Deep Aggregation of Regional Convolutional Activations for Content Based Image Retrieval

Konstantin Schall, Kai Uwe Barthel, Nico Hezel, and Klaus Jung

Visual Computing Group, HTW Berlin

visual-computing.com
Challenges in Image Retrieval

1. How to generate feature vectors?

\[ f(x_i) \quad \text{and} \quad f(x_j) \]
Activation Maps by CNNs

- Layers of Convolutional Neural Networks produce a set of activation maps $\mathcal{X}^l = \{\mathcal{X}_i^l | i = 1 \ldots C^l\}$, with $C^l$ being the number of filters in layer $l$.

- Number of filters $C$ is increasing while the width $W$ and the height $H$ are decreasing in deeper layers.
1. Extraction of activation maps $\mathcal{X}^l$ where $l$ is the last convolutional layer in a CNN

2. Compute the average of $C$ activation maps $A \in \mathcal{X}^l$ with $A(w, h)$ being a single activation

$$A(w, h)$$

One value of $C$-dimensional feature vector
1. Extraction of activation maps $\mathcal{X}^l$ where $l$ is the last convolutional layer in a CNN

2. Compute the average of $C$ activation maps $A \in \mathcal{X}^l$ with $A(w, h)$ being a single activation

Spatial information is completely lost
Regional Max Pooling

RMAC: Regional Maximal Activations of Convolutions (Tolias et al., 2016)

1. Extraction of activation maps $\mathcal{X}^l$

2. Search for maximal activation values in a set of 20 regions $R^A = \{R_i^A | i = 1 \ldots 20\}$, for each $A \in \mathcal{X}^l$
Regional Max Pooling

RMAC: Regional Maximal Activations of Convolutions (Tolias et al., 2016)

1. Extraction of activation maps $\mathcal{X}^l$

2. Search for maximal activation values in a set of 20 regions $R^A = \{R^A_i|i = 1 \ldots 20\}$, for each $A \in \mathcal{X}^l$
Regional Max Pooling

RMAC: Regional Maximal Activations of Convolutions (Tolias et al., 2016)

1. Extraction of activation maps $\mathcal{X}^l$

2. Search for maximal activation values in a set of 20 regions $R^A = \{R^A_i | i = 1 ... 20\}$, for each $A \in \mathcal{X}^l$
Challenges in Image Retrieval

1. How to generate feature vectors?
2. How to optimize model weights for the specific task of retrieval?
Challenges in Image Retrieval

1. How to generate feature vectors?

2. How to optimize model weights for the specific task of retrieval?

2D embeddings for MNIST trained with

*Softmax Cross Entropy*

2D embeddings for MNIST trained with

*Triplet Loss*
Finetuning for Retrieval

• DIR: Deep Image Retrieval (Gordo et al., 2017)
• End-to-End image retrieval system.
• Finetune a ResNet101 with Triplet loss
• Total training time: approximately 168 hours

Input image with 800 pixels at the smaller side

$X^L$ of size $(25 \times 25 \times 2048)$ Tensor of size $(20 \times 2048)$ Tensor of size $(20 \times 2048)$ Feature Vector of size 2048
1. Why only regional max pooling?
2. Are all regions equally important?
3. Is Triplet loss the ideal loss function?
4. Is an increased image size necessary during training?
Average vs. Max Pooling

- ResNet50 pretrained on ImageNet
- 6 differently pooled FV’s
- Larger image size improves retrieval (in most cases)
21 Regions

• Make use of average and max pooling
• In total we use 21 regions (global is added):
RAMAC: Regional Average and Maximal Activations of Convolutions

1. Extraction of activation maps $\mathcal{X}^l$

2. Search for max. and avg. activation values in a set of 21 regions $R^A = \{R_i^A|i = 1 \ldots 21\}$, for each $A \in \mathcal{X}^l$
RAMAC: Regional Average and Maximal Activations of Convolutions

1. Extraction of activation maps $\mathcal{X}^l$

2. Search for max. and avg. activation values in a set of 21 regions $R^A = \{R^A_i| i = 1 \ldots 21\}$, for each $A \in \mathcal{X}^l$
Aggregation of Pooled Values

- DARAC: Deep Aggregation of Regional Activations of Convolutions

Input Image (299x299x3) → Conv5_3 10x10x2048 → Feature Maps 42x1x2048 → Feature Maps 16x1x2048 → Feature Vector 2048D

- pre-trained ResNet50 version 2 → Regional Avg+Max Pooling → Conv 16x42x1x1 → Conv 1x16x1x1

Feature Map Aggregation
NRA Loss Function

\[ J = -\frac{1}{m} \sum_{i=1}^{m} \left( \log(s_{i,\text{max}}^+) + \log(1 - s_{i,\text{min}}^-) \right) \]

Approximates nonlinear ranks to contract similar and disperse dissimilar images
Proposed System

- ResNet50 v2 pretrained on ImageNet
- Training with Google Landmarks dataset
- DARAC aggregation and train with NRA Loss
- Total training time: approximately 24 hours

Input image with 299 pixels at both sides

\[ \mathbf{x}^L \text{ of size } (10 \times 10 \times 2048) \]

Tensor of size \( (42 \times 2048) \)

Feature Vector of size 2048

After training
Multi Resolution

Input image size = 299

Input image size = 540

Input image size = 1020

Vector of size 2048
### Evaluation

**Mean Average Precision scores for the proposed steps**

GL = Google Landmarks. CL = Cleaned Landmarks. W = PCA Whitening

<table>
<thead>
<tr>
<th>FV</th>
<th>Training Set</th>
<th>Image Size</th>
<th>Oxford</th>
<th>Paris</th>
<th>Holidays</th>
</tr>
</thead>
<tbody>
<tr>
<td>Glob. Avg.</td>
<td>ImageNet</td>
<td>299</td>
<td>48.7</td>
<td>68.9</td>
<td>86.1</td>
</tr>
<tr>
<td>Glob. Avg.</td>
<td>GL</td>
<td>299</td>
<td>75.3</td>
<td>87.6</td>
<td>91.6</td>
</tr>
<tr>
<td>DARAC</td>
<td>GL</td>
<td>299</td>
<td>77.6</td>
<td>89.4</td>
<td>92.8</td>
</tr>
<tr>
<td>DARAC + W</td>
<td>GL</td>
<td>299</td>
<td>81.4</td>
<td>90.8</td>
<td>93.7</td>
</tr>
<tr>
<td>DARAC + W</td>
<td>GL</td>
<td>540</td>
<td>82.2</td>
<td>91.9</td>
<td>95.5</td>
</tr>
<tr>
<td>DARAC + W</td>
<td>GL</td>
<td>MR</td>
<td>83.4</td>
<td>93.0</td>
<td><strong>96.9</strong></td>
</tr>
<tr>
<td>DARAC + W</td>
<td>CL</td>
<td>MR</td>
<td><strong>88.2</strong></td>
<td><strong>94.1</strong></td>
<td>95.5</td>
</tr>
</tbody>
</table>
## Comparison to Similar systems

*Mean Average Precision* scores for the proposed steps

GL = Google Landmarks. CL = Cleaned Landmarks.

<table>
<thead>
<tr>
<th>System</th>
<th>Matching</th>
<th>MR</th>
<th>Oxford</th>
<th>Paris</th>
<th>Holidays</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELF (GL)</td>
<td>local</td>
<td>7 (0.25 – 2)</td>
<td>83.8</td>
<td>85.0</td>
<td>-</td>
</tr>
<tr>
<td>GeM</td>
<td>global</td>
<td>5 (0.25 – 1)</td>
<td>87.8</td>
<td>92.7</td>
<td>93.9</td>
</tr>
<tr>
<td>DIR (CL)</td>
<td>global</td>
<td>550, 800, 1050</td>
<td>86.1</td>
<td>94.5</td>
<td>94.8</td>
</tr>
<tr>
<td>Ours (GL)</td>
<td>global</td>
<td>299, 540, 1020</td>
<td>83.4</td>
<td>93.0</td>
<td>96.9</td>
</tr>
<tr>
<td>Ours (CL)</td>
<td>global</td>
<td>299, 540, 1020</td>
<td>88.2</td>
<td>94.1</td>
<td>95.5</td>
</tr>
</tbody>
</table>
Comparison to Similar systems

**Mean Average Precision** scores for the proposed steps.

GL = Google Landmarks. CL = Cleaned Landmarks.

<table>
<thead>
<tr>
<th>System</th>
<th>Matching</th>
<th>MR</th>
<th>Oxford</th>
<th>Paris</th>
<th>Holidays</th>
</tr>
</thead>
<tbody>
<tr>
<td>DELF (GL)</td>
<td>local</td>
<td>7 (0.25 – 2)</td>
<td>83.8</td>
<td>85.0</td>
<td>-</td>
</tr>
<tr>
<td>GeM</td>
<td>global</td>
<td>5 (0.25 – 1)</td>
<td>87.8</td>
<td>92.7</td>
<td>93.9</td>
</tr>
<tr>
<td>DIR (CL)</td>
<td>global</td>
<td>550, 800, 1050</td>
<td>86.1</td>
<td><strong>94.5</strong></td>
<td>94.8</td>
</tr>
<tr>
<td>Ours (GL)</td>
<td>global</td>
<td>299, 540, 1020</td>
<td>83.4</td>
<td>93.0</td>
<td><strong>96.9</strong></td>
</tr>
<tr>
<td>Ours (CL)</td>
<td>global</td>
<td>299, 540, 1020</td>
<td><strong>88.2</strong></td>
<td>94.1</td>
<td>95.5</td>
</tr>
</tbody>
</table>

- Fastest inference and training
- Lowest memory footprint
- Very good retrieval quality
Thank you very much!

More Information at

www.visual-computing.com