Lost in Binarization:
Query-Adaptive Ranking for Similar Image Search with Compact Codes

Yu-Gang Jiang
Jun Wang
Shih-Fu Chang

Columbia University
IBM T.J. Watson Research

ACM ICMR 2011, Trento, Italy, April 2011
Growth of Visual Data

- Explosive growth of the amount of visual data
- The Internet boosts up information overload
Large Scale Visual Search

• Nearest neighbor search
• Challenges
  – Feature must fit in memory
    • Disks are too slow...
  – Matching needs to be fast enough

Facebook has around 20 billion images \((2 \times 10^{10})\)
PC can have 20 Gbytes of memory \((2 \times 10^{11} \text{ bits})\)

**Budget of** \(10^1\) **bits/image**

YouTube has over a trillion video frames \((10^{12})\)
Good cluster can have 10 Tbytes memory \((10^{14} \text{ bits})\)

**Budget of** \(10^2\) **bits/frame**

Budget numbers from slide of Rob Fergus
Scalable Search Methods

• **Inverted file**
  – Indexing structure is expensive; typically still requires hundreds of bytes for each image

• **Tree-based approaches**
  – E.g., kd tree
    • Works well on low dimension, but can not handle high dimensional data very well

• **Hashing or binary embedding methods**
  – locality sensitive hashing, spectral hashing, deep learning...
  – Attracted a lot of attention in recent years
Hashing Based Indexing

- Hyperplane partitioning

- Linear projection based hashing

\[ h_k(x) = \text{sgn}(f(w_k^T x + b_k)) \]

\[ y_k(x) = (1 + h_k(x))/2 \]

\[ d_H(x_i, x_j) = \sum_k |y_k(x_i) - y_k(x_j)| \rightarrow \text{Hamming Distance} \]
Visual Search by Compact Codes

10110110101

Visual Query

Limitation
- Coarse ranking

Modified from slide of Rob Fergus
Visual Search by Compact Codes

101101110101

Visual Query

Limitation
- Coarse ranking

220 different codes with Hamming distance 3
66 different codes with Hamming distance 2
12 different codes with Hamming distance 1
How to produce better ranking?

• Assume we use binary codes with \( n \) bits
  
  – There will be \( n \) different Hamming distances
    
    • Original \# levels of ranking: \( n \)

<table>
<thead>
<tr>
<th>Query</th>
<th>Image 1</th>
<th>Image 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1 0</td>
<td>1 1 1 1 0 (HD=1)</td>
<td>1 0 1 1 1 (HD=1)</td>
</tr>
</tbody>
</table>

Bit-wise weights: 0.1 0.3 0.5 0.2 0.6

• \#levels of ranking increase from \( n \) to \( 2^n \)!
• The weights are computed adaptively for each query
Auxiliary database: semantic concept classes
- image compact codes and learned class-specific weights

Cityscape
[0.12 0.11 0.42 ... 0.10]

Plane
[0.22 0.11 0.12 ... 0.15]

Water
[0.05 0.15 0.21 ... 0.46]

Tree
[0.08 0.17 0.02 ... 0.19]

Person
[0.02 0.24 0.22 ... 0.08]

Sunset
[0.22 0.04 0.62 ... 0.02]
Learning Concept-Specific Weights

- Weight vector for concept $k$
  \[ f(a_1, \ldots, a_k) = \sum_{i=1}^{k} \sum_{x \in X_i} ||a_i \odot x - a_i \odot c^{(i)}||^2 \]

- Binary code of an image

- Center of binary codes of concept $i$

- Intra-class compactness

- Concept class similarity in raw feature space
  \[ g(a_1, \ldots, a_k) = \sum_{i,j=1}^{k} s_{ij} ||a_i \odot c^{(i)} - a_j \odot c^{(j)}||^2 \]

- Inter-class relationship

- Final objective function
  \[
  \min_{a_1, \ldots, a_k} f(a_1, \ldots, a_k) + \lambda g(a_1, \ldots, a_k)
  \]
  \[
  \text{s.t. } a_i^\top 1 = 1, \ i = 1, \ldots, k,
  \]
  \[
  a_i \geq 0, \ i = 1, \ldots, k,
  \]
Rewrite the objective function in quadratic form:

\[ f(a_1, ..., a_k) + \lambda g(a_1, ..., a_k) = \frac{1}{2} a_i^\top Q_i a_i + p_i^\top a_i + t_i \]

where

\[ Q_i = 2A_i + 4\lambda (\sum_l s_{il} - s_{ii}) C_{ii}, \]

\[ p_i = -4\lambda \sum_{j \neq i} s_{ji} C_{ji} a_j, \]

\[ t_i = \sum_{j \neq i} a_j^\top A_j a_j + 2\lambda \sum_{j \neq i, l} s_{jl} a_j^\top C_{jj} a_j - 2\lambda \sum_{j \neq i, l \neq i} s_{jl} a_j^\top C_{jl} a_l. \]
Learning Concept-Specific Weights

• Rewrite the objective function in quadratic form:

\[ f(a_1, ..., a_k) + \lambda g(a_1, ..., a_k) = \frac{1}{2} a_i^\top Q_i a_i + p_i^\top a_i + t_i \]

Repeat
For \( i = 1, ..., k \)
    Compute \( Q_i, p_i, \) and \( t_i \)
    Solve the following QP problem:

\[ a_i^* = \arg \min_{a_i} \frac{1}{2} a_i^\top Q_i a_i + p_i^\top a_i + t_i \]

s.t. \( a_i^\top 1 = 1 \) and \( a_i \geq 0; \)

Set \( a_i = a_i^*; \)
End for
Until convergence
The framework (Recall)

**Auxiliary database:** semantic concept classes
- image compact codes and learned class-specific weights

<table>
<thead>
<tr>
<th>Concept</th>
<th>Compact Codes</th>
<th>Learned Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cityscape</td>
<td>[0.12 0.11 0.42 ... 0.10]</td>
<td>[0.13 0.05 0.51 ... 0.06]</td>
</tr>
<tr>
<td>Plane</td>
<td>[0.22 0.11 0.12 ... 0.15]</td>
<td></td>
</tr>
<tr>
<td>Water</td>
<td>[0.05 0.15 0.21 ... 0.46]</td>
<td></td>
</tr>
<tr>
<td>Tree</td>
<td>[0.08 0.17 0.02 ... 0.19]</td>
<td></td>
</tr>
<tr>
<td>Person</td>
<td>[0.02 0.24 0.02 ... 0.08]</td>
<td></td>
</tr>
<tr>
<td>Sunset</td>
<td>[0.22 0.04 0.62 ... 0.02]</td>
<td></td>
</tr>
</tbody>
</table>

**Query:***

1. Feature extraction
2. Binary embedding to compact code
3. Image database (compact codes)
4. Query-adaptive weights

[0.13 0.05 0.51 ... 0.06]
Experimental results

- 260,000 Flickr images from NUS
- 81 fully labeled classes
- Randomly sampled 8,000 query images
- Evaluation: normalized (mean) average precision
Two supervised binary coding methods

• Semi-Supervised Hashing
  • J. Wang, S. Kumar, S.-F. Chang, CVPR & ICML 2010

• Deep Belief Network
  • Hinton & Salakhutdinov
    • Science 2006
Overall performance

DBN, entire list

SSH, entire list

DBN, upper half

SSH, upper half
Per-category performance

- Divide the queries into 81 groups according to their semantic label(s)
Result example

Query

Baseline

Ours

Query

Baseline

Ours
Summary

• A query-adaptive ranking approach for compact code image search
  • Finer-grained ranking!

• Future work
  • Consider more semantic classes in the auxiliary database
Thank you!