Method	Extraction latency (ms)	Memory (GB) $\mathcal{R}$Oxf+1M	$\mathcal{R}$Par+1M
(A) Local feature aggregation			
DELF-R-ASMK⋆ [10]	2260	27.6	–
(B) Global features			
R50-GeM [8]	100	7.7	7.7
R101-GeM [8]	175	7.7	7.7
(C) Unified global + local features			
R50-DELG (3 scales global & local) [ours]	118	439.4	440.0
R50-DELG [ours]	244	485.5	486.2
R101-DELG (3 scales global & local) [ours]	193	437.1	437.8
R101-DELG⋆ (3 scales global & local) [ours]	193	21.1	21.1
R101-DELG [ours]	416	485.9	486.6
R101-DELG⋆ [ours]	416	22.6	22.7
Local features			
DELF (3 scales) [5]	98	434.2	434.8
DELF (7 scales) [5]	201	477.9	478.5

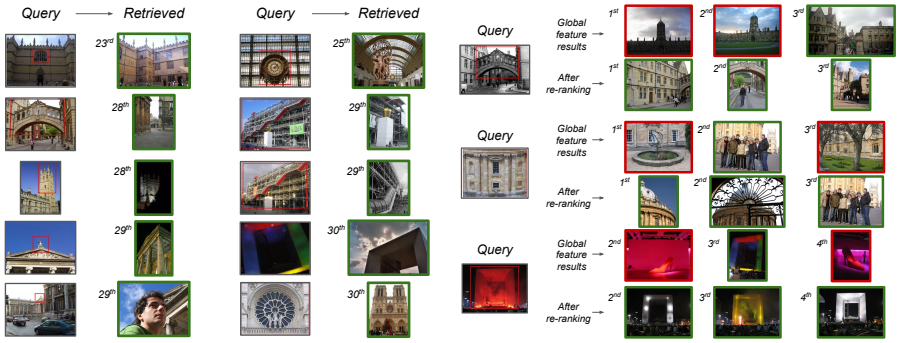
Table 1: Feature extraction **latency** and database **memory** requirements for different image retrieval models. Latency is measured on an NVIDIA Tesla P100 GPU, for square images of side 1024. (A) DELF-R-ASMK⋆ measurements use the code and default configuration from [10]; (B) ResNet-GeM variants use 3 image scales; (C) DELG and DELG⋆ are compared with different configurations. As a reference, we also provide numbers for DELF in the last rows.

Appendix A. Additional experiment results

Latency and memory. Tab. 1 reports feature extraction latency and index memory footprint for state-of-the-art methods; as a reference, we also present numbers for DELF (which uses an R50 backbone). Joint extraction with DELG allows for substantial speed-up, compared to running two separate local and global models: when using 3 local feature scales, separately running R50-GeM and DELF would lead to 198ms, while the unified model runs with latency of 118ms (40% faster). For the R50 case with 7 local scales, the unified model is 19% faster. The binarization technique adds negligible overhead, having roughly the same latency.

Storing unquantized DELG local features requires excessive index memory requirements; using binarization, this can be reduced significantly: R101-DELG⋆ requires 23GB. This is lower than the memory footprint of DELF-R-ASMK⋆. Note also that feature extraction for DELG⋆ is much faster than for DELF-R-ASMK⋆, by more than 5×. R50-DELG with 3 scales is also faster than using a heavier global feature (R101-GeM [8]), besides being more accurate. As a matter of fact, several of the recently-proposed global features [8,2] use image pyramids with 3 scales; our results indicate that their performance can be improved substantially by adding a local feature head, with small increase in extraction latency, and without degrading the global feature.

Qualitative results. We give examples of retrieval results, to showcase the DELG model. Fig. 1a illustrates difficult cases, where the database image shows a very different viewpoint, or significant lighting changes; these images can still achieve relatively high ranks due to effective global features, which capture well the similarity even in such challenging scenarios. In these cases, local features do not produce sufficient matches.



(a) **Global feature retrieval: high recall.** (b) **Local feature re-ranking: high precision.**

Fig. 1: Sample DELG results on $\mathcal{R}$Oxf-Hard and $\mathcal{R}$Par-Hard. **(a)** Examples of difficult, high-ranked relevant retrieved images for 10 different queries. These retrieved database images have a low number of inliers after geometric verification (if any), which means that their similarity is mainly captured by the global feature. **(b)** Examples illustrating performance improvements from the re-ranking stage using local features. For each query (left), two rows are presented on the right, the top one showing results based on global feature similarity and the bottom one showing results after re-ranking the top 100 images with local features. Correct results are marked with green borders, and incorrect ones in red. While top retrieved global feature results are often ranked incorrectly, local feature matching can effectively re-rank them to improve precision.

Fig. 1b shows the effect of local feature re-ranking for selected queries, for which substantial gains are obtained. Global features tend to retrieve images that have generally similar appearance, but which sometimes do not depict the same object of interest; this can be improved substantially with local feature re-ranking, which enables stricter matching selectivity. As can be observed in these two figures, global features are crucial for high recall, while local features are key to high precision.

Appendix B. Training cost

One of the advantages of DELG is that local and global features can be jointly trained in one shot, without the need of additional steps. In practice, we have observed that the training time for DELG is roughly the same as for the associated global feature, both taking approximately 1.5 day.

This is because the additional cost of learning the attention and autoencoder layers is small: there are only 5 extra trainable layers (2 in the attention module, 2 in the autoencoder, plus the attention loss classifier), and their gradients are not backpropagated to the network backbone. Other factors play a much more significant role in the training speed, e.g.: reading data from disk, transferring batch to GPU memory, applying image pre-processing operations such as resizing/cropping/augmentation, etc.

In short, training our local image features comes at a very small cost on top of global feature learning. Let us clarify, though, that while the training cost is roughly the same between the global and joint models, the inference cost for the joint model has an

overhead for local feature detection (e.g., selecting the top local features across different scales in the image pyramid).

We can also compare DELG's training cost to DELF's: DELF would require one additional run for attention learning (6 hours), followed by a PCA computation step (3 hours). The advantage of DELG, compared to DELF, is that the attention and autoencoder layers are already adapting to the network backbone while it is training; for DELF, this process only happens after the backbone is fully trained.

Appendix C. Model selection and tuning

We provide more detail on how our models were selected/tuned, and specify chosen parameters which were not mentioned in the main text. For more information, please refer to our released code/models.

Model selection. Our models are trained for 1.5M steps, corresponding to approximately 25 epochs of an 80% split of the GLD training set. We attempt three different initial learning rates for each configuration, and select the best one based on the performance on $\mathcal{R}$Oxf/$\mathcal{R}$Par (it is a convention in recent work [9,10,4] to perform ablations/tuning on these datasets). In all cases, we pick the model at the end of training and do not hand-pick earlier checkpoints which may have higher performance. These selected models are then used in all large-scale experiments, on $\mathcal{R}$Oxf+1M, $\mathcal{R}$Par+1M, GLDv2-retrieval and GLDv2-recognition.

Tuning image matching. For local feature-based matching with DELG, we used a simple distance criterion for proposing putative feature correspondences, before feeding them to RANSAC. RANSAC is used with 1k iterations, and any returned match is used for re-ranking (minimum number of inliers is zero). We tuned two thresholds: local descriptor distance threshold and the RANSAC residual threshold. These thresholds are tuned on $\mathcal{R}$Oxf/$\mathcal{R}$Par, and then fixed for $\mathcal{R}$Oxf+1M and $\mathcal{R}$Par+1M; similarly, they are tuned on the validation set of GLDv2-retrieval/GLDv2-recognition, then fixed for experiments on the testing set of GLDv2-retrieval/GLDv2-recognition. For DELG on $\mathcal{R}$Oxf/$\mathcal{R}$Par/GLDv2-retrieval, the local descriptor distance threshold is set to 1.0, and the RANSAC residual threshold is set to 20.0; for DELG*, we adjust the former threshold to 1.1 due to quantization errors. For DELG on GLDv2-recognition, the local descriptor distance threshold is set to 0.9, and the RANSAC residual threshold is set to 10.0 (since the recognition task has a stronger focus on precision, we find that these tighter matching parameters perform better).

Appendix D. Memory footprint

For reporting the memory footprint of DELG/DELG*, we follow a similar convention to previous work [7,10] and report total storage required for local and global descriptors. Note that local feature geometry information is not counted in the reported numbers, and would add some overhead.

Our main focus in this paper is to propose a new model for unified local and global feature extraction, so we did not thoroughly explore techniques for efficient quantization. As the DELG* results show, there is great promise in aggressively quantizing local

descriptors, leading to reasonable storage requirements – which can likely be improved with more effective quantizers. Similarly, global descriptors and local feature geometry could be quantized to improve the total memory cost substantially.

Appendix E. Local feature matching visualizations

We present more qualitative results, to illustrate local feature matching with DELG: Fig. 2 presents examples of correct matches, and Fig. 3 presents examples of incorrect matches. For these examples, we use the R50-DELG model and the $\mathcal{R}$Oxf/$\mathcal{R}$Par datasets. These visualizations depict the final obtained correspondences, post-RANSAC.

For each row, one query and two index images are shown, and on the right their local feature matches are shown, with lines connecting the corresponding keypoints. Fig. 2 showcases DELG's robustness against strong viewpoint and illumination changes: for example, matches can be obtained across different scales and day-vs-night cases. Fig. 3 presents overtriggering cases, where a match is found even though different objects/scenes are presented: these tend to occur for similar patterns between query and index images (ie, similar windows, arches or roofs) which appear in similar geometric configurations. Generally, these do not affect retrieval results much because the number of inliers is low.

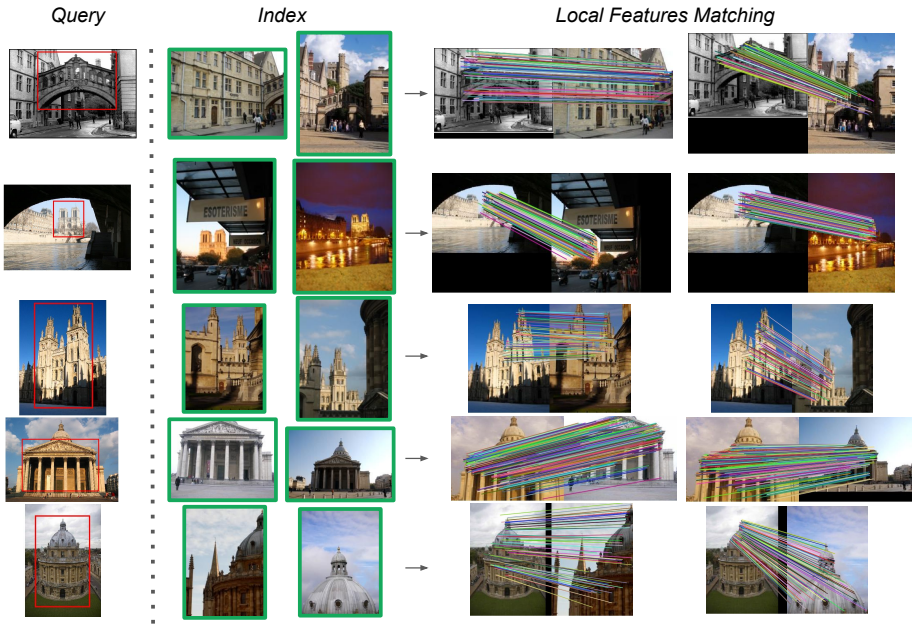


Fig. 2: Examples of correct local feature matches, for image pairs depicting the same object/scene.

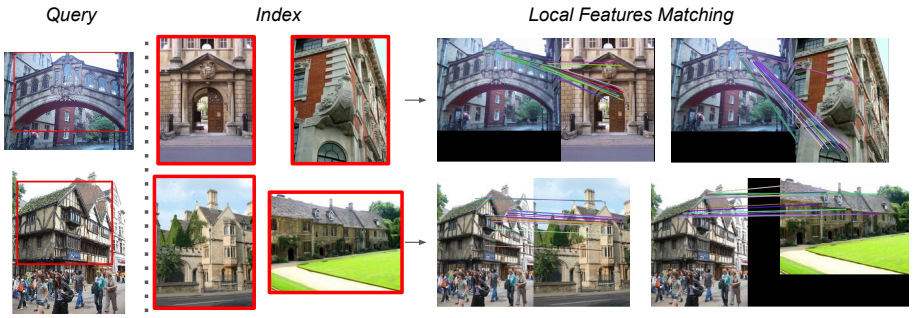


Fig. 3: Examples of incorrect local feature matches, for image pairs depicting different objects/scenes.

Appendix F. Feature visualization

We provide visualizations of the features learned by the DELG model. This is useful to understand the hierarchical representation which we rely on for extraction of local and global features. We explore two types of visualizations, based on dataset examples with high activations and by optimizing input images with respect to a given layer/channel.

F.1 Feature visualization by dataset examples

For this experiment, we run our R50-DELG model over 200k images from the Google Landmarks dataset [5], and collect the images and feature positions with largest value in each channel of several activation maps. We specifically consider the activation maps at the outputs of the *conv2*, *conv3*, *conv4* and *conv5* blocks of layers in our model's ResNet architecture [3]. The feature positions with maximum activations can be mapped back to the relevant input image regions by computing the model's receptive field parameters [1]. Note that the region may partly fall outside the image, in which case we apply zero-padding for the visualization.

Fig. 4 presents image patches that produce the highest activations for selected channels of the above-mentioned layers. The activation values are noted in each subfigure's title and can be read by zooming in. For each selected channel of a specific layer, the 9 patches with largest activations are shown. The receptive field sizes (both horizontally and vertically) for each of these layers are: 35 (*conv2*), 99 (*conv3*), 291 (*conv4*) and 483 (*conv5*); these correspond to the sizes of image patches. One can notice that the types of patterns which maximally activate specific layers grow in complexity with network depth. This agrees with observations from previous work, where the hierarchical nature of CNN features is discussed [11,6]. Shallower layers such as *conv2* tend to focus on edges and simple textures; *conv3* responds highly to more complex shapes, such as edges resembling palm trees (channel 15) and arches (channel 48); *conv4* focuses on object parts, such as green dome-like shapes (channel 2), or arches (channel 30); *conv5* shows strong activations for entire objects, with substantial invariances to viewpoint and lighting: entire buildings (channel 3), islands (channel 23) or towers (channel 52) are captured.

These visualizations help with intuitive understanding of our proposed method, which composes local features from a shallower layer (*conv4*) and global features from a deeper layer (*conv5*). Features from *conv5* present high degree of viewpoint invariance, being suboptimal for localized matching and more suitable to global representations. In contrast, *conv4* features seem more grounded to localizable object parts and thus can be effectively used as local feature representations.

F.2 Feature visualization by optimization

In this experiment, we consider the same layers and channels as above, but now adopt a visualization technique by optimizing the input image to maximally activate the desired feature. First, the input image is initialized with random noise. Given the desired layer/channel, we backpropagate gradients in order to maximally activate it. Regularizers can be useful to restrict the optimization space, otherwise the network may find ways to activate neurons that don't occur in natural images.

We reuse the technique from Olah et al.[6], with default parameters, and the results are presented in Fig. 5. Again, we notice that a hierarchical representation structure forms, with more complex patterns being produced as the network goes deeper from *conv2* to *conv5*. As expected, the produced images agree very well with the patches from Fig. 4, in terms of the types of visual contents. For example, for *conv3* channel 48, the optimized image shows arch-like edges while the dataset examples present image patches where those types of patterns occur.

Note also how deeper layers tend to specialize for the target task, by detecting image patches with parts and texture that are common to landmarks. For example, *conv4* shows detection of green dome-like shapes (channel 2) and arches (channel 30), which are common in these types of objects; *conv5*, on the other hand, shows building walls (channel 3) and rocky patterns that are common in ancient buildings or islands (channel 23).

References

1. Araujo, A., Norris, W., Sim, J.: Computing Receptive Fields of Convolutional Neural Networks. Distill (2019), https://distill.pub/2019/computing-receptive-fields
2. Gordo, A., Almazan, J., Revaud, J., Larlus, D.: End-to-end Learning of Deep Visual Representations for Image Retrieval. IJCV (2017)
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Proc. CVPR (2016)
4. Ng, T., Balntas, V., Tian, Y., Mikolajczyk, K.: SOLAR: Second-Order Loss and Attention for Image Retrieval. In: Proc. ECCV (2020)
5. Noh, H., Araujo, A., Sim, J., Weyand, T., Han, B.: Large-Scale Image Retrieval with Attentive Deep Local Features. In: Proc. ICCV (2017)
6. Olah, C., Mordvintsev, A., Schubert, L.: Feature Visualization. Distill (2017), https://distill.pub/2017/feature-visualization
7. Radenović, F., Iscen, A., Tolias, G., Avrithis, Y., Chum, O.: Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking. In: Proc. CVPR (2018)

conv2, channel 0 | *conv2*, channel 12 | *conv2*, channel 49

conv3, channel 15 | *conv3*, channel 29 | *conv3*, channel 48

conv4, channel 1 | *conv4*, channel 2 | *conv4*, channel 30

conv5, channel 3 | *conv5*, channel 23 | *conv5*, channel 52

Fig. 4: Visualization of patterns detected by specific feature maps / channels, by presenting image patches that produce high activations.

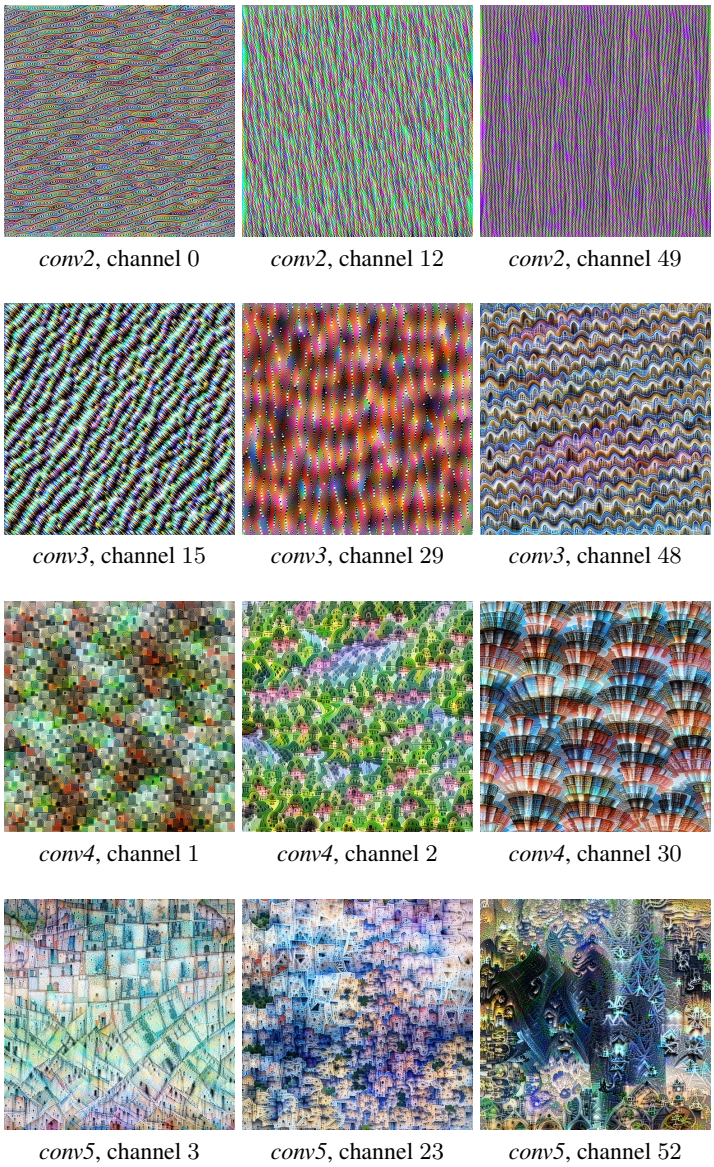


Fig. 5: Visualization of patterns detected by specific feature maps / channels, by optimizing the input image to produce high activations.

8. Radenović, F., Tolias, G., Chum, O.: Fine-tuning CNN Image Retrieval with No Human Annotation. IEEE Transactions on Pattern Analysis and Machine Intelligence (2018)
9. Revaud, J., Almazan, J., de Rezende, R.S., de Souza, C.R.: Learning with Average Precision: Training Image Retrieval with a Listwise Loss. In: Proc. ICCV (2019)
10. Teichmann, M., Araujo, A., Zhu, M., Sim, J.: Detect-to-Retrieve: Efficient Regional Aggregation for Image Search. In: Proc. CVPR (2019)
11. Zeiler, M.D., Fergus, R.: Visualizing and Understanding Convolutional Networks. In: Proc. ECCV (2014)