



Gobi Desert from ISS



1 mm

(2014) LEARNING FINE-GRAINED IMAGE SIMILARITY WITH DEEP RANKING

PAPER REVIEW

ELENA TERENZI

PAPER REVIEW

Absolutely recommend it!

Great learning experience!

When you think you have understood... did you really understand?

You will find out many details you did discard!

And learn so much more!!

WHY THIS PAPER?

Insurance damage similarity detection problem (with Chew-Yean Yam and Nuno Silva as lead SEs)

Could also be interesting for other applications where a large inventory of images needs to be searched and similar images must be retrieved based on a query image (e.g. Fashion image similarity case, CY lead SE)

Same/similar damage



Figure 3: Query image and target image.

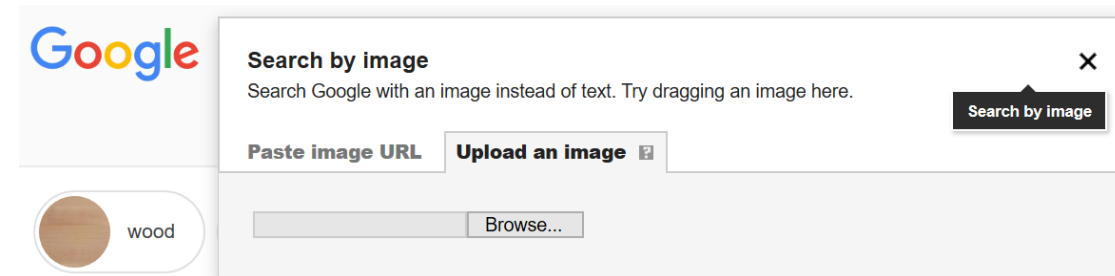
WHY THIS PAPER (AUTHOR'S MOTIVATIONS)

Sometimes high-level information (low resolution/less granular) doesn't capture well the image semantics

Search-by-example, i.e. finding images that are similar to a query image, is an indispensable function for modern image search engine

Most existing image similarity models consider *category-level* image similarity.

category-level image similarity is not sufficient for the search-by-example image search application. Search-by-example requires the **distinction of differences between images within the same category**, therefore *fine-grained image similarity*

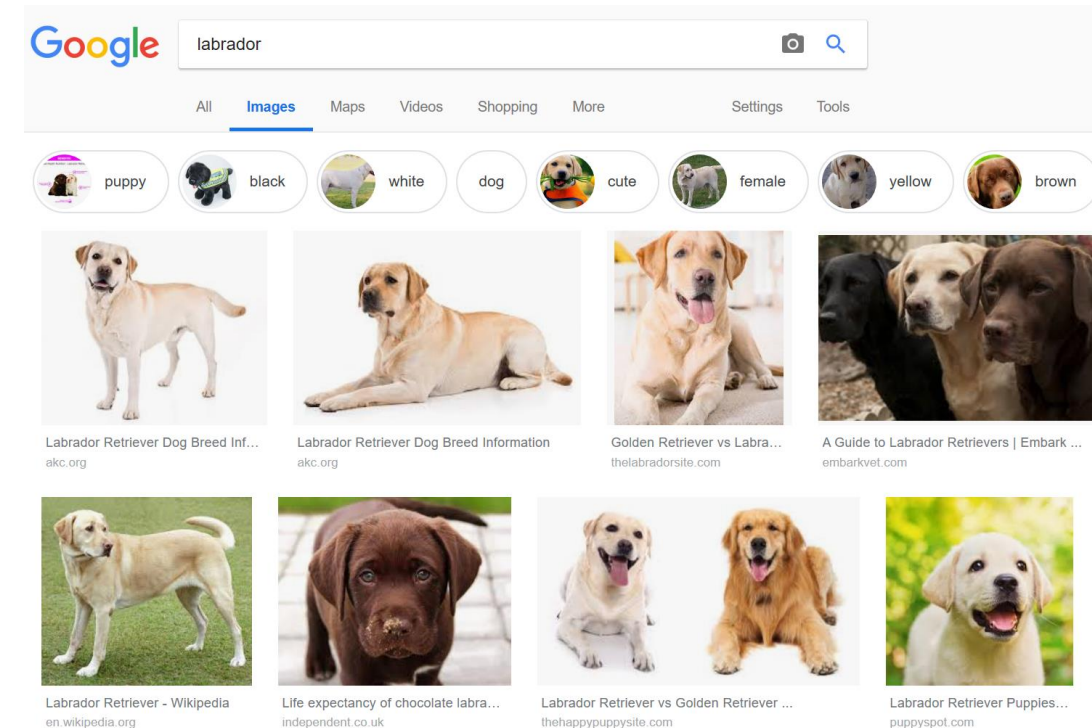
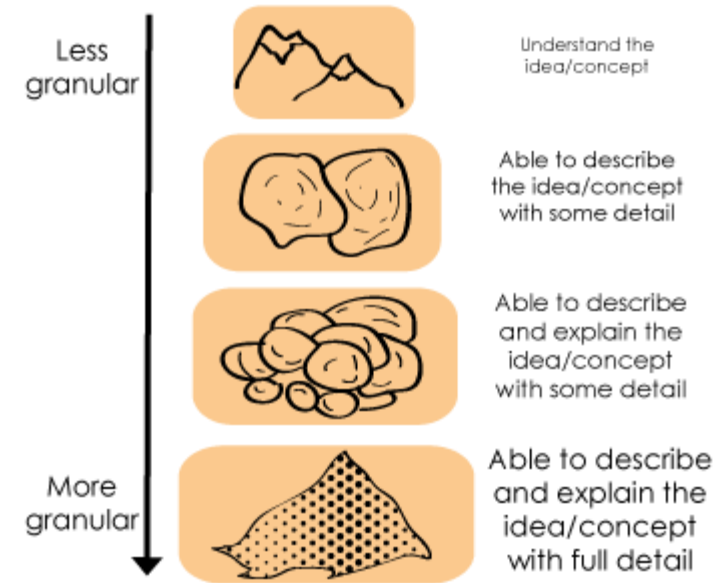


WHY FINE-GRAINED IMAGE SIMILARITY

Search-by-example requires the **distinction of differences between images within the same category**

Same category: in this paper a category is represented by all (top N) pictures returned by Google Search given a **query text**

To achieve their goal (distinguish differences between similar images) researchers believed they needed to capture **fine-grained image information**

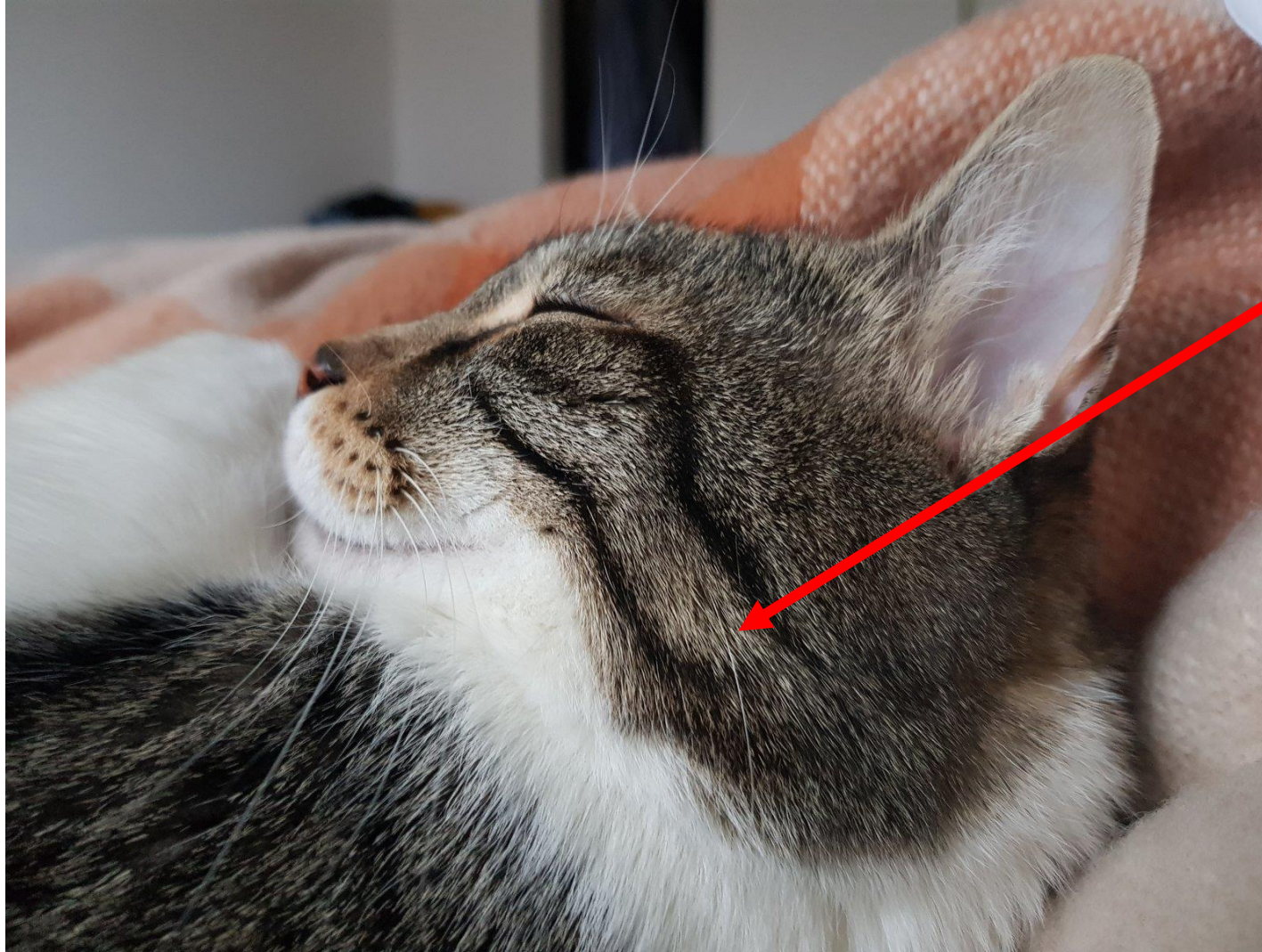


INFORMATION GRANULARITY

Different Resolution: More/less pixels to represent the same object (pixel density)

Different Scale: same picture at different scale => different picture shape/size

EFFECTS OF INFORMATION GRANULARITY: WHAT IS THIS?






CLASSIFICATION VS SIMILARITY

For image classification, “black car”, “white car” and “dark-grey car” are all cars, **unless we define more granular classes**

For similar image ranking, if a query image is a “black car”, we usually want to rank the “dark gray car” higher than the “white car”.

Due to the intrinsic difference between image classification and similar image ranking tasks, a good network for image classification may not be optimal for distinguishing fine-grained image similarity

	Classifier g(.)	Similarity ranking f(.)
	“car” => g(p) = 0110101001	100% f(p) = 0110101001 distance.cosine(f(q),f(p))=0
	“car” => g(p) = 0110101001	85% f(p) = 0110111011 distance.cosine(f(q),f(p))=0.15
	“car” => g(p) = 0110101001	73% f(p) = 0010111011 distance.cosine(f(q),f(p))=0.27

SIMILARITY FUNCTION

Our similarity measure is defined as the **squared Euclidean distance** of any two given images P and Q in the **embedding space**

$$D(f(P), f(Q)) = \|f(P) - f(Q)\|_2^2$$

Where $f(.)$ is the embedding function that we need to learn.

This definition formulates the similar image ranking problem as nearest neighbor search problem in Euclidean space, which can be efficiently solved via **approximate nearest neighbor search algorithms** (e.g. hashing methods, such as Locality-Sensitive Hashing or LSH)

RANKING, TRIPLETS, RELEVANCE AND LOSS

Ranking is a supervised task (leverages humans for the task). Since it is error-prone for human labelers to directly label the image ranking which may consist tens of images, the similarity relationship of images is labeled with **triplets**.

Let's assume that $r_{q,p} = r(q_i, p_i)$ defines the **pairwise relevance** for any two given images q_i and p_i

Then:

$$D(f(q_i), f(p_i)) < D(f(q_i), f(n_i))$$

for any triplet $t_i = (q_i, p_i, n_i)$ such that $r(q_i, p_i) > r(q_i, n_i)$

where q_i is the query image, p_i is the positive image and n_i is the negative image

To increase the robustness of our model (reduce **overfitting**) a gap parameter g is introduced and the loss (error) of our embedding function can be defined as:

$$l(q_i, p_i, n_i) = \max\{0, g + D(f(q_i), f(p_i)) - D(f(q_i), f(n_i))\}$$

Never clearly defined
Uses "golden feature"

Never defined!!

OBJECTIVE FUNCTION

Minimizing our loss keeping the **model parameters** as small as possible

$$\min \sum_i l_i + \lambda \|\mathbf{W}\|_2^2$$

λ is another regularization parameter, in this case $\lambda=0.001$

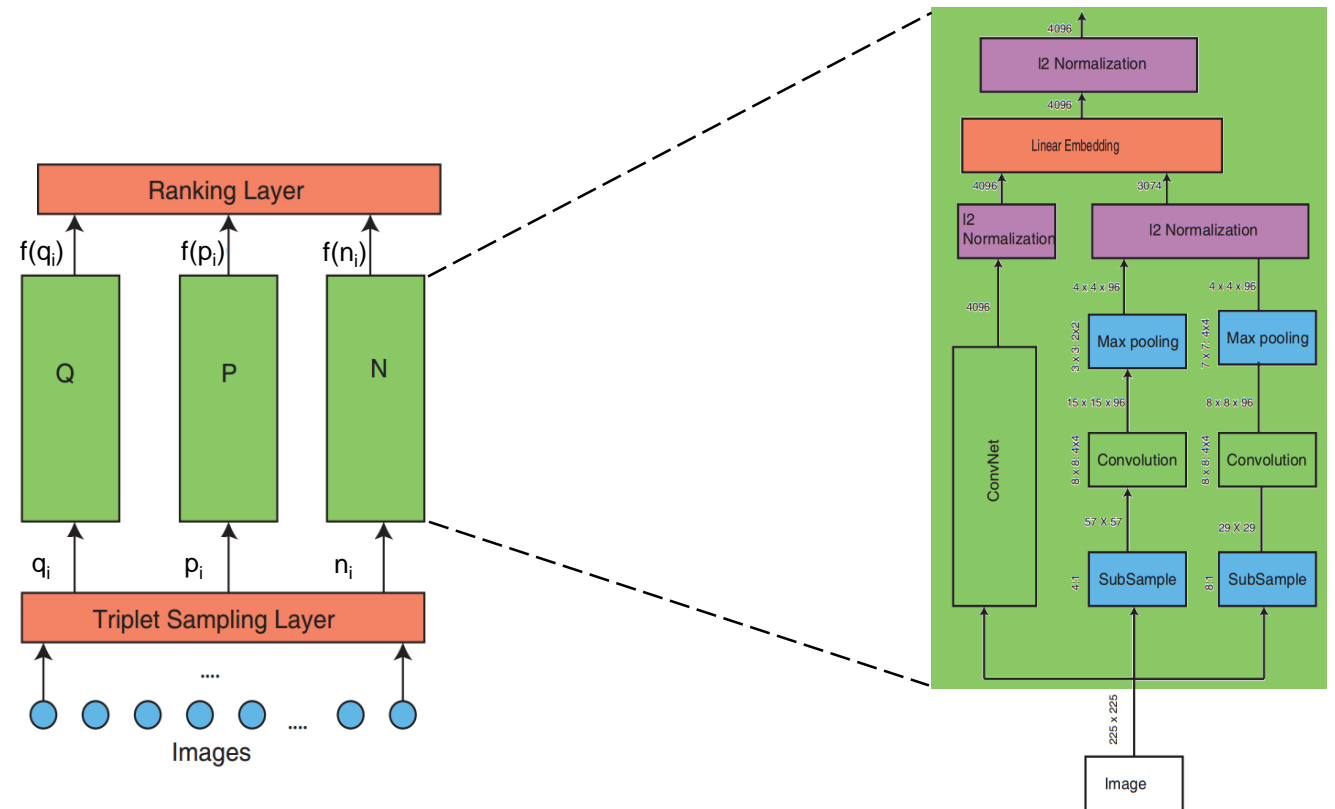
NETWORK ARCHITECTURE

Triplet, Multiscale network

Q,P and N share the same architecture and same weights (weights are always identical at each step).

The ConvNet captures image semantics. Think ResNet for image classification

The subsampling (low-res) paths capture image features at **varying scale** (remember: information granularity).



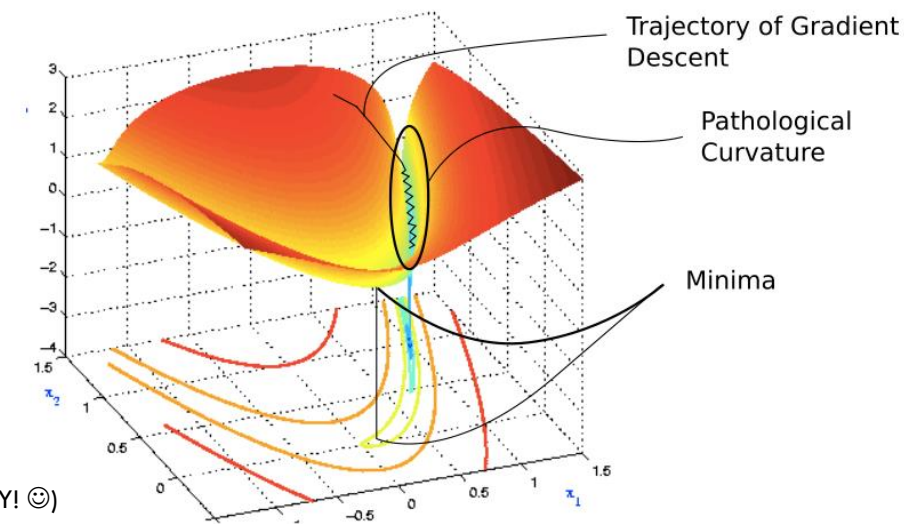
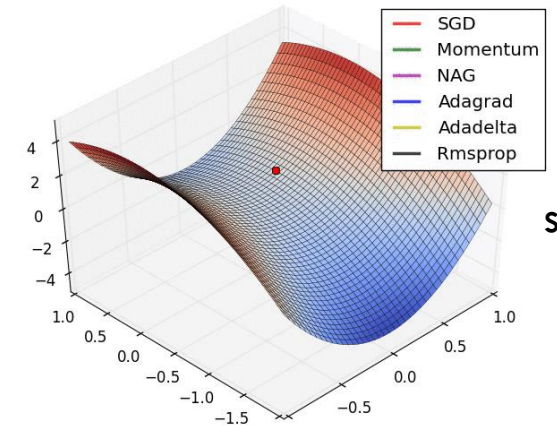
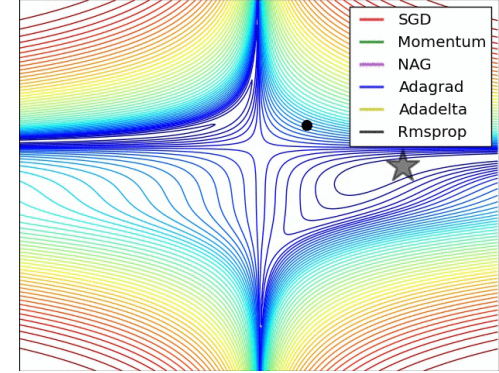
SGD + NESTEROV'S MOMENTUM

Weights are updated using SGD (stochastic gradient descent) with Nesterov's Momentum.

SGD + Momentum: instead of using only the gradient of the current step to guide the search, **momentum also accumulates the gradient of the past steps to determine the direction to go**

SGD + Nesterov's Momentum: same as SGD + Momentum except momentum is not calculated with respect to the current value of our weights but with respect to their **approximate future position**

Converges faster when loss function has **pathological curvatures** such as ravines



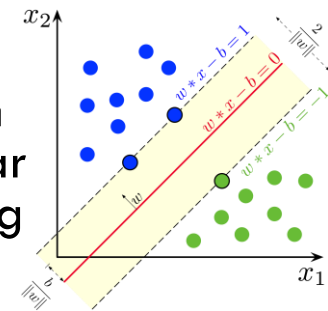
TRAINING DATA

two sets of training data

1) ImageNet to train the ConvNet => one could think of leveraging pretrained models such as VGG, Xception, ResNet, ...

2) *Relevance training data*, responsible for learning fine-grained visual similarity. Dataset is collected from 100,000 search queries (using Google image search), with the top 140 image results from each query: overall 12 million images (with **24 million triplets**). They employ a **golden feature** to compute the relevance $r_{i,j}$ for the images from the same search query and set it to 0 for images belonging to other queries. **My assumption: Google image ranking cannot be fully trusted/they want to improve it**

The **golden feature** is a weighted linear combination of **27 features**. It incorporates both **visual appearance information and semantic information**, and it is of high performance in evaluation (see later). It also includes features learned through image annotation data (similar to Bing Visual Search). Linear weights are learned through max-margin linear weight learning using human rated data (i.e. **Google Image Search ranking?**).



Max-margin SVM

GOLDEN FEATURE

It incorporates both **visual appearance information and semantic information**, and it is of high performance in evaluation (see later).

Harr wavelet: L0 distance of Harr wavelet decomposition of image pairs

Color: Normalized L1 distance between color histograms

SIFT (Scale Invariant Feature Transform): SIFT-like features (they use L1 distance for pairs)

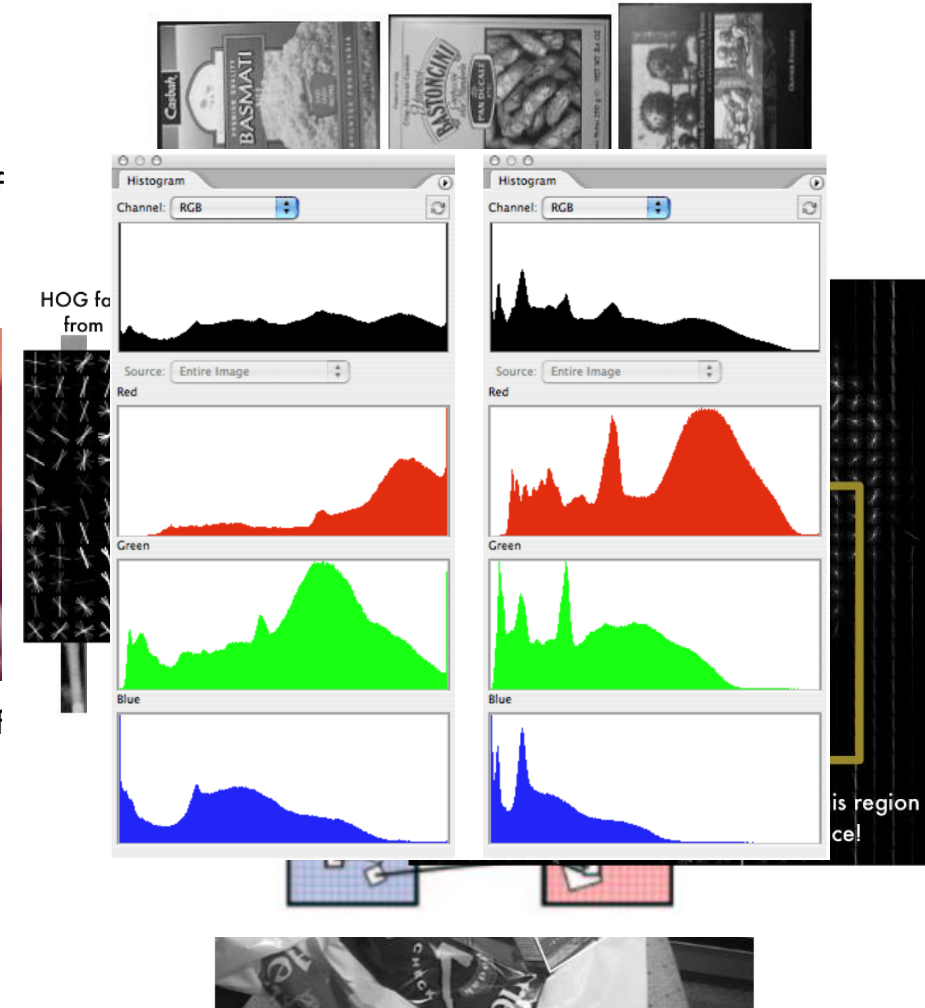
Fisher vectors on SIFT-like features: treats image features as words, generates a vocabulary and, from this, (visual word) frequency histograms. They use L2 distance for pairs

HOG: Histogram of Oriented Gradients. L1 distance for pairs

SPMK Texton features: Spatial Pyramid Matching Kernel (SPMK) is a way to aggregate features from coarse to fine spatial scales. Similar to Fisher+SIFT but uses **texture** instead of SIFT as features

It also includes features learned through image annotation data (from [this paper](#) Large scale image annotation: learning to rank with joint word-image embeddings)

And 20 more techniques which are not documented!!



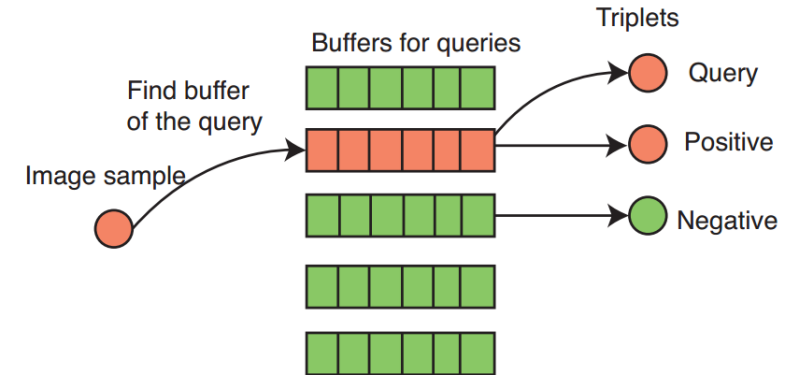
SO WHY NOT USE THE GOLDEN FEATURE ALONE?

Method	Precision	Score-30
Wavelet [9]	62.2%	2735
Color	62.3%	2935
SIFT-like [17]	65.5%	2863
Fisher [20]	67.2%	3064
HOG [4]	68.4%	3099
SPMKtexton1024max [16]	66.5%	3556
L1HashKPCA [14]	76.2%	6356
OASIS [3]	79.2%	6813
Golden Features	80.3%	7165
DeepRanking	85.7%	7004

There is still space for improvement (info granularity)

Table 1. Similarity precision (Precision) and score-at-top-30 (Score-30) for different features.

TRIPLER SAMPLING



Sampling all possible triplet combinations would be computationally prohibitive ($\sim 10^{21}$ triplets) and **sub-optimal** (we are not interested in all possible combinations!)

They define **total relevance** r = how relevant an image is in terms of its pairwise relevance to the other images in the same category

They define the concept of **in-class** (same category) and **out-of-class** negative images

For every category:

- 1) sample a query image q and keep it only if it's within the top- N **total-relevant images**
- 2) sample a positive image p and accept it with probability $\min(1, r_{qp}/r_p)$ ← Total relevance of positive image
- 3) sample negative images n : Pairwise relevance between query and positive image
 - 3.1) if **in-class**: sampled in a similar way to the positive image, but needs to have lower pairwise relevance than the positive image (a threshold is defined for robustness): $r_{qn} + T < r_{qp}$
 - 3.2) if **out-of-class**: drawn uniformly from all the images with different categories

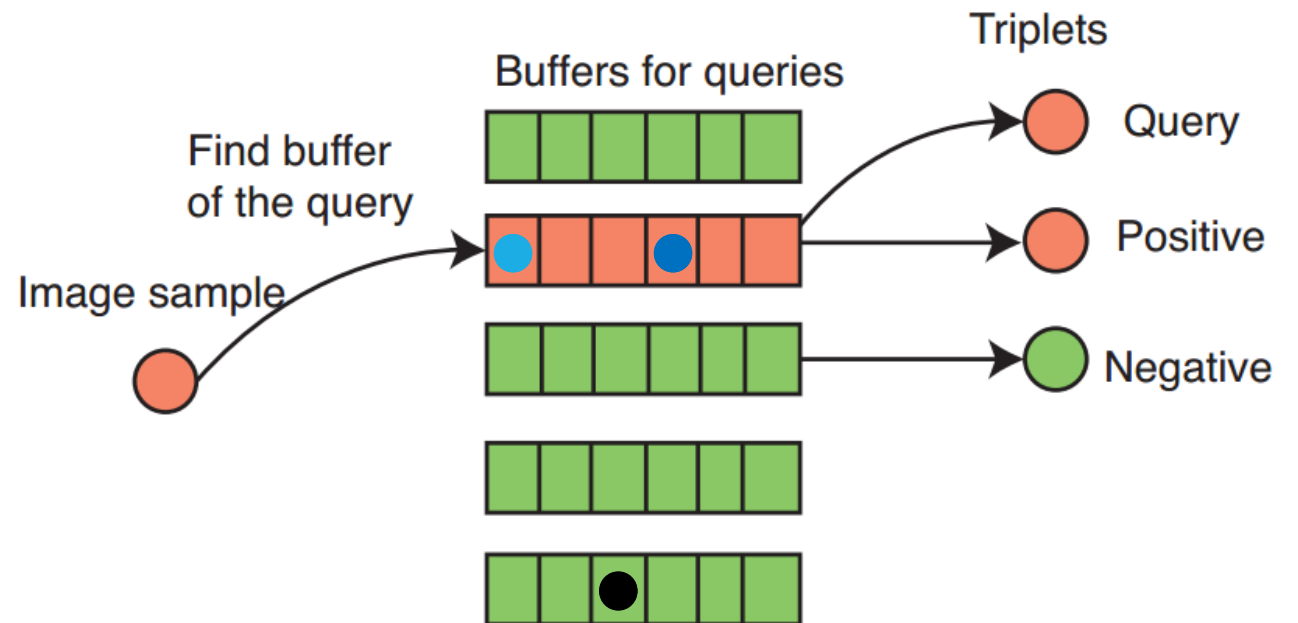
$$r_i = \sum_{j: c_j = c_i, j \neq i} r_{i,j}$$

TRIPLET SAMPLING EXAMPLE





Buffers are created for each query/category such that they contain highly relevant images for each category (top N)

- 1) Sample a query image q from category k
- 2) sample another image p from same category, and keep it in the triplet as a positive image with **probability** = $\min(1, r_{qp}/r_p)$
- 3) If negative *out-of-class*: randomly sample it from other categories. If *in-class* the negative image is sampled like 2)

Triplet is kept only if $r_{qn} + T < r_{qp}$



EXAMPLE OF IN-CLASS AND OUT-OF-CLASS TRIPLETS

Category = vehicle	Query	Positive	Negative
In-class			
Out-of-class			

IN-CLASS VS OUT-OF-CLASS (VS RANDOM TRIPLETS)

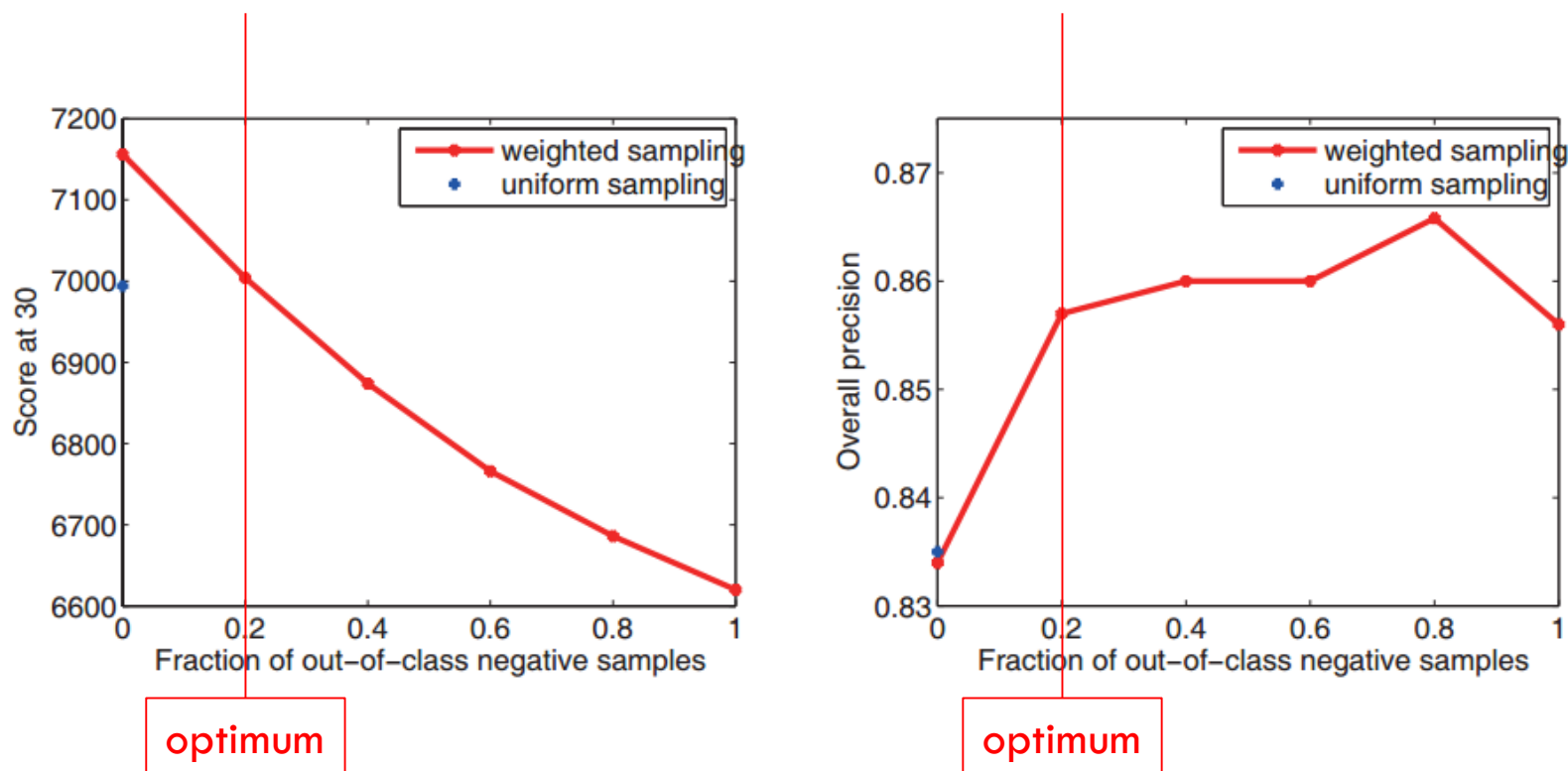


Figure 6. The relationship between the performance of the proposed method and the fraction of out-of-class negative samples.

EVALUATION DATA

This is the only dataset where they used humans to rank pictures... **even though the training dataset incorporates google search user feedback!!** (*golden feature uses human rated data*)

Dataset is built from 1 000 popular text queries. They sampled triplets (**Q, A, B**) from the top 50 search results for each query from the Google image search engine.

Each triplet is then rated by 3 raters. Only the triplets with unanimous scores from the three raters enter the final dataset.

In total the evaluation dataset contains around **14,000 triplets**. Those triplets are solely used for evaluation.

EVALUATION METRICS (1 OF 2)

Two evaluation metrics

Similarity precision: percentage of triplets being correctly ranked.

Given any triplet $t_i=(q_i, p_i, n_i)$ (unanimously ranked by humans) if the positive image is ranked higher than the negative ($r_{qp} > r_{qn}$) then the triplet is correctly ranked.

Remember: $D(f(q_i), f(p_i)) < D(f(q_i), f(n_i))$ if $r_{qp} > r_{qn}$

EVALUATION METRICS (2 OF 2)

Score-at-top-K: number of correctly ranked triplets minus the number of incorrectly ranked ones on a subset of triplets *whose ranks are higher than K*. They use **K = 30** because Google Image Search users will not typically look further.

Of the top 30 images which would be typically seen by Google users how many are actually correctly ranked (correct = confirmed by human visual inspection)? **Remember: images which made it in the top 30 might not always be “positives”, they might be of poor ranking quality! They made it to the top 30 but should not be there: hence they could be a negative!**

They don't share how many triplets make it to the “score-at-top-30” subset.

For each query image in the test set, retrieve 1000 images belonging to the same text query, and rank these images using the learned similarity metric. One triplet's rank is higher than K if its positive image or negative image is among the those which would typically be seen by a Google user (= *top K nearest neighbors of the query image*).

This metric is similar to the precision-at-top-K metric, which is widely used to evaluate retrieval systems.

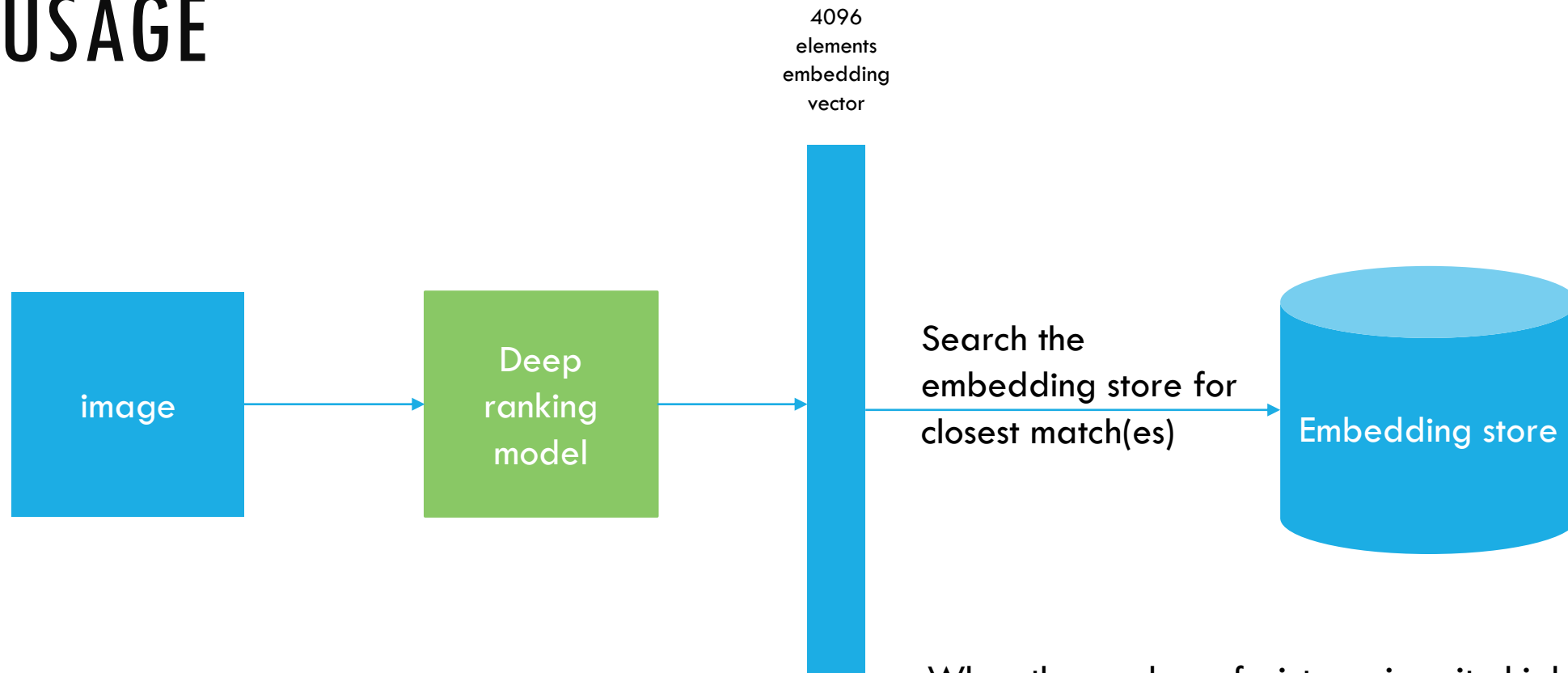
EVALUATION RESULTS

- 1) ConvNet: just the ConvNet (no triplets, no triplet loss)
- 2) ConvNet + triplets and triplet loss (single scale deep Ranking)
- 3) OASIS Deep CNN trained on top of 2)**
- 4) Embedding trained on top of ConvNet + Golden Features

Method	Precision	Score-30
ConvNet	82.8%	5772
Single-scale Ranking	84.6%	6245
OASIS on Single-scale Ranking	82.5%	6263
Single-Scale & Visual Feature	84.1%	6765
DeepRanking	85.7%	7004

Table 2. Similarity precision (Precision) and score at top 30 (Score-30) for different neural network architectures.

USAGE



When the number of pictures is quite high (depending on your available memory, think >4millions) search can be performed leveraging approximate nearest neighbor search algorithms (like LSH)

CODE

<https://github.com/akarshzingade/image-similarity-deep-ranking> (beware there is a typo in code see <https://github.com/akarshzingade/image-similarity-deep-ranking/issues/29>)

Further optimizations introduced in our version of the code: kudos to Nuno Silva!!



QUESTIONS?