Recent Image Search Techniques

CVPR 2016 Tutorial

Sung-Eui Yoon and Zhe Lin
Outline

- Indexing and Encoding Schemes for Large-Scale Image Search (45 min)
  - Product quantization and its variants
  - Inverted index, inverted multi-index, residual-shortlist

- Applications of Image Search (45 min)
  - Object retrieval and localization
  - Facial attribute recognition
  - Discriminative feature learning with CNN
  - Large-scale semantic search, recommendation
  - Large-scale image tagging
Indexing/Encoding Schemes

• Local descriptor-based method
  – Keypoints + Local invariant descriptors [Lowe99, Mikolajczyk04]
  – Bag of features model [Sivic and Zisserman03] [Philbin07]
  – Inverted index [Sivic and Zisserman03] [Nister06]
  – Geometric verification [Philbin07] [Jegou08]

• Global descriptor-based method
  – Product quantization [Jegou11]
  – Optimized product quantization [Ge13]
  – Distance-encoded product quantization [Heo14]
  – Inverted Multi-Index [Lempitsky13]
  – Shortlist computation [Heo16]
Local Feature-based Method

Interest points

Local descriptors

Photo source: Lempitsky's slides
Local Feature-based Method

• Mainly used for identical object/scene instance retrieval

Images from H. Jegou’s SSMS’12 Talk Slide

[X. Shen et al. CVPR 2012]
Local Descriptor-based Method

• How to index bag-of-features?

  – Build a codebook by k-means

  – Encode each local descriptor into a visual word

  – Store visual words into an *inverted file*
Inverted Index

• Organize bag-of-features w.r.t. visual words
Vocabulary Size

• Larger codebooks for efficiency?
  – Lower quantization errors with increasing the voc. size
  – Increased assignment complexity and memory requirement
  – Tradeoff between assignment accuracy and efficiency
Learning Large Vocabularies

• Hierarchical k-means [Nister06]
  – Quantize recursively

• Approximate k-means [Philbin07]
  – Flat k-means with fast approximate assignment step with randomized tree search
Geometric Verification

- RANSAC-based method [Philbin07]
- Weak geometric consistency [Jegou08]
- Geometrical min-hash [Chum 09]
- Bundling features [Wu 09]
- Spatial inverted file / Local BoW [Lin 10]
- Geometry preserving visual phrases [Zhang 11]
- Generalized Hough-voting-based [Shen 12]
Scalability

• Limited to a few million images due to memory cost

• How about larger scale search?
  – Search on a database of 10M to 1B
    • Global descriptor
    • Indexing (coarse quantizer) -- speed
    • Encoding (compact representation) -- memory
What is important for very large-scale search?

• Performance criteria
  – Search accuracy
  – Search speed
  – Memory/storage usage
Global Feature-based Method

Image Database → Feature Extractor → DB descriptors → Query descriptor → ANN Search → Similar Images
Nearest Neighbor Search

• Exhaustive NN Search
  – Euclidean distance metric

\[
\text{NN}(x) = \arg\min_{y \in \mathcal{Y}} \|x - y\|^2
\]

  – Very costly when the DB is very large, and the descriptor is very high dimensional: O(n*d)
  – Can be used as ”verification” method on a small candidate set
Efficient Solutions

• Dimensionality reduction
  – e.g. PCA

• Hierarchical methods (ex: kd-tree, HKM)
  – Efficient and accurate for a mid-range dataset
  – Suffer from ‘curse of dimensionality’
  – Do not provide a compact data representation
  – e.g. FLANN library [Muja & Lowe 09]

• Binary code embedding
  – LSH, spectral hashing, min-hash, hamming embedding, etc.
  – Very fast during search time
  – But, may not handle well very high-dim data
  – Requires raw features in verification, e.g. 128GB for 1B SIFT

• **Quantization-based methods (*)**
  – **Indexing:** inverted indexing, inverted multi-indexing, etc.
  – **Encoding:** vector quantization, product quantization, optimized product quantization, distance-encoded product quantization, etc.
Global Descriptor-based Method

1. Image Database
2. Feature Extractor
3. Query
4. DB descriptors
5. Query descriptor
6. Indexed Images
7. Search
8. Similar Images
Search Framework (Global desc.)

$q$ → Near cluster search → Inverted File

$T$ (shortlist size)

$R$ elements according to estimated distance

Top $R$ elements according to estimated distance
Indexing and Encoding

Indexer (ex: Coarse Quantizer)

Encoder

feature space

$\begin{align*}
&x \\
&c_1 \\
&c_2 \\
&c_k
\end{align*}$

Index

$x$

Compact Code

$\begin{align*}
&Data_{1,1} \hspace{1cm} Data_{1,2} \hspace{1cm} Data_{1,3} \hspace{1cm} \ldots \hspace{1cm} Data_{1,n_1} \\
&Data_{2,1} \hspace{1cm} Data_{2,2} \hspace{1cm} Data_{2,3} \hspace{1cm} \ldots \hspace{1cm} Data_{2,n_2} \\
&\vdots \\
&Data_{m,1} \hspace{1cm} Data_{m,2} \hspace{1cm} Data_{m,3} \hspace{1cm} \ldots \hspace{1cm} Data_{k,n_k}
\end{align*}$

Img-ID / Compact Code
Indexing and Encoding

- **Encoding**
  - Residual vector as input to encoder
  - Product quantization and its variants

- **Indexing**
  - Inverted (multi-)index
  - Residual-aware shortlist selection
Product Quantization [Jegou et al., TPAMI 2011]

• Vector quantization
  – For a very large codebook, e.g. $K = 4^{64}$
    $\rightarrow$ intractable as an encoder (speed and memory)

• Product quantization
  – Cartesian product of subspace quantization
  – Can generate an exponentially large codebook at
    very low memory/time cost
Product Quantization [Jegou et al., TPAMI 2011]
Product Quantization [Jegou et al., TPAMI 2011]

Distance estimation
(between encoded y and query x)

Subspace #1

Subspace #M

x: query, y: data
q(x)=c_x
q(y)=c_y

Figure from [Jegou et al., TPAMI 2011]
Distance Estimation Error

- Distance estimation error is statistically bounded by quantization error [Jegou PAMI 2011]
  \[ \text{MSDE}(q) \leq \text{MSE}(q) \]
  \[ \left( d(x, y) - d(x, q(y)) \right)^2 \leq d(y, q(y))^2 \]
Distance Estimation Bias

- Unbiased asymmetric estimator

\[ \tilde{e}(x, y) = \tilde{d}(x, y)^2 + \sum_j \xi_j(y) \]

Figures from [Jegou PAMI 2011]
Non-Exhaustive Search (IVFADC)

Inverted file structure

Database indexing

- coarse quantizer $q_c$
- $q_c(y)$
- compute residual $r(y)$
- product quantizer $q_p$
- $q_p(r(y))$
- inverted list $L_i$
- list entry $id$ code
- append to inverted list

Query processing

- compute residual $r(x)$
- coarse quantizer $q_c(x)$
- compute $d(r(x), q_p(r(y)))$
- select k smallest distances
- Search result

Figures from [Jegou PAMI 2011]
Optimized Product Quantization

• Problem of product quantization
  – Subspaces are assumed to be independent
  – Subspaces have unbalanced variances

• Subspace decomposition

Figures from Arandjelovic’s slides

<table>
<thead>
<tr>
<th>$m$</th>
<th>SIFT</th>
<th>GIST</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>natural</td>
<td>0.593</td>
<td>0.921</td>
</tr>
<tr>
<td>random</td>
<td>0.501</td>
<td>0.859</td>
</tr>
<tr>
<td>structured</td>
<td>0.640</td>
<td>0.905</td>
</tr>
</tbody>
</table>
Optimized Product Quantization

[Ge et al., CVPR 2013]
[Norozi and Fleet, CVPR 2013]

• Optimal subspace decomposition
  – **Estimate a rotation** projection matrix R to minimize quantization distortion
  – Rotation can **de-correlate** data and **balance** subspace variances well

From Arandjelovic’s slides
Optimized Product Quantization

[Ge et al., CVPR 2013]
[Norozi and Fleet, CVPR 2013]

- **Formulation**
  \[ \min_{R, C^1, \ldots, C^M} \sum_x \| x - c(i(x)) \|^2, \]
  \[ \text{s.t. } c \in C = \{ c \mid R c \in C^1 \times \cdots \times C^M, R^T R = I \} \]

- **Solutions**
  - **Nonparametric solution**
    - (step 1) Fix \( R \), estimate clusters \( c \) and assignment \( i(x) \) – k-means
    - (step 2) Fix \( C \)'s and optimize \( R \) – orthogonal procrustes problem
    - Alternate step 1 and step 2 until max iteration
  - **Parametric solution**
    - Assumes Gaussian distribution
    - Eigenvalue allocation algorithm
      - Align the data by PCA (make subspaces independent)
      - Allocate eigenvalues to buckets with balance
Optimized Product Quantization

- Results

Ge et al., CVPR 2013
Norozi and Fleet, CVPR 2013
Locally Optimized PQ

[Kalantidis et al., CVPR 2014]
In PQ, errors of estimated distances $d(x, y) - d(c_x, c_y)$ tends to be higher as $r_x$ and $r_y$ becomes larger.
Distance Encoded PQ (DPQ)

- Encode quantized distance from cluster center as well as the cluster index
Orthogonality in High Dim. Space

In high dimensional space, two randomly chosen vectors are highly likely to be orthogonal*.

\[
d(x, y)^2 = \|x - y\|^2 = \|(c_x - c_y) + (x - c_x) + (y - c_y)\|^2 \\
\approx \|c_x - c_y\|^2 + \|x - c_x\|^2 + \|y - c_y\|^2 \\
(\because c_x - c_y, x - c_x, \text{ and } y - c_y \text{ are mutually orthogonal.})
\]

*Mathematical foundations for the information age [J. Hopcroft, 2010]
Distance Estimation

Symmetric Distance

\[
\begin{align*}
\bar{r}_x & = r_x - C_x \\
d(C_x, C_y) & + \bar{r}_x^2 + \bar{r}_y^2
\end{align*}
\]

Asymmetric Distance

\[
\begin{align*}
\bar{r}_y & = r_y - C_y \\
d(C_x, C_y) & + \bar{r}_y^2
\end{align*}
\]
**Result (1M, 960-Dim GIST)**

<table>
<thead>
<tr>
<th>Method</th>
<th>Symmetric distance</th>
<th>Asymmetric distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours+OPQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ours+PQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SpH (SHD)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ITQ</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1000-nearest neighbor search mAP

- OPQ: Optimized PQ [Ge et al., CVPR 2013]
- SpH: Spherical Hashing [Heo et al., CVPR 2012]
- ITQ: Iterative Quantization [Gong and Lazebnik, CVPR 2011]
Result (1M, 1024-Dim BoW)

1000-nearest neighbor search mAP
SD: Symmetric distance
AD: Asymmetric distance
Result (Accuracy/Time/Memory)

- Tested on 4096-dimensional 11M CNN features
- Indexer: Vector Quantization with 4K centroids (4K lists)

![Graph showing Precision@100 vs. Search Time (ms) for different bit lengths and shortlist sizes](image-url)

- T = 100K (shortlist size)
- Bit lengths: 64 bits, 128 bits, 256 bits, 512 bits, 1024 bits

---

37
Indexing and Encoding

• Encoding
  – Residual vector as input to encoder
  – Product quantization or its variant

• Indexing
  – Inverted (multi-)index
  – Residual-aware shortlist selection
Inverted Index

- Generate a codebook by quantization
  - e.g. k-means clustering

- Build an inverted index
  - Quantize each descriptor into the closest word
  - Organize desc. IDs in terms of words

Construction time:

Figure from Lempitsky’s slides
Inverted Index

• Given a query,
  – Find its K closest words
  – Retrieve all the data in the K lists corresponding to the words

• Large K
  – Low quantization distortion
  – Expensive to find kNN words
Inverted Multi-Index [Babenko and Lempitsky, CVPR 2012]

- **Product quantization for indexing**
- **Main advantage:**
  - For the same K, much finer subdivision
  - Very efficient in finding kNN codewords
Inverted Multi-Index [Babenko and Lempitsky, CVPR 2012]

\[
\begin{array}{c|c|c|c|c|c}
\hline
i & u_{\alpha(i)} & r \\
\hline
1 & u_3 & 0.5 \\
2 & u_4 & 0.7 \\
3 & u_5 & 4 \\
4 & u_2 & 6 \\
5 & u_1 & 8 \\
6 & u_6 & 9 \\
\hline
\end{array}
\]

\[
\begin{array}{c|c|c}
\hline
j & v_{\beta(j)} & s \\
\hline
1 & v_4 & 0.1 \\
2 & v_3 & 2 \\
3 & v_5 & 3 \\
4 & v_2 & 6 \\
5 & v_6 & 7 \\
6 & v_1 & 11 \\
\hline
\end{array}
\]

Comparison:
- Inverted index
  - Number of entries: \( K \)
  - Operations to match to codebooks: \( 2K + O(1) \)
- Inverted multi-index
  - Number of entries: \( K^2 \)
  - Operations to match to codebooks: \( 2K + O(1) \)
### Inverted Multi-Index

[Babenko and Lempitsky, CVPR 2012]

#### multi-sequence algorithm

<table>
<thead>
<tr>
<th>$q^1$ vs. $U$</th>
<th>$q^2$ vs. $V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$i$</td>
<td>$u_\alpha(i)$</td>
</tr>
<tr>
<td>1</td>
<td>$u_3$</td>
</tr>
<tr>
<td>2</td>
<td>$u_4$</td>
</tr>
<tr>
<td>3</td>
<td>$u_5$</td>
</tr>
<tr>
<td>4</td>
<td>$u_2$</td>
</tr>
<tr>
<td>5</td>
<td>$u_1$</td>
</tr>
<tr>
<td>6</td>
<td>$u_6$</td>
</tr>
</tbody>
</table>

#### Matrix

<table>
<thead>
<tr>
<th>$u_\alpha(i)$</th>
<th>$v_\beta(j)$</th>
<th>$(i, j)$</th>
<th>$r(i) + s(j)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_3$</td>
<td>$v_4$</td>
<td>(1,1)</td>
<td>0.6 (0.5+0.1)</td>
</tr>
<tr>
<td>$u_4$</td>
<td>$v_4$</td>
<td>(2,1)</td>
<td>0.8 (0.7+0.1)</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$v_3$</td>
<td>(1,2)</td>
<td>2.5 (0.5+2)</td>
</tr>
<tr>
<td>$u_4$</td>
<td>$v_3$</td>
<td>(2,2)</td>
<td>2.7 (0.7+2)</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$v_5$</td>
<td>(1,3)</td>
<td>3.5 (0.5+3)</td>
</tr>
<tr>
<td>$u_4$</td>
<td>$v_5$</td>
<td>(2,3)</td>
<td>3.7 (0.7+3)</td>
</tr>
<tr>
<td>$u_5$</td>
<td>$v_4$</td>
<td>(3,1)</td>
<td>4.1 (4+0.1)</td>
</tr>
<tr>
<td>$u_5$</td>
<td>$v_3$</td>
<td>(3,2)</td>
<td>6 (4+2)</td>
</tr>
<tr>
<td>$u_3$</td>
<td>$v_2$</td>
<td>(1,4)</td>
<td>6.5 (0.5+6)</td>
</tr>
</tbody>
</table>

...
Inverted Multi-Index

[Babenko and Lempitsky, CVPR 2012]
Residual-Aware Shortlist Retrieval

[Jaepil et al., CVPR 2016]

Limitation of prev. methods

Neighbors could be missed due to the quantization error

Select promising subset in parallel from all the lists
Motivation: Better Distance Estimator?

• Distance estimator between \( y \) and \( x \) based on orthogonality in high-dimensional space:

\[
d(y, x)^2 \approx d(y, c)^2 + d(c, x)^2 = h^2 + r^2
\]

• Compute shortlist according to \( h^2 + r^2 \), instead of only \( h^2 \)
Distance Estimator

\[ d^2 = h_{xy}^2 + r_x^2 - 2h_{xy} r_x \cos \theta_{xy} \]

\[ = h_{xy}^2 + \left(1 - \frac{2h_{xy}}{r_x} \cos \theta_{xy}\right) r_x^2 \]

\[ \approx h_{xy}^2 + \alpha_K r_x^2 \]

\[ \alpha_K \] is a constant depending on \( K \) (= number of desired neighbors)

\[ = \text{avg}\left( \text{with K-NN pairs and with random pairs} \right) \]

e.g) \( \alpha_1 = 0.5 / \alpha_{10} = 0.55 / \alpha_{100} = 0.62 / \alpha_{1000} = 0.70 \)
Distance Estimator

\[ h^2 + \alpha_K r^2 \]
Shortlist with Inverted Index

- **Lookup Table Precomputation**
  - Sort and partition each inverted list according to $r_x^2$

- Compute a lookup table $W(i, j)$ — the number of data in $L_i$ whose $r_x^2$ are smaller than $R_j$

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>3</th>
<th>10</th>
<th>14</th>
<th>18</th>
<th>30</th>
<th>38</th>
<th>45</th>
<th>45</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>12</td>
<td>15</td>
<td>24</td>
<td>28</td>
<td>30</td>
<td>35</td>
<td>35</td>
<td>35</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>10</td>
<td>15</td>
<td>20</td>
<td>27</td>
<td>33</td>
<td>38</td>
<td>40</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>
Shortlist with Inverted Index

- **Runtime Shortlist Selection**
  - Given a query $y$, shortlist size $T$, and the target number of neighbors $K$
  - Compute squared distances to centroids $h_i^2$
  - Do binary search to find a distance threshold that corresponds to $T$

\[
W = \begin{array}{cccccccccccc}
0 & 0 & 3 & 10 & 14 & 18 & 30 & 38 & 45 & 45 \\
5 & 10 & 12 & 15 & 24 & 28 & 30 & 35 & 35 & 35 \\
0 & 4 & 10 & 15 & 20 & 27 & 33 & 38 & 40 & 40 \\
\end{array}
\]

$T=50$
Shortlist with Inverted Multi-Index

- **Residual-Aware Indexing**
  - Partition each cluster by $r_x$ to define index (subspace ID, cluster ID, distance ID)
  - Compute a representative residual distance $\bar{r}^2_{s,i,j}$ for each index $(s, i, j)$

<table>
<thead>
<tr>
<th>Subspace ID</th>
<th>Subspace #1</th>
<th>Subspace #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subspace #2</td>
<td>(1,1,1)</td>
<td>(1,1,1)</td>
</tr>
<tr>
<td></td>
<td>(1,1,2)</td>
<td>(1,1,2)</td>
</tr>
<tr>
<td></td>
<td>(1,2,1)</td>
<td>(1,2,1)</td>
</tr>
<tr>
<td></td>
<td>(1,2,2)</td>
<td>(1,2,2)</td>
</tr>
<tr>
<td>(2,1,1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,1,2)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,2,1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,2,2)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\bar{r}^2_{s,i,j}$

- $\bar{r}^2_{1,1,1} = 0.3$
- $\bar{r}^2_{1,1,2} = 0.4$
- $\bar{r}^2_{1,2,1} = 0.3$
- $\bar{r}^2_{1,2,2} = 0.8$
- $\bar{r}^2_{2,1,1} = 0.7$
- $\bar{r}^2_{2,1,2} = 0.6$
- $\bar{r}^2_{2,2,1} = 0.9$
- $\bar{r}^2_{2,2,2} = 1.0$
## Shortlist with Inverted Multi-Index

- **Runtime Shortlist Selection**
  - Compute \( h_{s,i}^2 \) (the distance to \( i \)-th centroid of \( s \)-th subspace), and sort the indices in each subspace according to \( h_{s,i}^2 \)
  - Traverse the table by using the multi-sequence algorithm

<table>
<thead>
<tr>
<th>Subspace #1</th>
<th>( h_{1,1}^2 ) = 0.8</th>
<th>( h_{1,2}^2 ) = 1.0</th>
<th>( h_{1,i}^2 + \alpha_{K,1} r_{1,i,j}^2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subspace #2</td>
<td>( h_{2,1}^2 ) = 1.1</td>
<td>( h_{2,2}^2 ) = 0.6</td>
<td></td>
</tr>
<tr>
<td>(2,1,1)</td>
<td>1.8</td>
<td>2.9</td>
<td>3.2</td>
</tr>
<tr>
<td>(2,1,2)</td>
<td>2.0</td>
<td>3.1</td>
<td>3.4</td>
</tr>
<tr>
<td>(2,2,1)</td>
<td>1.2</td>
<td>2.3</td>
<td>2.6</td>
</tr>
<tr>
<td>(2,2,2)</td>
<td>1.6</td>
<td>2.7</td>
<td>3.0</td>
</tr>
<tr>
<td>(1,1,1)</td>
<td>1.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,1,2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,2,1)</td>
<td>1.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1,2,2)</td>
<td>1.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2,2,1)</td>
<td></td>
<td>2.3</td>
<td>2.6</td>
</tr>
<tr>
<td>(2,2,2)</td>
<td></td>
<td>2.7</td>
<td>3.0</td>
</tr>
</tbody>
</table>
Shortlist Evaluation

Accuracy of Shortlist retrieving top 100 GT

- GIST-1M-960D
- CNN-1M-4096D
- VLAD-1M-2048D
- VLAD-1M-8192D
Shortlist Evaluation

End-to-End Accuracy

GIST-1M-960D, Re-ranked

VLAD-1M-8192D, Re-ranked
Applications

• Object retrieval and localization
• Product image recognition
• Face detection and attribute recognition
• Large-scale semantic image search
• Large-scale image tagging
• Free-text image search
Object Retrieval and Localization

[X. Shen et al., CVPR 2012]
Object Retrieval and Localization

- Local correspondence voting for non-rigid object matching

\[
D_{g_1} \rightarrow g_2 \quad D_{g_4} \rightarrow g_5
\]

\[
tf\text{-}idf \text{ pair voting score: } \frac{idf(k) \cdot idf(k)}{tf(Q,k) \cdot tf(D,k)}
\]

Choose the transformation with the highest score!
Object Retrieval and Localization

Examples of Voting Maps
Object Retrieval and Localization

Non-rigid cases
Product Image Recognition

[X. Shen et al., ECCV 2012]

Examples of product images in the database

Examples of query images taken by mobile phones
Product Image Recognition

(a)  (b)  (c)
(d)  (e)  (f)
Product Image Recognition

Images  Support map  Extraction  GrabCut
Face Detection by Image Retrieval

[X. Shen et al., CVPR 2013]
[H. Li et al., CVPR 2014]
Face Detection by Image Retrieval

Database Images

Voting Maps

Aggregation By Boosting
Face Detection by Image Retrieval

Example detection results
Facial Attribute Recognition

transfer landmark, pose, age, gender, expression...
Facial Attribute Recognition
Data-Driven Object Segmentation

[J. Yang et al. CVPR 2014]
Data-Driven Automatic Cropping

[A. Samii et al. CGF 2015]
Image Similarity Search
Image Similarity Search

Deep Convolutional Neural Network

Distance Encoded OPQ
Image Similarity Search
Automatic Image Tagging
Dataset

- 17 million Adobe stock images with tags

18k-100k tag vocabulary
Deep-kNN Tagging System

The system consists of the following components:

1. **Labeled Image DB (17M)**
2. **Neural Net Feature Extractor**
3. **Search Engine (DOPQ)**
4. **Search Index**
5. **kNN Voting**

The system takes an input image and processes it through the feature extractor, which then uses a search engine to query the database and extract relevant features. The search index helps in finding similar images. The kNN voting process then aggregates the results to tag the image with keywords such as:

- white tiger
- snow
- cute
- tiger
- beast
- cage
- zoo
- forest
- zebra

A keyword frequency table is shown:

<table>
<thead>
<tr>
<th>Keyword</th>
<th>freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>white tiger</td>
<td>3</td>
</tr>
<tr>
<td>snow</td>
<td>3</td>
</tr>
<tr>
<td>winter</td>
<td>2</td>
</tr>
<tr>
<td>cute</td>
<td>1</td>
</tr>
<tr>
<td>tiger</td>
<td>1</td>
</tr>
<tr>
<td>beast</td>
<td>1</td>
</tr>
<tr>
<td>cage</td>
<td>1</td>
</tr>
<tr>
<td>zoo</td>
<td>1</td>
</tr>
<tr>
<td>forest</td>
<td>1</td>
</tr>
<tr>
<td>zebra</td>
<td>1</td>
</tr>
</tbody>
</table>
Discriminative Feature Learning

- **Tag set similarities can reflect the visual similarities**

cute, fluffy, domestic, one, playful, curious, portrait, funny, paw, young, spots, black, pets, puppy, adorable, pretty, sitting, charming, fur, hair, lovely, animals, look, heartwarming, delectable, cuddly, veterinarian, small, mammal, soft, exquisite, ears, black puppy

cute, wreath, domestic, one, playful, bright, young, spots, black, pretty, orange, lovely, hawaiian, sitting, motley, charming, fur, flowers, animals, look, heartwarming, cuddly, veterinarian, small, mammal, soft, exquisite, ears, fluffy, hair, portrait, curious, funny, paw, artificial, pets, puppy, adorable, delectable, small black, black puppy
Tag Context Space

- We define pseudo classes from tag information

<table>
<thead>
<tr>
<th>Image-Tags</th>
<th>M dim sparse Bag of Tags</th>
<th>k pseudo classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_1 = {t^1_1, t^1_2, \ldots}$</td>
<td>$x_1 = [0,1,1,0,\ldots]$</td>
<td>$t_f/idf$ L₂ normalize</td>
</tr>
<tr>
<td>$I_2 = {t^2_1, t^2_2, \ldots}$</td>
<td>$x_2 = [1,0,0,0,\ldots]$</td>
<td>clustering</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>$I_n = {t^n_1, t^n_2, \ldots}$</td>
<td>$x_n = [0,0,1,1,\ldots]$</td>
<td>:</td>
</tr>
</tbody>
</table>
**Sudo-Classes**

<table>
<thead>
<tr>
<th>C/ID</th>
<th>Page #</th>
<th>Display</th>
<th>&lt;</th>
<th>&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

580 items (13 pops): waqtaal (0.3571), motacilla (0.2300), white waqtaal (0.1237), bird (0.1136), alba (0.1054), wildlife (0.0865), yellow waqtaal (0.0865), plumage (0.0795), ornithology (0.0765), fava (0.0723)
# Deep Feature Learning

<table>
<thead>
<tr>
<th>100 Tags</th>
<th>P@100</th>
<th>R@100</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet Feat (1000)</td>
<td>0.196</td>
<td>0.552</td>
</tr>
<tr>
<td>Deep Tagging Feat (1000)</td>
<td>0.204</td>
<td>0.573</td>
</tr>
<tr>
<td>Deep Tagging Feat (3500)</td>
<td><strong>0.220</strong></td>
<td><strong>0.616</strong></td>
</tr>
</tbody>
</table>
Automatic Image Tagging
Improving Tag Prediction

fish

Highly Similar User Tags

fish
cook
cooking
man
food
kitchen
homework
....
Search Results

cloudy

KNN

Div-KNN
Search Results

cloth

KNN

Div-KNN
KNN + Linear Classifier

Linearly combine the scores from KNN-retrieval and Logistic classifier
Evaluation

- Evaluation set
  - 82 tags: 30~50 images/tag
  - Include hard negatives
  - Top N Precision
Demo
Free-Text Image Search

sydney opera house
Free-Text Image Search

- Internet Image Search → Crawl Examples

- Multi-Query Visual Search → Retrieve Images
Demo

Google Reference Images

Our Database Images
Image Recommendation

[C. Fang et al. CVPR 2015]
### Image Recommendation

[C. Fang et al. CVPR 2015]

<table>
<thead>
<tr>
<th>Query</th>
<th>Tags</th>
<th>Nearest neighbors in latent factor space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>beauty portrait woman hair</td>
<td><img src="image1" alt="Nearest neighbors" /></td>
</tr>
<tr>
<td>2</td>
<td>wedding photography</td>
<td><img src="image2" alt="Nearest neighbors" /></td>
</tr>
<tr>
<td>3</td>
<td>elegant graceful neat refined</td>
<td><img src="image3" alt="Nearest neighbors" /></td>
</tr>
<tr>
<td>4</td>
<td>automotive classic</td>
<td><img src="image4" alt="Nearest neighbors" /></td>
</tr>
<tr>
<td>5</td>
<td>automotive design Industrial transportation</td>
<td><img src="image5" alt="Nearest neighbors" /></td>
</tr>
<tr>
<td>6</td>
<td>Casa La Encantada house</td>
<td><img src="image6" alt="Nearest neighbors" /></td>
</tr>
<tr>
<td>7</td>
<td>mascot logo gaming sport</td>
<td><img src="image7" alt="Nearest neighbors" /></td>
</tr>
<tr>
<td>8</td>
<td>shoe footwear</td>
<td><img src="image8" alt="Nearest neighbors" /></td>
</tr>
<tr>
<td>9</td>
<td>food pie food photography</td>
<td><img src="image9" alt="Nearest neighbors" /></td>
</tr>
</tbody>
</table>