Semantic Image Search

Alex Egg
Inspiration


Figure 4: (Left) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (Right) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.
Deep CNN

c) First Pooling Layer

d) second to last FC Layer
Nearest Neighbor Search

Images in 2D space
Semantic Search

Reverse Image Search: Image A -> Identical image of A
Semantic Image Search: Image A -> Any images containing A

The word "semantic" refers to the meaning or essence of something:

- Text Semantics: Sentiment & Meaning
- Image Semantics: Object Quantification
Theoretical Implementation

1. Setup DCNN (Image Feature Generator)
2. Setup Database
3. Index feature vectors in database
4. Query database using 1-Nearest Neighbor Search
Theoretical Impl. Problems

- **Accuracy**: Distance measure breakdown in high dimensions
- **Scalability**: Storage & Nearest Neighbor intractability

Naive Solution
Problem: Accuracy

- AlexNet: 4096D
- VGG: 4096D
- Inception V3: 2048D
- Inception V4: 1536D

After about 10D all points are equally far away -- distance measure break down: **Curse of Dimensionality**
Problem: Scalability

Storage Complexity: \(4096 \times 32 \text{ bits float} = 16.38 \text{ KB/vector} \times 1 \text{M images} = 16.38 \text{ GB}\)

Computational Complexity: \(1\text{NN} = O(n); 1\text{ms} \times 1e6 = 16.6m\)

We can mitigate most of these problems using a modern database system.
Solutions

- **Scalability**: Reduce search space
- **Accuracy**: Reduce Dimensionality
Dimensionality Reduction

- **PCA**: Directions of projection are data-dependent
- **Random Projections**: Direction of projections are data-independent

1. Data is so high dimensional that it is too expensive to compute PCA
2. *You don’t have access to the data all at once, as in streaming*
Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss Lemma: “A set of \( p \) points in high-dimensional space can be linearly embedded in \( m > (12 \log p) \) dimensions without distorting the distance between any two points by more than a factor of \((1 \pm \varepsilon)\).

\[
m > (4 \log 1e6)
\]

\[
m > 55
\]

- \( 2^5 = 32 \) splits
- \( 2^6 = 64 \) splits
Hash Table

Binary Tree Search is $O(\log n)$

Hash Table lookup is $O(1)$

Hashing Function: $h(v) = \text{sgn}(v \cdot r)$, that is $h(v) = \pm 1$ depending on what side of the hyperplane $v$ lies.

Bad Hashing Function (Maximizes Collisions)

```
In [125]: D=2
In [126]: p=np.random.randn(D)  # CNN Image Vector 4096D
In [127]: p
Out[127]: array([-0.77079265, 0.89362363])
In [128]: r=np.random.randn(4, D)  # 4 Random Projections
In [129]: projection=np.dot(r,p)
In [130]: projection
Out[130]: array([ 1.72679042, -1.18775683, -0.25176009,  1.32848232])
In [131]: np.sign(projection)
Out[131]: array([ 1., -1., -1.,  1.])
```
Locality Sensitivity Hashing (LSH)

Keep splitting until nodes are small enough

Median splits give nicely balanced trees

Build a forest
1. Same DCNN Feature Extractor
2. Database to store hashtable instead of vectors
3. Index Features in Database
4. Approximate Nearest Neighbors using LSH
Query: Giraffe & Zebra
Results: Giraffes and/or Zebras in various colors, varieties & orientations
Query: Human Face
Results: Human Faces in various colors, varieties & orientations
Query: Cat

Results: Cats in various colors, varieties & orientations
Query: Polar Bear

Results: Polar bears in various orientations + white sheep
Query: Grizzly Bear

Results: Bears in various colors, varieties & orientations

VGG16/3bit hash
Query: Orange Cat
Results: Cats in various colors, varieties & orientations
Query: Giraffe
Results: Giraffes various colors, varieties & orientations

Giraffes in different orientations

COCO_train2014_0000000007040.jpg  COCO_train2014_00000000510611.jpg  COCO_train2014_0000000326475.jpg  COCO_train2014_000000068894.jpg

COCO_train2014_000000068996.jpg  COCO_train2014_0000000281161.jpg  COCO_val2014_0000000517029.jpg  COCO_train2014_0000000399558.jpg
Future Work

● Text -> Image Search: Type in text phrase, then convert into a point in the same high-dimensional space as the images
● Deployment w/ kubernetes
Appendix
History

2005: No source control!

2010: Source control & continuous builds, yay! .. but not for ML :( 

2017: Great tools! .. still have a way to go for ML.
Distance Measures

Distance between two points in N dimensional space:

Euclidian

Cosine

Manhattan
Naive Implementation

- Feature Generator: TensorFlow Serving VGG16/FC6
- Database: PostgreSQL
- Frontend: Flask
- Deployment: Docker & Kubernetes
VGG16

Tensorflow Implementation

103 images/s on CPU

711 images/s on Tesla V100 GPU
Reverse image search in PostgreSQL using vector operations: rows are CNN image vectors. Euclidian Distance exhaustive search against query image. This is as opposed to the naive solution of doing exhaustive euclidian search in memory.