochs. The choice of scheduler should be based on the specific requirements and characteristics of the problem at hand, with ReduceLrOnPlateau being a promising option for achieving better performance. The results of these various schedulers are displayed in Table 7 and Figure Fig. 5.

The results of these various loss functions are displayed in Table 8 and Figure Fig. 6.

## 6. Conclusion

In conclusion, our study focused on the task of VG through retrieval, utilizing the GSV-cities dataset and evaluating its performance on the San Francisco extra small (SF-XS) and Tokyo extra small (Tokyo-XS) datasets. By examining various combinations of loss functions and optimizers, we discovered that employing the ContrastiveLoss function with specific hyperparameters (pos-margin=0, neg-margin=1) and utilizing the SGD optimizer with a hyperparameter of LR=0.0025 yielded superior R@1 and

| **Loss Function** | **R@1** | | | | | | | | | | **Test** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **00** | **01** | **02** | **03** | **04** | **05** | **06** | **07** | **08** | **09** | |
| TripletLoss | 44.0 | 50.3 | 53.3 | 54.7 | 56.5 | 58.3 | 58.8 | 60.7 | 60.6 | 60.8 | 32.4 |
| ContrastiveLoss | 49.2 | 54.3 | 57.5 | 59.6 | 60.8 | 61.5 | 62.2 | 62.7 | 63.4 | 63.6 | **48.3** |
| ArcFaceLoss | 10.1 | 12.2 | 14.1 | 15.9 | 17.6 | 19.6 | 20.5 | 21.9 | 23.7 | 24.8 | 15.2 |
| CircleLoss with Adam | 52.3 | 59.0 | 61.9 | 63.2 | 64.5 | 65.3 | 65.9 | 66.4 | 67.0 | 67.3 | 44.1 |
| CircleLoss with SGD | 63.6 | 65.8 | 66.8 | 62.7 | 67.6 | 67.7 | 67.8 | 68.3 | 68.1 | 68.9 | 37.8 |

Table 8. Comparison of R@1 in different loss functions

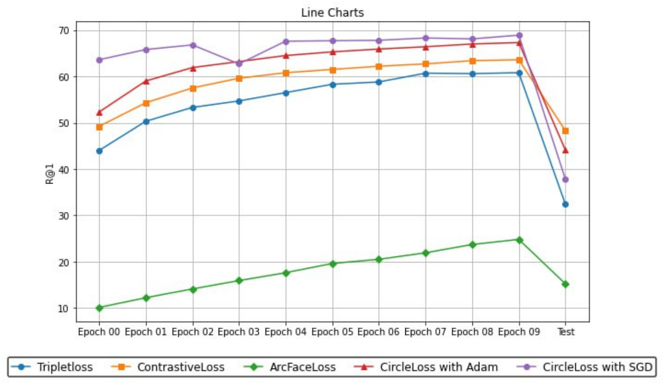

Figure 6. R@1 for different loss functions

R@5 results compared to other combinations, as demonstrated in the results section.

## 7. References

[1] A. R. Zamir, A. Hakeem, L. Van Gool, M. Shah, and R. Szeliski, *Introduction to large-scale visual geo-localization*. Springer, 2016. 1

[2] F. Warburg, S. Hauberg, M. Lopez-Antequera, P. Gargallo, Y. Kuang, and J. Civera, "Mapillary street-level sequences: A dataset for lifelong place recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2626–2635. 1

[3] G. Berton, C. Masone, and B. Caputo, "Rethinking visual geo-localization for large-scale applications," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4878–4888. 1, 2, 3

[4] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 5297–5307. 1

[5] J. Liu, K. Wang, and B. C. Fung, "Mining high utility patterns in one phase without generating candidates," *IEEE Transactions on knowledge and data engineering*, vol. 28, no. 5, pp. 1245–1257, 2015. 1

[6] A. Torii, J. Sivic, T. Pajdla, and M. Okutomi, "Visual place recognition with repetitive structures," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 883–890. 1

[7] V. Paolicelli, "Deep learning for visual place recognition large scale software and self-supervised approach to geo-localize a given photo," Master's thesis, Politecnico di Torino, Turin, Italy, 2020. 2

[8] G. Berton, R. Mereu, G. Trivigno, C. Masone, G. Csurka, T. Sattler, and B. Caputo, "Deep visual geo-localization benchmark," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5396–5407. 2

[9] H. Jin Kim, E. Dunn, and J.-M. Frahm, "Learned contextual feature reweighting for image geo-localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2136–2145. 2

[10] J. R. Smith and S.-F. Chang, "Visualseek: a fully automated content-based image query system," in *Proceedings of the fourth ACM international conference on Multimedia*, 1997, pp. 87–98. 2

[11] A. K. Yeung and G. B. Hall, *Spatial database systems: design, implementation and project management*. Springer Science & Business Media, 2007, vol. 87. 2

[12] A. Ali-bey, B. Chaib-draa, and P. Giguère, "Gsv-cities: Toward appropriate supervised visual place recognition," *Neurocomputing*, vol. 513, pp. 194–203, 2022. 3