Computer Vision

DATA.ML.300, 5 study credits

Esa Rahtu
Laboratory of Signal Processing, Tampere University

Content based image retrieval

What we would like to be able to do?

- Query by example
- Given:
  - An example **query image** illustrating the users needs
  - A very large dataset of images
- Task:
  - **Rank** all images in the dataset according to relevance to the query

Difference to classification

Query: This chair

Results from dataset classified as “chair”

Classification

Difference to classification

Query: This chair

Results from dataset ranked by similarity to the query

Retrieval

Many tasks can be casted as retrieval

No need to train separate classes for new categories
Avoid the need of defining exact category borders

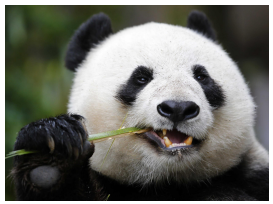
Examples:

- Face recognition (compare query face to dataset of know people)
- Place recognition (find similar image to determine location)

The retrieval pipeline

The retrieval pipeline

Query



The retrieval pipeline

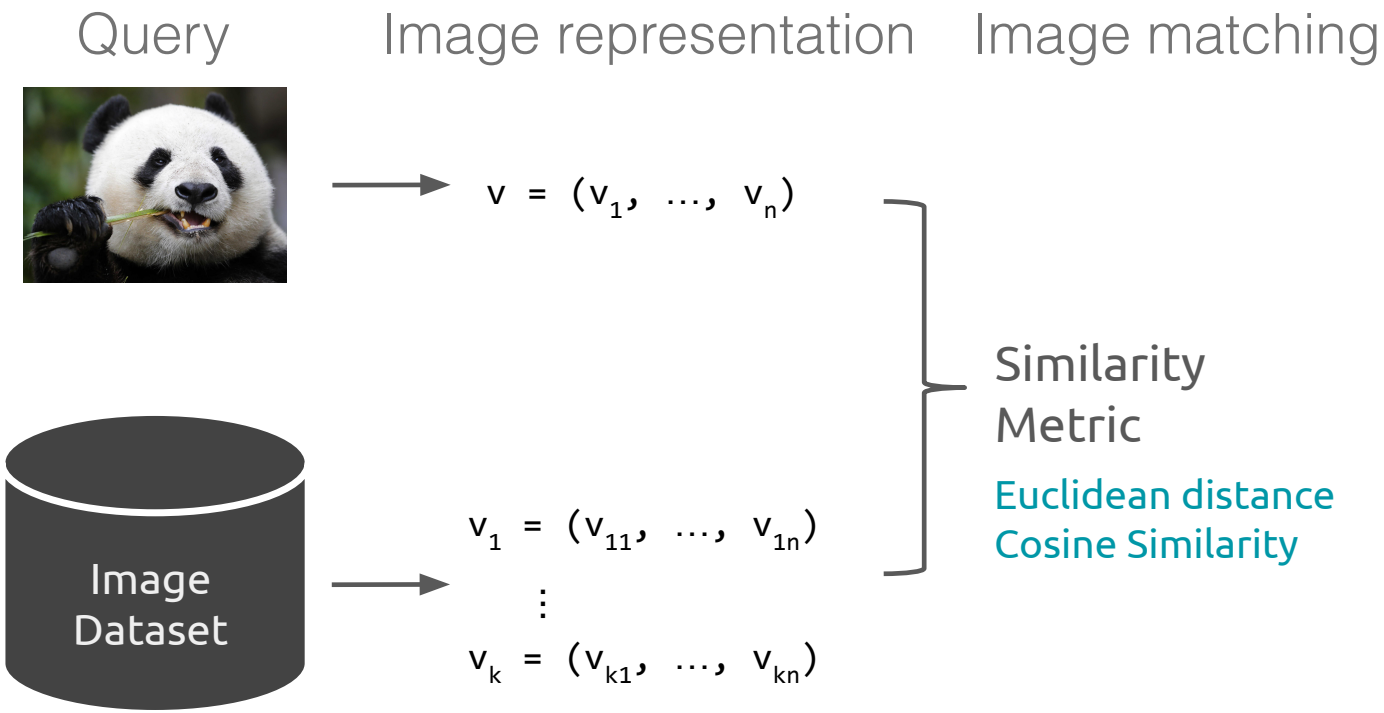
Query

Image representation

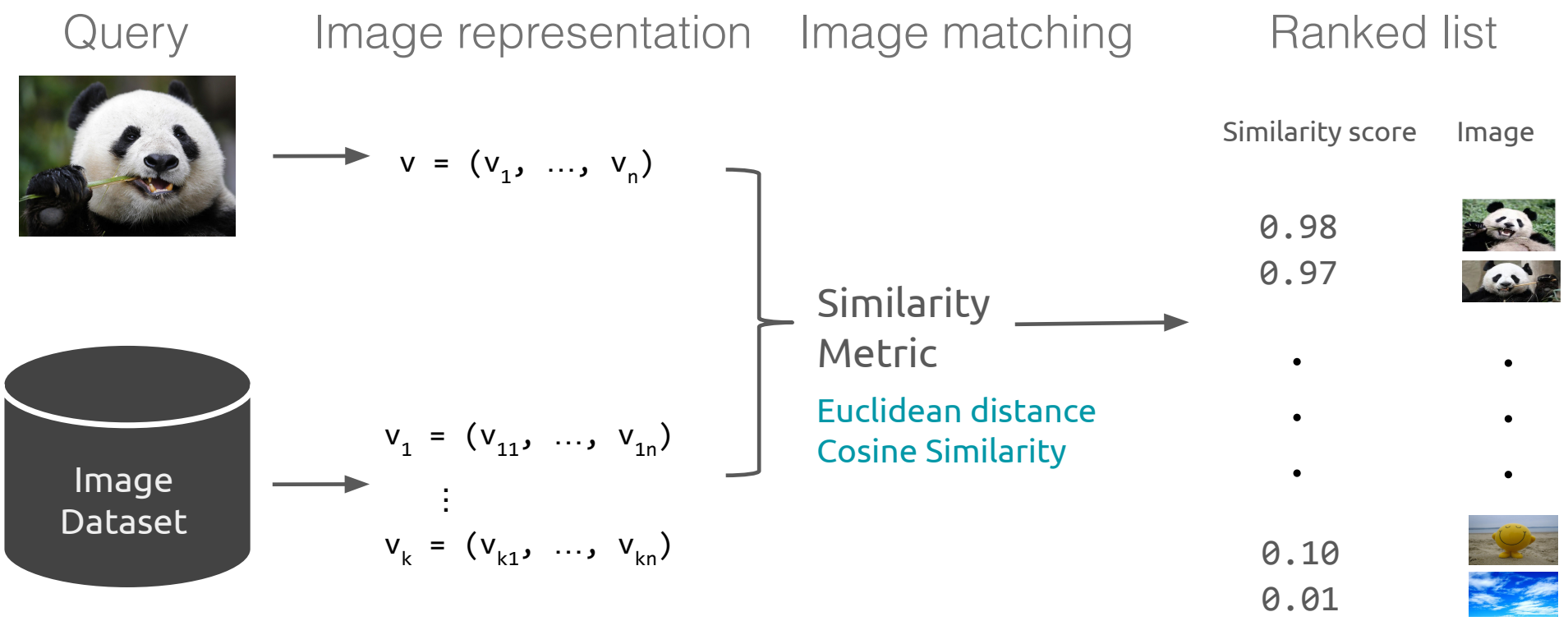
$$v = (v_1, \ldots, v_n)$$

$$v_1 = (v_{11}, \ldots, v_{1n})$$
$$\vdots$$
$$v_k = (v_{k1}, \ldots, v_{kn})$$

The retrieval pipeline

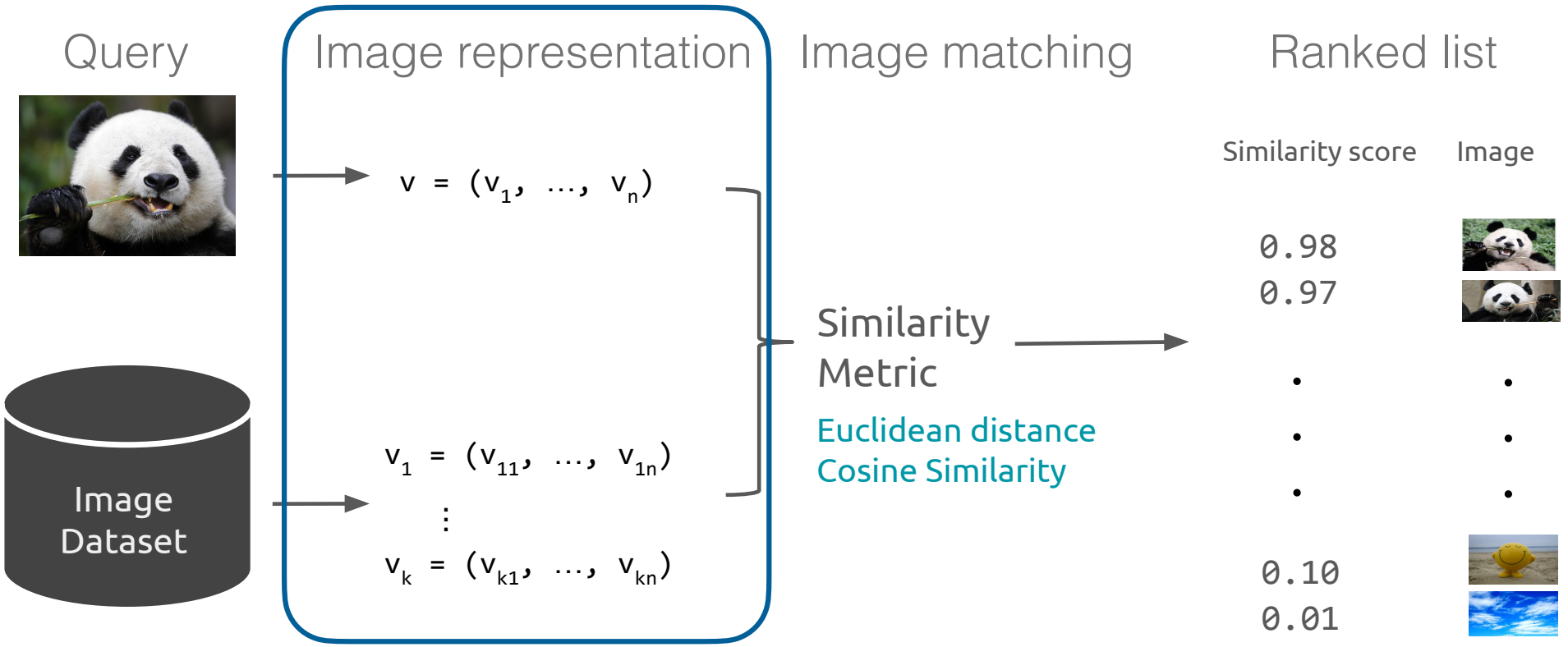


The retrieval pipeline



The image representation

The retrieval pipeline



Motivation

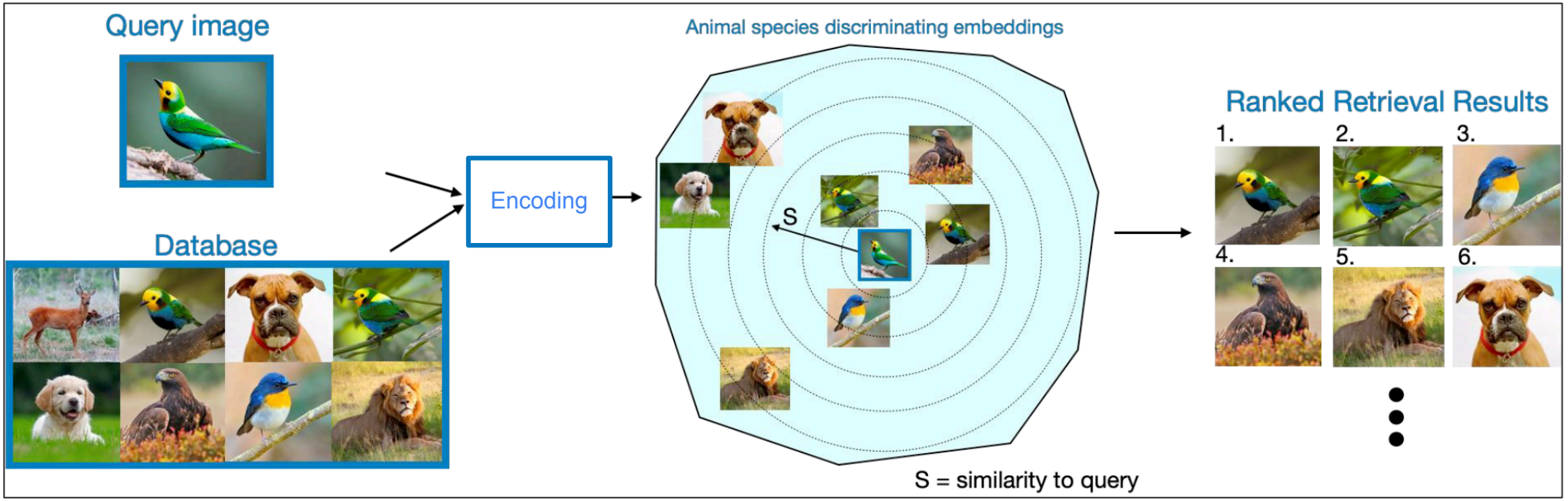
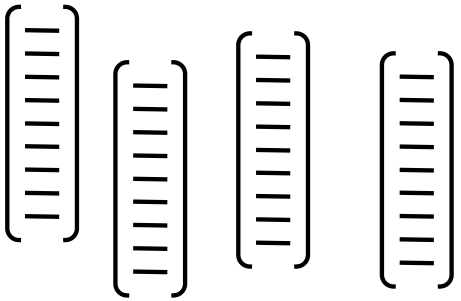


Figure: A. Zisserman

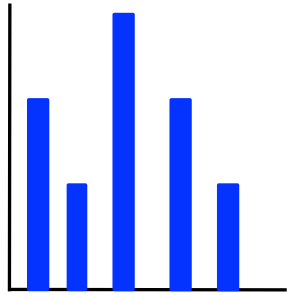
Classical approach

Detect variable number of local features (e.g. SIFT)

Compute descriptor (e.g. SIFT)

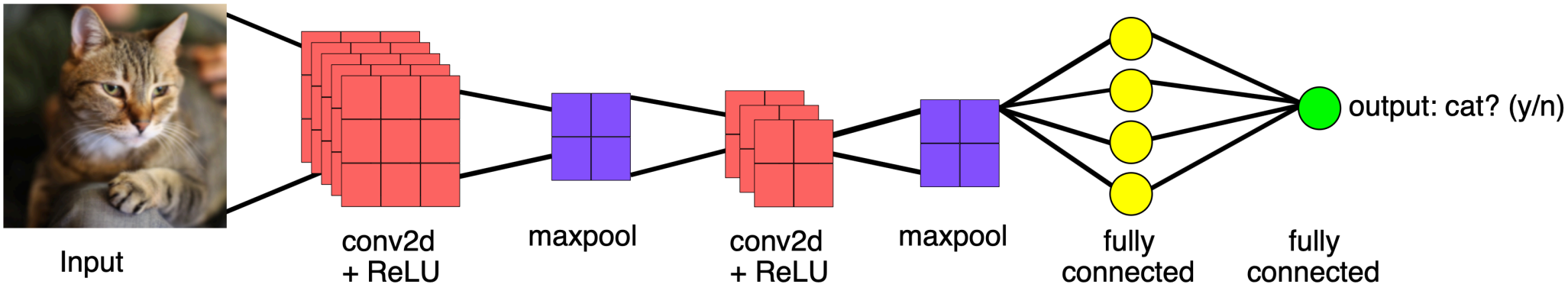


Quantise to form an image level descriptor



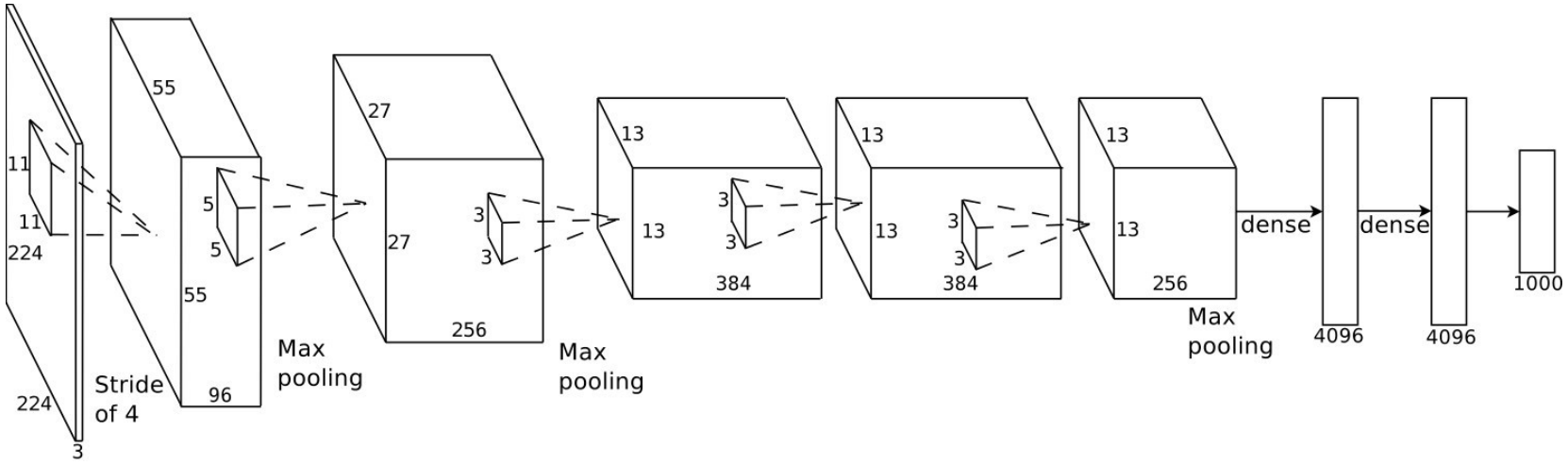
CNNs and retrieval

- CNNs are state-of-the-art in classification
- Could they be used for retrieval?



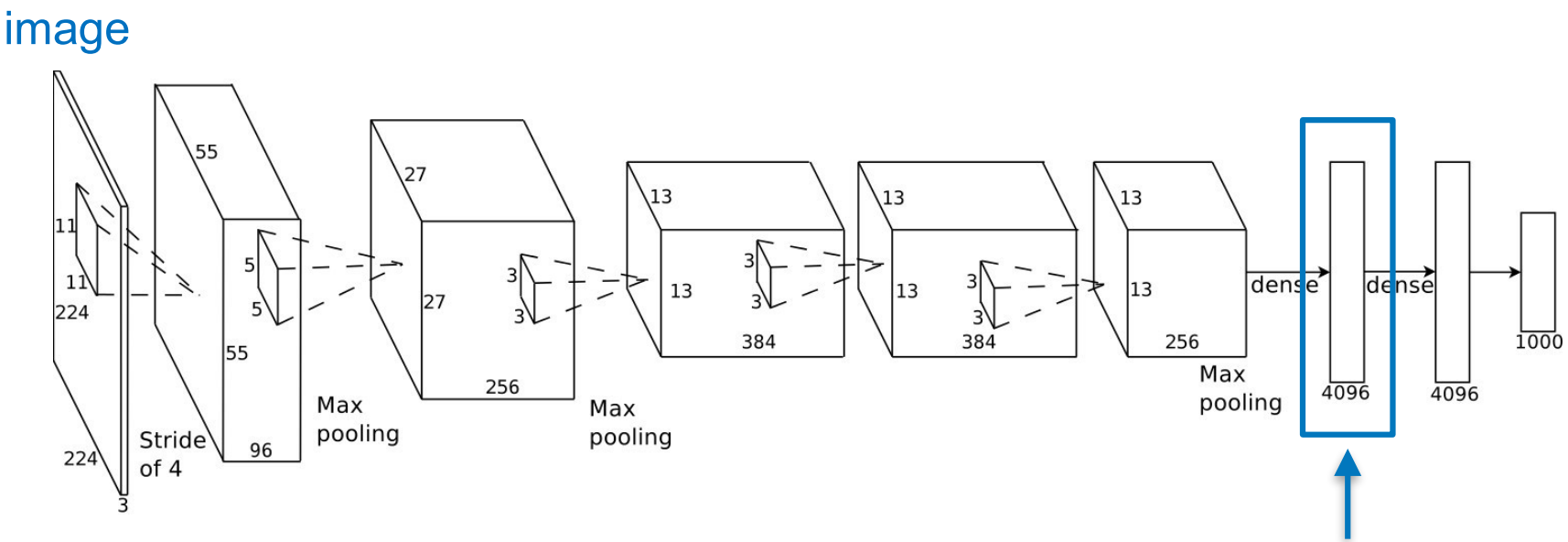
Using off-the-self CNN representations

image



AlexNet (Krizhevsky et al. 2012)

Using off-the-self CNN representations



AlexNet (Krizhevsky et al. 2012)

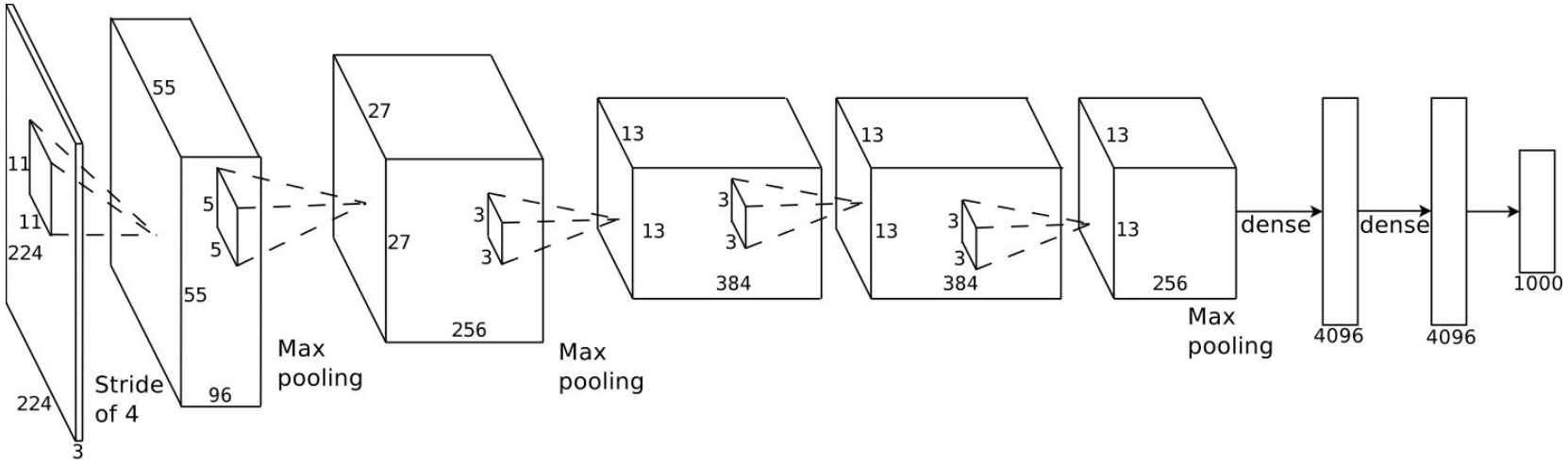
Using off-the-self CNN representations

- Babenko et al. [1]
  - FC7 layer features (4096D)
  - Euclidean distance
  - Slightly outperform traditional SIFT baseline after fine-tuning
- Razavian et al. [2]
  - Extracts features from several sub windows (sliding window)
  - Excellent results, but computationally impractically heavy

[1] Babenko et al, Neural codes for image retrieval, 2014, CVPR
[2] Razavian et al., CNN features off-the-shelf: an astounding baseline for recognition, 2014, CVPR

Using off-the-self CNN representations

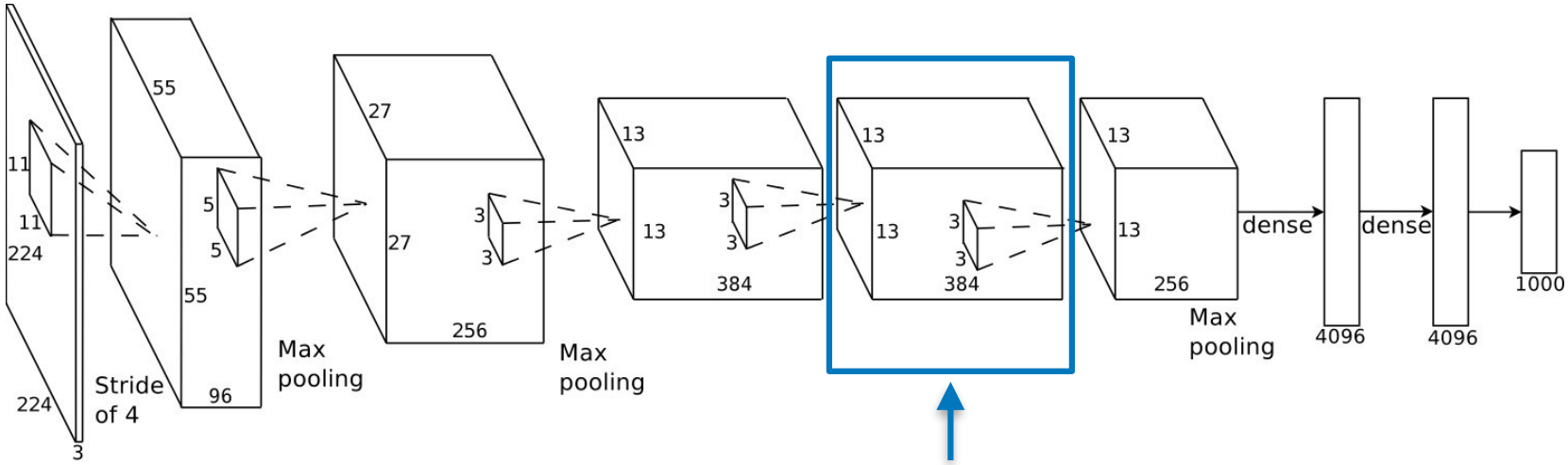
image



Could we obtain spatial information without explicit cropping small windows?

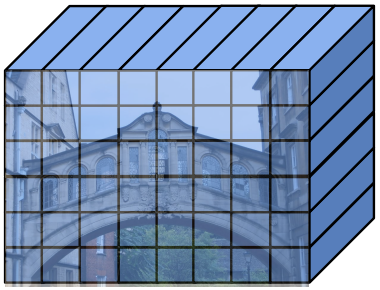
Obtaining descriptors from conv layers

image

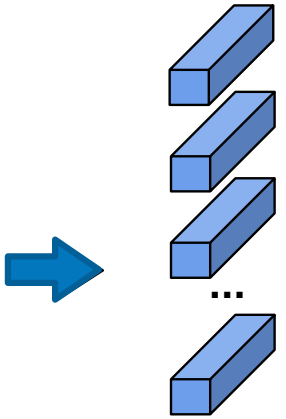


Obtaining descriptors from conv layers

Input image



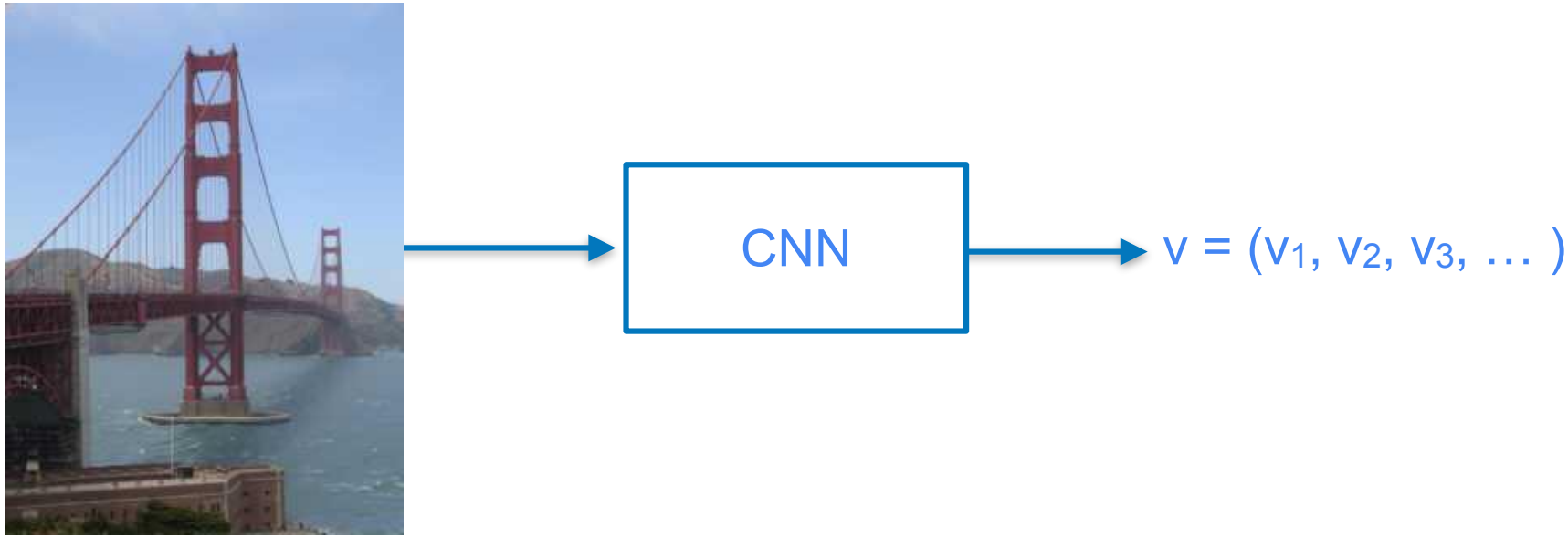
Conv Layer:
N feature maps of
dimension HxW



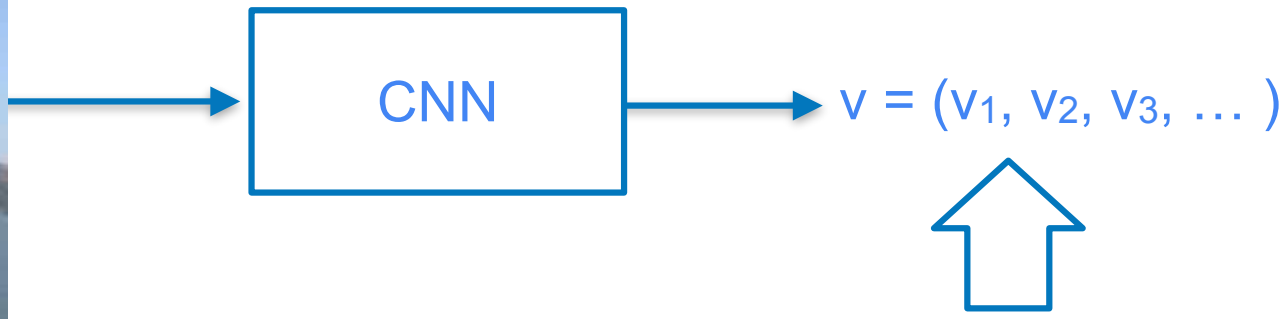
Local CNN features
HxW descriptors
of dimension N

Mohedano et al., Bags of local convolutional features for scalable instance search, ICMR 2016

Learning the descriptors

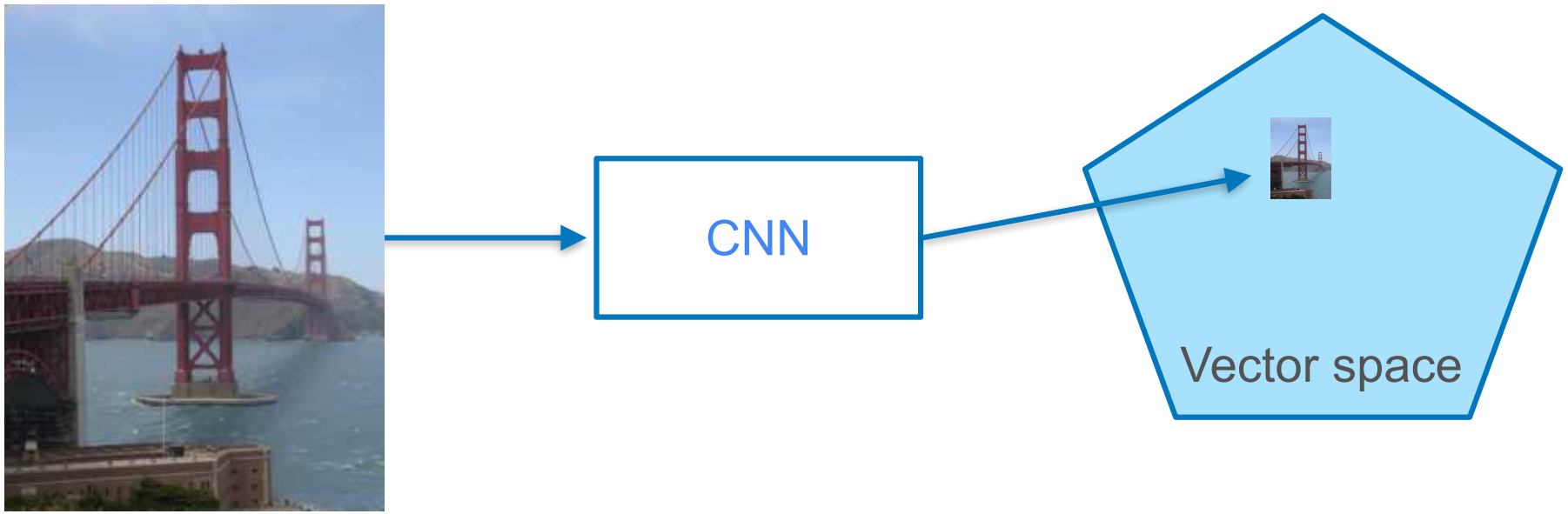


Learning the descriptors

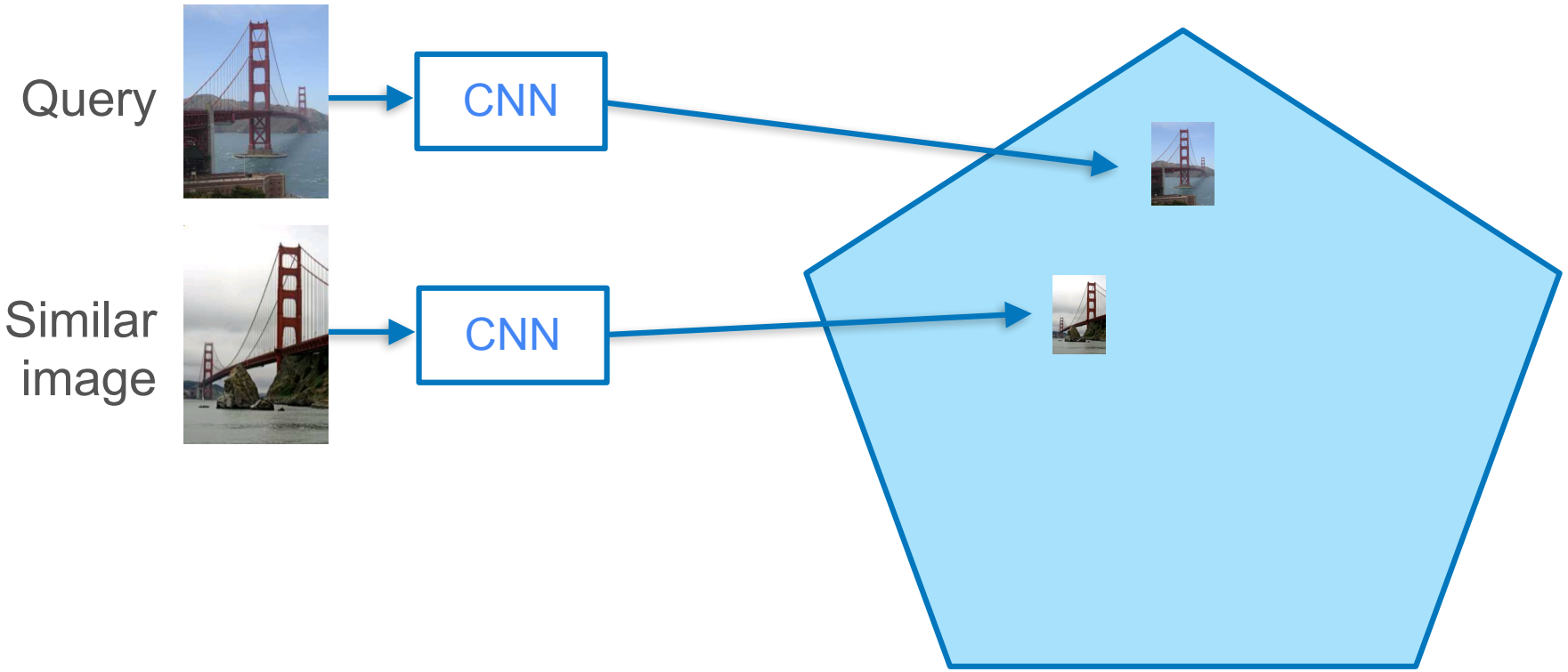


Unlike in classification, there is no ground truth for vector v

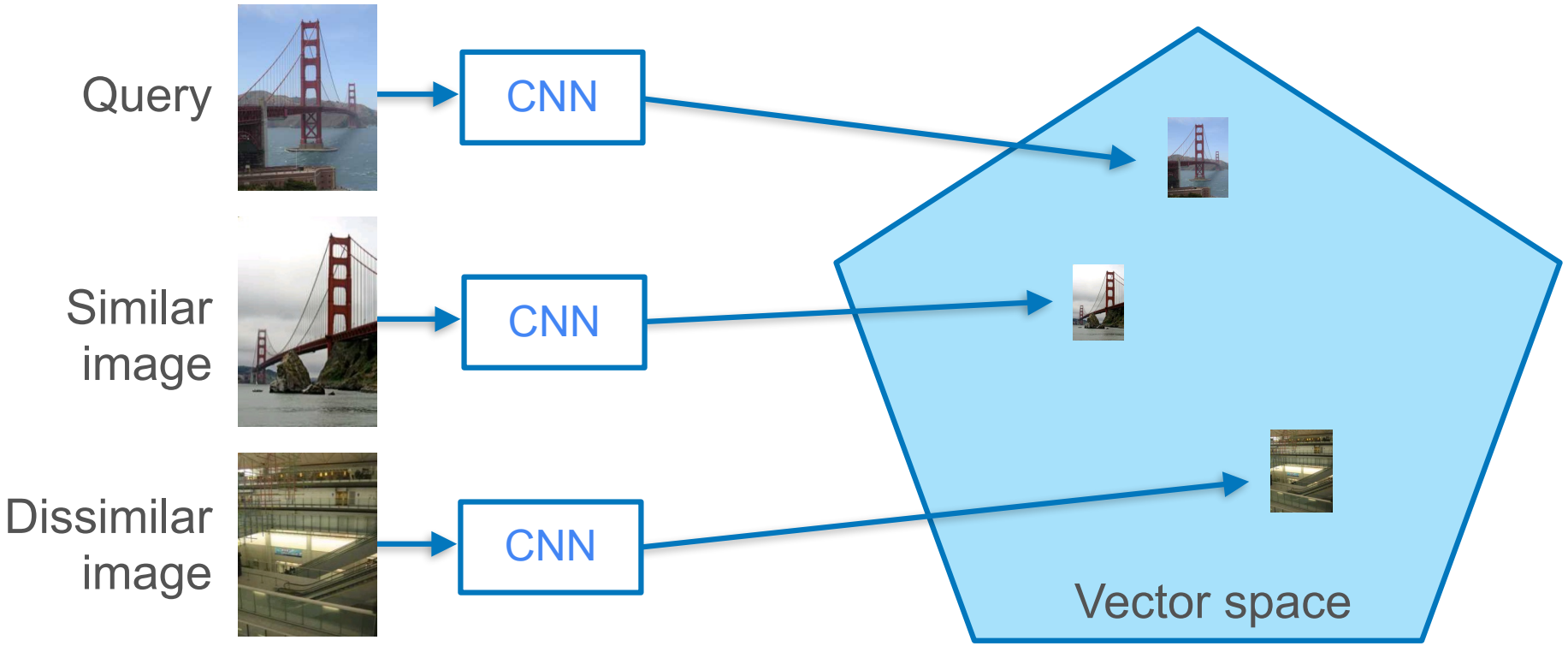
Learning the descriptors



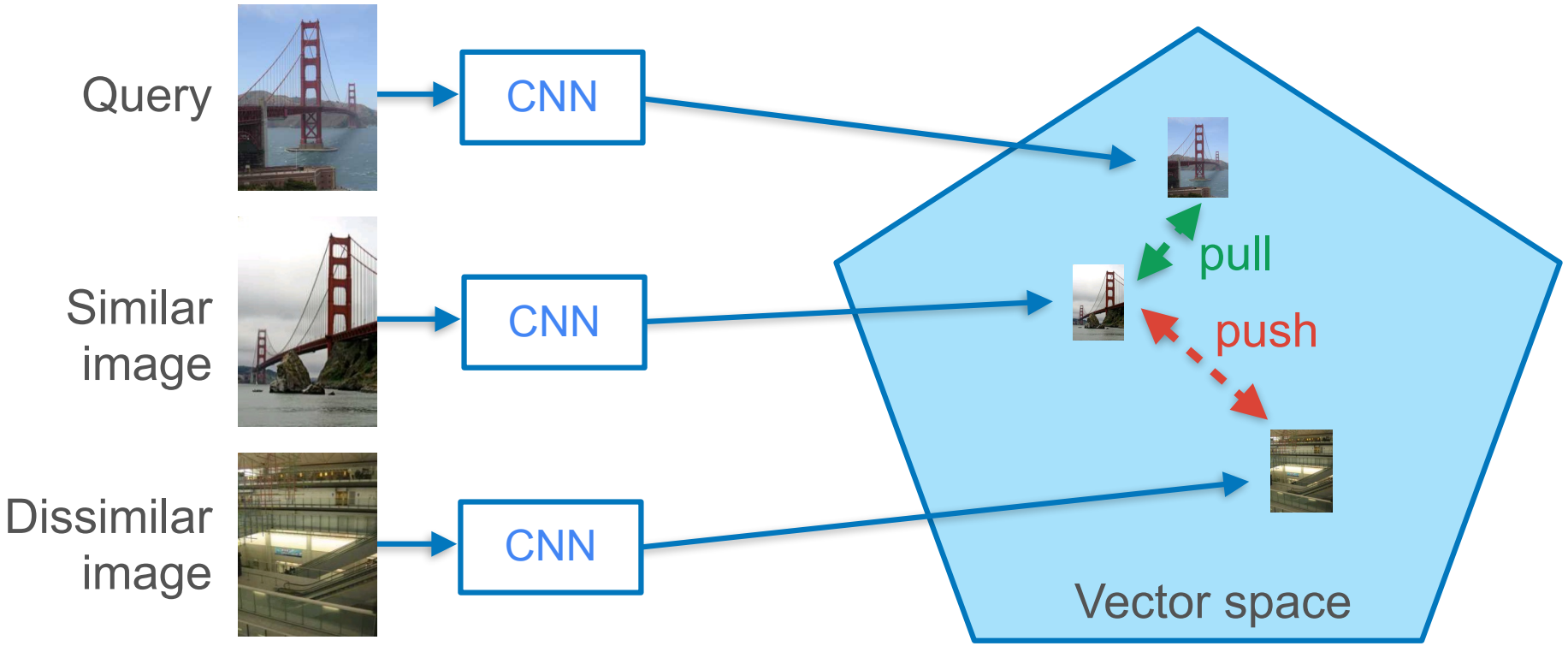
Learning the embedding



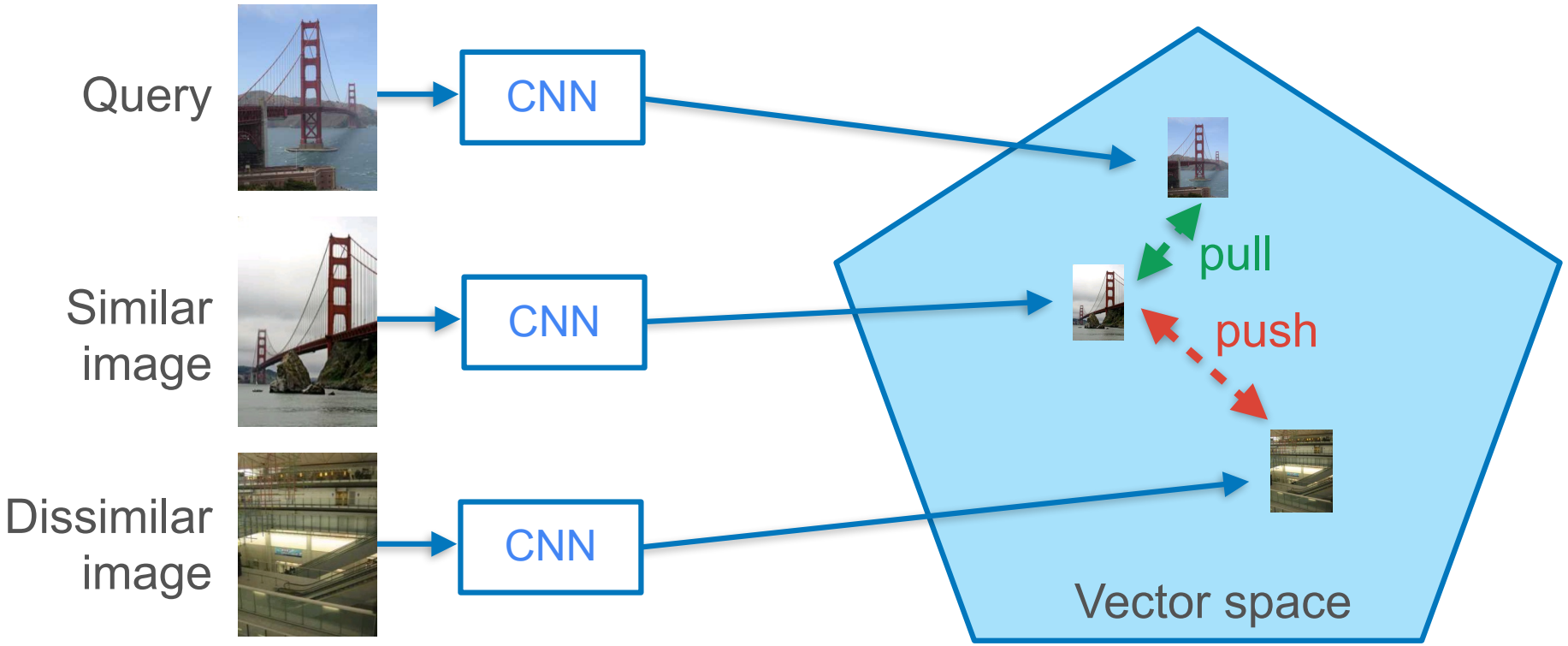
Learning the embedding



Learning the embedding

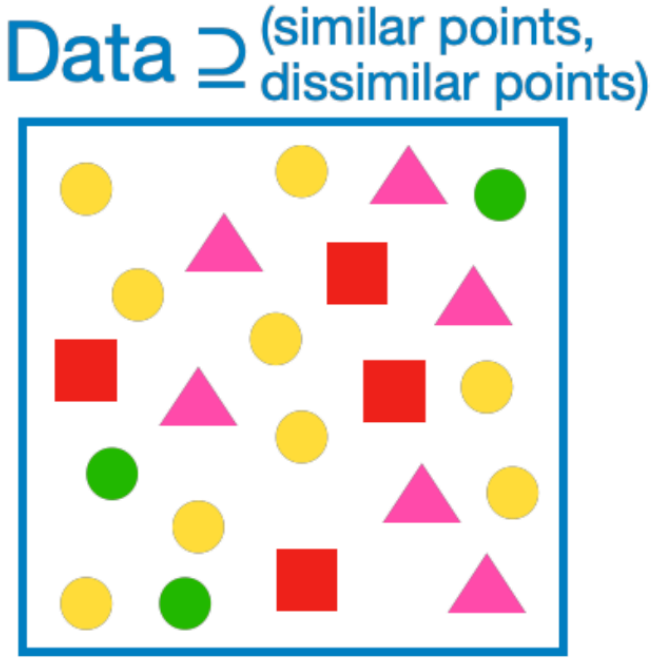


Learning the embedding



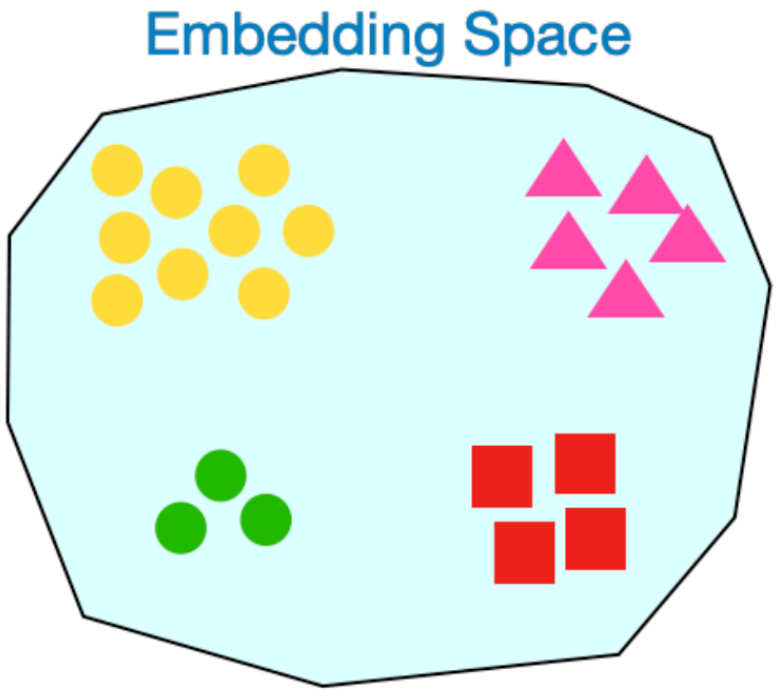
Implement a loss function to learn this from data

Learning the embedding



Nearest neighbours are not necessarily similar

Mapping



Nearest neighbours can be retrieved from this space

Figure: A. Zisserman

Learning the embedding

Contrastive loss function:

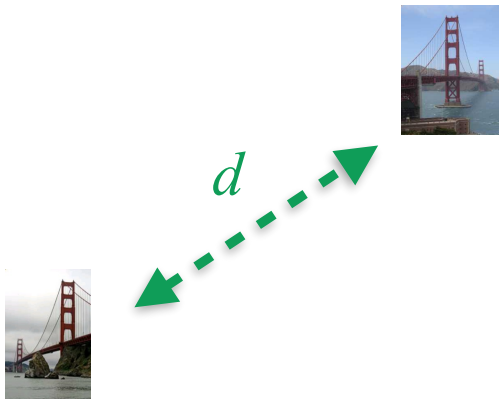
$$\text{loss}(d, y) = \frac{1}{2} y d^2 + (1 - y) \frac{1}{2} \max(0, m - d)^2$$

Where:

- d is the distance between two samples (e.g. Euclidean distance)
- y is the similarity label of the sample pair (1 if similar, 0 if not)
- m is the margin parameter

Learning the embedding

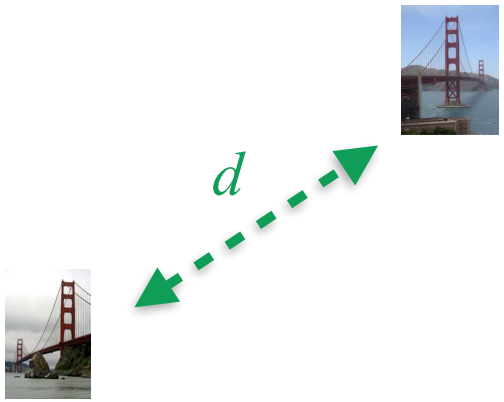
Similar (y=1)



$$\text{loss}(d, 1) = \frac{1}{2}d^2$$

Learning the embedding

Similar (y=1)

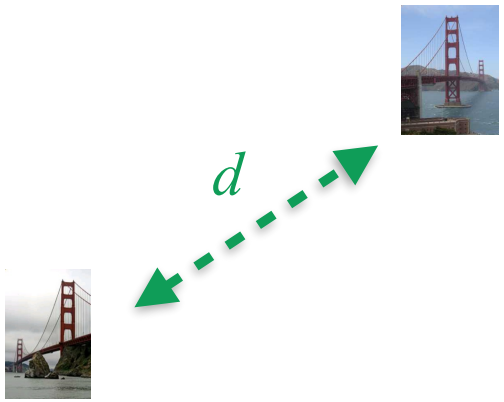


$$\text{loss}(d, 1) = \frac{1}{2}d^2$$

Similar images are pulled closer together by minimizing their distance.

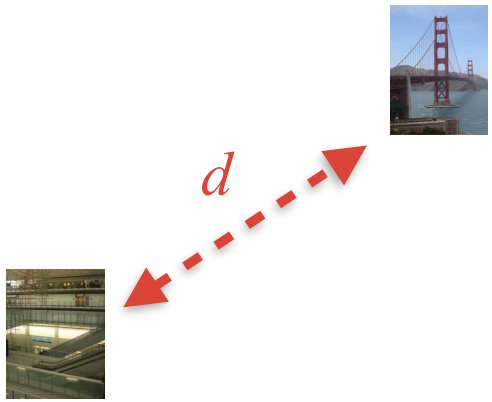
Learning the embedding

Similar (y=1)



$$\text{loss}(d, 1) = \frac{1}{2}d^2$$

Dissimilar (y=0)

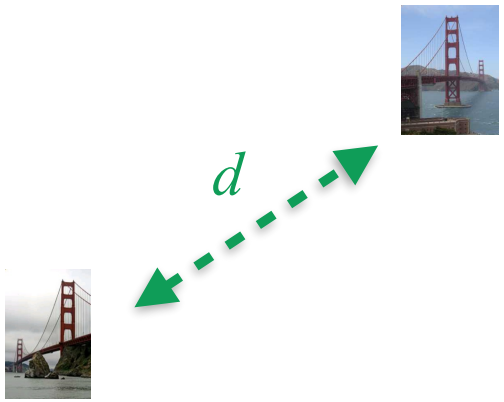


$$\text{loss}(d, 0) = \frac{1}{2}\max(0, m - d)^2$$

Similar images are pulled closer together by minimizing their distance.

Learning the embedding

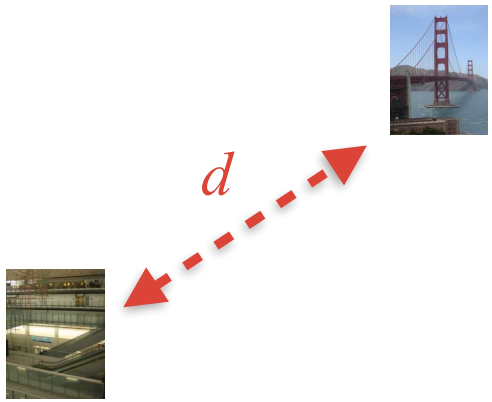
Similar (y=1)



$$\text{loss}(d, 1) = \frac{1}{2}d^2$$

Similar images are pulled closer together by minimizing their distance.

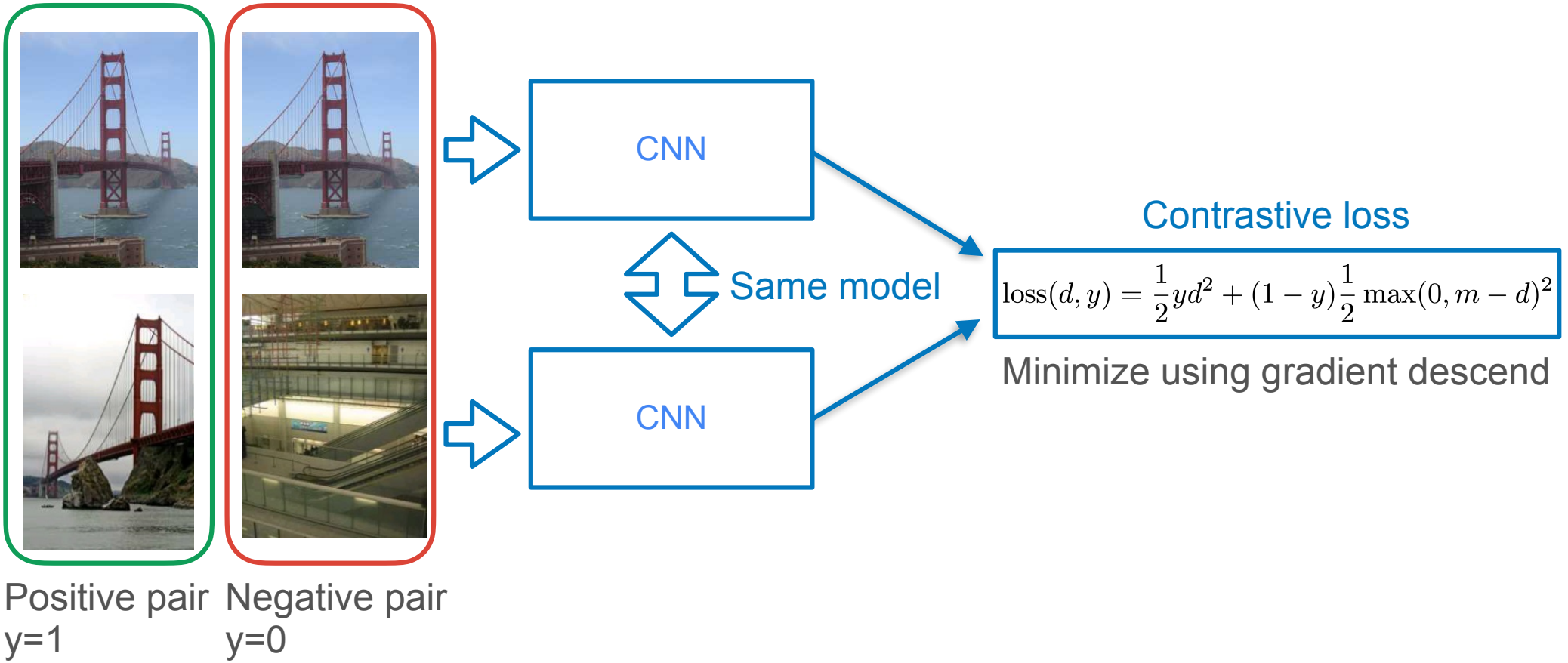
Dissimilar (y=0)



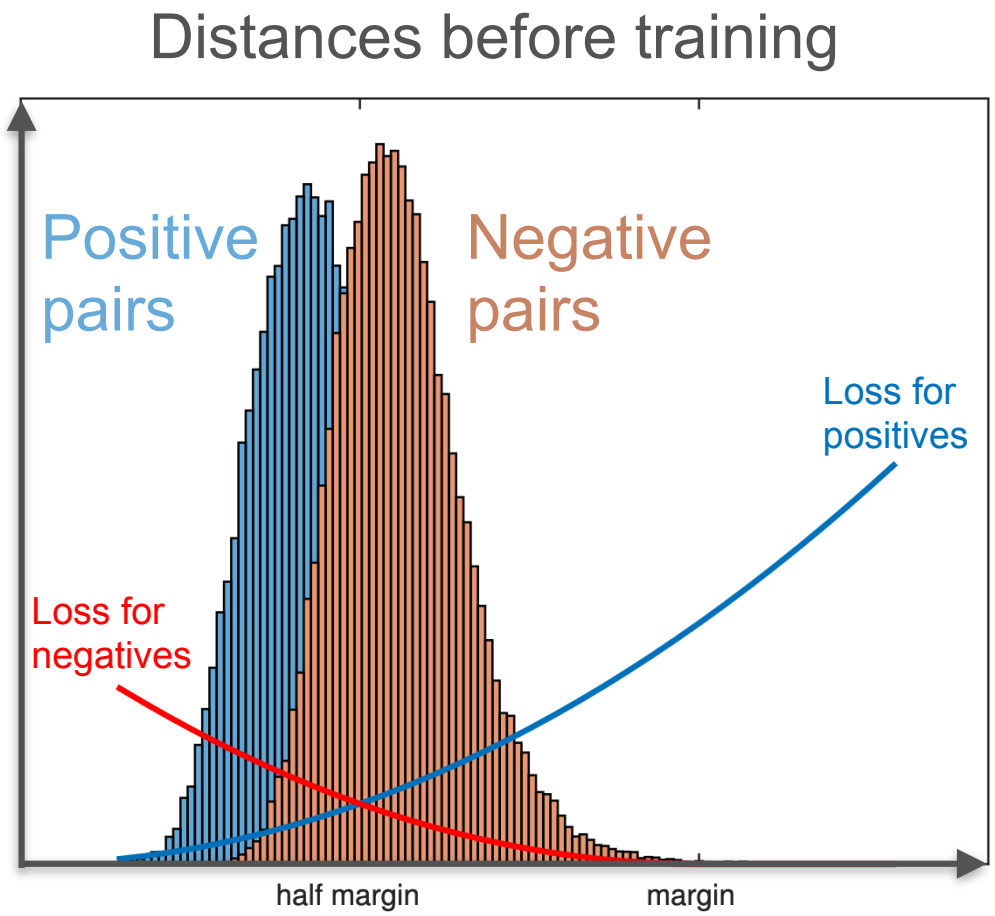
$$\text{loss}(d, 0) = \frac{1}{2}\max(0, m - d)^2$$

Dissimilar images are pushed apart until the distance is greater than the margin m

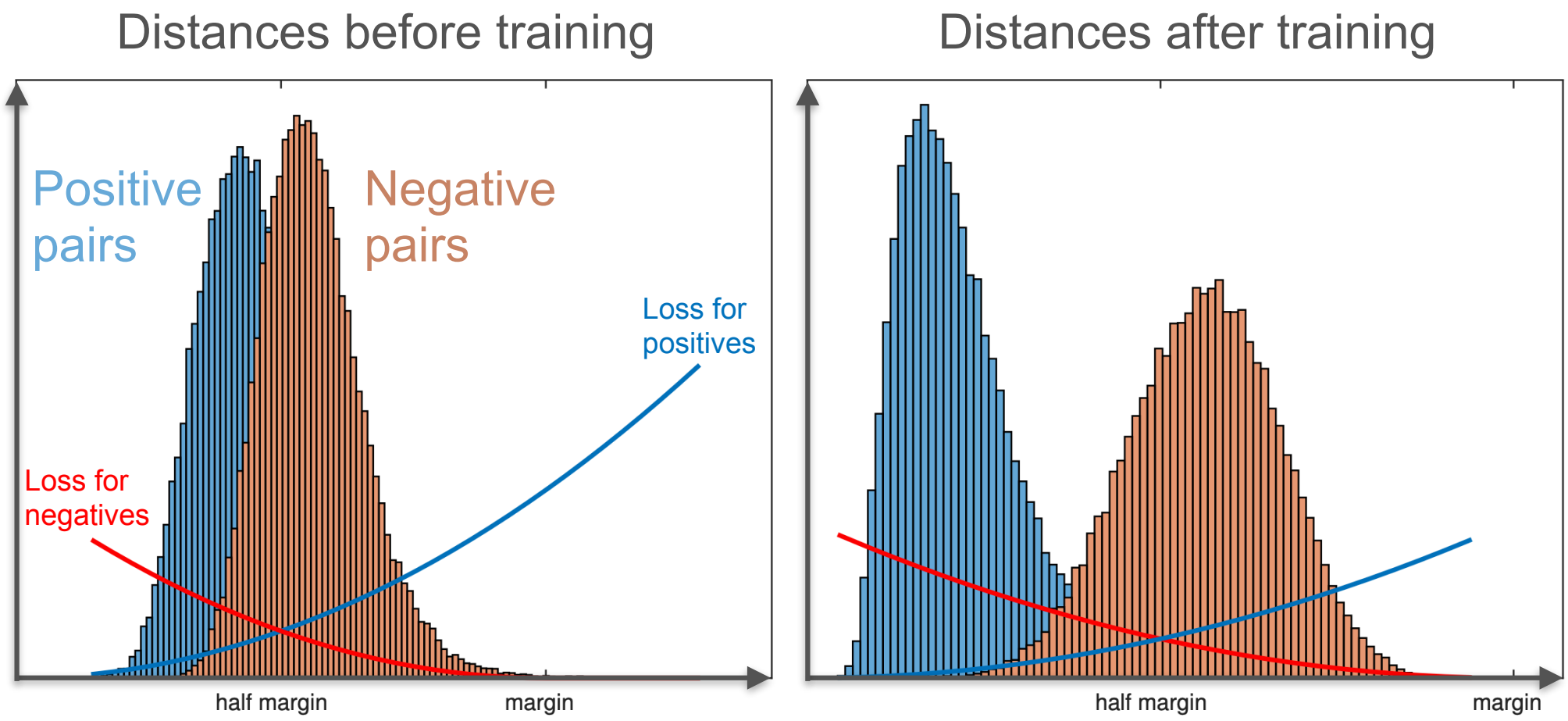
Learning the embedding



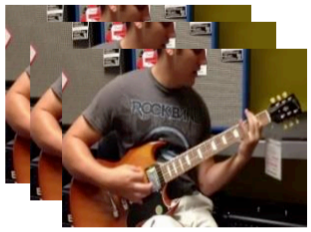
Learning the embedding



Learning the embedding



Applicable to mixed modalities



Visual encoder

A man playing an electric guitar

Text encoder

Contrastive loss

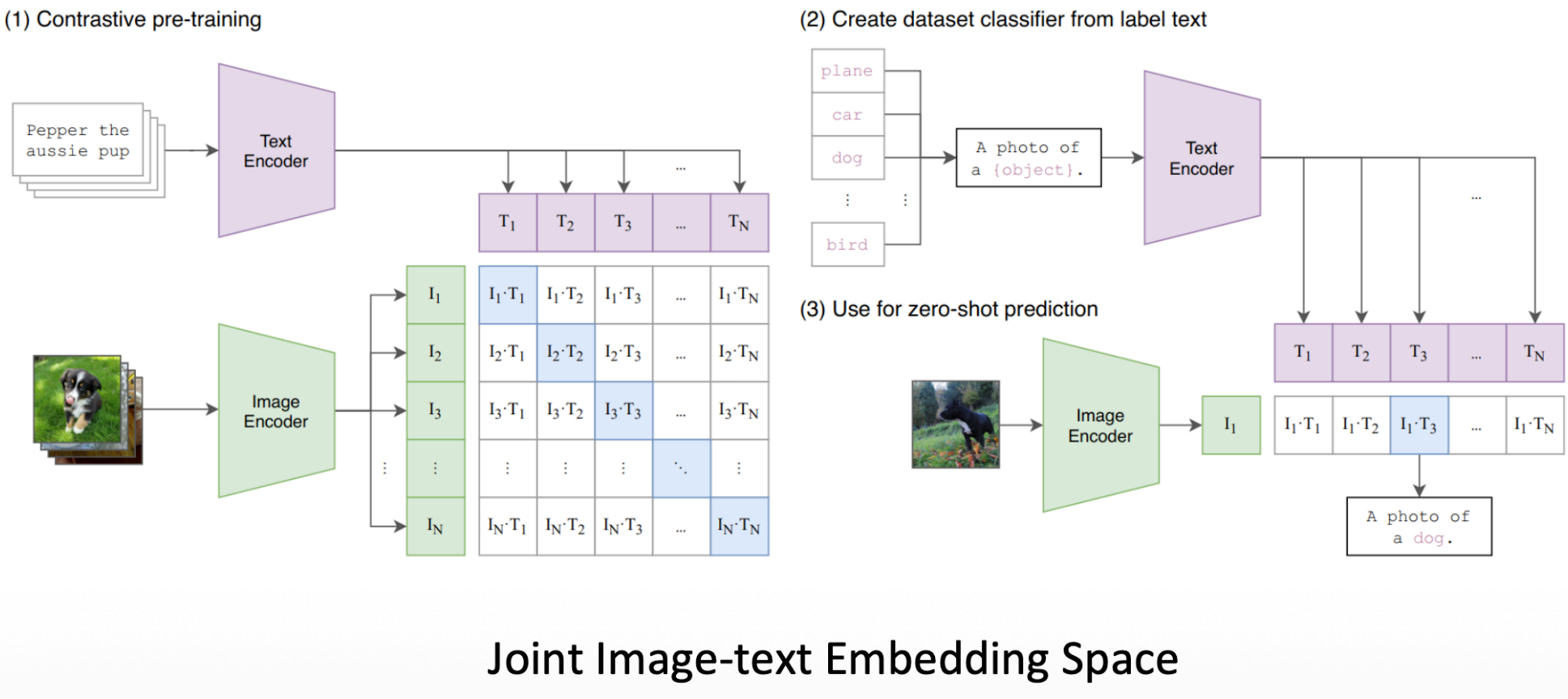
small distance if corresponding

large distance if not corresponding

Train on paired image-caption data

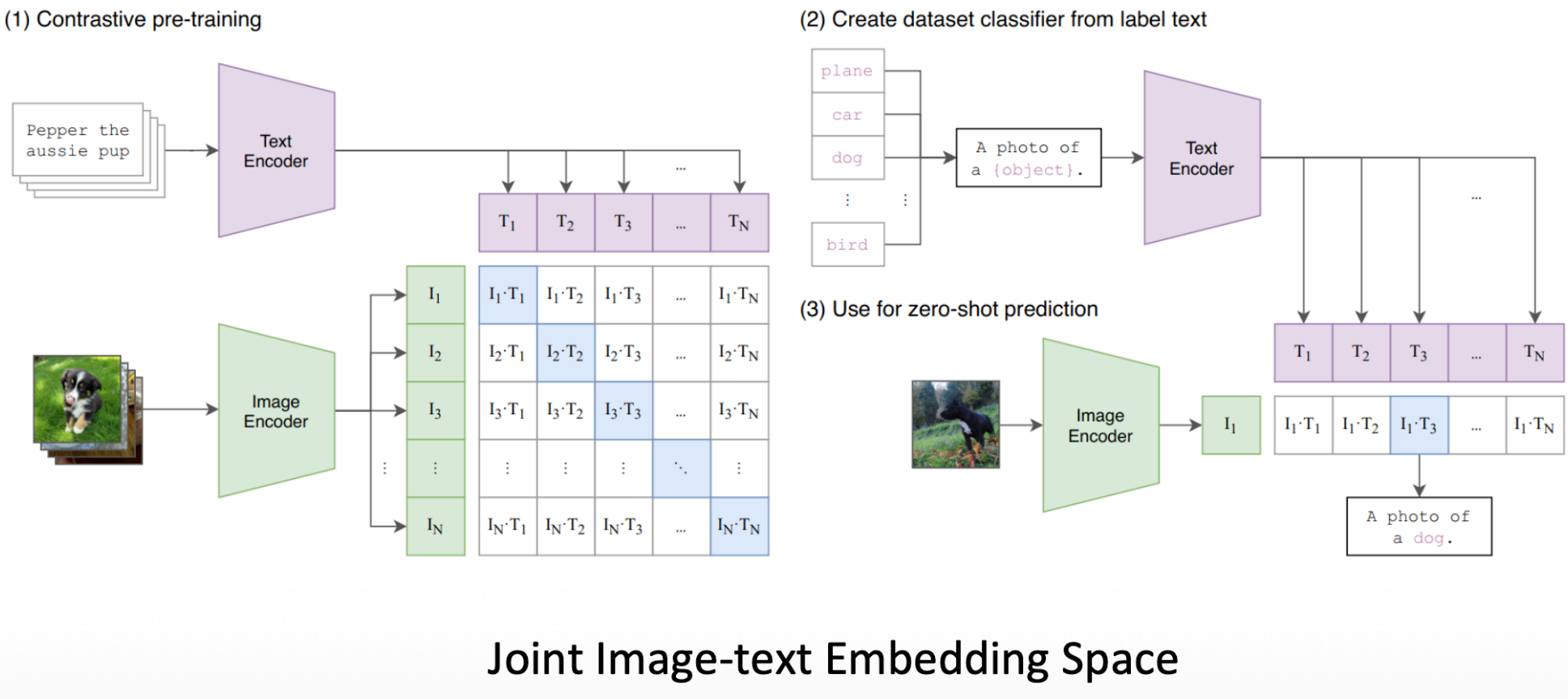
Figure: A. Zisserman

Contrastive Language-Image Pre-training (CLIP) Dual Encoder



OpenAI, https://openai.com/index/clip/

Contrastive Language-Image Pre-training (CLIP) Dual Encoder



OpenAI, https://openai.com/index/clip/

Shazam music search

HOW DOES SHAZAM WORK?

This is a question we get often asked. Here is a quick summary of the three main steps involved from the moment you Shazam until the magic happens.

Let's say you are in a shop and you like the music you're hearing. Start the app and tap the Shazam button.

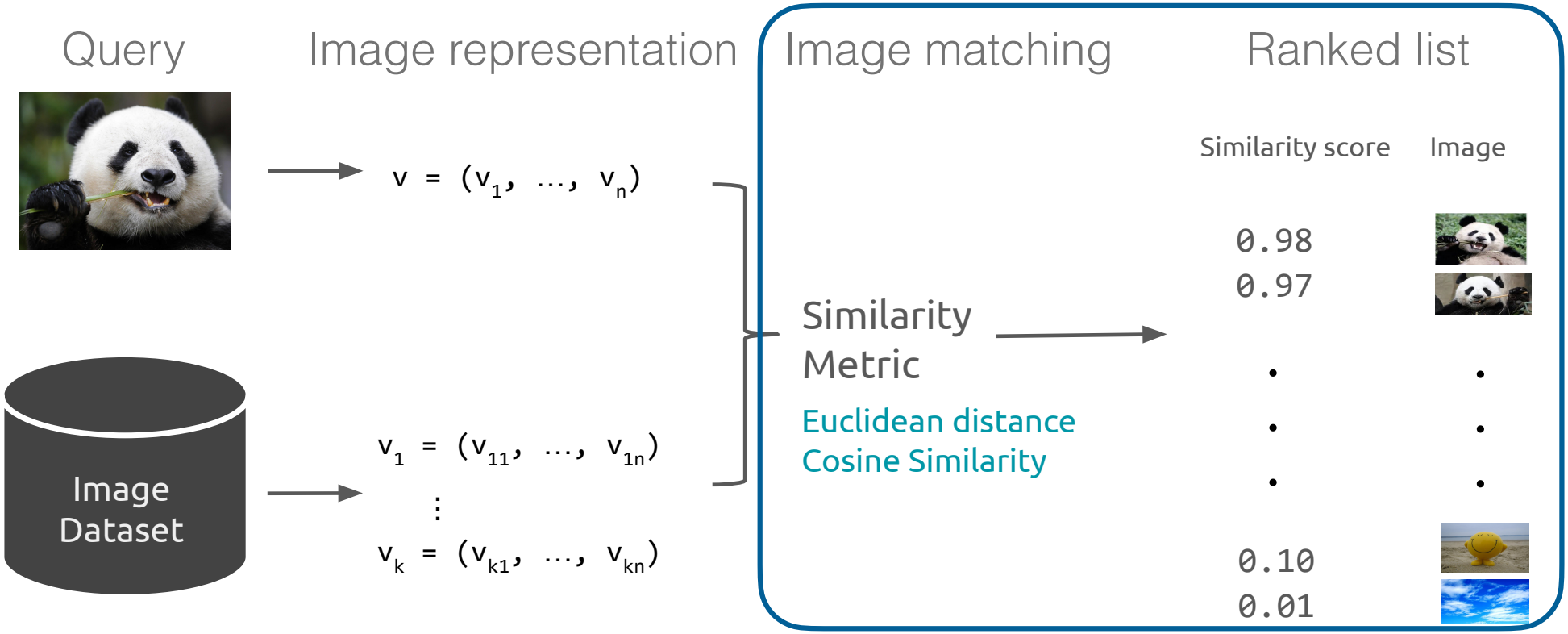
A digital fingerprint of the audio is created and, within seconds, matched against Shazam's database of millions of tracks.

You are then given the name of the track and the artist and information such as lyrics, video, artist biography, concert tickets and recommended tracks.

https://www.shazam.com/

Obtaining similarity and ranking

The retrieval pipeline



Calculating similarities

- Euclidean distance or cosine similarity between feature vectors

$$d_{L2}(x, a) = \sqrt{\sum_{i=1}^{n}(x_i - a_i)^2}$$

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

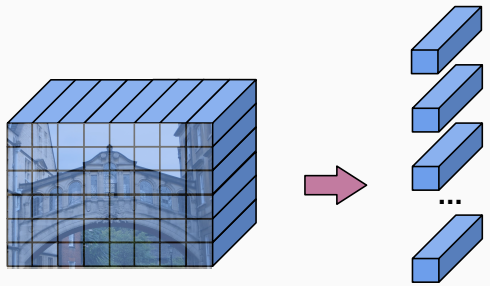
Calculating similarities

- Euclidean distance or cosine similarity between feature vectors

$$d_{L2}(x,a) = \sqrt{\sum_{i=1}^{n}(x_i - a_i)^2}$$

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Exhaustive evaluation unfeasible for large datasets
- What if we have multiple descriptors per image?



How large is large?

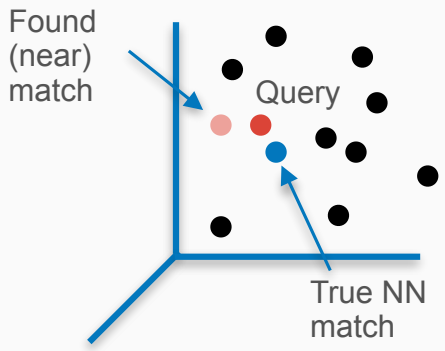
- Image search, e.g Meta
  - 1 billion images = 1 billion descriptors (e.g. 128D based on CNNs) per day
- Video search: thousands of hours of video
  - Billions of audio and video descriptors
  - 720,000 hours of videos (82 years) of videos are uploaded to YouTube every day [1]
- Music search: e.g. Shazam
- The order of magnitude is millions to billions

[1] YouTube official blog

Credits: Zisserman A.

Idea: approximate nearest neighbour

- Despite of efficient implementations NN search is still O(nd)
- Approximate methods improves speed, but is not guaranteed to find the true nearest neighbour

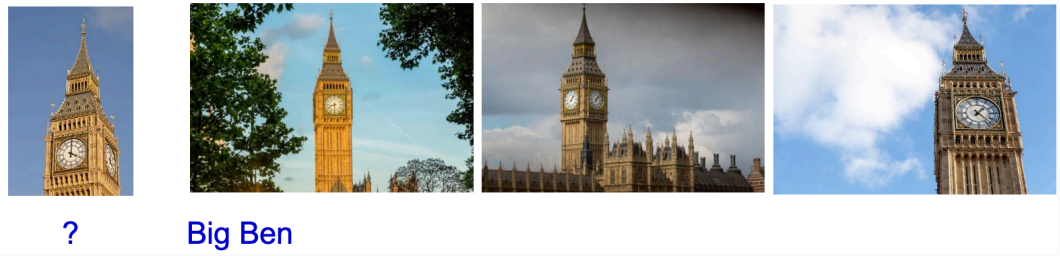


128D descriptor space

Credits: Relja A.

Idea: approximate nearest neighbour

- Despite of efficient implementations NN search is still O(nd)
- Approximate methods improves speed, but is not guaranteed to find the true nearest neighbour
- Is this acceptable?
  - Often there is no choice (use ANN or not = have Google or not)
  - Often it is good enough



Credits: Relja A.

Idea: Inverted file index

- For text documents, an efficient way to find all **pages** that contain a specific **word** is to use an index
- We want to find all **images** in which a **feature** occurs
- To apply the idea in practice, we need to map the features to "**visual words**"

Index

A

Abel, Micah 18
Adkins, Melinda 40, 55
Adndrick, Kayla 20
Akers, William 8, 51
Algebra, Advanced 34
Allen, Shyla 8
Allshouse, Damon 32
Anderson, Cassie 38, 59
Andrick, Kayla 20
Asterino, Thomas 66
Audia, Sidney 16

B

Baker, Catherine 36
Baker, Savannah 44, 69
Bartimus, Jerad 40
Basketball, Junior Varsity Boys' 50, 51
Basketball, Junior Varsity Girls' 50
Basketball, Varsity Boys' 48, 49, 52, 53
Basketball, Varsity Girls' 52
Baylor, Nicholas 14
Bell, Charles 22, 23
Benedum, Anastasia 16, 56
Bennett, Brittany 34, 50, 57
Bennett, Coach Greg 50
Bennett, Dylan 6
Brooks, Orry 34, 51
Brooks, Susan 46
Brown, Benjamin 14
Brown, Delante 6
Brown, Nikki 42
Brumage, Brett 30
Brumage, Brittiany 28
Brummage, Lindsey 18
Burks, Dylan 10
Burns, Tameka 10
Burton, Cherise 40, 52, 58, 67
Burton, Coach Ed 52
Burton, Julia 57
Burton, Linda 60
Burton, Lucille 18, 46
Butcher, Jack 34, 35
Butler, Alexis 14
Byrne, Devin 24

C

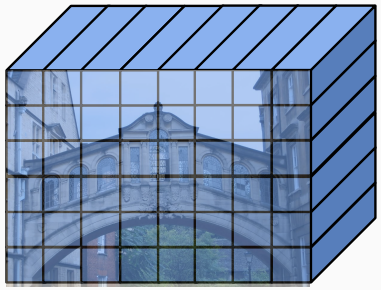
Canfield, Marisa 32
Carnes, Sara 14, 55
Carpenter, Phoenix 8
Centeno, Ashley 26
Cheerleaders, Junior Varsity 54, 55
Cheerleaders, Varsity 55
Clay, Jamie 6
Clevenger, Clayton 30
Club, Girls' 56, 57
Clutter, Jacob 20, 21
Cogar, Tyler 38
Cole, Anthony 10
Cole, Marcus 12
Denoon, Christian 30
Dent, Ashton 44
Dent, Bailee 12
des Mond, Adrian 68
Devalt, Jack 36, 37, 51
di Rosa, Rene 68
Dick, Jacob 8
Dick, Victoria 16
Dixon, Bethany 26
Donini, Jasmine 28, 29
Du Toit, Kim 68
Duckworth, Caleb 6
Duckworth, Mariah 10
Duckworth, Melyssa 36
Duckworth, Trenton 6
Dukich, Mikayla 32, 48
Duskey, David 26

E

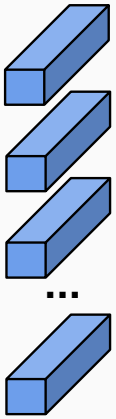
Echols, LC 34, 51
Eckles, Carson 30
Eddy, Brady 14
Efaw, Adrianna 26
Efaw, Breanna 36, 50, 54
Efaw, Coach Susie 54
Efaw, Danielle 14
Efaw, Hannah 42, 55, 57, 69
Elliott, Thomas 12
Eubank, Jacob 14, 15
Evan, Izaiha 28, 48, 49
Evans, Mason 32
Evanson, Jamie 42, 69
Evanson, Jenna 3, 5, 55, 69
Evanson, Jennifer 41

Obtaining visual words

Input image



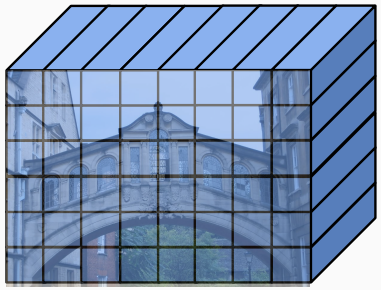
Conv Layer



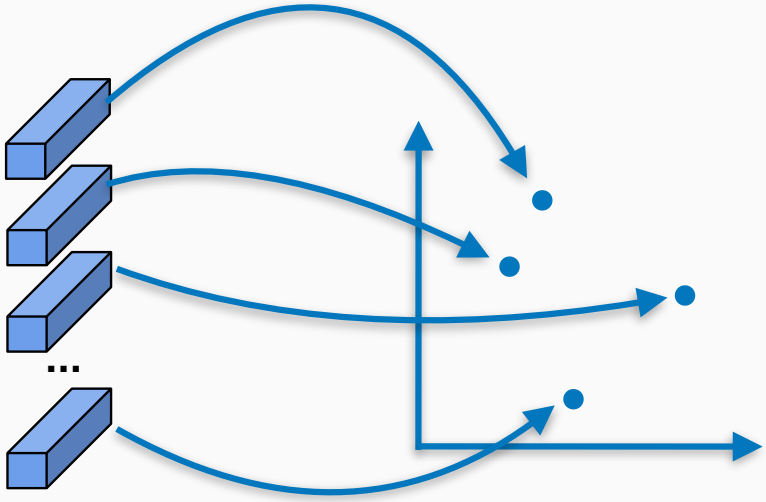
Local CNN descriptors of dimension N

Obtaining visual words

Input image



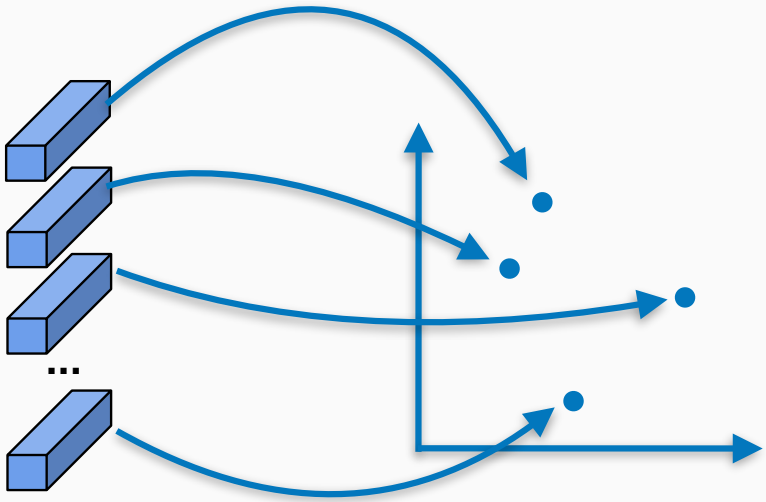
Conv Layer



Local CNN descriptors of dimension N

Represent as points in N dimensional feature space

Obtaining visual words

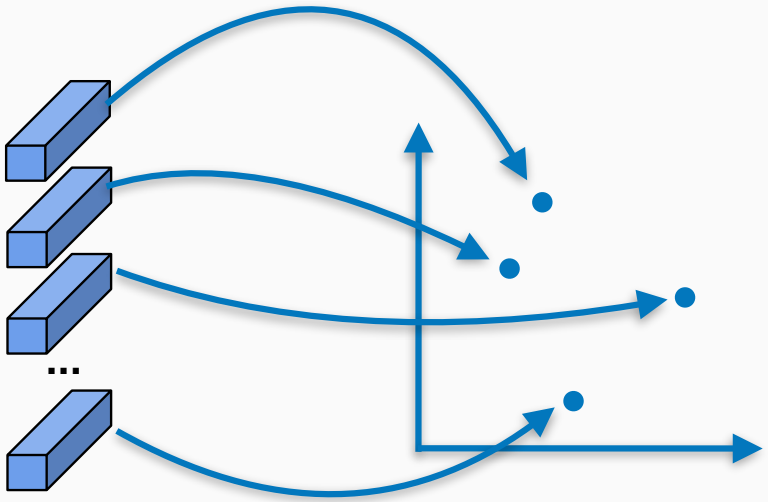


Local CNN descriptors of dimension N

Represent as points in N dimensional feature space

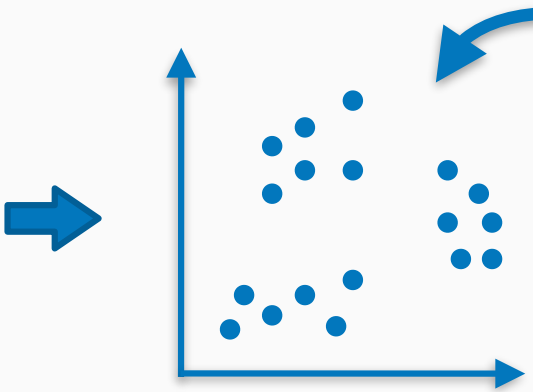
Training dataset

Obtaining visual words



Local CNN descriptors of dimension N

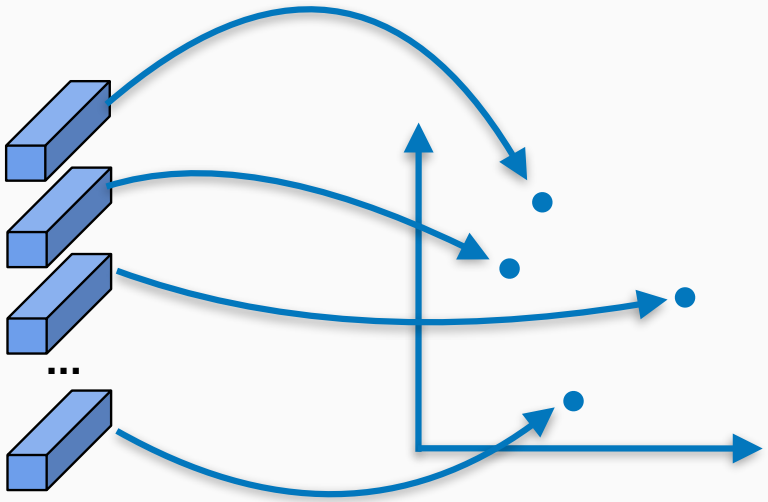
Represent as points in N dimensional feature space



Collect descriptor from the entire training set

Training dataset

Obtaining visual words



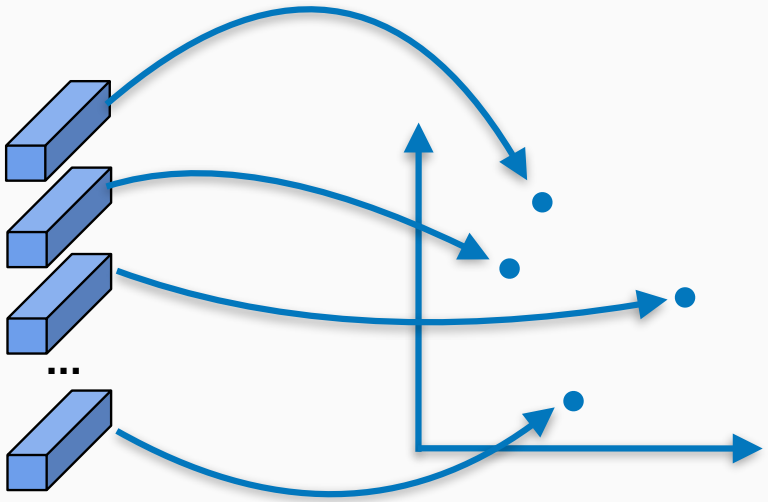
Local CNN descriptors of dimension N

Represent as points in N dimensional feature space

Quantise the feature space (clustering)

Training dataset

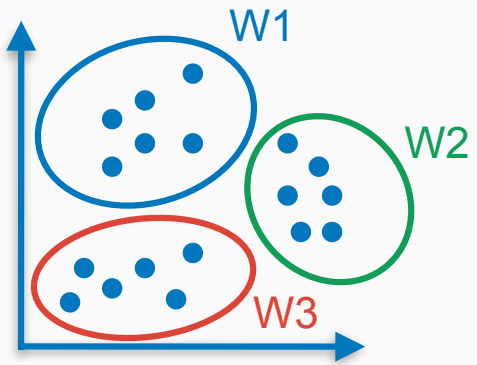
Obtaining visual words



Local CNN descriptors of dimension N

Represent as points in N dimensional feature space

Each cluster represents a “visual word” = **vocabulary**



Quantise the feature space (clustering)

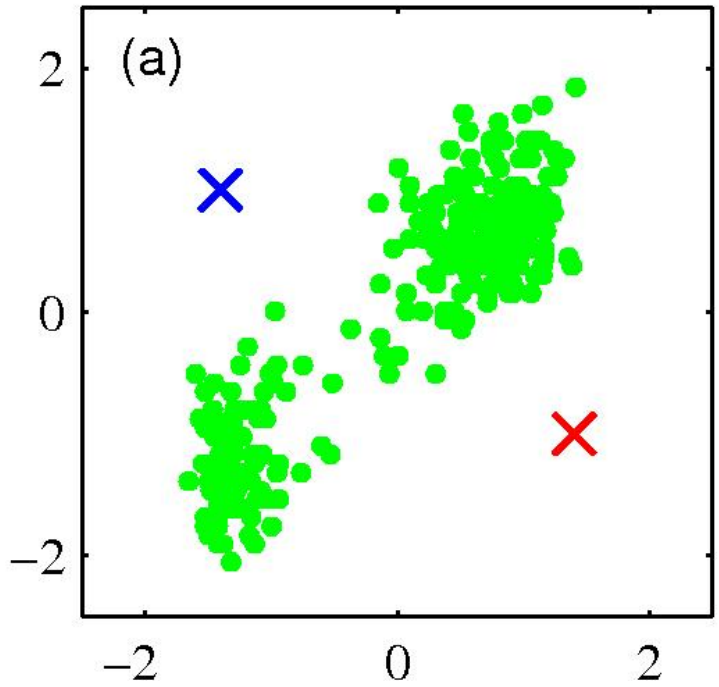
K-mean clustering

- Idea is to minimise sum of Euclidean distances between points x_i and their nearest cluster centres m_k

$$D(X,M) = \sum_{\text{cluster } k} \sum_{\substack{\text{point } i \text{ in} \\ \text{cluster } k}} (\mathbf{x}_i - \mathbf{m}_k)^2$$

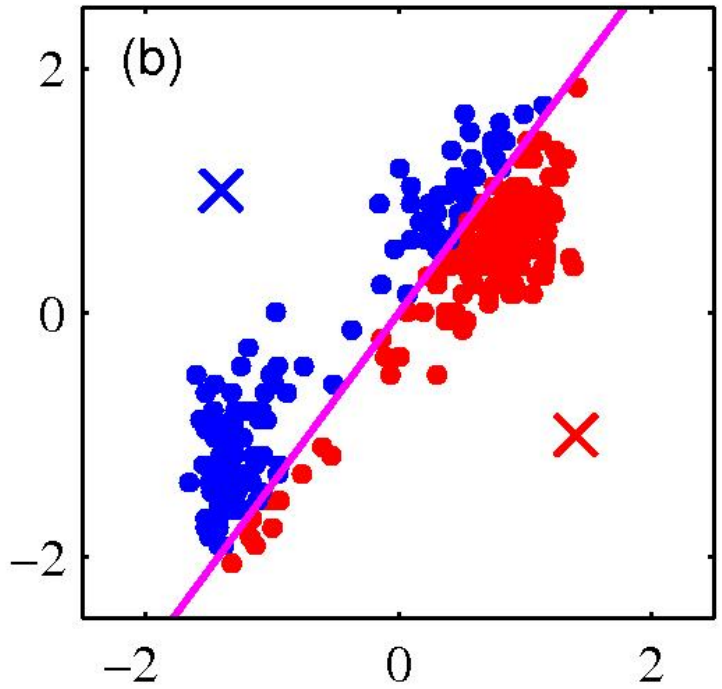
- Algorithm:
  1. Randomly initialise K cluster centres
  2. Assign all points to nearest cluster center
  3. Recompute cluster centres as the mean of all points assigned to it
  4. Return to 2 until convergence

K-means clustering: example



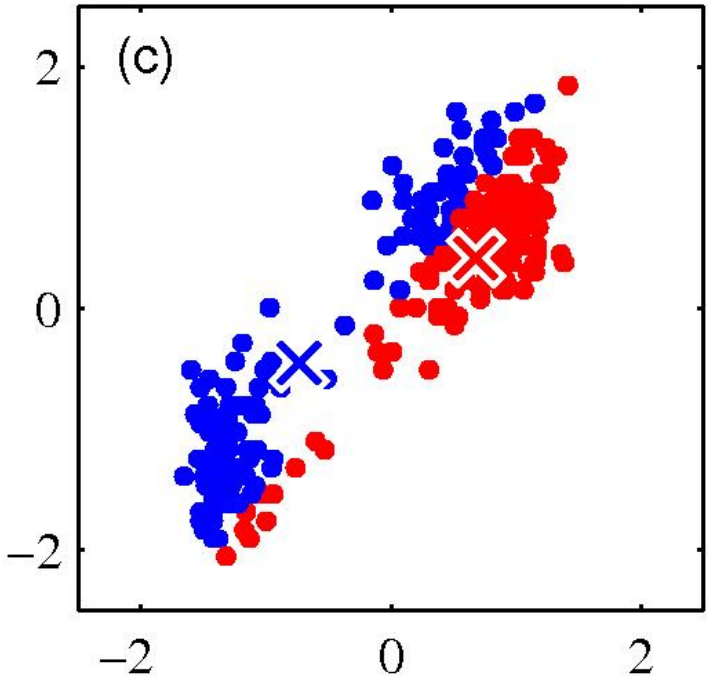
Initial state
2-dimensional space
K = 2

K-means clustering: example



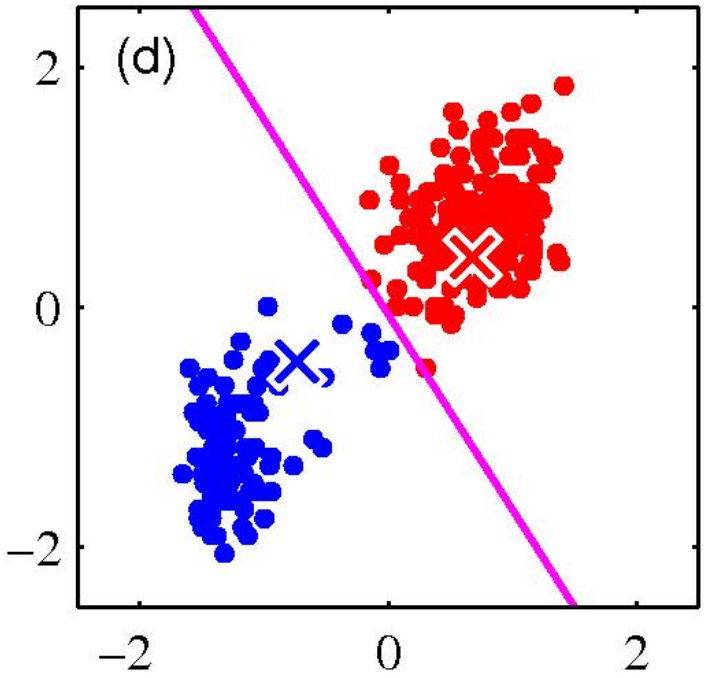
Result after step 3.
(assign data points to nearest cluster centre)

K-means clustering: example



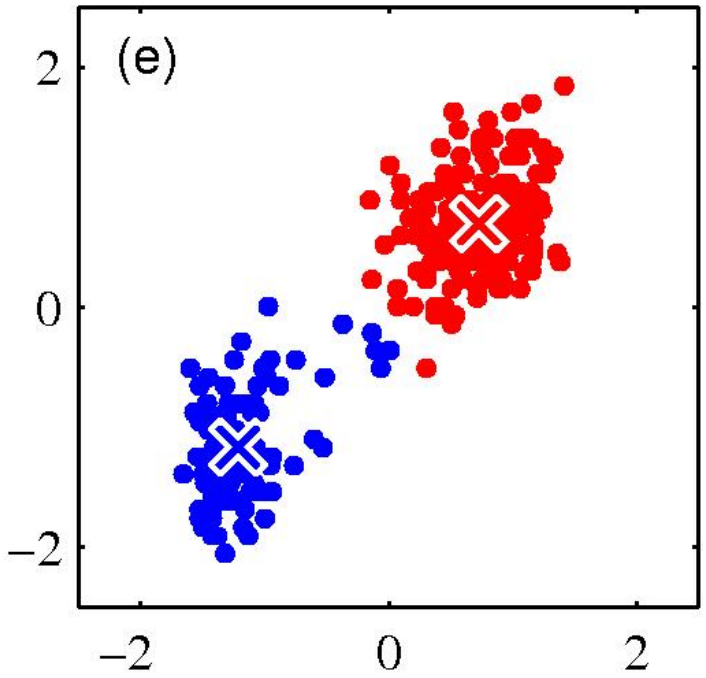
Result after step 4.
(move cluster centres
to the means of data points)

K-means clustering: example



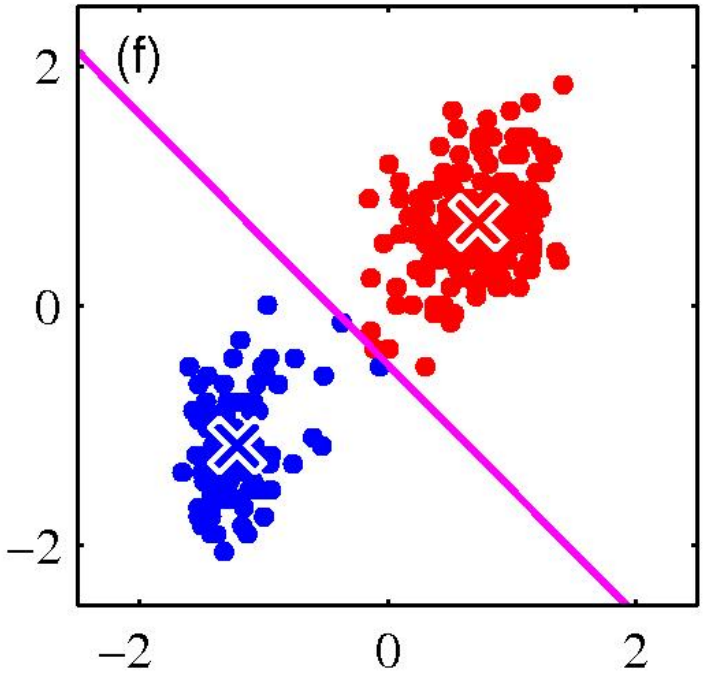
2nd iteration
result after step 3.

K-means clustering: example



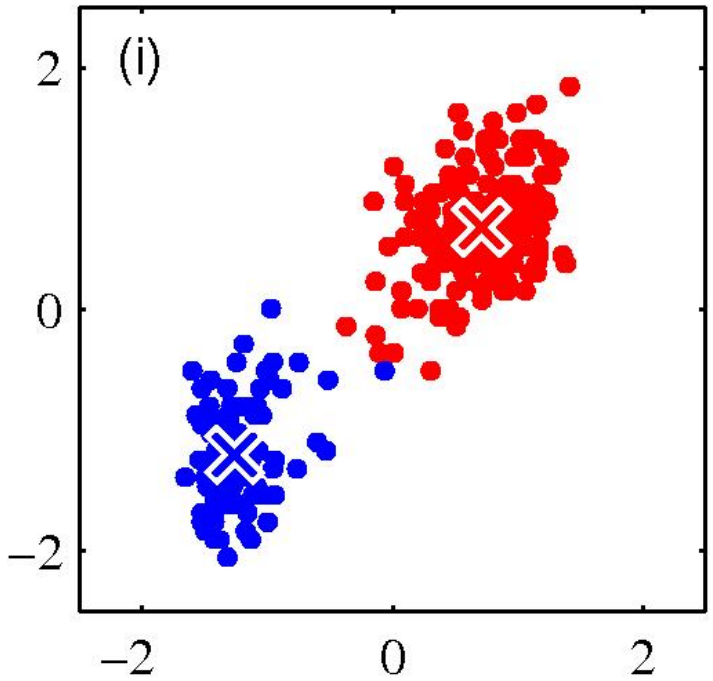
2nd iteration
result after step 4.

K-means clustering: example



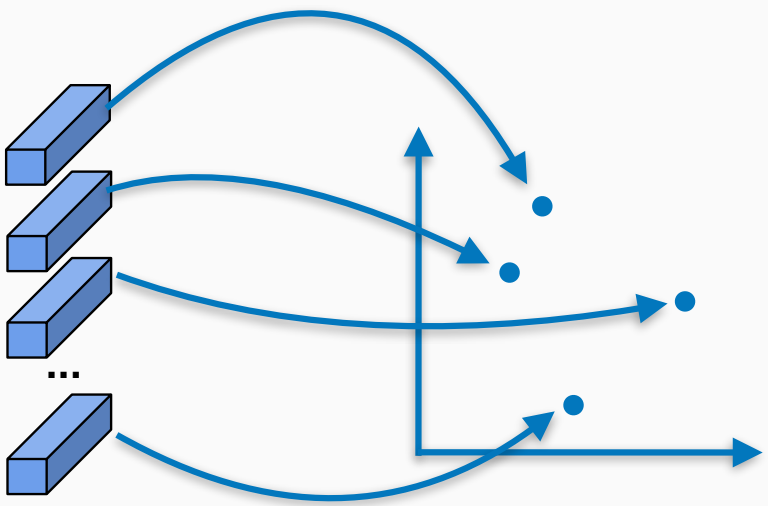
3rd iteration
result after step 3.

K-means clustering: example



Final clustering

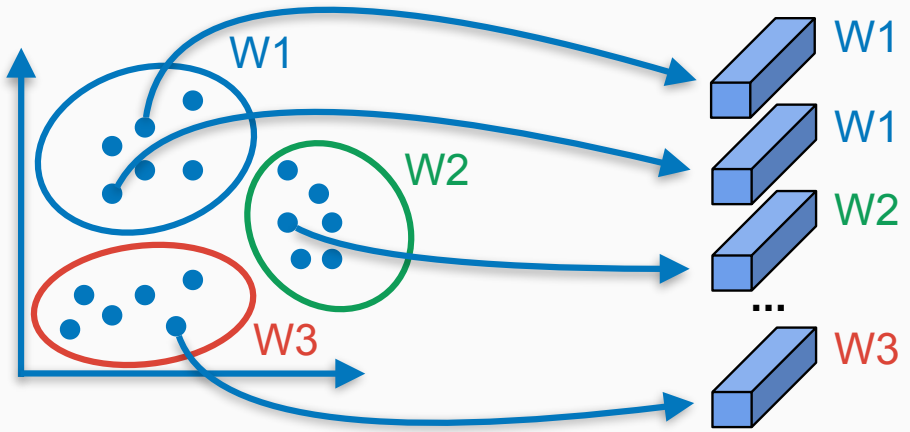
Obtaining visual words



Local CNN descriptors of dimension N

Represent as points in N dimensional feature space

Each cluster represents a "visual word" = **vocabulary**

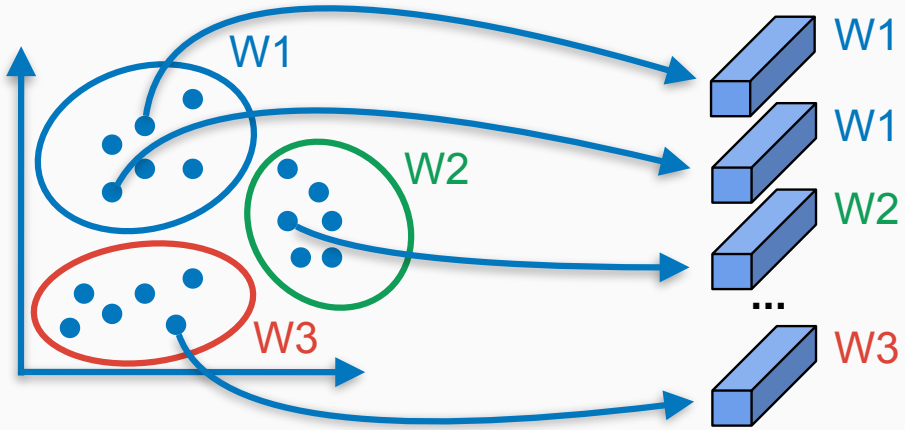


Quantise the feature space (clustering)

Assign the words to corresponding local descriptor

Forming inverted index

Each cluster represents
a “visual word” = **vocabulary**



Quantise the feature space (clustering)

Assign the words to corresponding local descriptor

Inverted index file

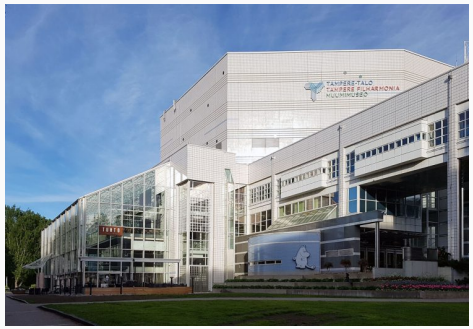
Word #	Image #
1	1,2,3
2	1,2
3	1
4	
5	2,3
6	
7	3
8	
…	…

Visual words in retrieval (pipeline)

- Training phase
  - Compute local (CNN) descriptors from the **training** images
  - Form a **vocabulary** by clustering the local descriptors from the training set
  - Assign visual words to each **database image** using vocabulary and form an index
- Query phase
  - At query time, obtain the local descriptors and visual words for the **query image**
  - **Search** matching images from the index

Example

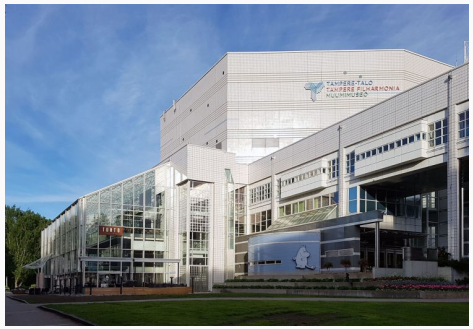
Database images



...

Example

Database images



W1 W1 W3 …

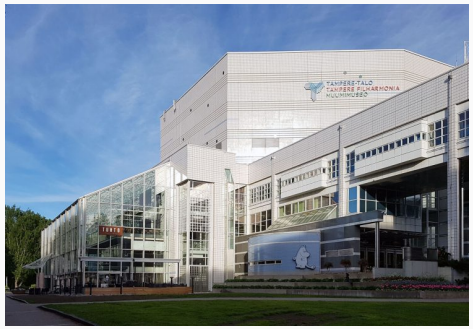
W2 W5 W1 …

W7 W5 W5 …

…

Example

Database images



…

W1 W1 W3 …

W2 W5 W1 …

W7 W5 W5 …

Inverted index file

Word #	Image #
1	1,2
2	2
3	1
4	
5	2,3
6	
7	3
8	
…	…

Example

New query image

W5 …

Inverted index file

Word #	Image #
1	1,2
2	2
3	1
4	
5	2,3
6	
7	3
8	
…	…

Example

New query image

W5 …

Inverted index file

Word #	Image #
1	1,2
2	2
3	1
4	
5	2,3
6	
7	3
8	
…	…

Example

New query image

W5 …

Inverted index file

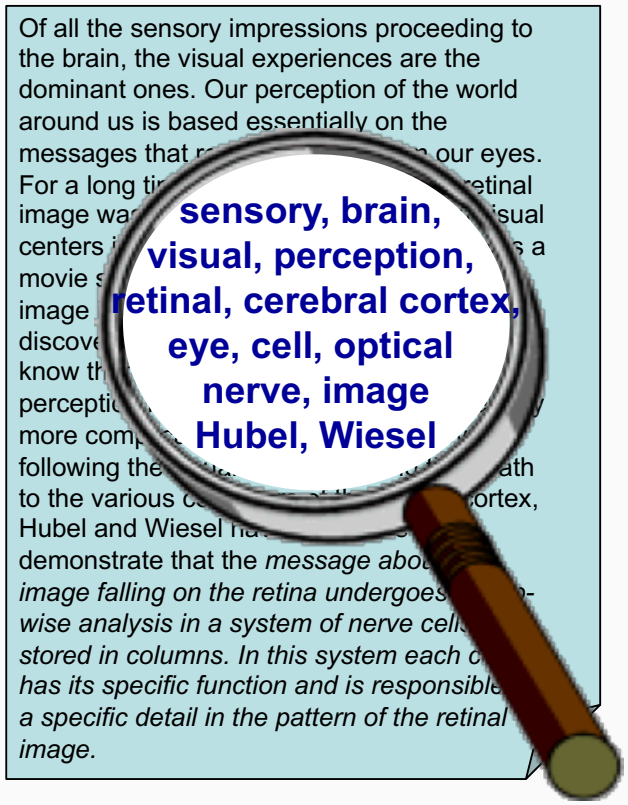
Word #	Image #
1	1,2
2	2
3	1
4	
5	2,3
6	
7	3
8	
…	…

Inverted file index

- Inverted file index is efficient only if it is **sparse**
- If most pages/images contain most words, then indexing gives no advantage over exhaustive comparison to each dataset image
- Problem solved?
- But how to summarise and compare the content of an entire image?

Comparing entire images with visual words

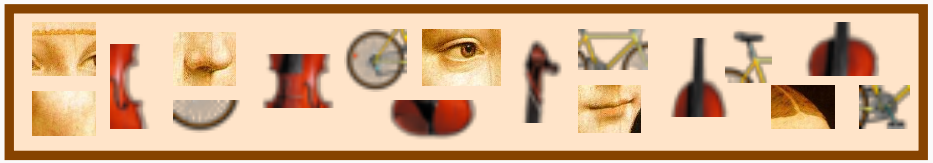
Analogy to documents

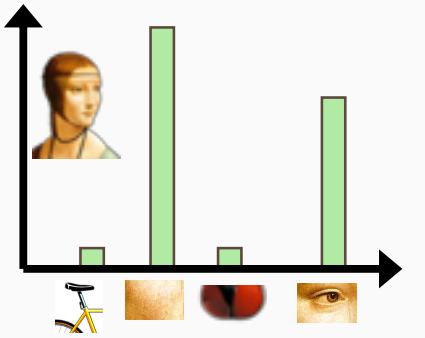


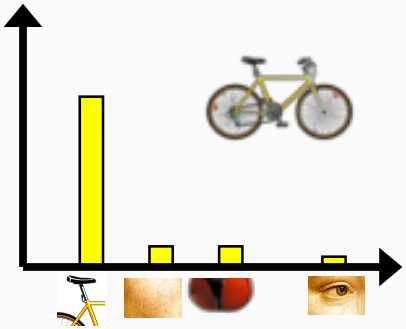
Credit: Fei-Fei Li

Bags of visual words

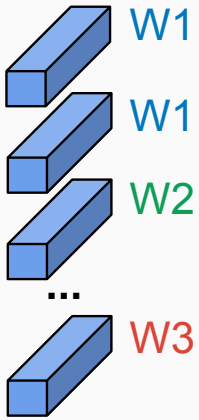
- Summarise the image using the histogram of word occurrences
- Analogous to bag of words representation used for documents





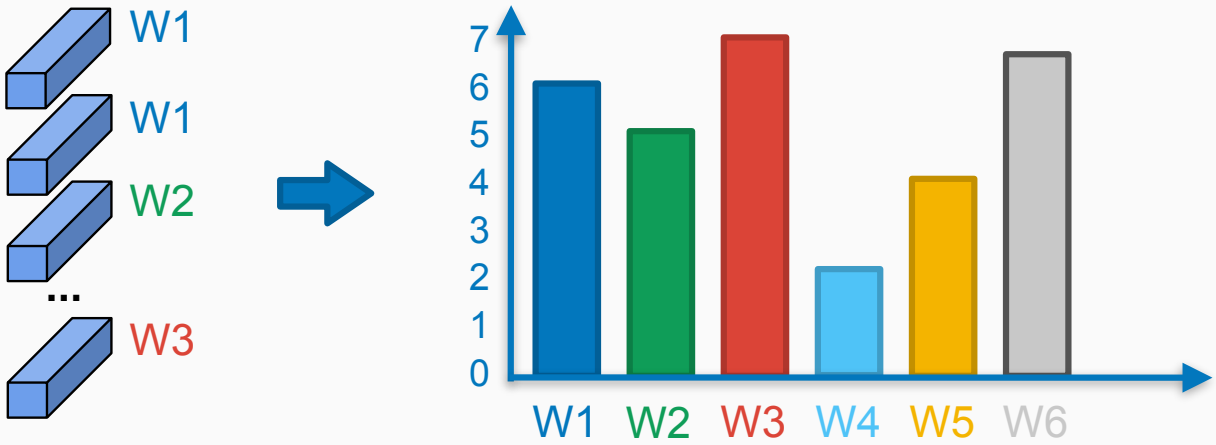


Bags of visual words



Quantised local
(CNN) descriptors

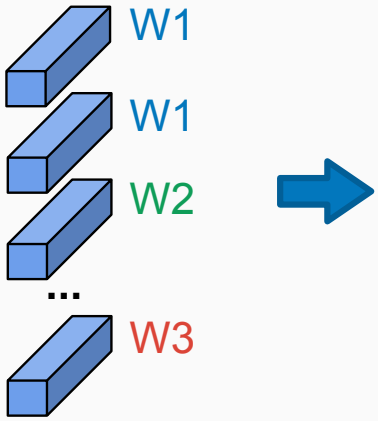
Bags of visual words



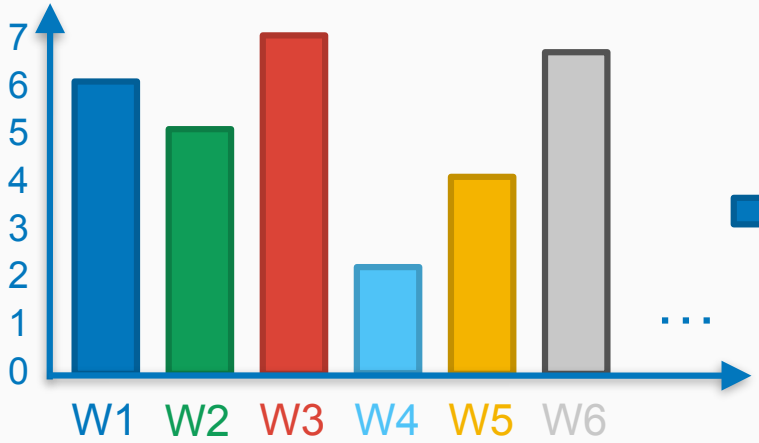
Quantised local (CNN) descriptors

Count the occurrences of each word in the image

Bags of visual words



Quantised local (CNN) descriptors



Count the occurrences of each word in the image

[6, 5, 7, 2, 4, 6, …]

Quantised descriptor for the entire image content

Comparing bags of words

- Measure similarity using normalised scalar product between the bags of words descriptor vectors

$$sim(d_j, q) = \frac{\langle d_j, q \rangle}{\|d_j\| \|q\|}$$

- Rank dataset images based on similarity to the query image

Inverted file index and bags of words similarity

1. Extract words from the query image

Query image

W5,W9,…

Inverted index file

Word #	Image #
1	1,2
2	2
3	1
4	8
5	2,3
6	
7	3
8	
9	8,3
…	…

Inverted file index and bags of words similarity

1. Extract words from the query image
2. Find relevant database images from inverted file index

Query image

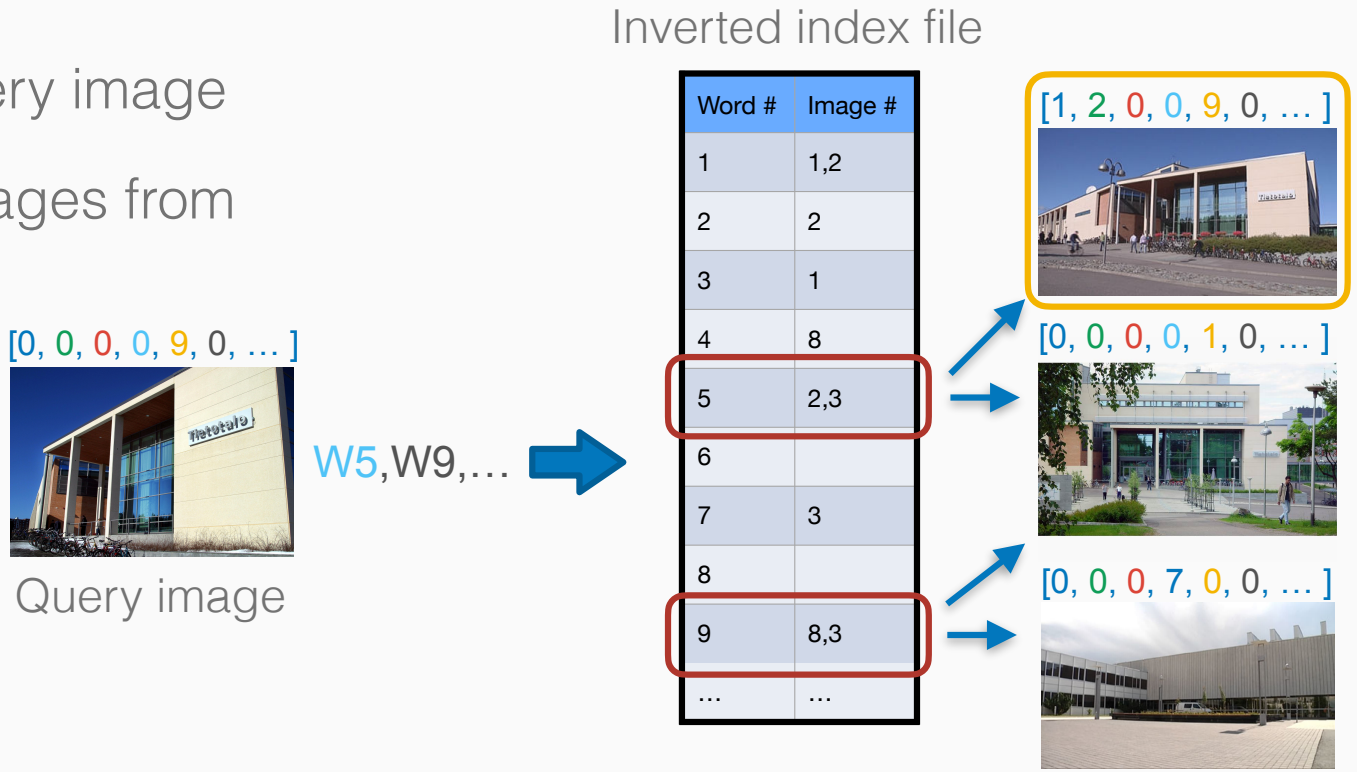
W5,W9,…

Inverted index file

Word #	Image #
1	1,2
2	2
3	1
4	8
5	2,3
6	
7	3
8	
9	8,3
…	…

Inverted file index and bags of words similarity

1. Extract words from the query image
2. Find relevant database images from inverted file index
3. Compare word counts



How to evaluate retrieval results?

How to evaluate retrieval quality?

Query

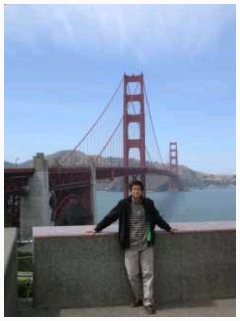
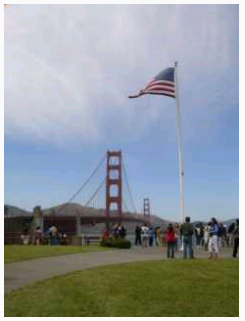
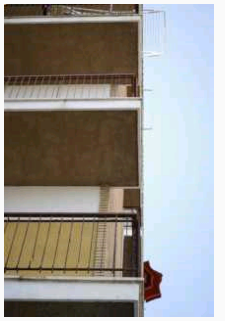
Dataset size: 10 images
Relevant (total): 5 images

Credit: O. Chum

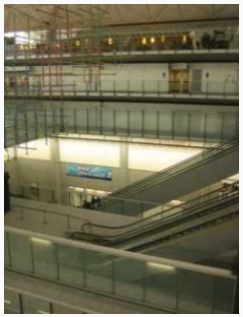
How to evaluate retrieval quality?

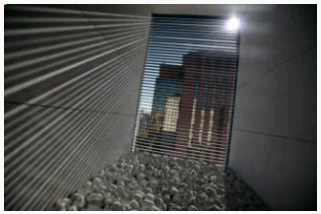
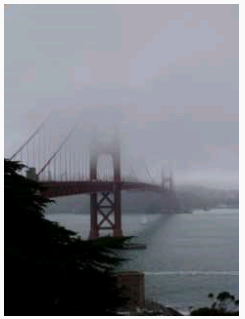
Query

Dataset size: 10 images
Relevant (total): 5 images





Results (ordered)

Credit: O. Chum

How to evaluate retrieval quality?

Query

Dataset size: 10 images
Relevant (total): 5 images

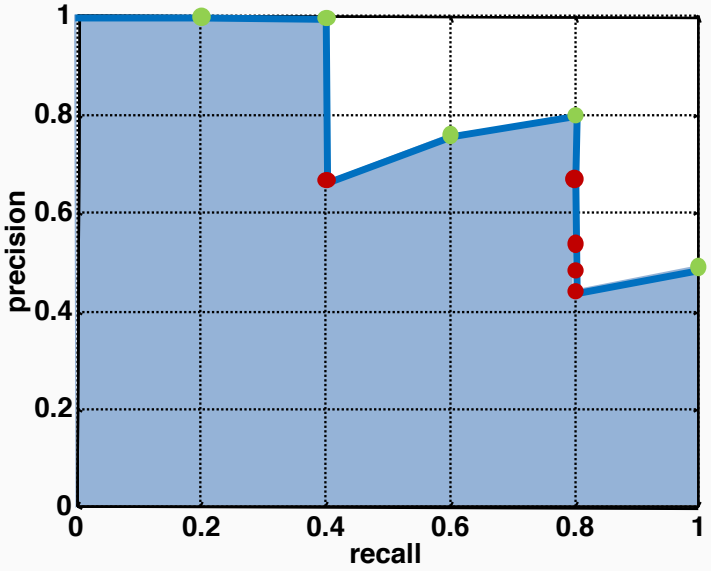
Results (ordered)

Credit: O. Chum

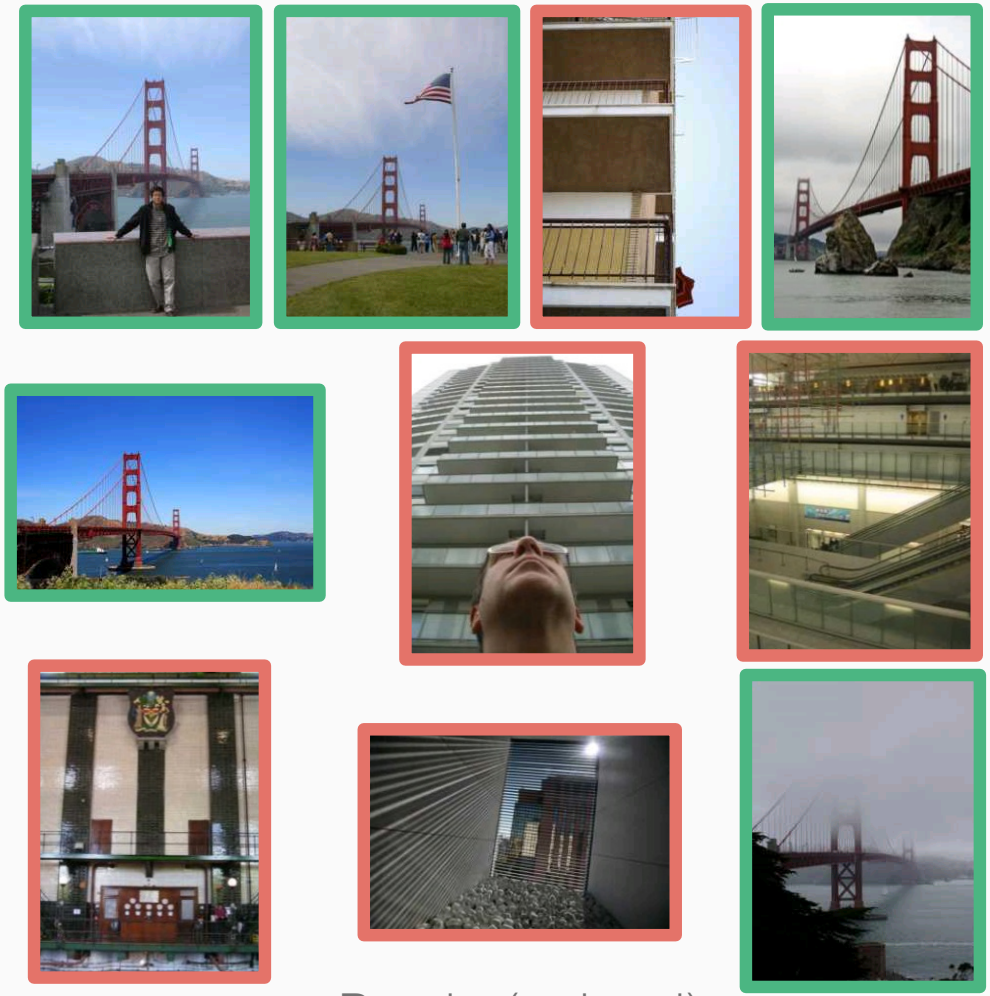
How to evaluate retrieval quality?

Query

Dataset size: 10 images
Relevant (total): 5 images



Precision = #relevant / #returned
Recall = #relevant / #total relevant



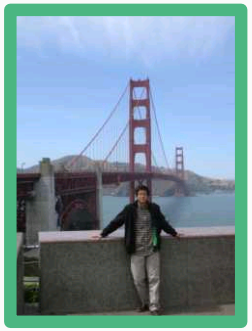
Results (ordered)

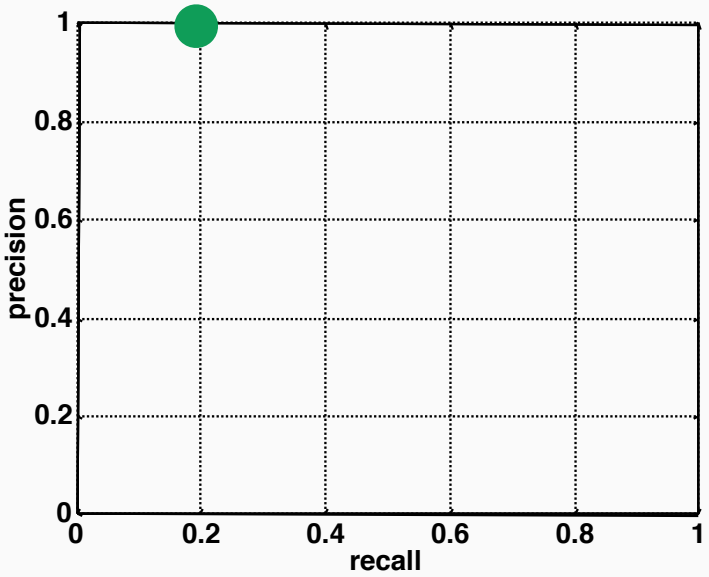
Credit: O. Chum

How to evaluate retrieval quality?

Query

Dataset size: 10 images
Relevant (total): 5 images





Precision = #relevant / #returned
Recall = #relevant / #total relevant

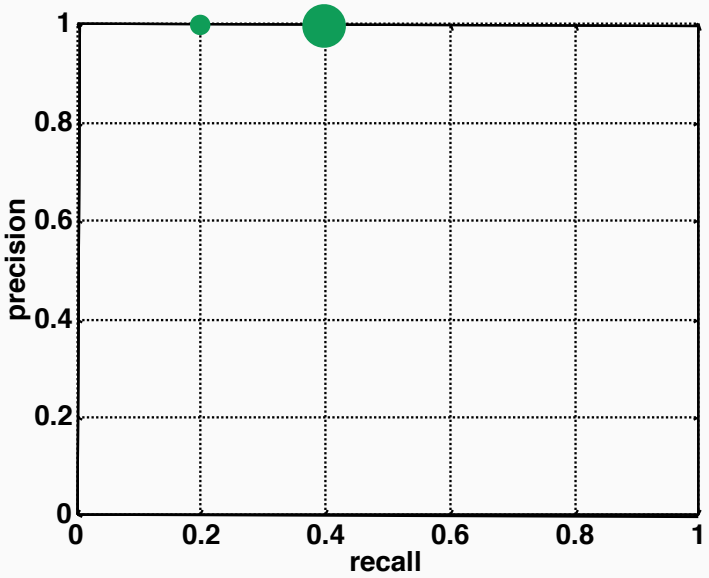
Results (ordered)

Credit: O. Chum

How to evaluate retrieval quality?

Query

Dataset size: 10 images
Relevant (total): 5 images



Precision = #relevant / #returned
Recall = #relevant / #total relevant

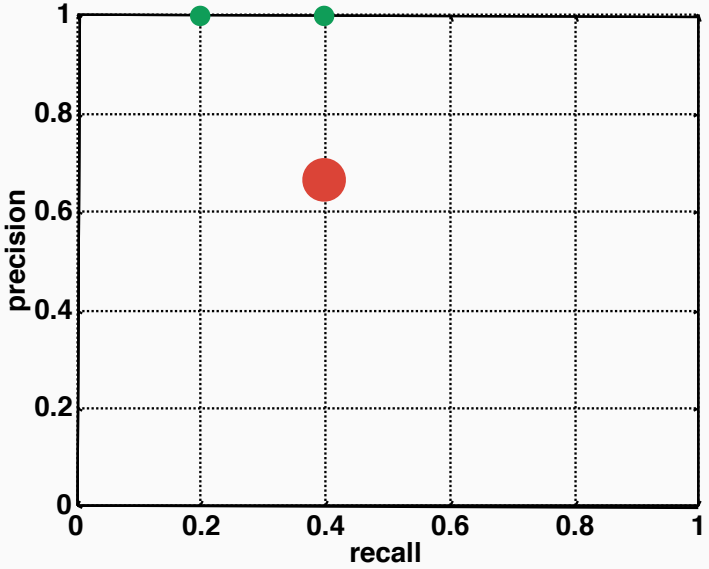
Results (ordered)

Credit: O. Chum

How to evaluate retrieval quality?

Query

Dataset size: 10 images
Relevant (total): 5 images



Precision = #relevant / #returned
Recall = #relevant / #total relevant

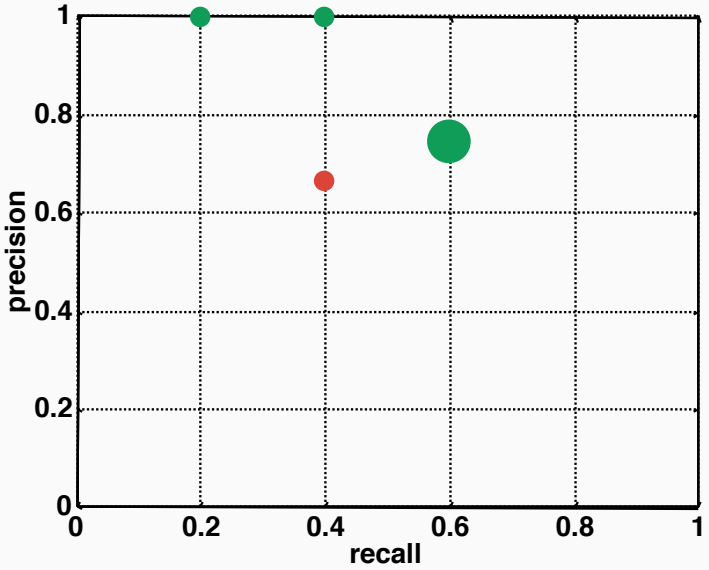
Results (ordered)

Credit: O. Chum

How to evaluate retrieval quality?

Query

Dataset size: 10 images
Relevant (total): 5 images



Precision = #relevant / #returned
Recall = #relevant / #total relevant

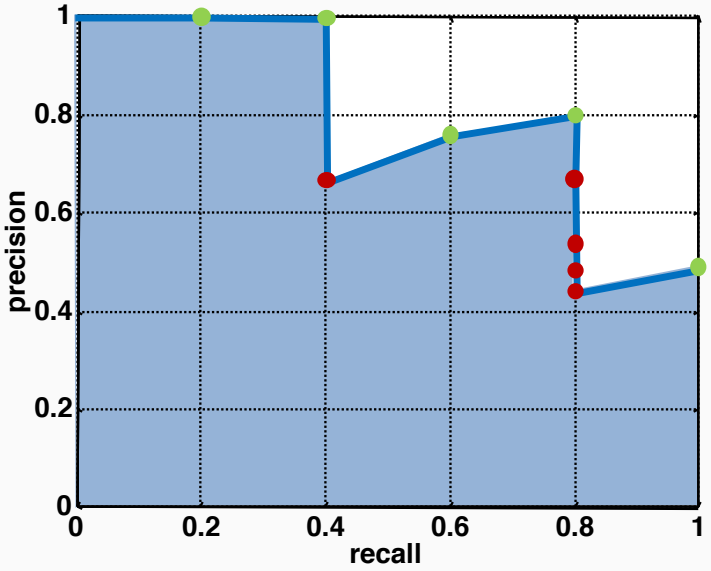
Results (ordered)

Credit: O. Chum

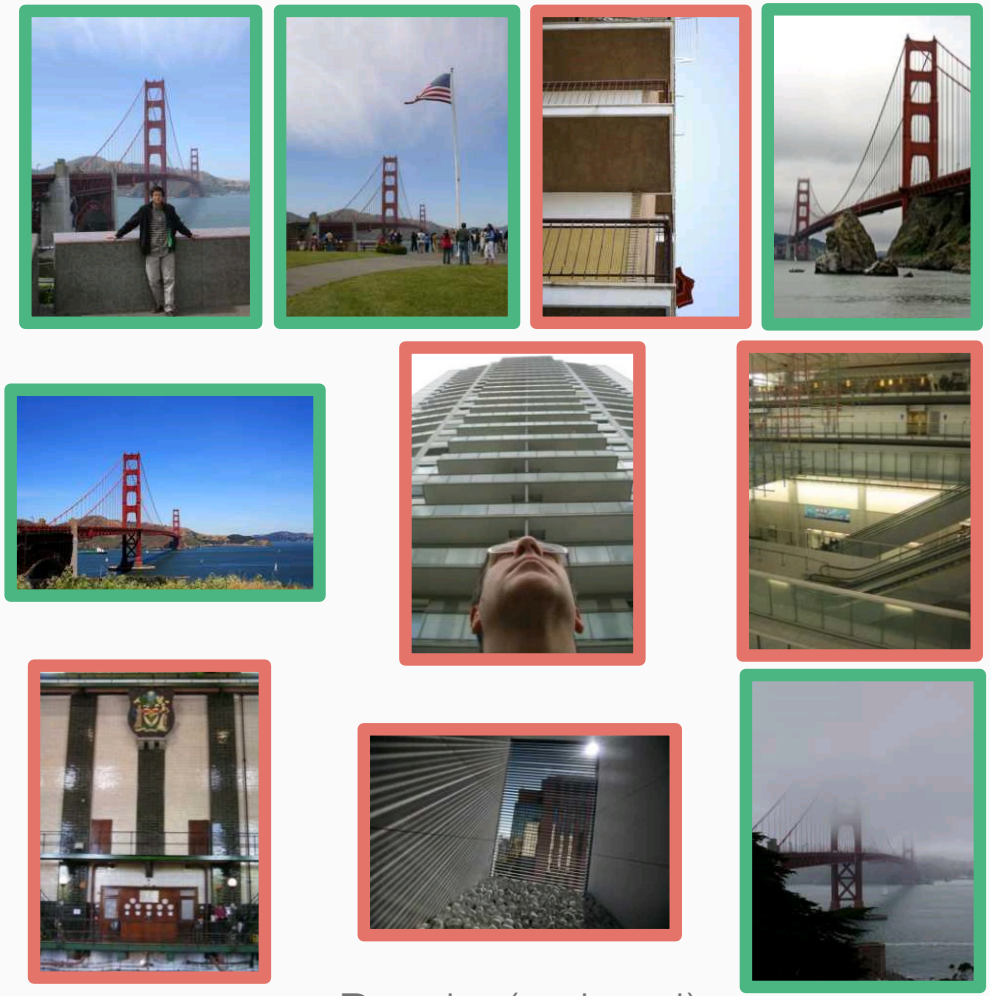
How to evaluate retrieval quality?

Query

Dataset size: 10 images
Relevant (total): 5 images



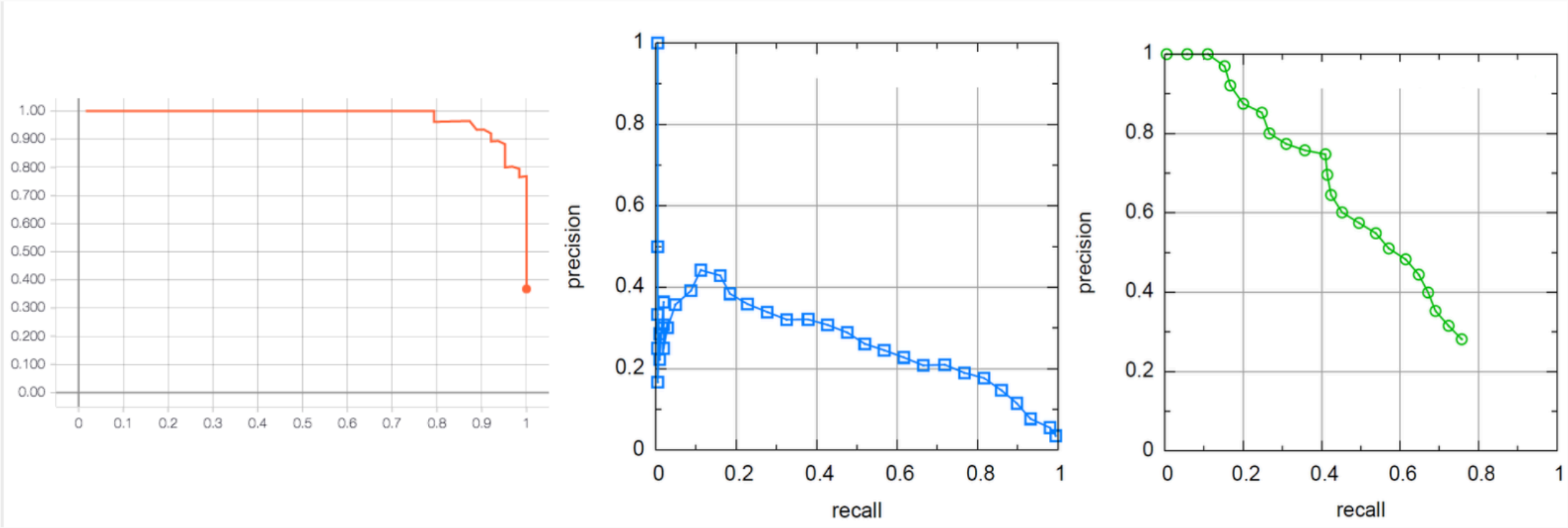
Precision = #relevant / #returned
Recall = #relevant / #total relevant



Results (ordered)

Credit: O. Chum

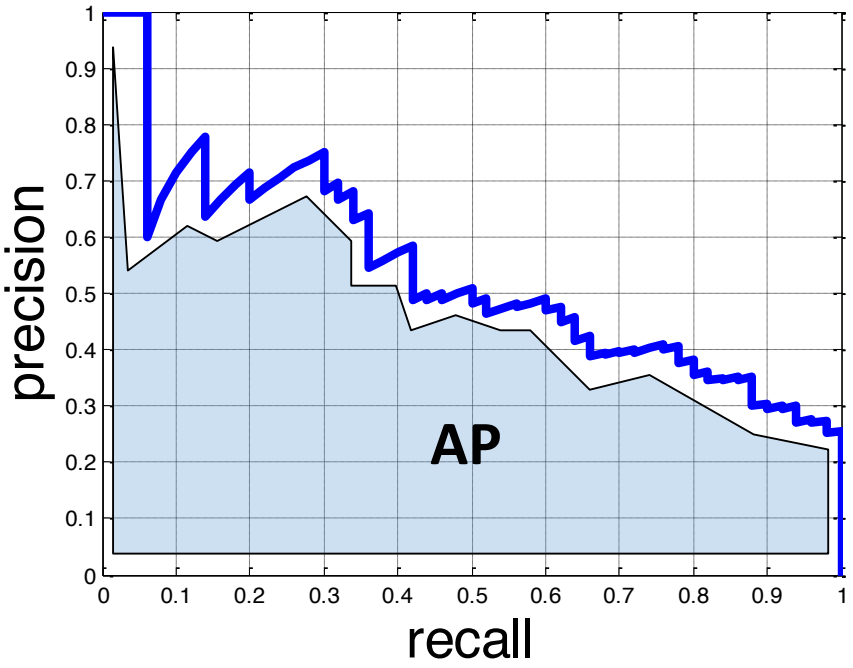
A pot-pourri of PR curves

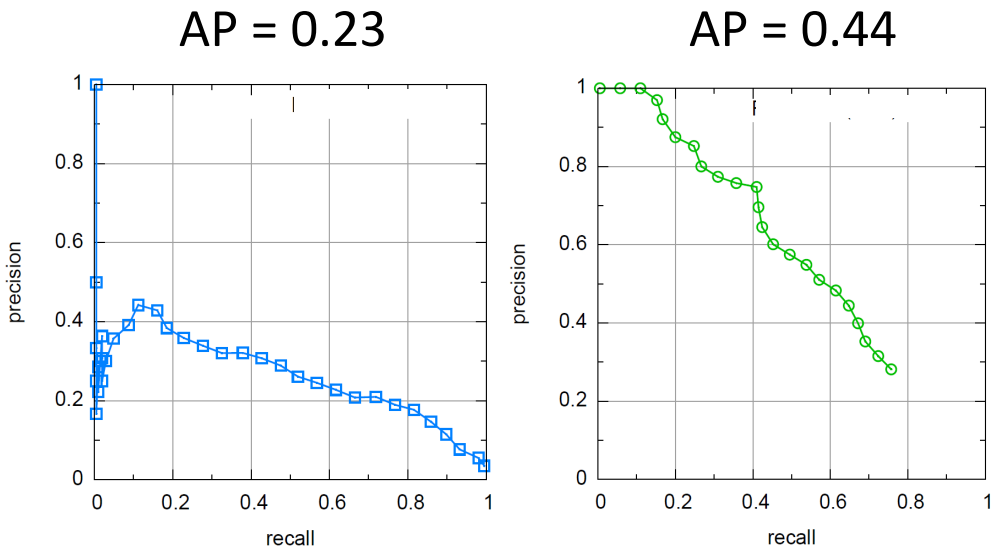


Credit: A. Zisserman

Average precision

- Area under Precision-Recall curve
- Single score to assess performance





A good score requires both high recall and high precision

Credit: A. Zisserman

Things to remember

- Two phases of image retrieval:
  - Obtain image descriptors for database and query image
  - Calculate similarity metrics and rank the database images according to similarity with query
- (Pre-trained) CNNs can form powerful image descriptors for retrieval
- Descriptors can be quantised to “visual words” for efficient search
  - Learn a vocabulary from a training set by clustering
  - Summarise image by distribution of words (bag of words)
- Precompute index to enable fast search at query time (inverted index)
- Retrieval quality can be evaluated using precision vs recall plot

That’s it folks!