3A Image Geolocalization Through Retrieval

TA: Gabriele Berton, Gabriele Trivigno

NOTE: This document might undergo some small changes in the next days. Do not download this document, it will **always** be available at the same link.

SUMMARY:

1) OVERVIEW

Visual geo-localization (VG), also called Visual Place Recognition (VPR), is the task of coarsely finding the geographical position where a given photograph was taken. This task 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 that represent the known world. The N-top retrieved images, with their geo-tag (typically GPS coordinates), provide the hypothesis for the query's geographical location.

The retrieval is performed as a k Nearest Neighbour search in a learned embedding space that well represents the visual similarity of places. Namely, each image is passed through a network composed of a feature extraction backbone and a head that aggregates or pools the features to create a global descriptor of the image. By using a contrastive learning approach, the model learns to extract descriptors that are discriminative of locations. The similarity search is then implemented as a pairwise comparison among descriptors, e.g. using a cosine similarity.

In this project, you will become familiar with the task of visual geo-localization and with how a visual geo-localization system works. You will start by experimenting with the baselines using a codebase provided by us that contains all the most common methods. Subsequently you will focus on improving one or more aspects of the system of your choice.

We already provide you with the code to implement the baseline, so that you do not have to waste time in re-implementing them; in this way you can focus on understanding the code, and then try to tackle real research problems. If you obtain any significant results and if you are interested you can bring the project to a publication to a computer vision conference or scientific journal.

2) DATASETS

We provide you with multiple datasets to experiment with:

- GSV-XS dataset: we provide a small version (XS for eXtra Small) of the GSV-cities dataset [21] that we created for your convenience to adapt to Colab resources. This dataset is used for **training**.
- San Francisco eXtra Small (SF-XS): a subset of the SF-XL dataset [19]. This is used for **validation** (SF-XS val) and **testing** (SF-XS test). Note that SF-XS also has a training set, but you will not use it in this project, and you can simply ignore it.
- Tokyo eXtra Small (Tokyo-XS): a subset of the Tokyo 24/7 [16] dataset. This is used only for **testing**.

The reason for using two test sets is to understand how your changes to the model affect the geolocalization for different datasets: for example, some extensions might improve results for Tokyo-XS, but worsen results on SF-XS (test).
Note that validation and test datasets have two directories, one for database and one for the queries: this is because you need both for evaluation. On the other hand at training time the code uses a single set of images (although some older papers [1] used database and queries also for training).
The datasets are available at this link, although in the code there is a notebook example to download the directly to your Google Drive, so you can use them for Colab

3) CODE

The code is available at this GitHub repo. Leaving a Star on the repo is highly appreciated.

4) STEPS

1. Study the literature and get familiar with the task

As a preliminary step to get familiar with the task, start by understanding what is image retrieval, and how it is used for visual geolocalization / visual place recognition. At first use Google, ChatGPT and YouTube to find relevant resources. Then read some papers to get an idea on how the task is solved. Understand what are the challenges (e.g. night-time photos are difficult to localize!), how they are solved, what is a triplet loss and what is mining.

2. Time to play with the code!

Download the datasets, download the code, and run your first training! Your neural network will learn how to solve the task, perhaps it will be able to geolocalize images even better than you! Make sure to set the right paths (GSV-XS for training, SF-XS val for validation, and SF-XS test for testing). Are the results different on SF-XS val from SF-XS test? Why?
Have a look at the images in the datasets, and make sure that you understand why the results are different on the two sets! What are the main differences between SF-XS val and SF-XS test?

3. Use the trained model to test on Tokyo-XS

Do you really need to train your model again for this step? The answer is no, the weights of the trained models are saved in the logs, you can resume those to test with your trained model! Are the results different on Tokyo-XS than on SF-XS test?
Have a look at the images in Tokyo-XS, what are the main differences between Tokyo-XS and SF-XS test?
At test time, the query image is deemed correctly localized if at least one of the top N retrieved database images is within d = 25 meters from the ground truth position of the query. You will have to report the values of recall@N (for N=1,5), i.e. the percentage of queries for which at least one of the first N predictions is within a 25

meters distance from the query.
Your first table should look something like this

SF-XS val	SF-XS test	Tokyo-XS
R@1/R@5	…	

4. Make your first changes to the code

What is the final pooling layer in the model that you trained?
Now try to change it with a GeM layer [2]. Did the results improve? If you don't know what it is, have a look at the paper, and look for the code to implement it. It's openly available online!

5. Visually analize the results

There is a parameter in the code that you can pass to the training script that will save some queries and their predictions (i.e. the images from the database that the model thinks are most similar to the query). Find it and use it! How do the predictions look? Does your model make any strange mistakes?

6. Now it's time to do some real research!

Are you ready to be a Deep Learning researcher? Then choose **at least** one of the following steps and work on it! Always make sure to understand what you are doing, make some guesses on how your changes could influence the neural net, and then launch your experiment to see if your guesses were correct. Also, don't forget to visually inspect the predictions. The objective of your contributions should be to improve results or speed up training. Test your trained models on all test datasets.

a. Miners

The GSV-XS dataset is the first dataset for VPR that has a distinct concept of 'classes', since the same place is depicted by several images, and different places are geographically apart from each other. This enables borrowing many techniques from the Image Retrieval field, where there usually are distinct categories. Using this library link, you find already implemented lots of alternatives from the literature. You can start from there and experiment with many different types of mining. Before you choose any option from the library, make sure that you fully understand what the technique entails, the implication on the training, and the relationship with the kind of data that you have in your dataset. You can also try Proxy mining as shown in https://arxiv.org/abs/2302.14217

b. Losses

What loss function have you used until now? How does it work? What is the input? Which other losses can be used?
The same considerations for mining hold for losses. Thanks to the structure of GSV-XS you can experiment with retrieval losses such as CosFace, ArcFace, MultiSimilarity, and many others that you find in that library. Also in this case before you choose any option from the library, make sure that you fully understand what the technique entails and the implication on the training

c. Self-supervised training

In this same library link you also find implemented several self-supervised losses like VicReg, and many others that are obtained combining one of the contrastive losses (like the triplet) with the self-supervised paradigm. In this case, you would not be using the labels anymore.
To exploit these self-supervised strategies, you need to define some augmentations, so that then you can ask the network to have embeddings robust to that kind of augmentation. Therefore augmentations design, and the

choice of the loss, are the most important design choices. Regarding augmentations, you can try RandomPerspective, cropping, adding patches: look for the list of available augmentations on the torchvision documentation. Design the augmentation based on what you think is useful for the task.

d. **Sampling strategy**
As training progresses, the fraction of informative triplets (or pairs) within the randomly sampled mini-batches becomes limited (i.e., the network becomes good at distinguishing hard negatives), and thus in many works [21,22] a large batch size is used. This means requiring a lot of GPU memory. A sampling strategy in practice can be either an offline mining step, or a policy to choose samples from classes. The latter could be a stratagem to reduce the need for large batch sizes. In the library that we mentioned you find some samplers available, like PerClassSampler and HierarchicalSampler. You can try them, or come up with your own strategy

e. **Aggregation modules**
The last part of the model is a pooling layer and a fully connected (i.e. linear) layer. Are there better alternatives? Try to implement the MLP-mixer style aggregation presented in [22], and study its behavior with different losses and miners tried above.

f. **Optimizer & Schedulers**
As the base parameter for training, the default is the Adam optimizer with LR 1e-5. Try to experiment with different recently proposed optimizers (beyond SGD) available in torch, like AdamW, ASGD. Try to find the better LR, weight decay parameters and understand what happens changing them. You can also try different schedulers like ReduceLrOnPlateau, or CosineAnnealingLR.

5) DELIVERABLES

To conclude the project you will need to:
- Deliver PyTorch scripts for the required steps as a zip file.
- Write a complete pdf report following a standard paper format. The report should contain a brief introduction, a related works section, a methodological section for describing the algorithms you are going to use, an experimental section with all the results and discussions. End the report with a brief conclusion.

Some questions you should be able to answer in the end:

- What is contrastive learning and how it relates to image retrieval?
- What is the meaning of the 'mining' procedure and what are its downsides?

6) REFERENCES

[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla and J. Sivic, *NetVLAD: CNN architecture for weakly supervised place recognition*, TPAMI 2018

[2] F. Radenovic, G. Tolias, and O. Chum, *Fine-tuning CNN Image Retrieval with No Human Annotation*, TPAMI 2018.

[3] C. Masone and B. Caputo, *A survey on deep visual place recognition*, IEEE Access 2021

[4] L. Liu, H. Li, and Y. Dai, *Stochastic Attraction-Repulsion Embedding for Large Scale Image Localization*, ICCV 2019.

[10] H. J. Kim, E. Dunn, and J.-M. Frahm, *Learned contextual feature reweighting for image geo-localization*, CVPR 2017

[13] X. Wang, R. Girshick, A. Gupta and K. He, *Non-local neural networks*, CVPR 2018.

[14] F. Warburg, S. Hauberg, M. Lopez-Antequera, P. Gargallo, Y. Kuang and J. Civera, *Mapillary Street-Level Sequences: A Dataset for Lifelong Place Recognition*, CVPR 2020

[16]A. Torii, R. Arandjelovi ́c, J. Sivic, M. Okutomi, and T.Pajdla. 24/7 place recognition by view synthesis. IEEETransactions on Pattern Analysis and Machine Intelligence,40(2):257–271, 2018.

[17]Yang, Min, Dongliang He, Miao Fan, Baorong Shi, Xuetong Xue, Fu Li, Errui Ding and Jizhou Huang. “DOLG: Single-Stage Image Retrieval with Deep Orthogonal Fusion of Local and Global Features.” *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021): 11752-11761.

[18] Cao, Bingyi, Andre F. de Araújo and Jack Sim. “Unifying Deep Local and Global Features for Image Search.” *ECCV* (2020).

[19] Berton, G. and Mereu, R. and Trivigno, G. and Masone, C. and Csurka, G. and Sattler, T. and Caputo, B. “Deep Visual Geo-localization Benchmark.” CVPR (2022).

[20] Berton, G. and Masone, C. and Caputo, B. “Rethinking Visual Geo-localization for Large-Scale Applications.” CVPR (2022).

[21] Ali-bey, Amar, Brahim Chaib-draa, and Philippe Giguère. "GSV-Cities: Toward appropriate supervised visual place recognition." *Neurocomputing* 513 (2022): 194-203.

[22] Ali-bey, Amar, Brahim Chaib-draa, and Philippe Giguère. "MixVPR: Feature Mixing for Visual Place Recognition." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2023.