MLDL Project Report
Image Geolocalization Through Retrieval

Michele Sgrillo - S304046,
Rosario Interlandi - S334202,
Yasaman Yazdanbakhsh- S327684

DAUIN, Polytechnic University of Turin, Turin, Italy
July 2024

Abstract

In this paper, we address the problem of large-scale visual place recognition, focusing on the rapid and accurate identification of the location where a picture was taken. Our model is trained on the GSV-XS dataset, a compact version of the GSV-cities dataset designed to be compatible with Google Colab resources.
ResNet-18, a classical convolutional neural network, serves as the backbone layer, while AvgPooling, GemPooling, and MixVPR are used as aggregator layers.
For validation, we employed the San Francisco eXtra Small Val (SF-XS), a subset of the SF-XL dataset, while SF-XS Test and the Tokyo eXtra Small (Tokyo-XS) subset of the Tokyo 24/7 dataset were used exclusively for testing.
We tested the effects of AVGPooling, GemPooling, and Feature Mixer in different configurations, in combination with various miners, loss functions, optimizers, and learning rate schedulers to assess their impact on the overall performance on the listed datasets. Findings are reported using the recall@1 and recall@5 metrics.
Among the tested configurations, the Mix-VPR architecture coupled with the multi-similarity loss function significantly outperformed other approaches and provided interesting results with extra small images and challenging datasets such as "San Francisco-XS Test" and "Tokyo-XS Test". Code and trained models are publicly available for research purposes at:
`https://github.com/michelesgrillo/mlexam`.

1. Introduction

Visual Place Recognition (VPR) plays a crucial role in computer vision for identifying and matching locations based on images or video frames, despite challenges posed by variations in environmental conditions, lighting, and viewpoints because it is essential for agents to navigate and understand their surroundings autonomously in a wide array of applications so robust algorithms are required for accurate feature extraction and matching.
This study adopts a classical deep learning approach for VPR, utilizing ResNet-18 as the backbone with aggregation layer. The objective is to extract features summarizing the pictures content and store them in a database to compare query images with reference images using KNN for location identification. ResNet-18 was selected for its balance between performance and memory usage, making it suitable for Google Colab Notebooks.
The GSV-Cities dataset, adapted for computational constraints in Colab is used for training. For validation and testing, specific subsets from larger datasets such as SF-XS and Tokyo-XS are utilized due to their challenging variations of poses and light conditions.
The paper also investigates the effect of various loss functions, miners, learning rate schedulers, optimizers, and aggregation modules tailored for VPR to enhance performance. This report outlines the adaptation and enhancement processes, experimental setup, and findings, aiming to enhance the reliability and effectiveness of VPR systems across different environmental conditions.

2. Related Work

The VPR task is well established research field on which lot of techniques have been developed in recent years. Probably the most important results for our work are related to [1] in which GSV-Cities dataset was presented creating one of the milestones of VPR Training.
In fact, even if also NetVlad [7] is still one of the most important milestones for VPR it makes use weakly super-

vised dataset meanwhile GSV-Cities is the first huge, labelled dataset: it significantly simplifies sample mining, and training batch creation.
For our experiments the second milestone is [2] because Mix VPR results are impressive in terms of its tradeoff between performance and numbers of the aggregator. It allowed to achieve our better results.
Other important references are [11] that introduced the Multi-Similarity loss and [9] that introduced the Circle Loss: in fact, even if Multi-Similarity provided best results also Circle loss provided good performance that have been mentioned.
[3] about Deep Visual Geolocation Benchmark was foundamental and also [12] about how ASGD and [13] AdamW Optimizers were important to achieve our best result.
Even if re-ranking application of Netvlad like [4] are important results it has been considered far from our approach because of limited resources employed in our experiments and the 24 hours limit of batch running limitation in Colab Pro+ and so no re-ranking was tested.
Other resources were really useful like [5] about learning rate schedulers and [6] which is a collection of Pytorch loss and miners codebase.
Even if they were less important for our final results also [10] about contrastive learning and [8] were used during our tests.

3. Methods

3.1. Codebase creation

First step of presented works has been the creation of Colab Codebase for VPR experiments.
This step has been splitted into two subtasks:

CodeBase Update
Since the original GSV-Cities publication code has been released in October 2022 the Pyothon packages used were less recent respect default packages of Colab so we were initially forced to reimport old packages version to let the code to work properly: we choose to use latest packages version and some modifications on training and validation methods used by Pytorch Lightning were required.

Dataset code adaptation
Our datasets version were different from the original ones: pictures size was 224X224 size instead of the original 320X320 and .npy files used in FAISS api were not present. These files containes query list, reference list and ground truth association. We rebuilt them using pictures information contained in provided filenames with particular attention on class names for GSV-XS Cities Dataset and UTM coordinates for SF-XS Val and Test and Tokio Test. Instead of the classical GSV Cities dataset that make use of specific .csv files for cities and pictures association we directly extract these associations during the creation of the dataset in the memory of Colab.

3.2. Dataset Analysis

Knowledge of used dataset is fundamental to understand challenges that and limits of the VPR Model.
The following is a quick analysis of the datasets used:
GSV-XS or Google Street View extra small dataset comprises street-level images from 23 different cities. This dataset captures buildings, places, and surrounding environments at various times, providing a diverse range of images. This diversity enables the training of models that can generalize better to unseen environments: it was used for training only.
For the validation and testing of our model, we used two datasets containing different images of San Francisco: **SF-XS Val** and **SF-XS Test**. As the name suggests SF-XS Val was used for validation meanwhile SF-XS Test was used for testing only.
Despite both datasets being from the same city, we observed different results between validation and testing datasets. The difference likely stems from the image content: the validation set includes wider shots of buildings and surroundings, similar to the training dataset, whereas the test dataset focuses mainly on buildings queries.
We also tested the model with the **Tokyo-XS** dataset which contains images of Tokyo's environments, buildings, and streets. The results on this dataset were better than those on the SF-XS Test dataset. This is likely because the Tokyo images are more similar to the training dataset images, but lower than those on the SF-XS validation dataset because, unlike San Francisco datasets, the Tokyo dataset includes nighttime images, strong shifts, and the buildings are more detailed. While San Francisco's buildings have simple shapes and low details, Tokyo's buildings are taken with closer focus.

3.3. AVG and GEM-Pooling

The initial setup of our image processing pipeline involved the usage of a pre-trained ResNet-18 network truncated at the third convolutional step. The Resnet was retrained using the AVG-Pooling as our initial aggregation layer becaus it reduces the spatial dimensions of feature maps by averaging the values in each channel, producing a compact representation that retains essential information and is invariant to small translations.
To enhance performance, we later implemented GEM Pooling, a more sophisticated technique. GEM Pooling refines the pooling operation by applying a parameterized power function to the average values within each channel of the feature maps. This allows adaptive adjustment, capturing more nuanced patterns compared to standard AVG Pooling.

By varying the power parameter, Gem Pooling can emphasize different aspects of the feature map, providing flexibility in feature extraction.
Both AVG Pooling and GEM Pooling effectively reduced the dimension of the feature maps, creating a fixed-size output for each channel, which streamlined subsequent processing steps. This configuration simplified downstream operations and served as a robust benchmark for evaluating different feature extraction strategies.
GEM pooling provided better results then AVG pooling but it was still not satisfactory.

3.4. FeatureMixer

The FeatureMixer module played a crucial role in our Visual Place Recognition (VPR) system because it aimed the enhancement of the accuracy and robustness of location recognition by aggregating multi-level features extracted from the ResNet-18 architecture.
The primary objective of the FeatureMixer is to create a more discriminative and comprehensive feature representation by integrating features from different layers of the ResNet-18 network. This approach is designed to address the challenges posed by varying environmental conditions, perspectives, and occlusions commonly encountered in VPR tasks.
MixVPR integrates features in a holistic manner for Visual Place Recognition (VPR). It starts by extracting feature maps $F \in \mathbb{R}^{c \times h \times w}$ from Resnet-18. These maps are treated as 2D features $\{X_i\}, i = \{1, \ldots, c\}$ and flattened into $F \in \mathbb{R}^{c \times n}$, with $n = h \times w$.

A cascade of L MLP blocks, processes these flattened features. Each block enhances X_i by applying fully connected layers with ReLU activation, incorporating global relationships and utilizing skip connections for gradient flow.

After L blocks, the output $Z \in \mathbb{R}^{c \times n}$ undergoes dimensionality reduction via W_d and W_r layers to produce $O \in \mathbb{R}^{d \times r}$, flattened and L2-normalized for VPR.

To summarize, MixVPR enhances feature integration for VPR through iterative Feature-Mixer blocks, leveraging the full receptive field of feature maps to generate global descriptors efficiently.

3.5. Model Tuning

To further optimize the model's performance, we employed fine-tuning of the parameters of Mix-VPR.
We tested the behavior of different mix depths, MLP ratios, and output channel multiplier. The mix depth determined the number of FeatureMixer layers stacked together, while the MLP ratio controlled the size of the intermediate projections within the MLP. The output rows defined the dimensionality of the final output embedding space.
We moreover experimented AVG, GEM Pooling and Mix VPR with various loss functions to identify the most effective one for our task. This included testing:
Triplet Margin Loss: Optimizes embeddings so images of the same place are closer together compared to images of different places. Uses triplets (query, positive, negative) to enforce a margin between positive and negative pairs. It has been used with triplet margin miner.
NTXent Loss: Similar to triplet loss but normalizes scores before applying the loss, aiding training stability, especially on large datasets. It has been used with multi similarity miner.
Circle Loss: Improves on triplet loss by using a learnable circular margin. Penalizes images outside this margin more heavily, which is beneficial for varying dissimilarities in VPR. It has been used with distance-weighted mining.
Contrastive Loss: Minimizes distance between similar image embeddings and maximizes distance between dissimilar ones using image pairs. Simpler than triplet-based methods but might be less effective as it doesn't enforce a specific margin. It has been used with pair margin mining.
Multi-Similarity Loss: It considers three types of similarities for the weighting of pairs: self-similarity and so the similarity of an example with itself, positive similarity as the relative similarity among pairs of similar examples;negative similarity: the relative similarity among pairs of dissimilar examples. These similarities are used to assign a weight to each pair, which directly influences how the examples will contribute to the model's parameter updates during training. A greater weight indicates that the pair is considered more informative and thus has a more significant impact on the learning of the embedding. In practice, this approach allows for a more principled gathering and weighing of informative pairs, enhancing the effectiveness of deep metric learning and achieving state-of-the-art performance.

Multi-similarity was used to achieve our better results like in the original GSV-Cities publication.

3.6. Model Optimization

We explored various optimizers and different learning rates and weight decay rates (if applicable).
Initially, we employed ADAM as a starting optimizer like it was done in GSV-Cities publication and we tested it in the first execution series of test.
Later, once Mix-VPR has been tuned we made a comparison of:
SGD (Stochastic Gradient Descent): A simple and computationally efficient optimization algorithm that updates model parameters in the opposite direction of the gradient. It can be slow to converge on complex data or poorly scaled gradients.
Adam (Adaptive Moment Estimation): An adaptive learning rate optimizer that uses moving averages of past

gradients and squared gradients to adjust learning rates. It performs well across different architectures and datasets due to its adaptive nature.
ASGD (Asynchronous Stochastic Gradient Descent): a variant of SGD in which multiple workers compute gradients independently and asynchronously, which can be faster than plain SGD.
AdamW: An Adam variant that decouples the gradient of the regularizer from the update rule of Adam, improving generalization performance and preventing overfitting, particularly useful in large-scale datasets.

Different optimizers were tested in combination with different learning rate schedulers that control how learning rate changes during different epochs and in particular we tested:
Multi-Step LR: It adjusts the learning rate by a fixed factor at specific milestones. The learning rate is reduced by a gamma factor at each milestone epoch. It's useful when the training process has distinct phases, and you need to reduce the learning rate at specific intervals to ensure fine-tuning and convergence.
Cosine Annealing: It gradually reduces the learning rate following the cosine function from the initial learning rate to a minimum value over a specified number of epochs. The learning rate decreases following a cosine curve, reaching the minimum value at the end of the cycle, and can optionally restart (warm restarts). It helps to escape local minima and improves convergence by periodically varying the learning rate.
LR On Plateau: It reduces the learning rate when the monitored metric (e.g., validation loss) stops improving for a specified number of epochs (patience). If the monitored metric does not improve for 'patience' epochs, the learning rate is reduced by a factor. It automatically adjusts the learning rate based on model performance, making it useful for adaptive learning rate control.
One Cycle LR: It varies the learning rate in a single training cycle, first increasing it linearly from a lower bound to an upper bound, then decreasing it back to a lower bound. The learning rate starts low, gradually increases to a peak (upper bound), then decreases back to a minimum value. Momentum is adjusted inversely. It aims to achieve better convergence and training speed by allowing the model to explore various learning rates within one cycle.

3.7. Model Checkpoints

We implemented a robust checkpointing mechanism to save all models based on specific performance metrics (e.g., sfxsR1, sfxsR5). This ensured that the best models configurations were preserved for further evaluation and potential deployment.
Through this iterative fine-tuning process, we aimed to maximize the model's robustness and accuracy. By systematically testing and refining different combinations of settings, we sought to identify the optimal configuration that delivered the best performance on our specific visual place recognition task. The fine-tuning efforts were guided by performance metrics evaluated during testing, allowing us to make data-driven decisions to enhance the model's overall effectiveness.
We didn't always choose the highest R@1 checkpoint looking for a tradeoff between loss result and r@1 result to prevent the choice of overfitted checkpoint.

4. Experiments

4.1. General Setup

During all the experiments batch size has been fixed to 64 even it has been observed that in GSV Cities publication it was fixed to 100. We made this choice to keep a good tradeoff between execution time and performance.
Generally, fist 2 layer of Resnet has been kept frozen unless differently specified and according to what has been observed no relevant differences has been obtained against the fully unfrozen version of the backbone.
During AVG, GEM Pooling and non-optimized version of Mix VPR Adam Optimizer has been used with learning rate=0.0002 and multistep learning rate scheduler has been used as in the original GSV Cities publication.
Number of epochs has been settled differently for each of described test and it has written in each test details.
Number of Random augumentations has always been kept to 3 as the original publication.

4.2. AVG and GEM Pooling Results

AVG Pooling achieved the results in Table1 running for 30 epochs and using Multi Step LR configured to split by 3 the learning rate each 3 epochs.

	Sf-Xs Val		Sf-Xs Test		Tokyo Test	
Loss	**R@1**	**R@5**	**R@1**	**R@5**	**R@1**	**R@5**
MultiSimilarity	48.43	63.18	13.1	24.8	22.54	36.83
Triplet	52.5	68.26	11.6	21.7	22.22	41.59
NTXent Loss	53.58	72.78	16.5	29.5	28.25	40.95
Contrastive	**61.25**	**74.18**	**21.6**	**36.7**	**34.6**	**50.79**
Circle	59.99	73.25	20	33.6	29.52	43.81

Table 1. Performance metrics for different similarity measures with AVG Pooling

Using same configuration of Learning Rate schedulers we achieved the results in Table 2 using GEM Pooling.

As it can be oserved GEM Pooling provided a interesting performance improvement but the overall result was still unsatisfactory even changing loss and miners: in particular with both SF-XS Test and Tokio Test, 30% in R@1 was

	Sf-Xs Val		Sf-Xs Test		Tokyo Test	
Loss	**R@1**	**R@5**	**R@1**	**R@5**	**R@1**	**R@5**
MultiSimilarity	58.45	73.84	20	32	27.62	45.71
Triplet	60.29	75.88	19.4	30.7	31.75	50.16
NTXent Loss	67.31	80.46	25.5	39.3	34.6	55.24
Contrastive	**68.96**	80.87	**28.7**	43.1	**41.9**	**60.63**
Circle	68.7	**81.91**	27.7	**44.4**	40.63	58.73

Table 2. Performance metrics for different similarity measures with GeM Pooling

never reached.
As it can be seen in the table, best loss was the Contrastive loss that in these fist two tests outperformed the Multi Similarity Loss.

4.3. Feature Mixer

As third step we tested Feature Mixer in its base configuration using width ratio, depth and channel multipliers of MixVPR equal to 1.
More interesting results were obtained and have been reported in Table3. Also in this case we kept same configuration of Multi Step LR and optimizer but we reached 38.7% of R@1 and 53.97% with Tokio-XS. In this test the best loss function for San Francisco datasets was the Multi Similarity Loss, meanwhile Contrastive Loss provided our best result with Tokio Test dataset.
Mix VPR surpassed both AVG and GEMPooling so it has been chosen as the best candidate for further tunings: as it can be seen we tested it with same losses and miners of AVG and GEM Pooling but we obtained a remarkable performance improvement with both test datasets: it's related to the ability of Mix VPR to capture relevant features in the pictures because of the presence of another MLP level on top of the backbone: according to this initial hyptotesis the Mix VPR parameters have been tuned.

	Sf-Xs Val		Sf-Xs Test		Tokyo Test	
Loss	**R@1**	**R@5**	**R@1**	**R@5**	**R@1**	**R@5**
MultiSimilarity	**74.43**	**85.22**	**38.7**	**53.2**	50.79	66.35
Triplet	60.03	74.62	18.1	30.8	31.75	46.98
NTXent Loss	70.11	82.36	28	42.6	40.32	57.14
Contrastive	70.04	80.58	33.2	47.5	**53.97**	**69.84**
Circle	73.18	84.19	31.8	49.2	43.49	62.86

Table 3. Performance metrics for different similarity measures with base feature mixer

4.4. Regularizers and reducers

To further enhance the model's performance, we also experimented with the effects of LP regularizer and reducers in different configurations.
In this case, the configuration was tested with a frozen backbone version, with a depth equal to 5, a ratio of 5, and an output channel multiplier equal to 1.
Despite these efforts, regularizer didn't introduce remarkable performance improvements. The same applies to reducers, so neither regularizer nor reducers were used, likely because in more challenging pictures it worsened the ability to capture relevant features.

	Sf-Xs Val		Sf-Xs Test		Tokyo Test	
Loss	**R@1**	**R@5**	**R@1**	**R@5**	**R@1**	**R@5**
MultiSimilarity	**76.57**	**86.55**	**38.7**	**53**	**51.43**	**65.08**
Triplet	64.08	77.51	17.6	31.2	34.29	50.79
NTXent Loss	69.84	82.31	26.7	41.5	40.63	55.56
Contrastive	66.98	78.97	27.3	42.5	48.89	64.76
Circle	73.09	83.72	33.8	49.5	47.94	63.81

Table 4. Performance metrics for different similarity measures with base feature mixer and regularizer.

4.5. Mix VPR Tuning

Width	Depth	Mult.	R@1	R@5
7	7	1	74.8	85.17
1	4	5	76.13	83.8
7	1	5	76.53	85.05
8	3	4	77.18	85.01
5	2	4	77.32	85.24
4	4	5	77.39	85.47
6	7	7	78.01	86.38
5	2	8	78.11	86.34
5	7	5	78.12	85.5
8	2	7	78.26	86.31

Table 5. Sfxs_val Results with Different Depths and Configurations. the test was made for 5 epochs, with original GSV-Citeis Adam tuning of parameters and Multisimilarity loss using all cities in the dataset.

We tested the effect of tuning the MixVPR aggregator, which involved adjusting three parameters: depth, width, and initial channel multiplier of the MixVPR. These parameters respectively control the number of multilayer perceptrons stacked together, the size of each layer, and the size of the feature space produced for the embedding.
The goal was to identify the optimal configuration for optimal feature aggregation in our VPR tasks, based on the observed table results with the SF-XS Val Dataset.
For these test runs has been limited to 5 epochs only because the main goal was to observe the overall effect of parameters changing with no further fine tuning. It has been observed that generally speaking increasing output features number was beneficial and that the width was more relevant that the depth: the maximum number was fixed to (8,8,8) because of GPU Ram limitations: each layer of Mix VPR adds a relevant feature number to be tuned and the same apply for the width parameter. For paramters

selection optuna framework has been used: it makes use of bayesian estimator to furter improve the performance.

4.6. Learning rate and schedulers comparison

To find the best result we tested different learning rate optimizers and learning schedulers. The optimizers were SGD, Adam, AdamW, and ASGD, each with various learning rate profiles. The learning rate schedulers tested were MultiStepLR, ReduceLROnPlateau, CosineAnnealingLR, and OneCycleLR.
We tested these optimizers and schedulers on a reduced version of the dataset, focusing on a 2 cities version instead of the entire 23-city dataset. This approach allowed us to identify the best optimizer-scheduler combination more quickly and execute more tests in a limited amount of time. The best results were obtained using the Reduce LR On Plateau as learning rate scheduler with the AdamW optimizer.
The test was made using freezed backbone using mixVPR with parameters Depth=2, Ratio=8, Channel multiplier=7 because there are best parameters obtained during the Mix-VPR tuning test.
Each test ran for 20 epochs, LR on Plateu was set with patience parameter=1 and factor=0.1 with monitored metric= R@1. Even if more than 30 tests were made only best results have been reported in table6. The usage of One Cycle - LR didnt' introduced remarkable benefits with SGD and so none of these tests has been reported.

4.7. Best Results

Best results were obtained using Multi Similarirty loss and miners with the following settings:
Batch Size: **64**
Epochs: **30**
Optimizer: **AdamW**
Learning rate: **0,0002**
Weight decay: **0,0001**

We obtained the following results:

	Sf-Xs Val		Sf-Xs Test		Tokyo Test	
Loss	**R@1**	**R@5**	**R@1**	**R@5**	**R@1**	**R@5**
MultiSimilarity	**81.35**	**88.99**	**54.8**	**68.9**	**64.13**	72.38
Circle	77.66	86.19	45.8	59.4	62.22	**73.97**

Table 7. Performance metrics using Adam and Multi Step LR.

As it can be seen the tuning of Mix Vpr, losses and miners significantly improved the performance with test datasets of SF-XS Test and Tokio Test that were deeply affected by the tuning paramters process. In particular, even if no remarkable performance achieved in the Val dataset, R@1 improvement was 26% on Tokio Test Dataset and 41% on SF-XS Test Dataset.

5. Visual Analysis

Visual analysis shows that the model is often robust to viewpoint and light changing condition. We observed problems in case of pictures that are foreground on buildings details with very similar colors or shapes or in case of very similar scenarios than can be misclassified also by human hand. In the appendix we reported some examples of visual analysis.

6. Ablation Studies

For a complete testing process it has been evaluated also the effect of Learning Rate Schedulers and Optimizers separately: using same configuration of Adam and Lr Scheduler we achieved:

	Sf-Xs Val		Sf-Xs Test		Tokyo Test	
Loss	**R@1**	**R@5**	**R@1**	**R@5**	**R@1**	**R@5**
MultiSimilarity	**80.1**	**87.56**	**53.10**	**67.20**	**64.44**	**76.19**

Table 8. Base optimizer and Learning Rate Scheduler tuning with tuned Feature Mixer

As it can be seen the difference is not relevant respect to the best solution so according to what can be seen the original tuning in [1] is already a really good tuning.

For completeness also a test with batch size to 100 has been made to look for performance decay related to tuning choice but, also in this case, no remarkable differences have been observed and the results have not been report here but are present in the code in github only.

7. Conclusion and Limitation

We investigated different pooling methods (Avg Pooling, Gem Pooling) and the FeatureMixer architecture (MixVPR) for effective feature extraction and aggregation. By meticulously tuning hyperparameters, loss functions, and optimizers, we significantly enhanced the model's performance in recognizing places from images, even with challenging datasets.
Despite these improvements, the system's performance is constrained by the trade-off between computational efficiency and representational power inherent to the ResNet-18 architecture, as well as the small resolution of the images and the small dimension of the dataset used for the training (compared to datasets used from state of art models). These factors can lead to misclassifications, especially with images that exhibit significant scene complexities or require recognition of subtle details.
Visual analysis reveals that many misclassifications are

Optimizer	lroptimizer	Learning Rate	Weight Decay	Momentum	R@1	R@5
ADAM	Reduce LR OnPlateau	0.00005			69.59	79.06
ASGD	Reduce LR OnPlateau	0.5	0.001		69.69	79.22
SGD	Reduce LR On Plateau	0.03	0.001	0.9	70.57	79.37
ASGD	Cosine Annealing LR	0.5	0.001		71.06	80.21
ADAMW	Cosine Annealing LR	0.005	0.001		71.8	80.8
ADAMW	Reduce LR OnPlateau	0.0002	0.0001		72.74	81.33

Table 6. Top 5 best hyperparameter

challenging even for human operators. Considering the limitations of Google Colab resources, the overall performance of our model is commendable.

References

[1] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. GSV-Cities: Toward appropriate supervised visual place recognition. *arXiv preprint arXiv:2210.10239*, 2022. Neurocomputing 2022.

[2] Amar Ali-bey, Brahim Chaib-draa, and Philippe Giguère. MixVPR: Feature mixing for visual place recognition. *arXiv preprint arXiv:2303.02190*, 2023. Accepted at WACV 2023.

[3] Gabriele Berton, Riccardo Mereu, Gabriele Trivigno, Carlo Masone, Gabriela Csurka, Torsten Sattler, and Barbara Caputo. Deep visual geo-localization benchmark. *arXiv preprint arXiv:2204.03444*, 2022. CVPR 2022 (Oral).

[4] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition. *arXiv preprint arXiv:2103.01486*, 2021.

[5] Leonie Monigatti. A visual guide to learning rate schedulers in pytorch. `https://towardsdatascience.com/a-visual-guide-to-learning-rate-schedulers-in-pytorch-24bbb262c863`, 2022.

[6] Kevin Musgrave. Pytorch metric learning. `https://kevinmusgrave.github.io/pytorch-metric-learning/`, 2024.

[7] Filip Radenović, Giorgos Tolias, and Ondřej Chum. Fine-tuning cnn image retrieval with no human annotation. *arXiv preprint arXiv:1711.02512*, 2017. TPAMI 2018. arXiv admin note: substantial text overlap with arXiv:1604.02426.

[8] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv preprint arXiv:1708.07120*, 2017.

[9] Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. Circle loss: A unified perspective of pair similarity optimization. *arXiv preprint arXiv:2002.10857*, 2020.

[10] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[11] Xun Wang, Xintong Han, Weiling Huang, Dengke Dong, and Matthew R. Scott. Multi-similarity loss with general pair weighting for deep metric learning. *arXiv preprint arXiv:1904.06627*, 2021.

[12] Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. *arXiv preprint arXiv:1609.08326*, 2016. International Conference on Machine Learning, 2017: 4120-4129.

[13] Zhenxun Zhuang, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona. Understanding adamw through proximal methods and scale-freeness. *arXiv preprint arXiv:2202.00089*, 2022.

8. Appendix

In the appendix examples of correctly and incorrectly classified pictures are reported and in both cases we reported examples in the order from top to bottom: SF-XS Val, SF-XS Test and Tokio Test dataset. In case of SF-XS Val and Test it can be observed that failures are caused by the lack of significant discriminative elements, for Tokio Test sample the failure has been caused by the challeling scenario: in this case the model has been misleaded by distracting elements like cartels with the same orientation or building details.

Figure 1. Success examples of the model, The first picture on the left is the query, correctly associated image have been framed in green

Figure 2. Failure examples of the model, The first column contains query images, Ground Truth images have been framed in blue, misclassified images are framed in red