Global features: CNN Image Retrieval

This Python toolbox implements the training and testing of the approach described in the papers:

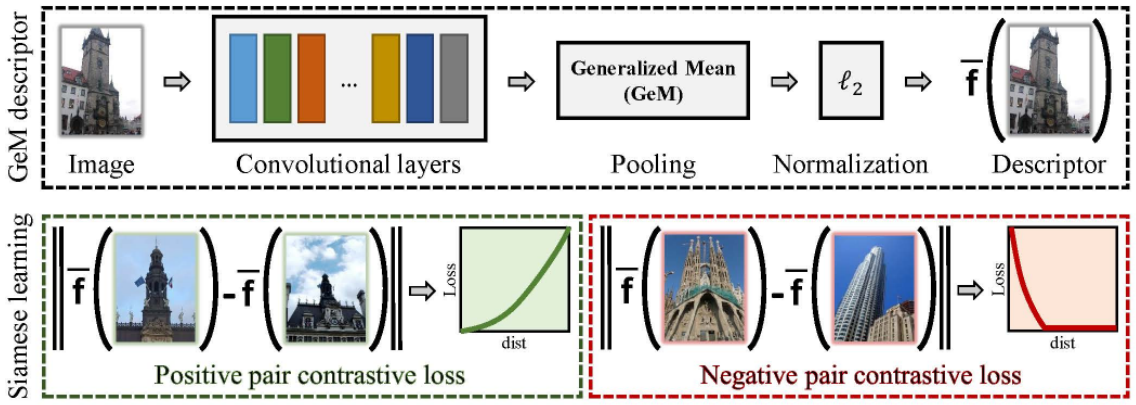
paper arXiv.2001.05027

```
"Fine-tuning CNN Image Retrieval with No Human Annotation",
Radenović F., Tolias G., Chum O.,
TPAMI 2018
```

paper arXiv.2001.05027

```
"CNN Image Retrieval Learns from BoW: Unsupervised Fine-Tuning with Hard Examples",
Radenović F., Tolias G., Chum O.,
ECCV 2016
```

Fine-tuned CNNs are used for global feature extraction with the goal of using those for image retrieval. The networks are trained on the *SfM120k* landmark images dataset.



When initializing the network, one of the popular pre-trained architectures for classification tasks (such as ResNet or VGG) is used as the network's backbone. The fully connected layers of such architectures are discarded, resulting in a fully convolutional backbone. Then, given an input image of the size [W × H × C], where C is the number of channels, W and H are image width and height, respectively; the output is a tensor X with dimensions [W' × H' × K], where K is the number of feature maps in the last layer. Tensor X can be considered as a set of the input image's deep local features. For deep convolutional features, the simple aggregation approach based on global pooling arguably provides the best results. This method is fast, has a small number of parameters, and a low risk of overfitting. Keeping this in mind, we convert local features to a global descriptor vector using one of the retrieval system's global poolings (MAC, SPoC, or GeM). After this stage, the feature vector is made up of the maximum activation per feature map with dimensionality equal to K. The final output dimensionality for the most common networks varies from 512 to 2048, making this image representation relatively compact.

Vectors that have been pooled are subsequently L2-normalized. The obtained representation is then optionally passed through the fully connected layers before being subjected to a new L2 re-normalization. The finally produced image representation allows comparing the resemblance of two images by simply using their inner product.

Install DELF library

To be able to use this code, please follow these instructions to properly install the DELF library.

Usage

▶ **Training**

▶ **Training logic flow**

Exporting the Trained Model

Assuming the training output, the TensorFlow checkpoint, is located in the `--directory` path. The following code exports the model:

```
python3 model/export_CNN_global_model.py \
        [--ckpt_path PATH] [--export_path PATH] [--input_scales_list LIST]
        [--multi_scale_pool_type TYPE] [--normalize_global_descriptor BOOL]
        [arch ARCHITECTURE] [pool POOLING] [whitening BOOL]
```

NOTE: Path to the checkpoint must include .h5 file.

Testing the trained model

After the trained model has been exported, it can be used to extract global features similarly as for the DELG model. Please follow these instructions.

After training the standard training setup for 100 epochs, the following results are obtained on Roxford and RParis datasets under a single -scale evaluation:

```
>> roxford5k: mAP E: 74.88, M: 58.28, H: 30.4
>> roxford5k: mP@k[1, 5, 10] E: [89.71 84.8  79.07],
                             M: [91.43 84.67 78.24],
                             H: [68.57 53.29 43.29]

>> rparis6k: mAP E: 89.21, M: 73.69, H: 49.1
>> rparis6k: mP@k[1, 5, 10] E: [98.57 97.43 95.57],
                            M: [98.57 99.14 98.14],
                            H: [94.29 90.   87.29]
```