Deep Visual Geo-localization Benchmark: Supplementary Material

Gabriele Berton Politecnico di Torino
Riccardo Mereu Politecnico di Torino
Gabriele Trivigno Politecnico di Torino
Carlo Masone CINI
Gabriela Csurka NAVER LABS Europe
Torsten Sattler CIIRC, Czech Technical University in Prague
Barbara Caputo Politecnico di Torino

This supplementary material contains additional information that could not fit within the main paper due to a lack of space:

- Section 1 describes in detail the datasets used in the benchmark.
- Section 2 explains the organization of the open-source software that implements the benchmark.
- Section 3 provides extended results and discussions for the experiments presented in the main paper.
- Section 4 contains additional experiments and discussions that complement the tests presented in the main paper.

1. Datasets

Pitts30k [1] is a subset of Pitts250k [26], split in train, val and test set. It is collected from Google Street View imagery from the city of Pittsburgh cropping equirectangular panoramas into tiles, and applying a gnomonic projection to the tiles. Database and queries are collected two years apart, and there are no noticeable weather variations.

Mapillary Street Level Sequence (MSLS) [28] spans multiple cities across six continents, covering a large variety of domains, cameras and seasons. As for Pitts30k, it is split in train, val and test set, although the test set's ground truths are not currently released. We therefore report the validation recalls, following previous works [8]. Only Pitts30k and MSLS provide a train set with temporal variability, which is necessary for training a VG model [1].

Tokyo 24/7 [25] presents a relatively large database (from Google Street View) against a smaller number of queries, which are split into three equally sized sets: day, sunset and night. The latter are manually collected with phones. In some cases [1, 15, 31] Tokyo Time Machine (Tokyo TM) is used as a training set for Tokyo 24/7.

San Francisco [4], similarly to Tokyo 24/7, is composed of a large database collected by a car-mounted camera and orders of magnitude less queries taken by phone. Among the multiple Structure from Motion reconstructions available, we use the one from [14, 27] as it offers the most accurate query 6 DoF coordinates, thus referring to it as Revisited San Francisco.

Eynsham [5] consists of grayscale images from cameras mounted on a car going around around a loop twice, in the city and countryside of Oxford. We use the first loop as database, and second as queries. The cameras collected equirectangular panoramas, and each panorama was split in five crops.

St Lucia [17] is collected by driving a car with a forward facing camera around the riverside suburb of St Lucia, Brisbane. Of the nine drives, we use the first and the last one as database and queries. Given the high density of the images (extracted from videos), we select only one frame every 5 meters. Note that all these pre-processing steps (as well as downloading) are performed automatically with our open source codebase (see Section 2).

In Fig. 1 we show relevant query-database image pairs from each of the used datasets. These examples illustrate view, environmental, and acquisition condition variability between query and database images as well as across the datasets making the generalization between datasets hard. In Tab. 1 we provide a summary of the number of database and query images as well as the area and perimeter covered by the respective datasets. Figure 2 shows the density of the images in the respective geographical areas.

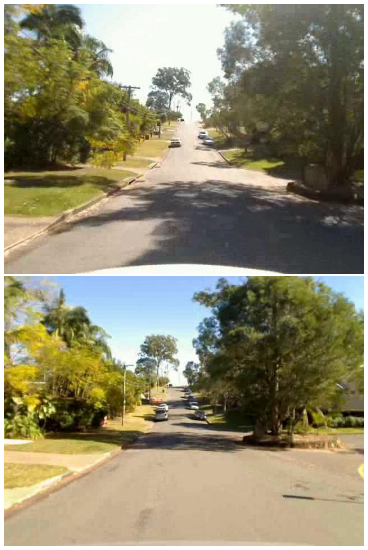
(a) Pitts30k

(c) San Francisco

(e) Eynsham

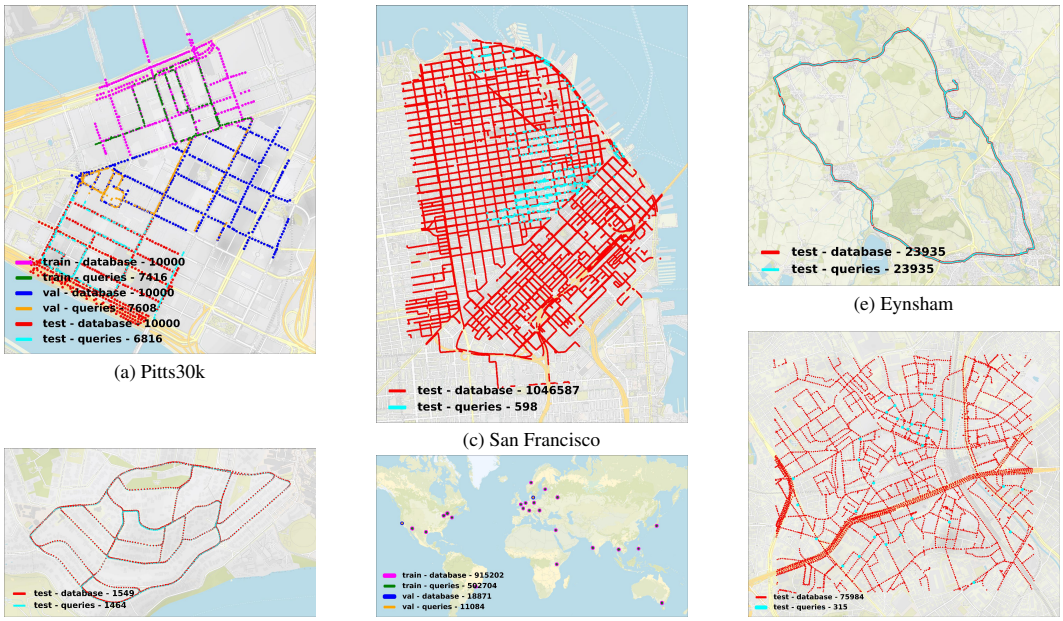
(b) Tokyo 24/7

(d) MSLS



(f) St Lucia

Figure 1. **Examples of a query and a positive** for each of the used dataset.



(a) Pitts30k (c) San Francisco (e) Eynsham

(b) St Lucia (d) MSLS (f) Tokyo 24/7

Figure 2. **Maps of used datasets,** self-generated with our open source codebase.

	# database	# queries	Dataset size	Area (Km^2)	Perimeter (Km)	Environment	Day/night changes	Long-term variations
Pitts30k	30K	21.8K	2.0 GB	0.615	3.42	Urban	N	Y
MSLS	973K	541K	56 GB	N/A	N/A	Urban + Suburban	Y	Y
Tokyo 24/7	75K	315	4.0 GB	2.1	5.8	Urban	Y	Y
R-SF	1.05M	598	36 GB	13.6	14.0	Urban	N	Y
Eynsham	24K	24K	1.2 GB	N/A	N/A	Urban + Suburban	N	N
St Lucia	1.5K	1.5K	124 MB	0.69	3.5	Suburban	N	N

Table 1. **Summary of datasets used.** Long-term variations refers to images taken at least one year apart.

2. Software

We aim to create and maintain an organized open-source repository where existing and new VG methods will be integrated in the future. Our site [1] will be used to show the performances of these methods with different VG datasets.

Following these main motivations, we designed the software aiming to create a modular and easy expandable framework that provides the users with a common playground (i) to train, test, and fairly compare the impact of different components of a VG model, (ii) to ease the reproducibility of the results, and (iii) to evaluate the performances with datasets of different scales.

We organized the software into three distinct modules:

- `benchmarking_vg`: a general and expandable template for training and evaluating VG models;

- `dataset_vg`: a dataset utility to automatically download most of the datasets described in Section 1 and format them according to a standardized methodology;

- `pretrain_vg`: a template to pretrain neural networks backbones used in the VG task.

We mainly consider VG techniques that tackle the Visual Geo-localization problem through an image retrieval approach using Deep Learning (DL). For this reason, the first and main module (`benchmarking_vg`) of our framework follows a common structure for all the models, which are composed of a neural network backbone and a pooling layer on top. We integrated existing PyTorch open-source implementations of VG models or self-implemented them when they were unavailable. For the similarity search we use the implementations from the highly optimized FAISS [10] library. Unless otherwise specified, we use an exhaustive kNN search. Further techniques can be easily integrated and work under our environment.

The `benchmarking_vg` module further allows the user to choose which VG model, training dataset, and min-

[1] https://deep-vg-bench.herokuapp.com/

ing techniques to use to evaluate its performance. Even external trained models can be loaded and evaluated on VG datasets. Section 4.1 of this supp. material shows the results obtained by integrating the models of [19] into our framework.

The `pretrain_vg` module constitutes a template to pretrain backbones on the Landmark Recognition and Classification datasets. In the current version, the Google Landmark v2 dataset [29] and Places 365 [32] are available.

3. Extended Results

3.1. CNN Backbones

In Tab. 2 we show comparative results obtained by cropping a ResNet-18 and a ResNet-50 to the *conv4_x* layer (used in the experiments in the main paper) or alternatively cropping to the *conv5_x* (refer to the ResNets paper [9] for more details on the layers). We see that on average cropping the ResNet backbone at the lower level *conv4_x* leads to better results while being somewhat lighter in size.

3.2. Aggregation and Descriptors Dimensionality

In Tab. 3, we show a more comprehensive set of results than in the main paper, comprising all the aggregation methods that can be attached to the different backbones using our software. As seen in the literature, GeM pooling [19] outperforms in general SPOC [2], MAC [20], R-MAC [24], RRM [13].

3.3. Visual Transformers: full table

Tab. 4 includes results using Transformer-based backbones when trained on Pitts30k that could not fit into the main paper. In general, it can be seen that these architectures confer better generalization capabilities, outperforming both a ResNet-18 and a much more costly ResNet-50. Additionally, directly using the CLS token yields worse results than SeqPool, GeM, or NetVLAD. A possible explanation is that using the CLS is the only strategy that does not consider the whole set of tokens. This consideration could indicate that the CLS token provides a less robust representation, especially when trained on small-scale datasets.

3.4. Negative Mining

Since the inception of the triplet loss, considerable attention has been paid to finding the best possible negative images. Using negatives too different from the query will cause a drop in the loss to low values (even to zero if using a triplet margin loss), severely hindering the learning process of the model. For this reason, mining for the hardest negatives w.r.t. a given query is an important step in learning representation in general and, hence, in Visual Geo-localization. Therefore several hard negative mining techniques were proposed in the literature. In [1] the authors propose to compute offline features for all images (cache) periodically and to use such features to find the most difficult negatives. We refer to this as "full database mining". While this has proven to produce good results, its time and space complexities grow linearly with the database size, making it extremely costly to use it with large dimensional descriptors and large-scale datasets. In [28] the authors presented a new large scale dataset, for which the mining proposed by [1] would be rather time-consuming, and they performed an approximation of it considering only a small subset of the database (1000 images), making it a more suitable choice for large scale datasets. We refer to this as "partial database mining".

Discussion. Tab. 5 shows results when training with different mining methods. Full database mining performs the best when training on Pitts30k, although the less expensive partial mining performs similarly. Surprisingly, when training on Pitts30k, choosing random negatives without performing any mining operation results in only a 5% drop in recall@1 (on average over all datasets) compared to the training with partial database mining, although the gap grows to 12% when training on the MSLS dataset. This is probably due to the huge variety in domains of MSLS (as it spans over multiple continents), making a random negative likely to be very different from the query. On the other hand, Pitts30k is collected in a small area of Pittsburgh, with little to no weather variations, making random negatives a suitable choice for the triplet loss.

Training on MSLS, the results favour partial mining over full database mining because of the large scale of this dataset. While all other experiments converge in less than 24 hours, training a network on the MSLS dataset using full database mining is computationally very expensive, and therefore we stopped training after 5 days. Moreover, training a model that outputs a descriptor with high dimensionality (such as the NetVLAD layer) converges slowly and also requires intractable amounts of RAM, as it requires all images' descriptors to be periodically computed and stored in RAM. These results show that full database mining is impractical when working on large-scale problems.

3.5. Data Augmentation

In Fig. 3 we report the same plots shown in the main paper at Fig. 2, at a bigger and more readable size.

3.6. Query pre/post-processing and Predictions Refinement

In a real-world geo-localization system, the queries fed to the software at production time may have different resolutions than the database images. A handful of datasets (*e.g.*, R-SF [14, 27], Tokyo 24/7 [25]) incorporates this variability, and the solution often used in previous works to handle such cases [1, 7, 19, 21, 24] is to use a batch size

Backbone	Aggregation Method	Features Dim	FLOPs	Model Size	Training Dataset	R@1 Pitts30k	R@1 MSLS	R@1 Tokyo 24/7	R@1 R-SF	R@1 Eynsham	R@1 St Lucia
ResNet-18 *conv4_x*	GeM	256	17.29 GF	10.63 MB	Pitts30k	77.8 ± 0.2	35.3 ± 0.5	35.3 ± 1.1	34.2 ± 1.7	64.3 ± 1.2	46.2 ± 0.4
ResNet-18 *conv4_x*	NetVLAD	16384	17.27 GF	10.76 MB	Pitts30k	**86.4 ± 0.3**	**47.4 ± 1.2**	**63.4 ± 1.2**	**61.4 ± 1.5**	**76.8 ± 1.2**	**57.6 ± 3.3**
ResNet-18 *conv5_x*	GeM	512	22.33 GF	42.67 MB	Pitts30k	77.9 ± 0.3	34.4 ± 0.4	34.4 ± 0.6	36.9 ± 0.3	59.1 ± 1.3	51.2 ± 1.3
ResNet-18 *conv5_x*	NetVLAD	32768	22.28 GF	42.92 MB	Pitts30k	79.6 ± 0.5	47.1 ± 1.8	48.9 ± 2.5	49.1 ± 3.6	70.5 ± 1.0	54.4 ± 2.7
ResNet-50 *conv4_x*	GeM	1024	40.61 GF	32.71 MB	Pitts30k	82.0 ± 0.3	38.0 ± 0.1	41.5 ± 1.8	45.4 ± 2.0	66.3 ± 2.5	59.0 ± 1.4
ResNet-50 *conv4_x*	NetVLAD	65536	40.51 GF	33.21 MB	Pitts30k	**86.0 ± 0.1**	**50.7 ± 2.0**	**69.8 ± 0.8**	**67.1 ± 2.3**	**77.7 ± 0.4**	**60.2 ± 1.6**
ResNet-50 *conv5_x*	GeM	2048	50.54 GF	89.88 MB	Pitts30k	79.8 ± 0.5	41.5 ± 0.7	48.0 ± 2.5	44.3 ± 1.0	65.2 ± 1.4	57.5 ± 1.5
ResNet-50 *conv5_x*	NetVLAD	131072	50.35 GF	90.88 MB	Pitts30k	79.6 ± 0.2	46.2 ± 0.5	54.7 ± 2.6	51.2 ± 2.5	69.8 ± 1.0	53.0 ± 4.1
ResNet-18 *conv4_x*	GeM	256	17.29 GF	10.63 MB	MSLS	71.6 ± 0.1	65.3 ± 0.2	42.8 ± 1.1	30.5 ± 0.8	80.3 ± 0.1	83.2 ± 0.9
ResNet-18 *conv4_x*	NetVLAD	16384	17.27 GF	10.76 MB	MSLS	**81.6 ± 0.5**	**75.8 ± 0.1**	**62.3 ± 1.6**	**55.1 ± 0.9**	**87.1 ± 0.2**	**92.1 ± 0.7**
ResNet-18 *conv5_x*	GeM	512	22.33 GF	42.67 MB	MSLS	73.5 ± 0.5	68.4 ± 0.8	41.0 ± 0.8	38.6 ± 1.8	79.4 ± 0.5	84.7 ± 0.7
ResNet-18 *conv5_x*	NetVLAD	32768	22.28 GF	42.92 MB	MSLS	75.7 ± 0.7	75.7 ± 0.6	49.9 ± 1.6	41.3 ± 0.2	84.1 ± 0.4	91.3 ± 0.4
ResNet-50 *conv4_x*	GeM	1024	40.61 GF	32.71 MB	MSLS	77.4 ± 0.6	72.0 ± 0.5	55.4 ± 2.5	45.7 ± 1.0	83.9 ± 0.6	91.2 ± 0.7
ResNet-50 *conv4_x*	NetVLAD	65536	40.51 GF	33.21 MB	MSLS	**80.9 ± 0.0**	**76.9 ± 0.2**	**62.8 ± 0.9**	**51.5 ± 1.2**	**87.2 ± 0.3**	**93.8 ± 0.2**
ResNet-50 *conv5_x*	GeM	2048	50.54 GF	89.88 MB	MSLS	74.7 ± 0.4	70.6 ± 0.6	46.3 ± 1.3	42.1 ± 0.5	82.5 ± 0.5	89.8 ± 0.4
ResNet-50 *conv5_x*	NetVLAD	131072	50.35 GF	90.88 MB	MSLS	74.7 ± 0.2	75.2 ± 0.5	52.4 ± 0.8	44.0 ± 1.1	85.5 ± 0.4	91.3 ± 0.7

Table 2. **ResNets:** The advantages of cropping the ResNets at *conv4_x* for visual geo-localization.

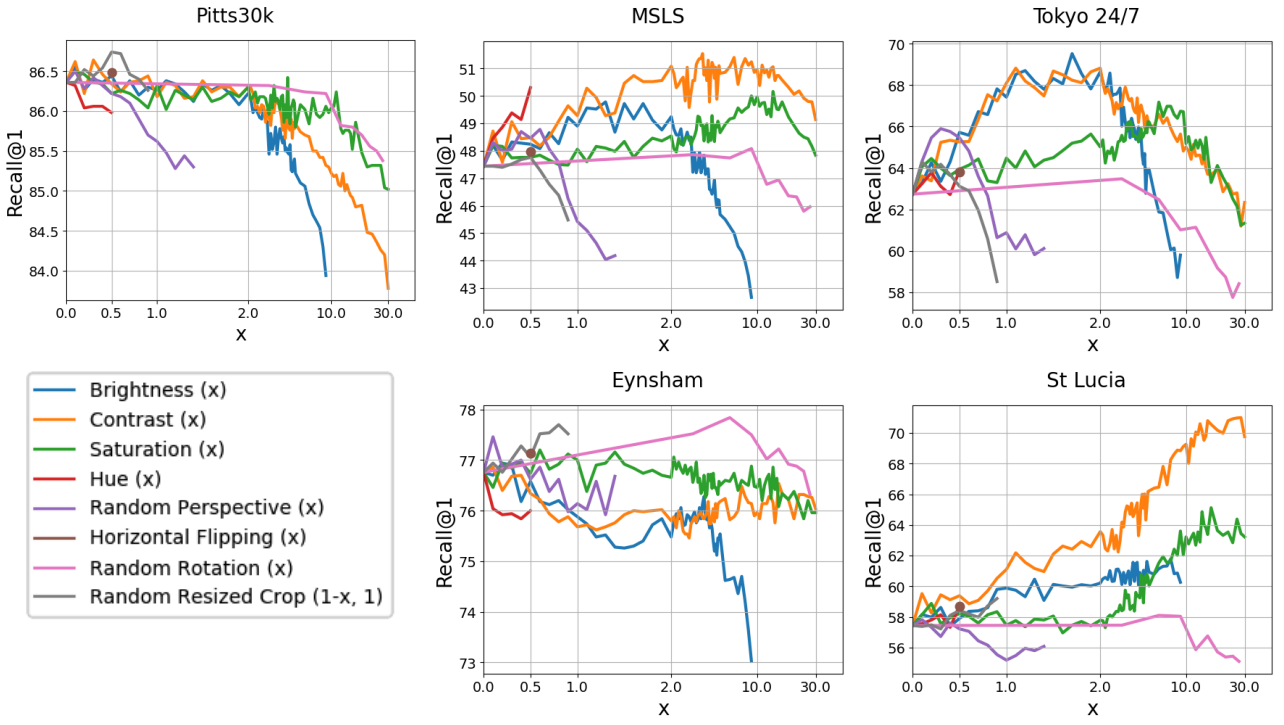


Figure 3. **Data Augmentation.** Results obtained when applying a number of popular data augmentation techniques during the training. We used PyTorch's transforms classes, and the x-axis relates to the parameter passed to the class. Brightness, contrast, saturation and hue are all performed with `ColorJittering()`. For `RandomPerspective()` and `RandomRotation()`, the parameter refers to the first argument (`distortion_scale` and `degrees` respectively). Regarding `RandomResizedCrop()`, we use the value as $(1-x, 1)$ for `scale` so that all transformations have their origin in the same point (*i.e.* $x = 0$ equals to the identity transformation), and the crops are then resized to the original resolution. When used, `HorizontalFlipping()` is applied with a probability of 0.5. Please refer to the PyTorch documentation for further information.

of 1 when extracting query descriptors at inference time. This approach can give good results at the cost of slower computation, which may or may not be an issue depending on the application's scalability requirements. Besides this common choice, we experiment with other engineering solutions that allow stacking multiple queries in a batch, in-

Backbone	Aggregation Method	Features Dim	Training Dataset	R@1 Pitts30k	R@1 MSLS	R@1 Tokyo 24/7	R@1 R-SF	R@1 Eynsham	R@1 St Lucia
ResNet-18	SPOC [2]	256	Pitts30k	60.6 ± 0.9	16.5 ± 0.5	15.2 ± 1.1	10.4 ± 0.3	41.0 ± 2.0	29.0 ± 1.5
ResNet-18	MAC [20]	256	Pitts30k	57.3 ± 0.5	25.6 ± 0.4	15.2 ± 1.3	15.5 ± 0.3	49.6 ± 0.7	26.6 ± 1.0
ResNet-18	RMAC [24]	256	Pitts30k	63.2 ± 0.4	28.7 ± 0.6	22.7 ± 2.3	30.5 ± 1.4	64.0 ± 0.7	42.8 ± 1.3
ResNet-18	RRM [13]	256	Pitts30k	68.2 ± 0.5	21.4 ± 0.8	25.4 ± 1.4	21.7 ± 1.8	51.9 ± 0.8	33.7 ± 0.3
ResNet-18	GeM [19]	256	Pitts30k	77.8 ± 0.2	35.3 ± 0.5	35.3 ± 1.1	34.2 ± 1.7	64.3 ± 1.2	46.2 ± 0.4
ResNet-18	GeM + FC 256	256	Pitts30k	72.4 ± 0.7	26.4 ± 0.5	27.5 ± 1.2	29.0 ± 1.2	59.3 ± 1.0	39.1 ± 0.8
ResNet-18	NetVLAD + PCA 256	256	Pitts30k	80.7 ± 0.7	38.3 ± 1.2	41.7 ± 0.8	35.9 ± 1.8	68.9 ± 1.1	45.4 ± 2.2
ResNet-18	CRN + PCA 256	256	Pitts30k	**82.0 ± 0.7**	**43.6 ± 0.7**	**47.7 ± 0.9**	**45.1 ± 0.3**	**71.3 ± 0.8**	**51.3 ± 3.4**
ResNet-18	GeM + FC 2048	2048	Pitts30k	75.0 ± 0.4	29.9 ± 0.6	34.5 ± 0.4	36.1 ± 0.2	63.7 ± 0.3	45.1 ± 2.1
ResNet-18	NetVLAD + PCA 2048	2048	Pitts30k	85.0 ± 0.4	45.0 ± 1.5	56.6 ± 0.7	53.2 ± 2.4	75.4 ± 1.1	54.6 ± 3.0
ResNet-18	CRN + PCA 2048	2048	Pitts30k	**85.7 ± 0.3**	**50.6 ± 0.6**	**61.0 ± 1.6**	**62.8 ± 1.2**	**77.4 ± 0.5**	**61.1 ± 2.7**
ResNet-18	NetVLAD [1]	16384	Pitts30k	86.4 ± 0.3	47.4 ± 1.2	63.4 ± 1.2	61.4 ± 1.5	76.8 ± 1.2	57.6 ± 3.3
ResNet-18	CRN [12]	16384	Pitts30k	**86.8 ± 0.1**	**53.2 ± 0.7**	**68.8 ± 1.0**	**69.0 ± 0.6**	**79.1 ± 0.3**	**64.8 ± 3.2**
ResNet-50	SPOC [2]	1024	Pitts30k	60.9 ± 0.5	19.2 ± 0.4	14.0 ± 0.5	9.0 ± 0.7	40.5 ± 2.3	27.1 ± 1.5
ResNet-50	MAC [20]	1024	Pitts30k	77.6 ± 0.2	36.2 ± 0.7	36.2 ± 1.4	34.8 ± 0.7	72.9 ± 0.3	51.3 ± 2.4
ResNet-50	RMAC [24]	1024	Pitts30k	74.9 ± 1.0	34.8 ± 0.8	41.8 ± 0.6	46.4 ± 1.0	73.1 ± 0.7	**68.7 ± 0.5**
ResNet-50	RRM [13]	1024	Pitts30k	72.8 ± 0.2	27.9 ± 0.6	28.3 ± 0.8	28.6 ± 1.0	65.9 ± 0.9	45.1 ± 1.7
ResNet-50	GeM [19]	1024	Pitts30k	82.0 ± 0.3	38.0 ± 0.1	41.5 ± 1.8	45.4 ± 2.0	66.3 ± 2.5	59.0 ± 1.4
ResNet-50	NetVLAD + PCA 1024	1024	Pitts30k	83.9 ± 0.7	46.5 ± 2.0	59.4 ± 1.2	53.2 ± 3.8	72.5 ± 0.3	57.7 ± 2.0
ResNet-50	CRN + PCA 1024	1024	Pitts30k	**84.1 ± 0.4**	**49.9 ± 0.8**	**64.6 ± 1.2**	**58.8 ± 0.1**	**74.3 ± 0.2**	63.4 ± 0.4
ResNet-50	GeM + FC 2048	2048	Pitts30k	80.1 ± 0.2	33.7 ± 0.3	43.6 ± 1.6	48.2 ± 1.2	70.0 ± 0.3	56.0 ± 1.7
ResNet-50	NetVLAD + PCA 2048	2048	Pitts30k	84.4 ± 0.4	47.9 ± 2.0	62.6 ± 1.7	56.0 ± 2.9	74.1 ± 0.4	58.9 ± 1.6
ResNet-50	CRN + PCA 2048	2048	Pitts30k	**84.7 ± 0.3**	**51.2 ± 0.8**	**67.1 ± 0.7**	**62.3 ± 0.3**	**75.8 ± 0.2**	**65.0 ± 0.1**
ResNet-50	NetVLAD [1]	65536	Pitts30k	**86.0 ± 0.1**	50.7 ± 2.0	69.8 ± 0.8	67.1 ± 2.3	77.7 ± 0.4	60.2 ± 1.6
ResNet-50	CRN [12]	65536	Pitts30k	85.8 ± 0.2	**54.0 ± 0.8**	**73.1 ± 0.3**	**70.9 ± 0.2**	**79.7 ± 0.1**	**65.9 ± 0.4**
ResNet-18	SPOC [2]	256	MSLS	44.2 ± 1.0	39.5 ± 0.5	20.3 ± 1.3	9.5 ± 0.9	62.3 ± 0.6	58.8 ± 0.8
ResNet-18	MAC [20]	256	MSLS	60.4 ± 1.1	54.7 ± 1.8	20.4 ± 2.6	18.9 ± 2.0	76.3 ± 1.2	69.2 ± 1.2
ResNet-18	RMAC [24]	256	MSLS	58.1 ± 1.2	48.9 ± 2.0	29.1 ± 2.0	34.3 ± 1.4	73.3 ± 1.1	63.7 ± 2.7
ResNet-18	RRM [13]	256	MSLS	60.8 ± 1.5	54.9 ± 2.6	**44.4 ± 2.1**	30.9 ± 2.8	75.7 ± 1.5	68.7 ± 1.4
ResNet-18	GeM [19]	256	MSLS	71.6 ± 0.1	65.3 ± 0.2	42.8 ± 1.1	30.5 ± 0.8	80.3 ± 0.1	83.2 ± 0.9
ResNet-18	GeM + FC 256	256	MSLS	68.6 ± 1.1	59.6 ± 2.6	41.9 ± 2.7	31.3 ± 0.5	78.5 ± 2.0	76.1 ± 3.4
ResNet-18	NetVLAD + PCA 256	256	MSLS	74.2 ± 0.2	70.6 ± 0.3	43.6 ± 0.5	34.7 ± 1.7	84.4 ± 0.4	89.8 ± 0.5
ResNet-18	CRN + PCA 256	256	MSLS	**74.5 ± 0.8**	**72.1 ± 0.1**	44.1 ± 1.4	**35.1 ± 2.4**	**84.8 ± 0.3**	**91.6 ± 0.4**
ResNet-18	GeM + FC 2048	2048	MSLS	71.9 ± 1.0	64.0 ± 1.2	51.8 ± 0.9	37.6 ± 1.3	81.1 ± 0.9	79.2 ± 0.9
ResNet-18	NetVLAD + PCA 2048	2048	MSLS	**80.4 ± 0.4**	74.6 ± 0.2	55.6 ± 1.2	47.4 ± 1.1	86.4 ± 0.3	92.2 ± 0.3
ResNet-18	CRN + PCA 2048	2048	MSLS	80.1 ± 0.8	**75.8 ± 0.1**	**57.2 ± 2.3**	**47.8 ± 2.7**	**86.8 ± 0.3**	**93.2 ± 0.4**
ResNet-18	NetVLAD [1]	16384	MSLS	**81.6 ± 0.5**	75.8 ± 0.1	62.3 ± 1.6	**55.1 ± 0.9**	87.1 ± 0.2	92.1 ± 0.7
ResNet-18	CRN [12]	16384	MSLS	81.3 ± 0.7	**76.8 ± 0.0**	**63.8 ± 1.4**	53.9 ± 2.0	**87.5 ± 0.2**	**93.7 ± 0.1**
ResNet-50	SPOC [2]	1024	MSLS	47.5 ± 1.3	47.9 ± 1.5	20.6 ± 1.6	8.9 ± 1.0	68.3 ± 0.5	68.6 ± 1.4
ResNet-50	MAC [20]	1024	MSLS	76.0 ± 0.2	67.4 ± 1.6	45.3 ± 1.0	44.4 ± 2.6	84.6 ± 0.4	86.0 ± 0.7
ResNet-50	RMAC [24]	1024	MSLS	70.1 ± 0.8	62.0 ± 0.5	52.1 ± 2.3	**54.3 ± 1.8**	80.6 ± 0.5	85.9 ± 1.0
ResNet-50	RRM [13]	1024	MSLS	69.3 ± 1.0	67.4 ± 0.4	53.7 ± 0.8	43.7 ± 1.0	84.3 ± 0.5	84.8 ± 1.1
ResNet-50	GeM [19]	1024	MSLS	**77.4 ± 0.6**	72.0 ± 0.5	**55.4 ± 2.5**	45.7 ± 1.0	83.9 ± 0.6	91.2 ± 0.7
ResNet-50	NetVLAD + PCA 1024	1024	MSLS	**77.4 ± 0.2**	74.8 ± 0.3	51.3 ± 1.3	39.0 ± 1.3	85.2 ± 0.3	92.9 ± 0.3
ResNet-50	CRN + PCA 1024	1024	MSLS	77.3 ± 0.3	**75.6 ± 0.0**	51.8 ± 1.1	38.8 ± 1.0	**85.7 ± 0.3**	**94.1 ± 0.2**
ResNet-50	GeM + FC 2048	2048	MSLS	**79.2 ± 0.6**	73.5 ± 0.8	**64.0 ± 3.9**	**55.1 ± 2.4**	86.1 ± 0.7	90.3 ± 1.0
ResNet-50	NetVLAD + PCA 2048	2048	MSLS	78.5 ± 0.2	75.4 ± 0.2	52.8 ± 0.4	42.6 ± 1.3	85.8 ± 0.3	93.4 ± 0.4
ResNet-50	CRN + PCA 2048	2048	MSLS	78.3 ± 0.3	**76.3 ± 0.1**	54.3 ± 0.7	42.8 ± 1.6	**86.2 ± 0.4**	**94.4 ± 0.2**
ResNet-50	NetVLAD [1]	65536	MSLS	**80.9 ± 0.0**	76.9 ± 0.2	62.8 ± 0.9	51.5 ± 1.2	87.2 ± 0.3	93.8 ± 0.2
ResNet-50	CRN [12]	65536	MSLS	80.8 ± 0.2	**77.8 ± 0.1**	**63.6 ± 0.5**	**53.4 ± 1.4**	**87.5 ± 0.4**	**94.8 ± 0.3**

Table 3. **Aggregation methods.** Full table of aggregation methods, grouped by backbone and features dimension.

Backbone	Aggregation Method	Features Dim	FLOPs [GF]	Model Size [MB]	Training Dataset	R@1 Pitts30k	R@1 MSLS	R@1 Tokyo 24/7	R@1 R-SF	R@1 Eynsham	R@1 St Lucia
ResNet-18	GeM	256	17.29	10.63	Pitts30k	77.8 ± 0.2	35.3 ± 0.5	35.3 ± 1.1	34.2 ± 1.7	64.3 ± 1.2	46.2 ± 0.4
ResNet-50	GeM	1024	40.61	32.71	Pitts30k	**82.0 ± 0.3**	38.0 ± 0.1	41.5 ± 1.8	45.4 ± 2.0	66.3 ± 2.5	59.0 ± 1.4
ViT	CLS	768	82.31	350.96	Pitts30k	79.2 ± 1.5	39.0 ± 0.8	44.5 ± 3.2	48.3 ± 2.5	67.6 ± 1.2	**69.6 ± 2.0**
CCT	CLS	384	22.34	190.39	Pitts30k	76.3 ± 1.4	39.5 ± 0.4	39.0 ± 1.7	44.4 ± 0.4	50.8 ± 2.1	57.3 ± 2.6
CCT	SeqPool	384	26.19	221.92	Pitts30k	81.1 ± 1.0	46.9 ± 1.2	51.5 ± 0.8	57.8 ± 1.5	**75.2 ± 1.1**	63.6 ± 2.6
CCT	GeM	384	22.36	191.24	Pitts30k	79.6 ± 0.3	**47.8 ± 0.7**	**52.3 ± 2.0**	**61.3 ± 0.1**	71.0 ± 0.8	59.1 ± 2.0
ResNet-18	NetVLAD	16384	17.27	10.76	Pitts30k	**86.4 ± 0.3**	47.4 ± 1.2	63.4 ± 1.2	61.4 ± 1.5	76.8 ± 1.2	57.6 ± 3.3
ResNet-50	NetVLAD	65536	40.51	33.21	Pitts30k	86.0 ± 0.1	50.7 ± 2.0	**69.8 ± 0.8**	67.1 ± 2.3	**77.7 ± 0.4**	**60.2 ± 1.6**
CCT	NetVLAD	24576	18.53	160.08	Pitts30k	84.6 ± 0.3	**52.5 ± 1.9**	69.1 ± 0.4	**73.5 ± 1.4**	72.6 ± 0.6	56.1 ± 3.3
ResNet-18	GeM	256	17.29	10.63	MSLS	71.6 ± 0.1	65.3 ± 0.2	42.8 ± 1.1	30.5 ± 0.8	80.3 ± 0.1	83.2 ± 0.9
ResNet-50	GeM	1024	40.61	32.71	MSLS	77.4 ± 0.6	72.0 ± 0.5	55.4 ± 2.5	45.7 ± 1.0	83.9 ± 0.6	91.2 ± 0.7
ViT	CLS	768	82.31	350.96	MSLS	**82.9 ± 0.6**	**73.5 ± 0.6**	**59.9 ± 4.4**	**65.0 ± 1.1**	84.5 ± 1.0	93.6 ± 0.7
CCT	CLS	384	22.34	190.39	MSLS	79.6 ± 0.3	71.1 ± 0.4	52.0 ± 1.1	49.9 ± 1.8	85.6 ± 0.1	**94.0 ± 0.3**
CCT	SeqPool	384	26.19	221.92	MSLS	81.4 ± 0.8	71.0 ± 0.9	59.1 ± 3.2	60.5 ± 1.5	**86.1 ± 0.6**	92.4 ± 1.1
CCT	GeM	384	22.36	191.24	MSLS	78.7 ± 0.6	72.0 ± 0.6	48.8 ± 1.2	48.6 ± 2.9	83.9 ± 0.1	92.9 ± 0.7
ResNet-18	NetVLAD	16384	17.27	10.76	MSLS	81.6 ± 0.5	75.8 ± 0.1	62.3 ± 1.6	55.1 ± 0.9	87.1 ± 0.2	92.1 ± 0.7
ResNet-50	NetVLAD	65536	40.51	33.21	MSLS	80.9 ± 0.0	76.9 ± 0.2	62.8 ± 0.9	51.5 ± 1.2	87.2 ± 0.3	93.8 ± 0.2
CCT	NetVLAD	24576	18.53	160.08	MSLS	**85.1 ± 0.2**	**79.9 ± 0.3**	**70.3 ± 2.0**	**65.9 ± 1.3**	**87.4 ± 0.2**	**98.4 ± 0.2**

Table 4. **Transformers** Comparison of traditional CNN architectures with novel Transformers-based approaches.

Backbone	Aggregation Method	Mining Method	Training Dataset	R@1 Pitts30k	R@1 MSLS	R@1 Tokyo 24/7	R@1 R-SF	R@1 Eynsham	R@1 St Lucia
ResNet-18	GeM	Random	Pitts30k	73.7 ± 0.7	30.5 ± 0.5	31.3 ± 0.8	24.0 ± 1.2	58.2 ± 1.4	41.0 ± 1.2
ResNet-18	GeM	Full database mining	Pitts30k	**77.8 ± 0.2**	**35.3 ± 0.5**	**35.3 ± 1.1**	**34.2 ± 1.7**	**64.3 ± 1.2**	**46.2 ± 0.4**
ResNet-18	GeM	Partial database mining	Pitts30k	76.5 ± 0.3	34.2 ± 1.3	33.9 ± 1.4	32.9 ± 0.7	64.0 ± 2.4	45.6 ± 0.9
ResNet-18	NetVLAD	Random	Pitts30k	83.9 ± 0.5	43.6 ± 0.5	55.1 ± 1.3	53.8 ± 1.1	76.3 ± 0.6	53.5 ± 1.4
ResNet-18	NetVLAD	Full database mining	Pitts30k	**86.4 ± 0.3**	**47.4 ± 1.2**	**63.4 ± 1.2**	61.4 ± 1.5	**76.8 ± 1.2**	**57.6 ± 3.3**
ResNet-18	NetVLAD	Partial database mining	Pitts30k	86.2 ± 0.3	47.3 ± 0.4	61.2 ± 0.5	**62.9 ± 0.3**	76.6 ± 0.5	57.1 ± 1.6
ResNet-50	GeM	Random	Pitts30k	77.9 ± 1.0	34.3 ± 1.3	40.1 ± 1.0	35.5 ± 3.0	63.8 ± 0.9	52.3 ± 1.4
ResNet-50	GeM	Full database mining	Pitts30k	82.0 ± 0.3	38.0 ± 0.1	41.5 ± 1.8	45.4 ± 2.0	66.3 ± 2.5	59.0 ± 1.4
ResNet-50	GeM	Partial database mining	Pitts30k	**82.3 ± 0.0**	**39.0 ± 0.4**	**43.5 ± 0.2**	**45.5 ± 1.7**	**67.7 ± 1.4**	**61.0 ± 2.0**
ResNet-50	NetVLAD	Random	Pitts30k	83.4 ± 0.6	45.0 ± 0.3	61.9 ± 2.1	55.8 ± 1.5	75.0 ± 1.8	52.6 ± 1.2
ResNet-50	NetVLAD	Full database mining	Pitts30k	**86.0 ± 0.1**	**50.7 ± 2.0**	**69.8 ± 0.8**	**67.1 ± 2.3**	**77.7 ± 0.4**	**60.2 ± 1.6**
ResNet-50	NetVLAD	Partial database mining	Pitts30k	85.5 ± 0.3	48.6 ± 3.1	66.7 ± 4.1	65.0 ± 4.3	77.6 ± 1.3	59.0 ± 4.1
ResNet-18	GeM	Random	MSLS	62.2 ± 0.3	50.6 ± 0.6	28.8 ± 0.8	17.1 ± 1.0	70.2 ± 0.6	71.4 ± 1.0
ResNet-18	GeM	Full database mining	MSLS	70.1 ± 1.1	61.8 ± 0.5	**42.8 ± 1.4**	**31.3 ± 1.2**	79.3 ± 0.2	81.0 ± 0.9
ResNet-18	GeM	Partial database mining	MSLS	**71.6 ± 0.1**	**65.3 ± 0.2**	**42.8 ± 1.1**	30.5 ± 0.8	**80.3 ± 0.1**	**83.2 ± 0.9**
ResNet-18	NetVLAD	Random	MSLS	73.3 ± 0.7	61.5 ± 1.4	45.0 ± 1.5	34.8 ± 0.2	84.9 ± 0.3	79.7 ± 1.7
ResNet-18	NetVLAD	Full database mining	MSLS	-	-	-	-	-	-
ResNet-18	NetVLAD	Partial database mining	MSLS	**81.6 ± 0.5**	**75.8 ± 0.1**	**62.3 ± 1.6**	**55.1 ± 0.9**	**87.1 ± 0.2**	**92.1 ± 0.7**
ResNet-50	GeM	Random	MSLS	69.5 ± 1.2	57.4 ± 1.1	43.5 ± 3.3	31.1 ± 0.9	78.8 ± 0.5	78.3 ± 1.2
ResNet-50	GeM	Full database mining	MSLS	77.3 ± 0.3	69.7 ± 0.2	52.4 ± 1.7	45.3 ± 0.2	**84.2 ± 0.0**	91.0 ± 0.2
ResNet-50	GeM	Partial database mining	MSLS	**77.4 ± 0.6**	**72.0 ± 0.5**	**55.4 ± 2.5**	**45.7 ± 1.0**	83.9 ± 0.6	**91.2 ± 0.7**
ResNet-50	NetVLAD	Random	MSLS	74.9 ± 0.4	63.6 ± 1.3	41.9 ± 1.6	34.6 ± 2.3	85.5 ± 0.2	80.9 ± 0.4
ResNet-50	NetVLAD	Full database mining	MSLS	-	-	-	-	-	-
ResNet-50	NetVLAD	Partial database mining	MSLS	**80.9 ± 0.0**	**76.9 ± 0.2**	**62.8 ± 0.9**	**51.5 ± 1.2**	**87.2 ± 0.3**	**93.8 ± 0.2**

Table 5. **Mining methods.**

vestigating if it is possible to simultaneously also improve the recalls. We group the methods into pre-processing, post-processing, and predictions refinement, according to where in the pipeline they are applied (see diagram in Fig. 1 of main paper).

In Tab. 6 we report the full results of our pre/post-processing and predictions refinement experiments, while the following is a thorough explanation of how such methods are applied. With respect to pre-processing approaches, in **Hard Resize** we perform an anisotropic resize of the

Backbone	Aggregation Method	Pre/Post-Processing Method	Pre-Proc.	Post-Proc.	Batch Parall.	Training Dataset.	R@1 Pitts30k	R@1 MSLS	R@1 Tokyo 24/7	R@1 R-SF	R@1 Eynsham	R@1 St Lucia
ResNet-18	GeM	Hard Resize	Y	N	Y	Pitts30k	**77.8 ± 0.2**	35.3 ± 0.5	31.8 ± 0.9	33.2 ± 2.1	**64.3 ± 1.2**	**46.2 ± 0.4**
ResNet-18	GeM	Single Query	Y	N	N	Pitts30k	**77.8 ± 0.2**	**35.6 ± 0.6**	35.3 ± 1.1	34.2 ± 1.7	**64.3 ± 1.2**	**46.2 ± 0.4**
ResNet-18	GeM	Central Crop	Y	N	Y	Pitts30k	**77.8 ± 0.2**	34.8 ± 0.5	**36.4 ± 1.1**	32.6 ± 1.4	**64.3 ± 1.2**	**46.2 ± 0.4**
ResNet-18	GeM	Five Crops Mean	Y	Y	Y	Pitts30k	75.4 ± 0.3	30.2 ± 0.2	35.9 ± 0.5	34.4 ± 2.0	59.1 ± 0.7	43.3 ± 0.8
ResNet-18	GeM	Nearest Crop	Y	Y	Y	Pitts30k	74.8 ± 0.1	28.3 ± 0.3	33.8 ± 1.3	**35.7 ± 1.6**	55.5 ± 0.8	39.4 ± 0.5
ResNet-18	GeM	Majority Voting	Y	Y	Y	Pitts30k	75.1 ± 0.0	29.1 ± 0.4	34.8 ± 1.5	35.3 ± 1.3	51.8 ± 0.2	41.3 ± 0.5
ResNet-18	NetVLAD	Hard Resize	Y	N	Y	Pitts30k	**86.4 ± 0.3**	47.4 ± 1.2	58.3 ± 1.4	58.9 ± 1.1	76.8 ± 1.2	**57.6 ± 3.3**
ResNet-18	NetVLAD	Single Query	Y	N	N	Pitts30k	**86.4 ± 0.3**	47.5 ± 1.3	63.4 ± 1.2	61.4 ± 1.5	76.8 ± 1.2	**57.6 ± 3.3**
ResNet-18	NetVLAD	Central Crop	Y	N	Y	Pitts30k	**86.4 ± 0.3**	**48.0 ± 1.3**	63.2 ± 0.2	57.8 ± 0.4	76.8 ± 1.2	**57.6 ± 3.3**
ResNet-18	NetVLAD	Five Crops Mean	Y	Y	Y	Pitts30k	85.1 ± 0.2	45.3 ± 1.3	63.0 ± 0.7	60.9 ± 1.7	**78.9 ± 0.9**	54.6 ± 2.8
ResNet-18	NetVLAD	Nearest Crop	Y	Y	Y	Pitts30k	84.8 ± 0.2	46.0 ± 1.5	**67.0 ± 1.4**	**64.8 ± 0.7**	75.7 ± 1.4	53.0 ± 2.5
ResNet-18	NetVLAD	Majority Voting	Y	Y	Y	Pitts30k	84.8 ± 0.3	45.2 ± 1.4	66.9 ± 1.1	64.7 ± 0.7	77.1 ± 1.1	53.4 ± 2.3
ResNet-50	GeM	Hard Resize	Y	N	Y	Pitts30k	**82.0 ± 0.3**	38.0 ± 0.1	34.6 ± 1.4	40.7 ± 1.8	**66.3 ± 2.5**	**59.0 ± 1.4**
ResNet-50	GeM	Single Query	Y	N	N	Pitts30k	**82.0 ± 0.3**	**38.2 ± 0.3**	41.5 ± 1.8	45.4 ± 2.0	**66.3 ± 2.5**	**59.0 ± 1.4**
ResNet-50	GeM	Central Crop	Y	N	Y	Pitts30k	**82.0 ± 0.3**	37.5 ± 0.3	40.4 ± 0.9	41.0 ± 2.6	**66.3 ± 2.5**	**59.0 ± 1.4**
ResNet-50	GeM	Five Crops Mean	Y	Y	Y	Pitts30k	80.4 ± 0.1	33.2 ± 0.1	39.8 ± 2.0	43.8 ± 0.9	65.0 ± 2.4	54.4 ± 1.3
ResNet-50	GeM	Nearest Crop	Y	Y	Y	Pitts30k	79.2 ± 0.2	30.8 ± 0.2	**43.5 ± 1.4**	**46.9 ± 1.4**	63.5 ± 2.2	52.6 ± 1.4
ResNet-50	GeM	Majority Voting	Y	Y	Y	Pitts30k	79.7 ± 0.0	31.5 ± 0.1	43.0 ± 2.0	44.8 ± 1.2	62.9 ± 2.3	52.8 ± 0.9
ResNet-50	NetVLAD	Hard Resize	Y	N	Y	Pitts30k	**86.0 ± 0.1**	50.7 ± 2.0	64.3 ± 1.9	64.3 ± 1.2	77.7 ± 0.4	**60.2 ± 1.6**
ResNet-50	NetVLAD	Single Query	Y	N	N	Pitts30k	**86.0 ± 0.1**	50.6 ± 1.9	69.8 ± 0.8	67.1 ± 2.3	77.7 ± 0.4	**60.2 ± 1.6**
ResNet-50	NetVLAD	Central Crop	Y	N	Y	Pitts30k	**86.0 ± 0.1**	**50.9 ± 1.9**	68.3 ± 1.4	64.6 ± 2.2	77.7 ± 0.4	**60.2 ± 1.6**
ResNet-50	NetVLAD	Five Crops Mean	Y	Y	Y	Pitts30k	84.7 ± 0.1	47.4 ± 1.9	68.0 ± 2.2	66.5 ± 1.5	**78.6 ± 0.3**	54.3 ± 2.8
ResNet-50	NetVLAD	Nearest Crop	Y	Y	Y	Pitts30k	84.2 ± 0.2	47.0 ± 1.7	72.3 ± 1.3	**68.4 ± 0.8**	76.8 ± 0.5	52.3 ± 2.3
ResNet-50	NetVLAD	Majority Voting	Y	Y	Y	Pitts30k	84.3 ± 0.2	47.1 ± 1.7	**72.8 ± 0.8**	68.1 ± 1.3	77.5 ± 0.4	53.4 ± 2.2

Table 6. **Query pre/post-processing.** Results with different pre/post-processing methods are shown in the table. The batch parallelization column indicates if images have to be processed one by one or if they can be stacked in a batch for parallel computation.

query to the same dimension as the database images (effectively leaving the query unchanged if the query and database images' dimensions already match); for **Single Query** we isotropically resize so that the query's shortest side is equal to the database images' shortest side, the aspect ratio is preserved, images are not padded, and if they are of varying resolutions, they cannot be stacked in a batch; in **Central Crop** we isotropically resize to the smallest resolution that can accommodate a rectangular region of the size of the database images, and a central crop of such size is taken; in **Five Crops** we produce five square crops of the database images shortest side.

Regarding post-processing and predictions refinement methods, with **Mean** we simply compute the mean of the descriptors of the five crops; in **Nearest Crop** we choose the prediction with shortest descriptors distance from at least one crop; with **Majority Voting** we implement a voting mechanism taking into account the distances from each crop's first 20 predictions.

Discussion. It can be seen in Tab. 6 that, as expected, *Hard Resize*, *Single Query* and *Central Crop* produce exactly the same results when queries have the same size as the database images (Pitts30k, Eynsham and St Lucia), as in these cases they correspond to an identity transformation. On the other hand, in Tokyo 24/7 and R-SF, where roughly half of the queries are vertical (*i.e.*, the height is longer than the width), more complex techniques, such as *Nearest Crop* and *Majority Voting*, on average yield better results. This is particularly noticeable with more robust networks. Furthermore, these methods allow multiple queries to be stacked in a single batch, as the crops they operate on all have the same dimensions. Finally, we can state that the ideal approach highly depends on the application: for robotics, if all images come from the same devices (and have the same resolution as database images), simply applying *Hard Resize* (which in this case results in no resize at all) leads to best results; in cases where queries can have unrestricted resolutions, *Single Query* represents a simple approach with acceptable results, while *Nearest Crop* produces the best R@1 and offers the possibility of batch parallelization, which is crucial for scalability.

3.7. Nearest Neighbor Search and Inference Time

As stated in the main paper, matching time can significantly impact inference time (see Sec. 4.7 of the main paper) and memory footprint. This section reports experiments with the different indexing techniques listed in the main paper, reporting extensive results on all datasets. The goal is to investigate how and if it is possible to make this computation more efficient.

Discussion. In Fig. 4 we show results for various methods. Among the most outstanding results, using an Inverted File

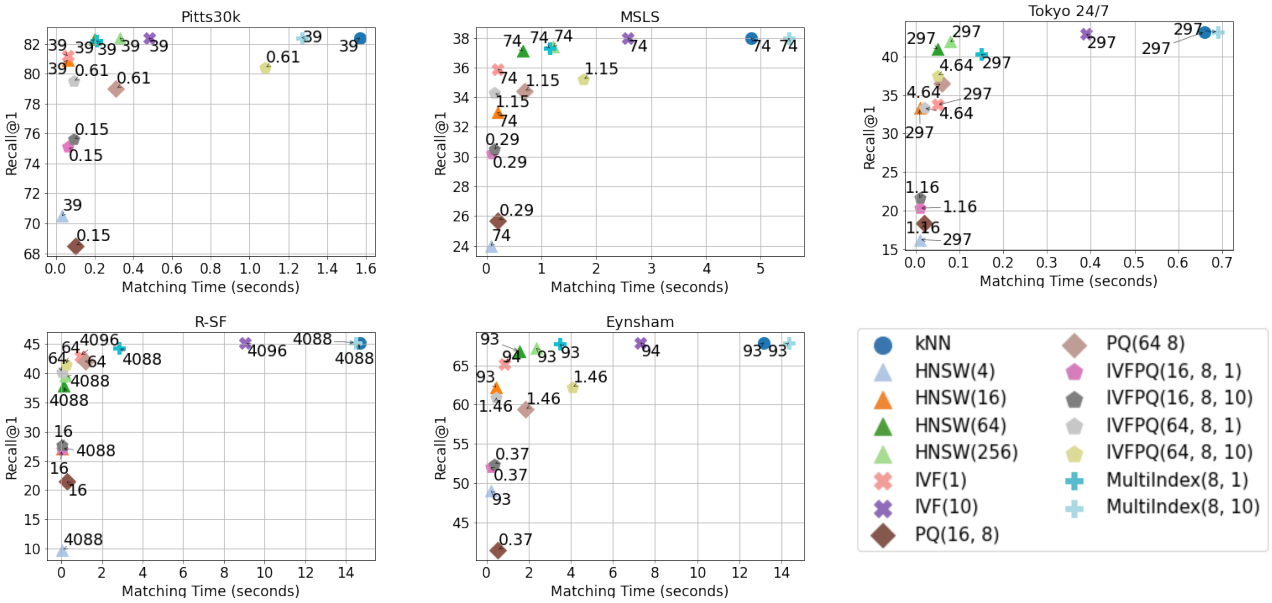


Figure 4. **Optimized kNN indexing: faster search & lower memory footprint.** The plots shows a number of efficient kNN variants, on different datasets, which are applied on features extracted with a ResNet-50 + GeM (features dimension 1024) trained on Pitts30k. On the x-axis is the matching time in seconds for all the queries in each dataset, while on the y-axis is the recall@1. The numbers next to the dots represent the RAM requirements of the method (memory footprint) in MB. Besides exhaustive kNN, we employ inverted file indexes (IVF) [23], product quantization (with and without inverted indexes, respectively PQ and IVFPQ) [11], the inverted multi index (MultiIndex) [3] and hierarchical navigable small world graphs (HNSW) [16]. In the legend, the parameters are shown for each method. The last parameter of IVFPQ, MultiIndex and IVF, which is either 1 or 10, represents the percentage of Voronoi cells to search, given that the search space has been split into 1000 Voronoi cells.

Index (IVF) [23] can lead to a reduction of matching time of roughly 20 times, while lowering the recall@1 of less than 2% on average. The Hierarchical Navigable Small World graphs (HNSW) [16] and the Inverted Multi-Index (Multi-Index) [3] bring similar achievements as the Inverted File Index, with slightly slower computation but higher recall. Regarding memory footprint, the Inverted File Index with Product Quantization (IVFPQ) [11] can reduce it by a factor of 64, and in the largest dataset, namely Revisited San Francisco (R-SF), it reduces matching time by 98.5%, with a drop in accuracy from 45.4% to 41.4%. However, the improvements become less obvious as the size of the database diminishes, as shown in the plots. Compared to the Inverted File Index with Product Quantization, the simpler Product Quantization leads to the same memory savings but lower recall-speed ratio, making Inverted File Index with Product Quantization the Pareto optimal solution concerning the recall@1 and the matching time when memory efficiency is an issue.

4. Additional Experiments

While the different components of a Visual Geo-localization system, as shown in Fig. 1 of the main paper, have been thoroughly studied in Sec. 4 of the main paper and Sec. 3 of this supp. material, in this section, we investigate several other factors. In Sec. 4.1, we aim to understand if models trained on large-scale landmark retrieval datasets can be reliably used for VG. In Sec. 4.2, we use images from those datasets as distractors to increase the size of the database of various orders of magnitude. The same landmark retrieval datasets are used in Sec. 4.4 to see if models pretrained on landmark retrieval can be easily fine-tuned for VG. Finally, in Sec. 4.5, we explore the use of different metrics and how they relate to the final results.

4.1. The role of the training dataset

In this section, we further investigate the role of the training dataset. To this end, we compute results with publicly-available state-of-the-art models for image retrieval[2], which have been trained on large scale landmark retrieval datasets, and we compare them with analogous networks trained on Pitts30k and MSLS. Note that higher recalls could have been achieved using a NetVLAD+PCA, but the GeM + FC aggregation was preferred to obtain a fair comparison with the models provided by third parties as state-of-the-art trained on the landmark datasets. Our framework easily

[2] https://github.com/filipradenovic/cnnimageretrieval-pytorch

Source	Loss	Training Dataset	Backbone	Aggregation Method	R@1 Pitts30k	R@1 MSLS	R@1 Tokyo 24/7	R@1 R-SF	R@1 Eynsham	R@1 St Lucia
[19]	Triplet	GLDv1	ResNet-50	GeM + FC 2048	**84.1**	69.5	**77.8**	**76.4**	61.8	77.3
[19]	Triplet	Sfm120k	ResNet-50	GeM + FC 2048	83.4	64.5	75.2	75.6	68.8	73.9
-	Triplet	Pitts30k	ResNet-50	GeM + FC 2048	80.1	33.7	43.6	48.2	70.0	56.0
-	Triplet	MSLS	ResNet-50	GeM + FC 2048	79.2	**73.5**	64.0	55.1	**86.1**	**90.3**
[19]	Triplet	GLDv1	ResNet-101	GeM + FC 2048	**85.1**	72.4	**77.8**	**79.8**	61.6	83.4
[19]	Triplet	Sfm120k	ResNet-101	GeM + FC 2048	83.9	64.7	77.5	78.3	62.8	76.3
-	Triplet	Pitts30k	ResNet-101	GeM + FC 2048	82.4	40.0	47.2	57.5	75.9	61.7
-	Triplet	MSLS	ResNet-101	GeM + FC 2048	79.1	**75.3**	61.9	54.9	**86.0**	**92.5**

Table 7. **The role of the training dataset.** The table shows results with models trained on large scale landmark retrieval datasets.

allows for retrieval models trained by [19] and [21] to be automatically downloaded from their GitHub repositories and used to perform experiments on Visual Geo-localization datasets.

Discussion. Results are reported in Tab. 7. The experiments confirm that the choice of the training dataset plays a significant role in a network's robustness and generalization capabilities. We notice that models trained on large-scale landmark retrieval datasets, especially on GLDv1 [18], offer a decent off-the-shelf solution for many Visual Geo-localization datasets. In general, we see that these models benefit from the wide variety of images present in the landmark retrieval datasets and are able to learn robust features that guarantee good generalization performances in the VG setting. As for the training directly on VG datasets, it is noticeable how models trained on MSLS, thanks to its bigger size and variability, achieve more robustness on all datasets except for R-SF and Tokyo 24/7 than the same models trained on Pitts30k. The reason behind this fact is that the images of these last two datasets are made up of 360° views, unlike the front view scenarios that the model sees during training on MSLS. Finally, the poor generalization performances obtained training on Pitts30k can be understood in relation to the use of the rather big (in terms of the number of parameters) FC layer that inevitably leads to overfitting on the small size of the said dataset; in fact, Tab. 3 shows how using as aggregator a NetVLAD + PCA method, significantly reducing the number of parameters, leads to a better generalization. The takeaway messages should be to use the training set that presents similar viewpoints, if known, or otherwise the more general one, and choose the model with a number of parameters proportional to the dataset size.

These results are directly comparable with results from Tab. 3.

4.2. Scaling datasets with distractors

While in the past few years, software and hardware improvements have allowed us to obtain better and faster results, common datasets are still on a small to medium scale, and their coverage is still shallow compared to a realistic real-world application (see Tab. 1). As an example, the San Francisco dataset, although being one of the largest with a database of 1 million images, still covers only 9% of the city of San Francisco. As in our work, we aim to investigate VG's possible applications, and then we built a large-scale dataset with up to 8 million distractors. To this end, we used the 315 queries from Tokyo 24/7, and first built a small-scale database with their positives and several random images from Tokyo 24/7 database (to reach a total of 10.000 pictures). A 10 times bigger set was then built using the whole Tokyo 24/7 database, as well as the ones from Pitts30k and MSLS test sets. By using the database of San Francisco, we reached 1 million images, and, finally, we scaled it to 8 million by including the whole Google Landmark v2 [29], Places 365 [32], and MSLS train database.

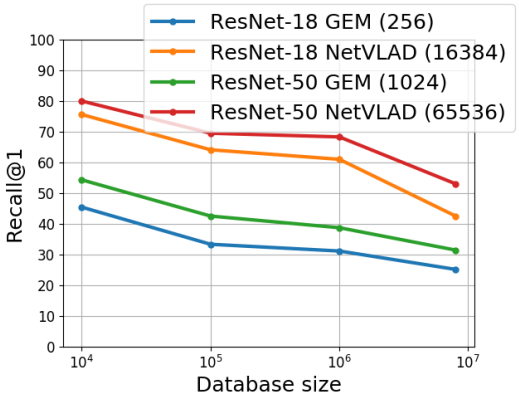


Figure 5. **Scaling datasets with distractors.** The plot shows the effects of exponentially increasing the size of the database up to 8M. In the legend, the descriptors dimensionality is shown between parentheses.

Discussion. Results are shown in Fig. 5. Unsurprisingly, we see that results steadily decrease as the size of the database increases, proving that the task is still far from

Model	Trained on Pitts30k		Trained on MSLS	
	R@1 Single DB	R@1 Multi DB	R@1 Single DB	R@1 Multi DB
ResNet-18 + GeM	57.9	42.2 (-27.1%)	74.4	65.1 (-12.5%)
ResNet-50 + GeM	60.9	53.4 (-12.3%)	79.4	71.6 (-9.82%)
ResNet-18 + NetVLAD	70.0	67.4 (-3.7%)	83.0	79.0 (-4.1%)
ResNet-50 + NetVLAD	71.4	68.7 (-3.8%)	83.2	79.0 (-5.0%)

Table 8. **All-data benchmark.** Using all queries from the six datasets, *Single DB* indicates the average result from matching the queries only to their respective database, *Multi DB* refers to matching the the queries to all six databases merged.

solved.

4.3. Testing on an ensemble of datasets

The experiment presented in Sec. 4.2 investigates how VG methods perform on a large-scale database built with 8M distractors, but with the queries all taken from a single dataset. However, in practice, the VG model may be tasked to geolocalize queries coming from different data distributions and geographical areas (e.g., Tokyo, San Francisco, *etc*.). To investigate how VG models fare in this situation, the benchmark also supports experiments considering an ensemble dataset that combines all the test queries and databases considered so far, *i.e*. Pitts30k, MSLS, Tokyo 24/7, R-SF, Eynsham, and St Lucia. In particular, here we report the results achieved matching the queries only to their respective database (*Single DB*) and matching the queries to all six databases merged (*Multi DB*).

Discussion. The results in Tab. 8 report the R@1 achieved by various methods both in the Single DB and Multi DB settings. In the Multi DB setting there is a clear decrease in recall with respect to averaging across the different datasets tested separately (Single DB), which demonstrates the difficulty of this scenario. Overall, the models trained on MSLS achieve better results than those trained on Pitts30k, which confirms that the larger number and variety of images of MSLS has a notable impact on the generalization capability of the model. We also observe that the percentile drop on the R@1 when going from the Single DB to the Multi DB setting is higher with GeM than with NetVLAD.

4.4. Pretraining the backbone on other datasets

In this section, we investigated whether pretraining the backbone of our VG system on datasets different from ImageNet can be beneficial for the training of the model. The datasets used for this purpose were Places 365 [32], a dataset for scene recognition, and Google Landmark v2 (GLDv2) [18, 29], a recent large-scale landmark retrieval/recognition dataset released by Google. The networks were trained with a standard classification approach for Places and using the ArcFace Loss [6] on GLDv2, following the idea proposed by [30].

Discussion. From the substantial number of experiments reported in Tab. 9 the evidence is that in the vast majority of the cases it is not convenient to choose a dataset other than ImageNet to pretrain the backbone; however, the scores are quite close except in the cases of R-SF and Tokyo 24/7 datasets, where using a different pretrain dataset in place of ImageNet on average leads to a drop in performance. Even in the few cases where GLDv2 or Places365 achieves the highest score, the gap is, in practice, negligible. Furthermore, if we take into account the off-the-shelf availability of ImageNet pretrained backbones with respect to the far less common alternatives, it is even more clear that the former is the more advantageous choice.

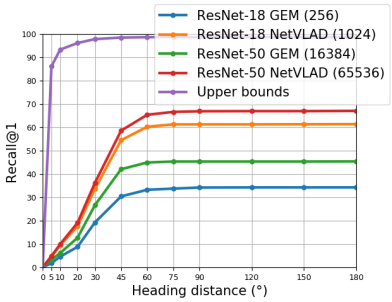


Figure 6. **Taking heading into account.** Recall@1 on R-SF when heading is taken into account. Different degrees of headings are taken into account on the x-axis. The "Upper bounds" curve refers to the percentage of queries that have at least one database image closer than 25 meters with a difference in heading lower than the given threshold. This corresponds to the upper bound of the recall@1.

4.5. Metrics

In previous experiments, we showed results based on recall@1, with a positive distance of 25 meters. In this section, we explore the use of different metrics. Specifically, we show how results would change if the heading is taken into account or setting the success threshold to other values than 25 meters. Moreover, we use various values of recalls. We compute these results using the models trained for Tab. 3 on Pitts30k and changing only the final metric at test time.

4.5.1 Taking heading/yaw/compass into account

As many datasets commonly used in Visual Geolocalization only have labels for GPS coordinates, it is often impossible to assess the difference between a query's heading and its predictions. While one might assume that a positive prediction is likely to have the same heading as its query, this might not always be the case, as in cities it is possible to find places that are self-similar in multiple directions (*e.g*. buildings facing each other with similar architectures). Moreover, two images representing the same scene might be taken from a very different viewpoint, within 25 meters from each other. To shed some light on this ques-

Backbone	Aggregation Method	Dataset	Training Dataset	R@1 Pitts30k	R@1 MSLS	R@1 Tokyo 24/7	R@1 R-SF	R@1 Eynsham	R@1 St Lucia
ResNet-18	GeM	ImageNet	Pitts30k	77.8 ± 0.2	35.3 ± 0.5	**35.3** ± 1.1	**34.2** ± 1.7	64.3 ± 1.2	46.2 ± 0.4
ResNet-18	GeM	GLDv2	Pitts30k	74.2 ± 0.4	30.9 ± 0.6	22.3 ± 1.9	20.4 ± 1.7	55.0 ± 2.0	43.3 ± 0.7
ResNet-18	GeM	Places 365	Pitts30k	**78.1** ± 1.0	**36.2** ± 0.9	31.8 ± 0.7	32.8 ± 1.6	**65.0** ± 2.1	**48.8** ± 2.1
ResNet-18	NetVLAD	ImageNet	Pitts30k	**86.4** ± 0.3	**47.4** ± 1.2	**63.4** ± 1.2	**61.4** ± 1.5	76.8 ± 1.2	**57.6** ± 3.3
ResNet-18	NetVLAD	GLDv2	Pitts30k	83.3 ± 0.5	39.9 ± 0.9	54.2 ± 2.3	41.1 ± 3.6	71.4 ± 2.6	46.8 ± 1.9
ResNet-18	NetVLAD	Places 365	Pitts30k	85.9 ± 0.4	**47.4** ± 0.6	57.9 ± 1.4	59.9 ± 3.2	**78.7** ± 0.7	50.4 ± 1.0
ResNet-50	GeM	ImageNet	Pitts30k	82.0 ± 0.3	38.0 ± 0.1	**41.5** ± 1.8	**45.4** ± 2.0	66.3 ± 2.5	59.0 ± 1.4
ResNet-50	GeM	GLDv2	Pitts30k	77.9 ± 0.5	35.2 ± 0.8	27.6 ± 2.1	37.2 ± 1.0	62.7 ± 1.6	48.4 ± 1.7
ResNet-50	GeM	Places 365	Pitts30k	**82.5** ± 0.4	**40.8** ± 0.3	41.3 ± 0.7	45.3 ± 0.6	**66.9** ± 1.3	**60.8** ± 1.6
ResNet-50	NetVLAD	ImageNet	Pitts30k	86.0 ± 0.1	**50.7** ± 2.0	**69.8** ± 0.8	**67.1** ± 2.3	**77.7** ± 0.4	**60.2** ± 1.6
ResNet-50	NetVLAD	GLDv2	Pitts30k	81.7 ± 0.6	43.5 ± 1.0	56.7 ± 0.9	54.1 ± 1.8	71.4 ± 0.6	42.3 ± 2.5
ResNet-50	NetVLAD	Places 365	Pitts30k	**86.2** ± 0.5	49.9 ± 2.0	66.3 ± 3.3	59.7 ± 3.5	75.4 ± 2.0	57.2 ± 5.5
ResNet-18	GeM	ImageNet	MSLS	**71.6** ± 0.1	**65.3** ± 0.2	**42.8** ± 1.1	**30.5** ± 0.8	**80.3** ± 0.1	**83.2** ± 0.9
ResNet-18	GeM	GLDv2	MSLS	60.7 ± 0.5	64.5 ± 0.7	30.9 ± 3.3	21.5 ± 0.8	79.2 ± 0.6	78.1 ± 1.0
ResNet-18	GeM	Places 365	MSLS	**71.6** ± 0.9	64.8 ± 1.1	36.6 ± 2.2	25.5 ± 0.3	80.1 ± 0.5	82.4 ± 0.6
ResNet-18	NetVLAD	ImageNet	MSLS	**81.6** ± 0.5	**75.8** ± 0.1	**62.3** ± 1.6	**55.1** ± 0.9	**87.1** ± 0.2	**92.1** ± 0.7
ResNet-18	NetVLAD	GLDv2	MSLS	73.3 ± 0.6	75.3 ± 0.3	53.4 ± 1.3	40.7 ± 2.9	86.1 ± 0.1	87.6 ± 0.9
ResNet-18	NetVLAD	Places 365	MSLS	79.7 ± 0.5	75.6 ± 0.2	61.5 ± 0.7	48.6 ± 1.5	86.5 ± 0.1	90.4 ± 0.4
ResNet-50	GeM	ImageNet	MSLS	77.4 ± 0.6	72.0 ± 0.5	**55.4** ± 2.5	**45.7** ± 1.0	83.9 ± 0.6	**91.2** ± 0.7
ResNet-50	GeM	GLDv2	MSLS	71.1 ± 1.7	72.4 ± 0.2	47.6 ± 0.4	35.8 ± 1.6	84.0 ± 0.4	86.1 ± 1.1
ResNet-50	GeM	Places 365	MSLS	**78.2** ± 1.1	**72.7** ± 0.6	51.8 ± 2.7	41.8 ± 2.2	**84.4** ± 0.2	89.3 ± 0.8
ResNet-50	NetVLAD	ImageNet	MSLS	**80.9** ± 0.0	76.9 ± 0.2	**62.8** ± 0.9	**51.5** ± 1.2	**87.2** ± 0.3	**93.8** ± 0.2
ResNet-50	NetVLAD	GLDv2	MSLS	74.7 ± 1.0	**77.4** ± 0.4	55.0 ± 1.7	45.4 ± 1.5	85.1 ± 0.5	87.7 ± 0.8
ResNet-50	NetVLAD	Places 365	MSLS	80.0 ± 1.1	75.6 ± 0.1	51.3 ± 3.3	44.8 ± 2.3	86.9 ± 0.1	91.3 ± 0.2

Table 9. **Pretraining the backbone on other datasets.**

tion, we compute recall@1 on the San Francisco dataset, for which heading labels are available, considering positives with a variable difference in heading from the query (Fig. 6). The distance threshold is fixed at 25 meters.

Discussion. We can see from Fig. 6 that roughly all (99.8%) correct predictions' heading are within 90° from the query's heading, 98% are within 60°, 90% within 45°, 57% within 30°, and only roughly 14% are within 10°. Moreover, we see that these results are pretty stable across all models. The figure clearly shows that post-processing techniques must be considered when an accurate pose estimation is needed.

4.5.2 Changing the positives' threshold distance

Although in most VG works [1, 12] the distance within which a database image is considered a positive is 25 meters, in the real world, one might require more or less accurate positions, depending on the task and final goal. Results are shown in Fig. 7, where we consider thresholds from 1 to 100 meters.

Discussion. Results are consistent across all models: as the threshold distance grows, we can see a fast rise in recall@1. Depending on the dataset, this rapid growth slows down somewhere between 10 and 25 meters. Recall@1 does not reach 90% for any dataset, even as the threshold grows as high as 50 meters.
The plots also give interesting insights into the datasets: looking at the Upper Bound line it is possible to understand which datasets are denser than others. For example, St Lucia has an upper bound of 88% at 5 meters, meaning that for 88% of the queries there is at least one positive within 5 a meters threshold. From the plots we see that a distance of 25 meters provides a reliable threshold for evaluation on all the considered datasets, as it ensures that close to 100% of queries have a relevant database image, and that random chance leads to recalls close to zero.

4.5.3 Other values of recall

In this section, we experiment using other values of N for the recall@N. Plots with recalls up to 100 are shown in Fig. 8.

Discussion. From Fig. 8 we can extract interesting in-

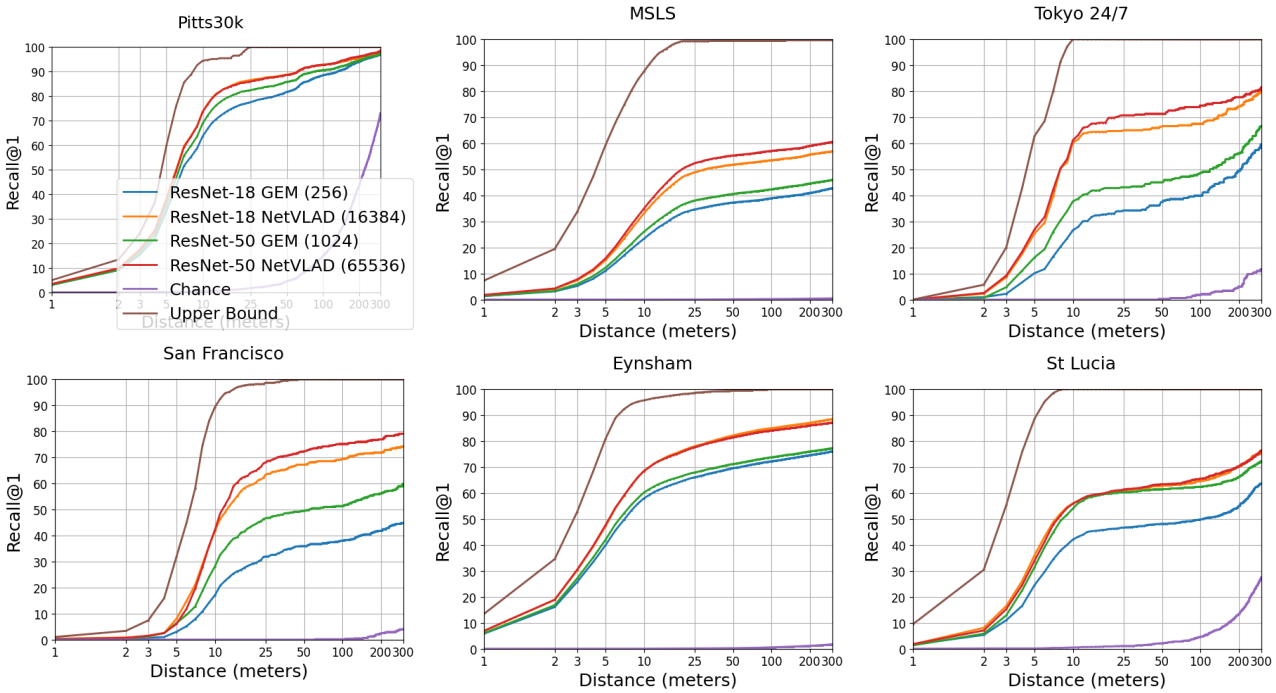
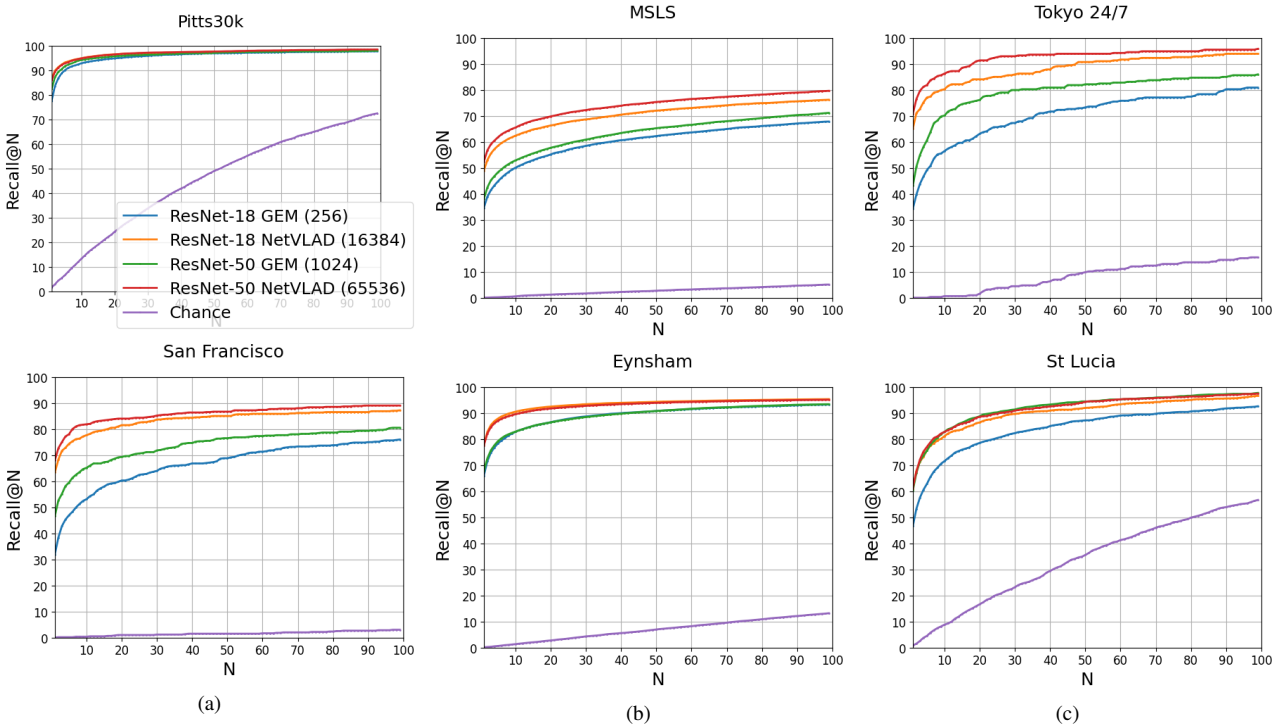
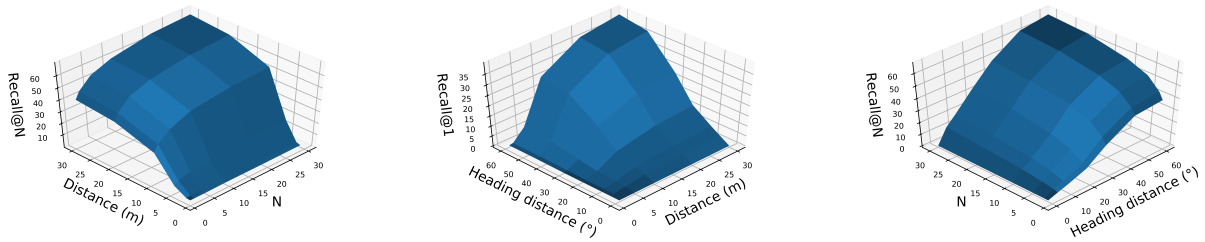


Figure 7. **Changing the positives' threshold distance.** Plots showing how Recall@1 changes when changing the positives' threshold (x-axis), expressed in meters. Moreover, we also show the upper bound (some queries might not have positives within a given distance) and chance, computed by choosing random predictions for each query. All models are trained on the Pitts30k dataset.



(a) (b) (c)

Figure 8. **Other values of recall.** Plots showing different values of recalls. On the x-axis is N $\in \{1, 2, 3... 100\}$, and on the y-axis the recall N.



(a) (b) (c)

Figure 9. **The relation between threshold distances, values of the recall and heading distance.** These 3D plots show how the three factors interact with each others. a) shows recall@N while changing the distances and values of the recall (N), and keeping heading distance at 180°; b) shows recall@1 varying heading angle and distance; c) keeping threshold distance at 25 meters, and varying the two other factors, shows the recall@N.

sights: we see that if a query's location is not found within 5 predictions, chances are rather low (35% on average over all methods on all datasets) that it is found within 20 predictions. This number is even lower for more challenging datasets: 19% for MSLS, 26% for San Francisco. Similarly, if a query's location is not found within one prediction, chances are 75% on average that it is found within 100 (53% for MSLS, 64% for San Francisco).
Moreover, the plot shows the upper bound for re-ranking methods like [8, 22]: these methods compute re-ranking over a limited number of predictions (usually 100) as the time complexity grows linearly with such number. For example, if 100 predictions are considered for re-ranking, the resulting recall@1 cannot be higher than the initial recall@100. These plots suggest that re-ranking over the top 20/30 predictions would give a similar performance at a much lower cost.

4.5.4 The relation between threshold distances, values of the recall and heading distance

In previous sections, we display results while changing one of the three factors between threshold distances, values of the recall, and heading distance at a time, here we investigate the relationships between any given pair of them. Fig. 9 shows how these factors interact with each other. To compute the results, we used a ResNet-50 + GeM trained on Pitts30k, and the recalls in the plots refer to the R-SF dataset (as it is the only one with heading labels).

5. Ethical implications

The technology of Visual Geo-localization can potentially be used to implement invasive forms of surveillance or social media monitoring, thus raising privacy concerns. The benchmark proposed in this manuscript is just a tool whose purpose is to provide a systematic and standardized approach to testing and comparing different Visual Geo-localization algorithms. As such, it cannot offer guarantees on the final use of the algorithms that it will help to evaluate. Therefore, we urge all researchers using this tool to be mindful of the potential misuses of their algorithms. For what concerns data, the framework relies only on pre-existing and publicly available datasets that are widely used in the community and focus on places rather than humans, and thus are considered safe to use.

References

[1] Relja Arandjelović, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(6):1437–1451, 2018. 1, 4, 6, 12

[2] Artem Babenko and Victor Lempitsky. Aggregating deep convolutional features for image retrieval. *ICCV*, 10 2015. 4, 6

[3] Artem Babenko and Victor S. Lempitsky. The inverted multi-index. In *CVPR*, pages 3069–3076. IEEE Computer Society, 2012. 9

[4] D. M. Chen, G. Baatz, K. Köser, S. S. Tsai, R. Vedantham, T. Pylvänäinen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 737–744, 2011. 1

[5] M. Cummins and P. Newman. Highly scalable appearance-only slam - FAB-MAP 2.0. In *Robotics: Science and Systems*, 2009. 1

[6] Jiankang Deng, J. Guo, and S. Zafeiriou. ArcFace: Additive angular margin loss for deep face recognition. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4685–4694, 2019. 11

[7] A. Gordo, J. Almazan, J. Revaud, and D. Larlus. End-to-end learning of deep visual representations for image retrieval. *IJCV*, 2017. 4

[8] Stephen Hausler, Sourav Garg, Ming Xu, Michael Milford, and Tobias Fischer. Patch-netvlad: Multi-scale fusion of locally-global descriptors for place recognition. In *Proceed-*

ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14141–14152, 2021. 1, 14

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 4

[10] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547, 2021. 3

[11] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128, 2011. 9

[12] Hyo Jin Kim, Enrique Dunn, and Jan-Michael Frahm. Learned contextual feature reweighting for image geo-localization. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3251–3260, 2017. 6, 12

[13] Giorgos Kordopatis-Zilos, Panagiotis Galopoulos, S. Papadopoulos, and Y. Kompatsiaris. Leveraging efficientnet and contrastive learning for accurate global-scale location estimation. *ACM International Conference on Multimedia Retrieval*, 2021. 4, 6

[14] Yunpeng Li, Noah Snavely, Daniel Huttenlocher, and Pascal Fua. Worldwide Pose Estimation using 3D Point Clouds. In *European Conference on Computer Vision*, 2012. 1, 4

[15] Dongfang Liu, Yiming Cui, Liqi Yan, Christos Mousas, Baijian Yang, and Yingjie Chen. DenserNet: Weakly supervised visual localization using multi-scale feature aggregation. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6101–6109, May 2021. 1

[16] Yu A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42:824–836, 2020. 9

[17] Michael Milford and G. Wyeth. Mapping a suburb with a single camera using a biologically inspired slam system. *IEEE Transactions on Robotics*, 24:1038–1053, 2008. 1

[18] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *IEEE International Conference on Computer Vision*, 2017. 10, 11

[19] F. Radenović, G. Tolias, and O. Chum. Fine-tuning CNN Image Retrieval with No Human Annotation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018. 4, 6, 10

[20] A. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Visual Instance Retrieval with Deep Convolutional Networks. *CoRR*, abs/1412.6574, 2015. 4, 6

[21] Jérôme Revaud, Jon Almazán, R. S. Rezende, and César Roberto de Souza. Learning with average precision: Training image retrieval with a listwise loss. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5106–5115, 2019. 4, 10

[22] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 14

[23] Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *ICCV*, pages 1470–1477. IEEE Computer Society, 2003. 9

[24] Giorgos Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of CNN activations. *CoRR*, abs/1511.05879, 2016. 4, 6

[25] A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):257–271, 2018. 1, 4

[26] A. Torii, J. Sivic, M. Okutomi, and T. Pajdla. Visual place recognition with repetitive structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(11):2346–2359, 2015. 1

[27] A. Torii, Hajime Taira, Josef Sivic, M. Pollefeys, M. Okutomi, T. Pajdla, and Torsten Sattler. Are large-scale 3d models really necessary for accurate visual localization? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:814–829, 2021. 1, 4

[28] Frederik Warburg, Soren Hauberg, Manuel Lopez-Antequera, Pau Gargallo, Yubin Kuang, and Javier Civera. Mapillary street-level sequences: A dataset for lifelong place recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2020. 1, 4

[29] Tobias Weyand, A. Araújo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2 – a large-scale benchmark for instance-level recognition and retrieval. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2572–2581, 2020. 4, 10, 11

[30] Shuhei Yokoo, Kohei Ozaki, Edgar Simo-Serra, and S. Iizuka. Two-stage Discriminative Re-ranking for Large-scale Landmark Retrieval. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 4363–4370, 2020. 11

[31] Jun Yu, Chaoyang Zhu, Jian Zhang, Qingming Huang, and Dacheng Tao. Spatial pyramid-enhanced netvlad with weighted triplet loss for place recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 31(2):661–674, 2020. 1

[32] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 4, 10, 11