SZTAKI @ ImageCLEF 2011

Bálint Daróczy

joint work with

András Benczúr, Róbert Pethes

Data Mining and Web Search Group

Computer and Automation Research Institute

Hungarian Academy of Sciences
Basic kernel extraction

Training/test images
- JPEG files
- Exif
- Flickr Tags

Low-level feature extraction
- Harris-Laplace corner detection
- scale-pyramid sampling
- SIFT and RGB Color descriptor
- dimension reduction with PCA

Gaussian Mixture Modeling
- codebook size: 256 clusters
- about 3 million training samples
- non-hierarchical codebook expansion

Flickr Tags
- ImageID1: bicycle, race, arena
- ImageID2: animal, little, lion
- Tag as probability distribution

Fisher vectors
- Normalization with Fisher information
- L2 and power normalization
- Fisher vector generation on 9 poolings per modality:
  - full image pooling
  - Harris-Laplacian detected points
  - 1x3 and 2x2 spatial pooling

Basic visual kernels
- Similarity matrix between images and the training set
- Manhattan distance
- averaged kernel for spatial poolings
- 4 kernels per modality:
  - Harris-Laplacian
  - full image
  - 1x3 spatial pooling
  - 2x2 spatial pooling

Basic textural kernel
- Jensen-Shannon divergence between the probability distributions of each image and the training set
- symmetric similarity matrix

Basic Kernels
Learning method

Basic visual kernels
- Similarity matrix between images and the training set
- Manhattan distance
- Averaged kernel for spatial poolings
- 4 kernels per modality:
  - Harris-Laplacian
  - Full image
  - 1x3 spatial pooling
  - 2x2 spatial pooling

Basic textual kernel
- Jensen-Shannon divergence between the probability distributions of each image and the training set
- Symmetric similarity matrix

Kernel aggregation
Independently for each category

Kernel weight determination
- Learning one-versus-all SVM models on the training set with cross-validation
- Search optimal weights over the predictions for each category
- Cut down high weights to prevent overfitting

SVM classifier
Binary classifier for each category

Predictions
Low level image descriptors

Sampling strategies:

- Dense pyramid
- Harris-Laplacian

Patch descriptors:

- Scale Invariant Feature Transform (SIFT) [1]
- RGB Colour moments (why not HSV?)
- Histogram of Oriented Gradients (HOG)
- Locale Binary Pattern (LBP)
Sampling strategies

**Dense pyramid:**

3 scales
(scale factor=1.6)
„global” type problems
sampling at 4 pixels on every scale
→ ~12-13k descriptors per image

**Harris-Laplacian:**

prevent homogeneous areas
„object” type problems
→ ~1.5-2k descriptors per image
Gaussian Mixture Modeling
Based on standard expectation maximization (EM)

- not hierarchical
- transformation of feature space → depends on the original space
- how to prevent zero variance?

---

**Algorithm 1** The GPU GMM algorithm

**Input:** data points \( \{x_i\}_{i=1}^N \), dimension \( D \), mixture number \( K \).

**Output:** \( \{P_i\}_{i=1}^K \), \( \{\mu_i\}_{i=1}^K \), \( \{\sigma_i\}_{i=1}^K \), where \( \mathcal{N}_i \) is normally distributed with parameters \( (\mu_i, \sigma_i) \) and \( \sigma_i \) is assumed to be diagonal.

1. Initialize the Gaussian distributions with random parameters
2. repeat
   1. for all \( n \) and \( k \) do
      1. Expectation: Compute likelihood \( p_{nk} = \frac{f_k(x_n)P_k}{\sum_{i=1}^K f_i(x_n)P_i} \) where \( f_i \) is the density of \( \mathcal{N}_i \)
   2. for all \( k \) and \( d \) do
      1. Maximization: compute \( P_k = \frac{\sum_{n=1}^N p_{nk}}{N} \), \( \mu_{kd} = \frac{\sum_{n=1}^N p_{nk}x_{nd}}{\sum_{n=1}^N p_{nk}} \) and \( \sigma_{kd}^2 = \frac{\sum_{n=1}^N p_{nk}(x_{nd}-\mu_{kd})^2}{\sum_{n=1}^N p_{nk}} \)
   3. until until converge
Implementation of GMM on GPGPU

Numerical bottleneck:
• Naive GMM implementation frequently underflows when computing density $\sigma_{kd}^2$
• Results in division by zero in next iteration BUT can be fully solved by using logarithms and “buckets”:

\[
\log(p_{nk}) = \log(f_k(x_n)) + \log(P_k) - \log \sum_{i=1}^{K} \exp(\log(f_i(x_n)) + \log(P_i))
\]

\[
\log(P_k) = \log \sum_{n=1}^{N} \exp \log(p_{nk}) - \log N
\]

\[
\mu_{kd} = \frac{\sum_{n=1}^{N} \exp(\log(p_{nk}) + \log(x_{nd}))}{\sum_{n=1}^{N} \exp \log(p_{nk})}
\]

\[
\sigma_{kd}^2 = \frac{\sum_{n=1}^{N} \exp(\log(p_{nk}) + 2 \log(x_{nd} - \mu_{kd}))}{\sum_{n=1}^{N} \exp \log(p_{nk})}
\]

For details see our freely downloadable code for CUDA and x86 CPU-s with test examples: http://datamining.sztaki.hu/?q=en/GPU-GMM
Fisher calculation on GPGPU

• Highly parallel algorithm [2]
• With K=256 and the original dimension is D=96 the number of independent calculations is \( D_{\text{fisher}} = K \times D \times 2 = 49152 \)
• Under reasonable conditions, time depends only on #low level features
• Sparsity? We applied the same methods to calculate \( p_{nk} \) as in our GMM algorithm → Fisher vectors are not sparse!
• Fisher, L2 and power normalization can be integrated efficiently into the Fisher calculation algorithm
• Our CUDA implementation is 4x faster as a good implementation on a single CPU core and 16x faster when the CPU is computing the same algorithm (\( K=256, N=10k, D=96 \))

Note: if we apply the faster, approximated CPU implementation, the Fisher vectors are sparser
• the advantage is higher with more Gaussians, more type of poolings or higher number of feature vectors

For details see our freely downloadable code for CUDA and x86 CPU-s with test examples: http://datamining.sztaki.hu/?q=en/GPU-GMM
Poolings and visual kernel calculation

We extracted Fisher vectors on four different poolings:
- all of the extracted features (full)
- only the features detected by the Harris-Laplacian
- spatial pyramids: 1x3 and 2x2 [3]

Dimensional problems:
- even the Fisher vector calculated over the lowest number of feature vectors (Harris-Laplacian) is not sparse
- the final dimension of extracted Fisher vectors is 9xK*2*D per image
- how to weight differently the higher dimensional spatial pyramid vectors and the full and the Harris-Laplacian vectors?

→ use pre-computed normalized kernels!
Calculate for each pooling $p$ $K_p(F_{i,p}, F_{j,p})$ where $F_{i,p}, F_{j,p}$ is the Fisher vectors extracted from the i-th image of the full dataset and the j-th image of the training set
- if we adopt kernels instead of Fisher vectors the dimension of the high level descriptor of an image will be lower (8k vs. 40K/49k.)
We can combine Fisher vectors with different modalities and visual methods using distance matrices as kernels.

Our preliminary results on the Pascal VOC 2007 dataset\[4\] indicated to use Manhattan distance for normalized Fisher vectors:

$$K_{k_{\text{visual}}}(X, I_t) = \frac{\text{dist}_{\text{Manhattan}}(F_k(X), F_k(I_t))}{\max_x \arg \max_t K_k(X, I_t)}$$

Preliminary results: The L1-kernel outperforms both the Fisher kernel and the Super-Vector[5]

<table>
<thead>
<tr>
<th>RGB color descriptors, dense sampling</th>
<th>Normalized Fisher vector #Gaussians=256</th>
<th>Super-Vector, K-means, s=0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dot product</td>
<td>Pre-computed kernel</td>
<td>K=256</td>
</tr>
<tr>
<td>Dimension</td>
<td>49152</td>
<td>8000</td>
</tr>
<tr>
<td>AvgMAP</td>
<td>0.4244</td>
<td>0.4501</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Curse of sampling
(pre-liminary results)

To select the optimal number of Gaussians we calculated Gaussian Mixture models with K=128/256/512/1024. We tested the resulted GMMs with Harris-Laplacian detected and dense sampled features on the Pascal VOC 2007 dataset:

<table>
<thead>
<tr>
<th>RGB color descriptor</th>
<th>K=128</th>
<th>K=256</th>
<th>K=512</th>
<th>K=1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harris-Laplacian</td>
<td>0.4019</td>
<td>0.4057</td>
<td>0.4070</td>
<td></td>
</tr>
<tr>
<td>Dense sampling</td>
<td>0.4377</td>
<td>0.4501</td>
<td>0.45803</td>
<td>0.4677</td>
</tr>
</tbody>
</table>

The results of Harris-Laplacian pooling are more disappointing if we calculate Fisher vectors for SIFT features:

<table>
<thead>
<tr>
<th>Grayscale SIFT</th>
<th>K=128</th>
<th>K=256</th>
<th>K=512</th>
<th>K=1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harris-Laplacian</td>
<td>0.4327</td>
<td>0.443</td>
<td>0.4295</td>
<td>0.4157</td>
</tr>
<tr>
<td>Dense sampling</td>
<td>0.4584</td>
<td>0.4669</td>
<td>0.4579</td>
<td></td>
</tr>
</tbody>
</table>
Textual kernel and combination

Textual kernel
To use the provided Flickr tags we thought of the tags of an image as probability distributions. To select the optimal similarity measure (kernel) for tag based methods we splitted the training set into two parts and trained linear SVM models with the libSVM package[6]. The best performing textual kernel:

\[ K_{k_{\text{textual}}}(X, I_t) = \text{dist}_{\text{Jensen-Shannon}}(X, I_t) \]

Combination
This splitting gave us the opportunity to establish a group of visual and textual kernels and pre-define weights over them. To extract weights we calculated the output scores of the basic kernel based svm classifiers → to prevent overfitting the resulted weights should be in [-11,11].

To treat all the kernels equally we calculated pre-computed kernels as distance matrices and transformed them to meet the following requirements:
- symmetric if possible
- normalized with the maximum
- \(K(X,X)=0\)

The final combined kernel:

\[ K(X, Y)_c = \frac{1}{|K|} \sum_{k=1}^{K} \alpha_{ck} \sum_{t=1}^{T} K_k(X, I_t) \ast K_k(Y, I_t) \]
We splitted the training set into two equal sized sets (2x4k images)
- the performance of the basic visual kernels are poor
- the Jensen-Shannon divergence overperforms the KL kernels
- the textual and visual kernels complement each other

<table>
<thead>
<tr>
<th>MAP/basic kernel</th>
<th>Avg</th>
<th>85-Baby</th>
<th>71-dog</th>
<th>10-Summer</th>
<th>6-Landscape</th>
<th>24-Mountains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color full</td>
<td>0.2519</td>
<td>0.0465</td>
<td>0.1487</td>
<td>0.3108</td>
<td>0.6203</td>
<td>0.2940</td>
</tr>
<tr>
<td>Color HL</td>
<td>0.2324</td>
<td>0.0509</td>
<td>0.1344</td>
<td>0.3145</td>
<td>0.5794</td>
<td>0.2103</td>
</tr>
<tr>
<td>Color sp1x3</td>
<td>0.2528</td>
<td>0.0365</td>
<td>0.1839</td>
<td>0.2853</td>
<td>0.5912</td>
<td>0.2712</td>
</tr>
<tr>
<td>Color sp2x2</td>
<td>0.2516</td>
<td>0.0301</td>
<td>0.1473</td>
<td>0.2779</td>
<td>0.6026</td>
<td>0.2560</td>
</tr>
<tr>
<td>SIFT full</td>
<td>0.2741</td>
<td>0.0294</td>
<td>0.1718</td>
<td>0.2597</td>
<td>0.7195</td>
<td>0.4003</td>
</tr>
<tr>
<td>SIFT HL</td>
<td>0.2414</td>
<td>0.0215</td>
<td>0.1916</td>
<td>0.2449</td>
<td>0.6737</td>
<td>0.2428</td>
</tr>
<tr>
<td>SIFT sp1x3</td>
<td>0.2827</td>
<td>0.0345</td>
<td>0.1817</td>
<td>0.2749</td>
<td>0.7226</td>
<td>0.3478</td>
</tr>
<tr>
<td>SIFT sp2x2</td>
<td>0.2778</td>
<td>0.0222</td>
<td>0.1927</td>
<td>0.2532</td>
<td>0.7228</td>
<td>0.3717</td>
</tr>
<tr>
<td>KL symm</td>
<td>0.2554</td>
<td>0.2975</td>
<td>0.6328</td>
<td>0.1722</td>
<td>0.4607</td>
<td>0.1571</td>
</tr>
<tr>
<td>KL nonsymm</td>
<td>0.2498</td>
<td>0.3227</td>
<td>0.6262</td>
<td>0.1653</td>
<td>0.4479</td>
<td>0.1648</td>
</tr>
<tr>
<td>JS stamped</td>
<td>0.3110</td>
<td>0.3380</td>
<td>0.7460</td>
<td>0.1719</td>
<td>0.5884</td>
<td>0.2410</td>
</tr>
<tr>
<td>JS nost</td>
<td>0.3140</td>
<td>0.3227</td>
<td>0.7743</td>
<td>0.1694</td>
<td>0.5550</td>
<td>0.2274</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Kernel aggregation</th>
<th>MAP</th>
<th>EER</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. visual + textual run3</td>
<td>weighted</td>
<td>0.438744</td>
<td>0.243574</td>
</tr>
<tr>
<td>1. visual + textual run3</td>
<td>weighted</td>
<td>0.436294</td>
<td>0.241691</td>
</tr>
<tr>
<td>1. visual + textual run3</td>
<td>averaged</td>
<td>0.420406</td>
<td>0.243885</td>
</tr>
<tr>
<td>4. visual run2</td>
<td>averaged</td>
<td>0.371795</td>
<td>0.261183</td>
</tr>
<tr>
<td>5. visual run1</td>
<td>averaged</td>
<td>0.367054</td>
<td>0.264328</td>
</tr>
<tr>
<td>6. textual run</td>
<td>only one</td>
<td>0.345616</td>
<td>0.338127</td>
</tr>
</tbody>
</table>

- the kernel weighting resulted 0.0183 gain in MAP over the averaged kernel
- despite the poor results of the basic visual runs the averaged visual run outperformed the textual run

Some examples for basic kernel weights → similarity to the basic splitted results?

<table>
<thead>
<tr>
<th></th>
<th>Color full</th>
<th>Color HL</th>
<th>Color 1x3</th>
<th>Color 2x2</th>
<th>SIFT full</th>
<th>SIFT HL</th>
<th>SIFT 1x3</th>
<th>SIFT 2x2</th>
<th>Jensen-Shannon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dog</td>
<td>-0.25</td>
<td>0.3</td>
<td>0.1</td>
<td>0.3</td>
<td>0.35</td>
<td>0.3</td>
<td>0.0</td>
<td>0.05</td>
<td>10.45</td>
</tr>
<tr>
<td>Baby</td>
<td>1</td>
<td>1.1</td>
<td>-0.1</td>
<td>-0.1</td>
<td>-0.2</td>
<td>-0.3</td>
<td>0.9</td>
<td>-0.1</td>
<td>10.0</td>
</tr>
<tr>
<td>Mounts</td>
<td>0.7</td>
<td>0.6</td>
<td>0.0</td>
<td>0.0</td>
<td>0.9</td>
<td>-0.15</td>
<td>-0.05</td>
<td>0.2</td>
<td>3.6</td>
</tr>
<tr>
<td>Lads</td>
<td>0.05</td>
<td>0.7</td>
<td>1.1</td>
<td>0.0</td>
<td>0.2</td>
<td>1.3</td>
<td>0.9</td>
<td>2.1</td>
<td>4</td>
</tr>
</tbody>
</table>
Confidence score and binary annotation

Since the output of our classifier was a summarized values of the weighted dot-products of the support vectors and the test instances, we calculated the confidence scores with the sigmoid function:

\[
Prediction_{\text{float}} = \frac{1}{1 + \exp(-1 \times \text{svm}_\text{output})}
\]

For the example-based evaluation we needed to define a mapping from the floating point predictions into a binary annotation. Threshold selection based on:

1. backsbsitution over training set
2. achieving the same +/- ratio as in training set

The previous had much higher F-score (0.593088 vs. 0.545341) and higher Semantic R-Precision (0.71928 vs. 0.70853).

Note: from our submissions the averaged visual kernel had the highest Semantic R-Precision (0.72902450).
Conclusions

1. We used Fisher vector as high-level image descriptor
   - two low level features: SIFT and RGB Color
   - Gaussian Mixture Model with 256 Gaussians

2. We adopted Jensen-Shannon divergence for Flickr tags

3. We calculated pre-computed kernels and combined them before the SVM based classification procedure

4. Future plans: segmentation? 100K+ categories? Combination with search engines?

Thank you!
References

5. Kai Yu. Tong Zhang Xi Zhou and Thomas Huang: ”Image Classification using Super-Vector Coding of Local Image Descriptors”, In 11th ECCV,2010, 201