# Image Geo-localization Through Retrieval

Farhad Yousefi Razin
Politecnico di Torino
s310027@studenti.polito.it

Hamed Goldoust
Politecnico di Torino
s309779@studenti.polito.it

Shayan Bagherpour
Politecnico di Torino
s313439@studenti.polito.it

## Abstract

*Visual Place Recognition (VPR), also known as Visual Geo-localization (VG), aims to determine the geographic location of photographs. This study develops and evaluates a VPR model using the ResNet-18 architecture on the GSV-XS cities training dataset. Evaluations were conducted on the SF-XS (test) and Tokyo-XS test datasets, with validation on the SF-XS (validation) dataset. Initially, the model did not include the GeM pooling layer, but was incorporated in subsequent executions. The research explores different loss functions, optimizers, and schedulers to enhance recall@N scores. Various data augmentation techniques—nightlight conversion, horizontal flipping, and color jitter—were tested, each demonstrating varied impacts across datasets. The objective is to achieve favorable recall@N metrics, highlighting the importance of these enhancements in improving VPR model performance. The project is available at this link.*

## 1. Introduction

Image geo-localization involves estimating the geographic locations where outdoor photos were captured, by querying a database to discover common locations. Recent works have exploited Convolutional Neural Networks (CNNs) in image geo-localization to enhance performance; however, the task remains challenging due to variations in image viewpoints, lighting conditions, and environment changes over time [17]. Therefore, current Visual Place Recognition (VPR) datasets for evaluating methods are limited in their deployment due to geographical and temporal constraints, sparse representations, and lack of representability [1].

In order to make these methods applicable to real-world scenarios, it is essential to develop large-scale dense datasets that capture a diverse range of environmental conditions. Scalability is also crucial, since methods must be able to cope with datasets of increasing magnitude. Moreover, the data requirements for computer vision tasks are large and have implications for data storage and handling, which needs to be addressed for visual geo-localization to advance. Our work aims to improve the robustness of location determination. To enhance model performance, we have explored various strategies. Additionally, we use different data augmentation techniques that simulate diverse conditions to improve robustness of the model.

## 2. Related Works

### 2.1. Visual Geo Localization

Visual geo-localization tackles the retrieval of images based on their geographic proximity to a query image, employing deep learning architectures like ResNet-18 for efficient feature extraction [2]. However, scalability remains a significant obstacle due to the computational intensity of feature extraction and similarity search algorithms. Recent advancements aim to optimize computational efficiency while ensuring robust performance across diverse datasets and environmental conditions [11].

Cross-domain adaptation is crucial in visual geo-localization to address challenges such as viewpoint variations and environmental changes. Techniques that generalize well across different datasets and conditions are essential for reliable performance in real-world applications. Evaluation benchmarks like VPR-Bench play a critical role in assessing the performance of visual geo-localization models [3]. These benchmarks provide standardized metrics and datasets for comparative analysis, guiding the development of effective algorithms and evaluation frameworks [8].

### 2.2. Image retrieval

Image retrieval, a fundamental practice in computer vision, involves retrieving relevant images from a database based on a query image. Content-Based Image Retrieval (CBIR) systems extract features directly related to image content, such as color, texture, and shape. In contrast, spatial-based image retrieval emphasizes spatial relationships and geometric characteristics, including object layout and positioning [16].

The objective is to represent images in a way that sup-

ports spatial search criteria, leveraging spatial relations and geometrical structures within images. Techniques developed for visual localization and landmark retrieval often contribute to visual geo-localization, adopting feature extraction and aggregation methods to enhance geolocation performance [15].

### 2.3. Deep Learning Approaches in VG

Deep learning has significantly impacted VG, with architectures like ResNet-18 playing a pivotal role. ResNet-18's residual connections facilitate effective gradient flow, making it suitable for tasks that require both accuracy and computational efficiency. In VG, ResNet-18 serves as a robust backbone for feature extraction, integrating with techniques such as generalized mean pooling (GeM) layer [7].

### 2.4. Enhancements in model training

Studies have led to improved performance metrics by refining how models learn and adjust. Advanced training techniques further enhance these results. Methods to simulate diverse conditions in input data help address challenges related to lighting and viewpoint variations.

## 3. Methodology

### 3.1. Datasets

Deep learning requires abundant data. In this project, we have four datasets, each available in two variations—a full version and an 'extra small' version—and for the purposes of this project, we have chosen to use the 'extra small' versions of each dataset. We use the GSV Extra Small (GSV-XS) dataset for training, the San Francisco Extra Small validation (SF-XS val) dataset for validation, and both San Francisco Extra Small test (SF-XS test) and Tokyo Extra Small (Tokyo-XS) datasets for testing. In our task, we ask the trained system to generate multiple predictions per test image. These predictions are ranked by the system probability score of the prediction matching the test image. Correct predictions are identified by a green line, while incorrect predictions are marked with a red line. This is shown in the Figure 1 and Figure 2.


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

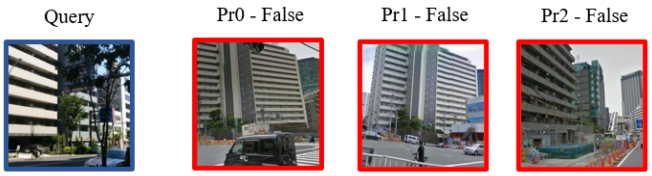


Figure 2. Tokyo dataset. The green surrounding line indicates the correct prediction and the red surrounding line is a false prediction.

- GSV-XS Dataset: This dataset includes over 530,000 images from about 62,000 locations across 23 cities, published in 2022. Each location features 4 to 20 images from various angles.
- SF-XS Dataset: This dataset contains geotagged images captured across various locations in San Francisco, offering a diverse range of urban scenes, landmarks, and environmental conditions. It is intended for evaluating image Geo-localization methods in urban settings [1].
- Tokyo-XS Dataset: Serving as a test set,it consists of a large collection of images captured in Tokyo, Japan. The dataset includes 76,000 images in the database, while there are 315 query images captured using mobile phone cameras. It presents a significant challenge as the query images were taken during different times of the day, including daytime, sunset, and nighttime. This variation in lighting conditions makes the dataset particularly demanding for visual Geo-localization algorithms [3].

### 3.2. Base Model

The ResNet-18 network is employed as the base model, commonly utilized as a backbone in retrieval-based geo-localization tasks [4]. This model functions as a feature extractor, producing embeddings that enable the comparison of image similarities [9]. ResNet-18 effectively addresses the vanishing gradient problem through the introduction of residual connections, facilitating gradient flow and simplifying the training of deep models.

ResNet-18 is advantageous due to its lightweight architecture, making it suitable for deployment on devices with limited resources. Additionally, it benefits from the availability of pretrained weights, particularly with ImageNet, which enhances transfer learning capabilities. This model strikes a good balance between performance and computational efficiency, making it a popular choice in various visual recognition applications.

In our configuration, the SGD optimizer is utilized with a learning rate of 1e-3, and weight decay of 1e-3. The loss function employed is ContrastiveLoss from PyTorch metric

| Model | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | SF test | Tokyo test |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base Model R@1 | 18.5 | 18.5 | 18.4 | 18.4 | 18.4 | 18.3 | 18.3 | 18.3 | 18.4 | 18.4 | 3.2 | 15.8 |
| Base Model + GeM R@1 | 19.0 | 18.8 | 18.8 | 18.8 | 18.8 | 18.8 | 18.8 | 18.8 | 18.8 | 18.8 | 3.3 | 16.2 |
| Base Model R@5 | 28.3 | 28.1 | 28.1 | 28.2 | 28.2 | 28.2 | 28.2 | 28.2 | 28.2 | 28.2 | 8.9 | 34.3 |
| Base Model + GeM R@5 | 29.9 | 29.7 | 29.7 | 29.7 | 29.7 | 29.7 | 29.7 | 29.8 | 29.8 | 29.7 | 8.2 | 34.9 |

Table 1. Comparison of Base Model with and without GeM(p=3) pooling layer at R@1 and R@5 across Epochs.
Loss function: Constrastive(positive margin:0 negative margin:-1) Optimizer: SGD(LR, WD:1e-3)

learning, with the positive margin set to 0 and the negative margin set to 1. We explore variations resulting from different loss functions and optimizers by modifying the configuration and analyzing the outcomes, allowing us to evaluate the performance impact of these changes on the baseline model. Table 1 shows the recall@N scores for this model.

### 3.3. Base Model + GeM Layer

The Generalized Mean (GeM) pooling layer introduces a versatile pooling mechanism in convolutional neural networks (CNNs) [6]. Traditionally, pooling layers like max pooling and average pooling simplify feature maps while preserving crucial information. GeM pooling extends this concept by introducing a power parameter $p$, allowing for a more flexible approach to pooling.

In CNNs, pooling layers condense feature maps spatially. Max pooling selects the highest value in a region, and average pooling computes the mean. GeM pooling adapts these methods by incorporating the power parameter $p$, which alters how values within the pooling region are processed.

Mathematically, for a feature map $X$ of dimensions $(C, H, W)$, the GeM pooling operation is defined as:

$$\text{GeM}(X) = \left( \frac{1}{|R|} \sum_{(i,j) \in R} (X_{ij})^p \right)^{\frac{1}{p}} \tag{1}$$

where $R$ denotes the pooling region, $X_{ij}$ represents the value at position $(i, j)$ in $X$, and $|R|$ counts the number of elements in $R$.

GeM pooling proceeds through several steps: first, each value $X_{ij}$ in the pooling region is raised to the power $p$; next, these powered values are averaged together; finally, the average result is raised to the power of $\frac{1}{p}$.

One of the main advantages of GeM pooling lies in its flexibility. By adjusting the power parameter $p$, GeM pooling can mimic different pooling behaviors. For instance, setting $p = 1$ makes GeM pooling equivalent to average pooling, while higher values of $p$ approach max pooling characteristics. This adaptability allows GeM pooling to emphasize different features within the data, depending on the chosen value of $p$.

In practical applications like ResNet-18, which traditionally employs average pooling, incorporating GeM pooling with an appropriate $p$ value can potentially enhance the network's ability to recognize and prioritize important features in the input data. Table 1 shows the recall@N scores for this model.

### 3.4. Loss Functions

A loss function evaluates the difference between the predicted output and the target output, influencing how parameters are adjusted throughout the training process. In optimizing feature embeddings, several loss functions manage similarity and dissimilarity among samples effectively. Contrastive Loss minimizes the distance between similar samples and maximizes it between dissimilar ones using binary labels and a margin parameter. Triplet Margin Loss ensures that anchor samples are closer to positives than negatives by a specified margin, thereby enhancing class discrimination [10]. Multi-Similarity Loss captures multiple levels of similarity within embeddings, promoting both intra-class compactness and inter-class separation [5]. Circle Loss takes an innovative approach using cosine similarities, dynamically adjusting weights for hard positive and negative samples to enhance discrimination [14]. The formula of Circle Loss function is as follows:

$$L = \log \left( 1 + \sum_i e^{\gamma(\cos(\theta_i^p) - \Delta_p)} \sum_j e^{\gamma(\Delta_n - \cos(\theta_j^n))} \right) \tag{2}$$

In the Circle Loss formula (2), $\gamma$ represents a scale factor that adjusts the influence of cosine similarities $\cos(\theta_i^p)$ and $\cos(\theta_j^n)$ for positive and negative samples, respectively. The parameters $\Delta_p$ and $\Delta_n$ act as margins, defining the acceptable range of similarity. This formulation dynamically weights the contributions of hard positive and negative samples, enhancing the model's ability to discriminate between classes by optimizing the embedding space effectively. The hyperparameters used for the mentioned loss functions are provided in Table 2.

### 3.5. Optimizers and Schedulers

Optimizers play a crucial role in neural network training, as they update model parameters to minimize the loss

| | pos margin | neg margin | margin | $\alpha$ | $\beta$ | base | $M$ | Gamma |
|---|---|---|---|---|---|---|---|---|
| **contrastive Loss** | 0 | 1 | - | - | - | - | - | - |
| **Triplet margin** | - | - | 1 | - | - | - | - | - |
| **Multi Similarity** | - | - | - | 2 | 40 | 0.5 | - | - |
| **circle Loss** | - | - | - | - | - | - | 0.125 | 64 |

Table 2. Hyperparameters of Loss functions

function. **Stochastic Gradient Descent (SGD)** employs fixed learning rates and current gradients for rapid updates. In contrast, **Adam (Adaptive Moment Estimation)** enhances SGD by incorporating adaptive learning rates and momentum. **AdamW** further extends Adam by incorporating weight decay directly into the updates, resulting in improved stability and convergence. **ASGD (Averaged Stochastic Gradient Descent)** averages parameters over multiple updates, enhancing generalization and stabilizing optimization. Each optimizer offers unique benefits, from rapid updates and adaptive learning rates to improved stability and generalization, catering to different training needs [12].

Schedulers, such as **CosineAnnealingLR** and **ReduceLROnPlateau**, dynamically adjust the learning rate during training, providing better control over the learning process. **CosineAnnealingLR** reduces the learning rate using a cosine function, gradually decreasing it to near-zero before increasing it again, which helps avoid local minimal. **ReduceLROnPlateau** decreases the learning rate when a specified metric, such as validation loss, ceases to improve for a defined patience period, thus fine-tuning the rate dynamically. These schedulers improve convergence and prevent stagnation, making them particularly useful when dealing with uncertain model behavior or when fine-tuning the learning rate is essential for optimal training [13]. The hyperparameters used for the mentioned optimizers and schedulers are provided in Table 3.

### 3.6. Data Augmentation

In this study, data augmentation techniques were employed to enhance the robustness and diversity of the training dataset, **GSV_XS**. Three types of augmentations were applied: horizontal flipping, daylight-to-nightlight transformation, and color jitter with contrast adjustment.

- **Flipping**: Horizontal flipping was used to increase the variability of the dataset by creating mirror images. This technique helps the model learn rotationally invariant features and prevents overfitting by introducing new perspectives of the same scene.

- **Daylight to Nightlight Transformation**: This augmentation simulated nighttime conditions by adjusting the color balance of the images. Scaling factors of **[0.1, 0.2, 0.5]** were applied to the red, green, and blue channels, respectively, darkening the images and emphasizing blue tones to mimic low-light conditions. This allowed the model to generalize across varying lighting environments.

- **Color Jitter with Contrast Adjustment**: To introduce further diversity, random noise within a range of **0 to 50** was added to the pixel values of each image, simulating different lighting and color conditions. Additionally, contrast adjustments were applied, with the contrast factor sampled from a normal distribution with a mean of **1.7** and a standard deviation of **0.1**. The contrast factor was clipped to a range of **[1.4, 2.0]** to maintain image quality. This dual approach helps the model learn to handle variations in lighting and contrast, improving its performance in different real-world conditions.

Each augmentation technique contributed to the overall robustness of the model by ensuring that it could generalize well across diverse environmental conditions encountered in geo-localization tasks. Figure 3 illustrates examples of the data augmentations used.


Figure 3. Examples of the data augmentations

## 4. Experiments

In this section, we present the experimental setup and results of our visual geo-localization project. We evaluate different components of the model, including loss functions, optimization algorithms, learning rate schedulers, and data augmentations, aiming to optimize the accuracy and robustness of our geo-location system.

### 4.1. Evaluation of Different Loss Functions

We start by investigating the impact of various loss functions on the performance of our geo-localization model.

| | Learning rate | Weight Decay | T_max | eta_min | mode | factor | Patience |
|---|---|---|---|---|---|---|---|
| SGD | 2.5e-3 | 1e-3 | - | - | - | - | - |
| ASGD | 2.5e-3 | 1e-3 | - | - | - | - | - |
| Adam | 2.5e-3 | 1e-3 | - | - | - | - | - |
| AdamW(1) | 2.5e-3 | 1e-3 | - | - | - | - | - |
| AdamW(2) | 5e-3 | 5e-4 | - | - | - | - | - |
| Cosine Annealing LR | - | - | 10 | 1e-6 | - | - | - |
| Reduce LR on Plateau | - | - | - | - | Max | 0.1 | 5 |

Table 3. Hyperparameters of Optimizers and Scedulers

Table 4 summarizes the results obtained using Contrastive Loss, Triplet Margin Loss, Multisimilarity Loss, and Circle Loss functions. Figure 4 illustrates the recall@N scores for epochs of validation data.

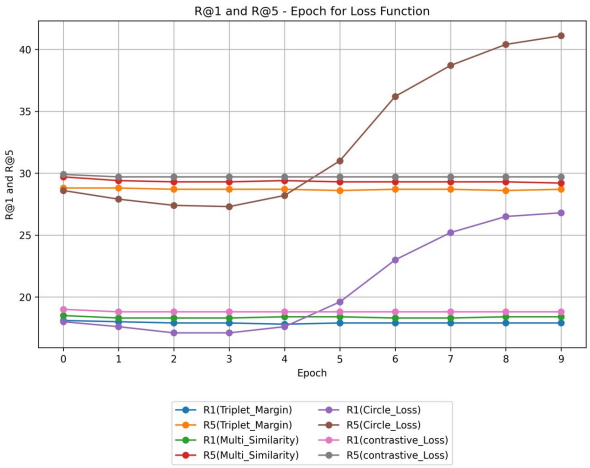


Figure 4. Evaluation on different Loss functions in the base model with GeM pooling layer. SGD optimizer (LR,WD: 1e-3) is used

## 4.2. Optimization Algorithms Comparison

Next, we analyze the performance of different optimization algorithms in conjunction with the Circle Loss function. We compare Stochastic Gradient Descent (SGD), Accelerated Stochastic Gradient Descent (ASGD), Adam, and AdamW with varying initial learning rates and weight decays. Table 5 summarizes the results obtained using different optimizers. Figure 5 illustrates the recall@N scores for epochs of validation data.

## 4.3. Evaluation of Learning Rate Schedulers

Subsequently, we evaluate the effectiveness of learning rate schedulers in improving model convergence and performance. Specifically, we examine the outcomes using schedulers such as CosineAnnealingLR and ReduceLROnPlateau with the Circle Loss and AdamW(1) optimizer. Table 6

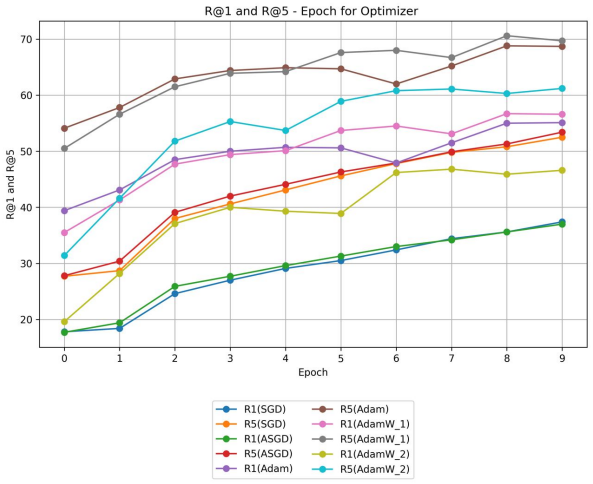


Figure 5. Evaluation on different Optimizeres with hyperparameters defined in table 3. Other details: GeM pooling layer, and Circle Loss function are used

summarizes the results obtained using scedulers. Fig.6 illustrates the recall@N scores for epochs of validation data.

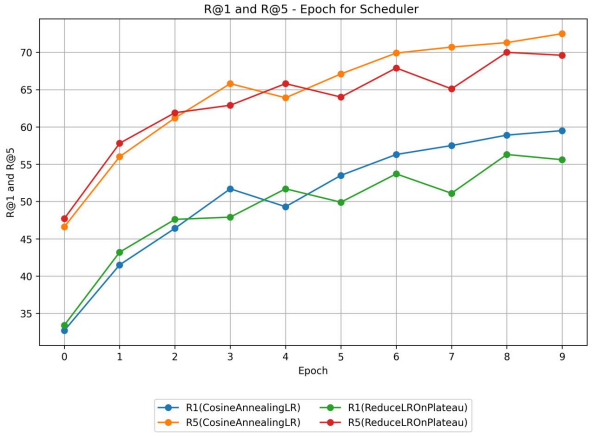


Figure 6. Evaluation on different scedulers with hyperparameters defined in table 3. Other details: GeM pooling layer, AdamW(1) optimizer, and Circle Loss function are used

### 4.4. Impact of Data Augmentations

Finally, we assess the impact of different data augmentation techniques on the model's accuracy and robustness. We investigate the effects of nightlight augmentation, flipped images, and color jitter when combined with Circle Loss, AdamW (initial learning rate and weight decay), and CosineAnnealingLR. Table 7 summarizes the results obtained using data augumentations. Figure 7 illustrates the recall@N scores for epochs of validation data.

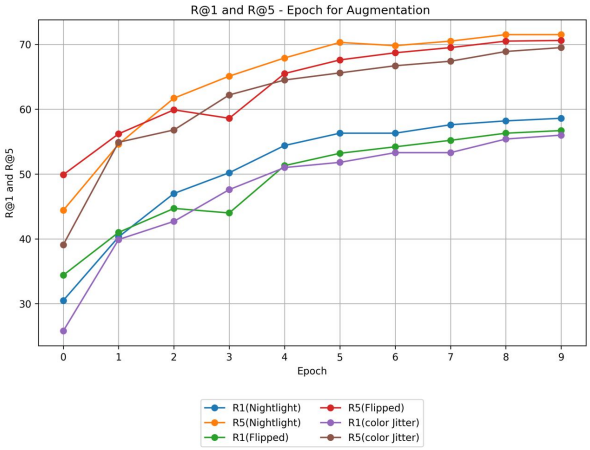


Figure 7. Evaluation on data augumentations with. Other details: GeM pooling layer, AdamW(1) optimizer, CosineAnnealingLR sceduler, and Circle Loss function are used

## 5. Results

We explored various techniques to enhance the performance of the visual geo-localization model. Through experimentation, we determined that the combination of the Generalized Mean (GeM) pooling layer with $p = 3$, Circle Loss function with parameters $m = 0.25$ and $\gamma = 64$, AdamW optimizer with a learning rate of $2.5 \times 10^{-3}$ and weight decay of $1 \times 10^{-3}$, and Cosine Annealing Learning Rate Scheduler with $T_{\text{max}} = 10$ and $\eta_{\text{min}} = 1 \times 10^{-6}$ yielded the best Recall@N scores for both the validation and test datasets.

Furthermore, our investigation into data augmentations revealed varied impacts. While data augmentations did not significantly benefit the *sf_xs_val* and *sf_sx_test* datasets, they demonstrated improvements for the *tokyo_xs* test dataset. Notably, all augmentations positively impacted the performance metrics of the *tokyo_xs* dataset, with the exception of the Recall@5 score for the color jitter contrast adjustment.

## 6. Conclusion

This study aimed to enhance the performance of our visual geo-localization model through various strategies and techniques. By systematically testing and optimizing different components and methods, we identified key factors that significantly improve model performance.

Our results demonstrate that a thoughtful selection and combination of these strategies lead to more accurate and robust visual geo-localization systems. The integration of appropriate techniques and optimizations, along with effective data augmentation, resulted in improved recall rates, highlighting their importance in the training process. Overall, the optimal combination of hyperparameters and data augmentations substantially enhanced the model's geo-localization capabilities.

## References


[1] A. Ali-bey, B. Chaib-draa, and P. Giguère. Mixvpr: Feature mixing for visual place recognition. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2997–3006, 2023. 1, 2

[2] Gabriele Berton, C. Masone, and B. Caputo. Rethinking visual geo-localization for large-scale applications. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4868–4878, 2022. 1

[3] Gabriele Berton, R. Mereu, Gabriele Trivigno, C. Masone, G. Csurka, Torsten Sattler, and Barbara Caputo. Deep visual geo-localization benchmark. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5386–5397, 2022. 1, 2

[4] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems*, pages 2787–2795, 2013. 2

[5] J. Che, Y. He, and J. Wu. Pedestrian multiple-object tracking based on fairmot and circle loss. *Scientific Reports*, 13, 2023. 3

[6] Y. Feng, F. Wu, Y. Ji, X. Jing, and J. Yu. Deep metric learning with triplet-margin-center loss for sketch face recognition. *IEICE Transactions on Information and Systems*, 103-D:2394–2397, 2020. 3

[7] F. Gu, K. Jiang, X. Hu, and J. Yang. Deep learning-based image geolocation for travel recommendation via multi-task learning. *J. Circuits Syst. Comput.*, 31:2250127:1–2250127:19, 2022. 2

[8] M. Karimian-kelishadrokhi, M. Ghattaei, and S. Fekri-Ershad. Innovative local texture descriptor in joint of human-based color features for content-based image retrieval. *Signal, Image and Video Processing*, 17:4009–4017, 2023. 1

[9] B. Ko, H. Kim, B. Heo, S. Yun, S. Chun, G. Gu, and W. Kim. Group generalized mean pooling for vision transformer. *ArXiv*, abs/2212.04114, 2022. 2

[10] X. Li, S. Liu, Y. Li, H. Liu, J. Zhao, L. Feng, G. Lao, and G. Li. Spatial-temporal attention network with multi-

similarity loss for fine-grained skeleton-based action recognition. pages 620–631, 2021. 3

[11] K. Liu and B. C. Fung. Mining high utility patterns in one phase without generating candidates. *IEEE Transactions on Knowledge and Data Engineering*, 28(5):1245–1257, 2015. 1

[12] I. Loshchilov and F. Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2016. 4

[13] X. Mao, M. Xie, and R. Zhu. Performance comparison of different convolutional neural network models and optimizers for dog breed classification. *Applied and Computational Engineering*, 2023. 4

[14] J. Robinson, C.-Y. Chuang, S. Sra, and S. Jegelka. Contrastive learning with hard negative samples. In *International Conference on Learning Representations*, 2020. 3

[15] R. Singh, N. Sharma, and R. Gupta. Classification and detection of corn leaf disease using resnet 18 transfer learning model. In *2023 8th International Conference on Communication and Electronics Systems (ICCES)*, 2023. 2

[16] H. Zhang, S. Luo, J. Shi, J. N. Yan, and W. Sun. Example-based spatial search at scale. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 539–551, 2022. 1

[17] Sijie Zhu, Taojiannan Yang, and Chen Chen. VIGOR: Cross-view image geo-localization beyond one-to-one retrieval. In *Computer Vision and Pattern Recognition*, Nov. 2020. 1

| Loss Function | Metric | Test Datasets | | Epochs on sf_xs Validation Dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sf_xs | tokyo_xs | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Triplet Margin | Recall@1 | 3.50 | 14.29 | 18.1 | 18.0 | 17.9 | 17.9 | 17.9 | 17.9 | 17.9 | 17.9 | 17.9 | 17.9 |
| | Recall@5 | 8.20 | 30.48 | 28.8 | 28.8 | 28.7 | 28.7 | 28.7 | 28.6 | 28.7 | 28.7 | 28.6 | 28.7 |
| Multi Similarity | Recall@1 | 3.00 | 15.56 | 18.5 | 18.3 | 18.3 | 18.3 | 18.4 | 18.4 | 18.3 | 18.3 | 18.4 | 18.4 |
| | Recall@5 | 8.30 | 32.38 | 29.7 | 29.4 | 29.3 | 29.3 | 29.4 | 29.3 | 29.3 | 29.3 | 29.3 | 29.2 |
| Circle | Recall@1 | 2.80 | 17.14 | 18.0 | 17.6 | 17.1 | 17.1 | 17.6 | 19.6 | 23.0 | 25.2 | 26.5 | 26.8 |
| | Recall@5 | 9.4 | 33.97 | 28.6 | 27.9 | 27.4 | 27.3 | 28.2 | 31.0 | 36.2 | 38.7 | 40.4 | 41.1 |

Table 4. Recall@1 and Recall@5 for Different Loss Functions using SGD(LR,WD: 1e-3) Optimizer

| Optimizer | Metric | Test Datasets | | Epochs on sf_xs Validation Dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sf_xs | tokyo_xs | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| SGD | Recall@1 | 6.00 | 25.4 | 17.8 | 18.4 | 24.6 | 27.0 | 29.1 | 30.5 | 32.4 | 34.4 | 35.6 | 37.4 |
| | Recall@5 | 13.7 | 42.2 | 27.7 | 28.7 | 38.0 | 40.6 | 43.1 | 45.6 | 47.8 | 49.8 | 50.8 | 52.5 |
| ASGD | Recall@1 | 6.60 | 23.17 | 17.7 | 19.4 | 25.9 | 27.7 | 29.6 | 31.3 | 33.0 | 34.2 | 35.6 | 37.0 |
| | Recall@5 | 14.2 | 41.59 | 27.8 | 30.4 | 39.1 | 42.0 | 44.1 | 46.3 | 47.9 | 49.9 | 51.3 | 53.4 |
| ADAM | Recall@1 | 9.6 | 24.76 | 39.4 | 43.1 | 48.5 | 50.0 | 50.7 | 50.6 | 47.9 | 51.5 | 55.0 | 55.1 |
| | Recall@5 | 18.4 | 46.66 | 54.1 | 57.8 | 62.9 | 64.4 | 64.9 | 64.7 | 62.0 | 65.2 | 68.8 | 68.7 |
| AdamW(1) | Recall@1 | 12.3 | 28.57 | 35.5 | 41.3 | 47.7 | 49.4 | 50.1 | 53.7 | 54.5 | 53.1 | 56.7 | 56.6 |
| | Recall@5 | 23.3 | 45.08 | 50.5 | 56.6 | 61.5 | 63.9 | 64.2 | 67.6 | 68.0 | 66.7 | 70.6 | 69.7 |
| AdamW(2) | Recall@1 | 6.9 | 28.32 | 19.6 | 29.2 | 37.1 | 40.0 | 39.3 | 38.9 | 46.2 | 46.8 | 45.9 | 46.6 |
| | Recall@5 | 15.1 | 37.78 | 31.4 | 41.6 | 51.8 | 55.3 | 53.7 | 52.9 | 60.8 | 61.1 | 60.3 | 61.2 |

Table 5. Recall@1 and Recall@5 for optimizers. Other details: GeM pooling layer, and Circle Loss function are used

| Scheduler | Metric | Test Datasets | | Epochs on sf_xs Validation Dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sf_xs | tokyo_xs | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| CosineAnnealingLR | Recall@1 | 13.2 | 32.06 | 32.7 | 41.5 | 46.2 | 51.7 | 49.3 | 53.5 | 56.3 | 57.5 | 58.9 | 59.5 |
| | Recall@5 | 24.7 | 48.25 | 46.6 | 56.0 | 61.2 | 65.8 | 63.9 | 67.1 | 69.9 | 70.7 | 71.3 | 72.5 |
| ReduceLrOnPlateau | Recall@1 | 11.3 | 26.18 | 33.4 | 43.2 | 47.6 | 47.9 | 51.7 | 49.9 | 53.7 | 51.1 | 56.3 | 55.6 |
| | Recall@5 | 21.7 | 42.86 | 47.7 | 57.8 | 61.9 | 62.9 | 65.8 | 64.0 | 67.7 | 65.1 | 70.0 | 69.6 |

Table 6. Recall@1 and Recall@5 for scedulers. Other details: GeM pooling layer, AdamW(1) optimizer, and Circle Loss function are used

| Scheduler | Metric | Test Datasets | | Epochs on sf_xs Validation Dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | sf_xs | tokyo_xs | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Flipped | Recall@1 | 12.2 | 35.24 | 34.4 | 41.0 | 44.7 | 44.0 | 51.3 | 53.2 | 54.2 | 55.2 | 56.3 | 56.7 |
| | Recall@5 | 24.0 | 50.48 | 49.9 | 56.2 | 59.9 | 58.6 | 65.5 | 67.6 | 68.7 | 69.5 | 70.5 | 70.6 |
| Nightlight transformation | Recall@1 | 12.1 | 33.65 | 30.5 | 40.3 | 47.0 | 60.2 | 54.4 | 56.3 | 56.3 | 57.6 | 58.2 | 58.6 |
| | Recall@5 | 23.3 | 50.79 | 44.4 | 54.6 | 61.7 | 65.1 | 67.9 | 70.3 | 69.8 | 70.5 | 71.5 | 71.5 |
| Contrast adjusment | Recall@1 | 12.1 | 28.89 | 25.8 | 39.9 | 42.7 | 47.9 | 51 | 51.8 | 53.3 | 53.3 | 55.4 | 56 |
| | Recall@5 | 22.9 | 50.16 | 39.1 | 54.9 | 56.8 | 62.2 | 64.5 | 65.6 | 66.7 | 67.4 | 68.9 | 69.5 |

Table 7. Recall@1 and Recall@5 for data augumentations. Other details: GeM pooling layer, AdamW(1) optimizer, CosineAnnealingLR sceduler, and Circle Loss function are used