Content Based Image Retrieval Using Deep Neural Networks

Prepared by:
Amel Jamal Yassin 144018
Hussein Mansour Mohammed 144026

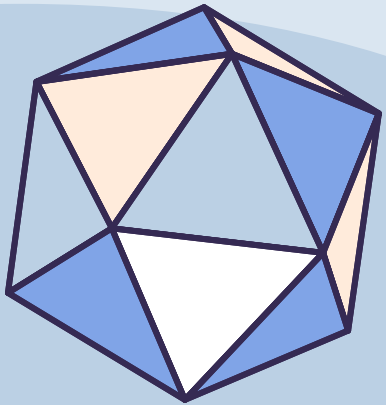


Table of Contents

01 Introduction

02 Efficient Nets

03 Metric & Data-set

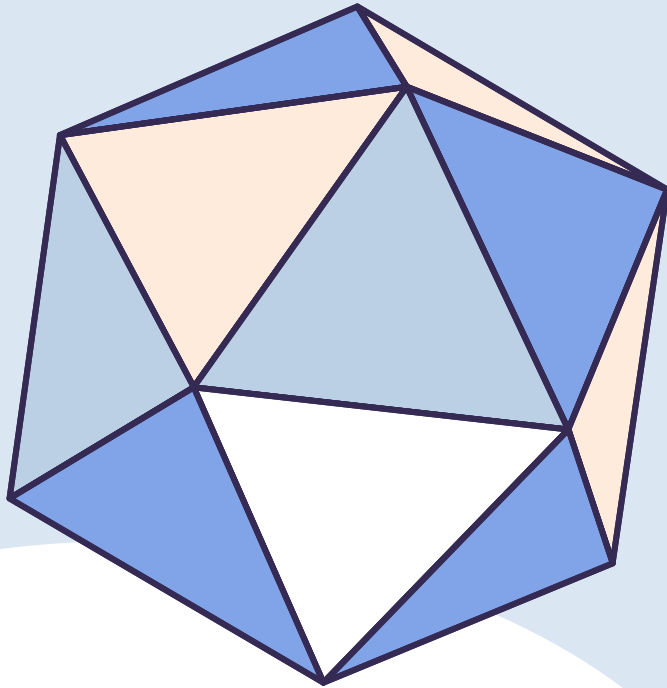
04 Improvements

05 Conclusion

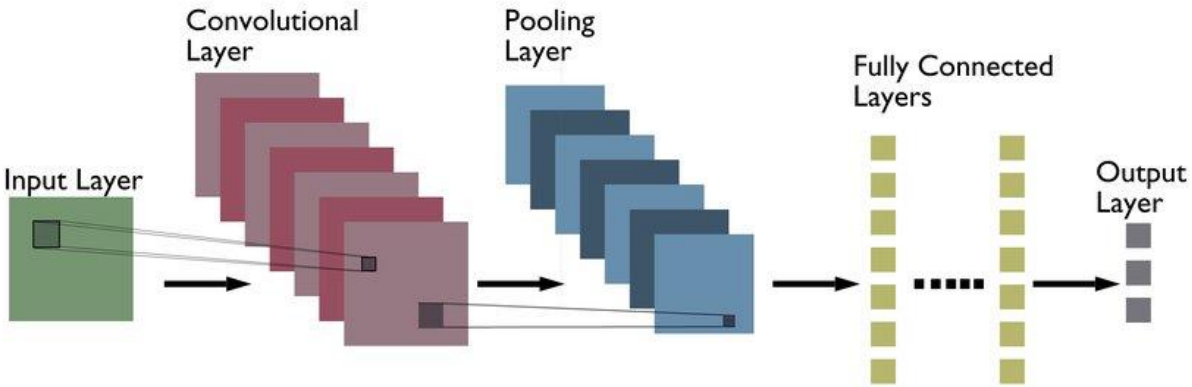
Approaches to Image Retrieval & why the fail

Text based image retrieval approaches	▪ Manual annotation ▪ No real understanding of image content
Low-level features based approaches	▪ Partial Representations ▪ Need of extensive pre and post processing
Local features based approaches	▪ Computationally Expensive

Efficient-Nets and convolutional Features



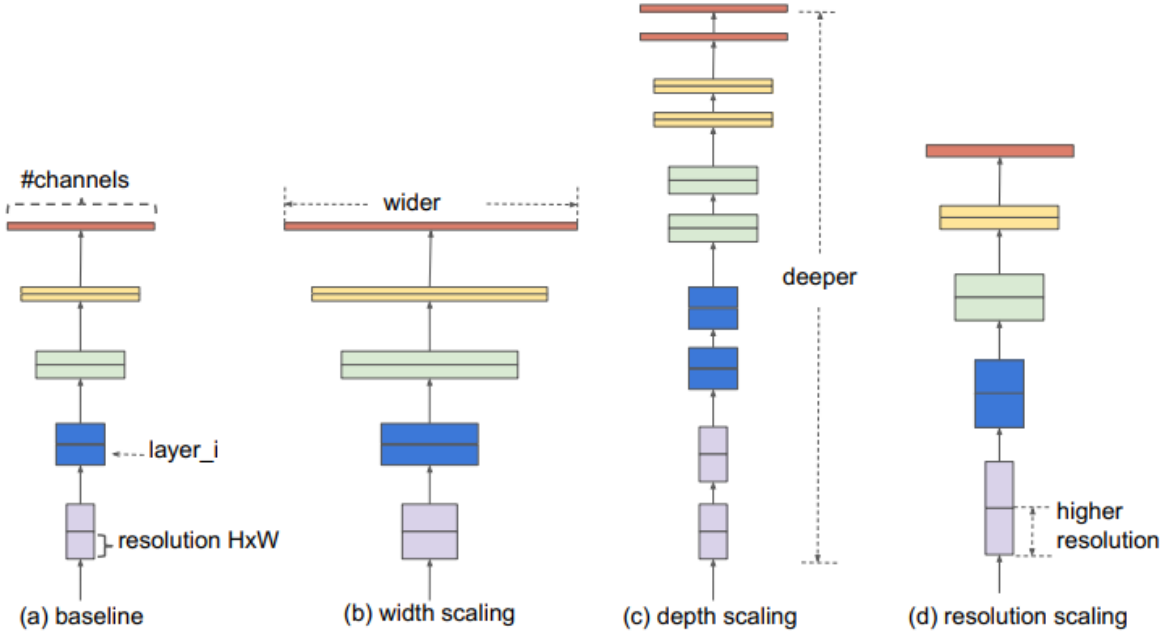
Convolutional Features?



Efficient Nets Key Concepts

1) Compound Model
2) Depth Wise separable Convolutions
3) Linear Bottlenecks
4) Inverted Residual Blocks

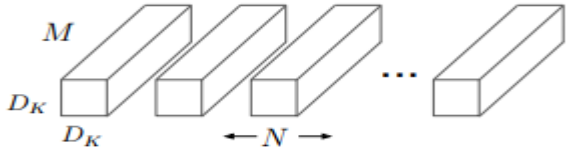
1. Compound model scaling



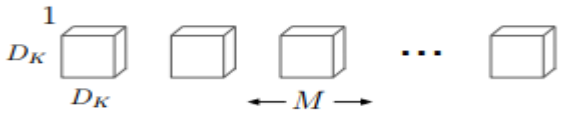
2. Depth wise separable convolutions

- Factorize the standard convolutions to:

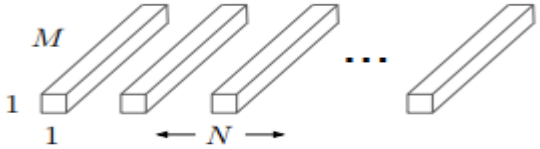
1) Depth-wise convolutions
2) Point-wise convolutions



(a) Standard Convolution Filters

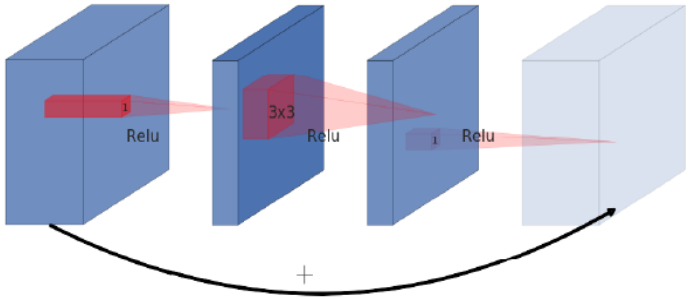


(b) Depthwise Convolutional Filters

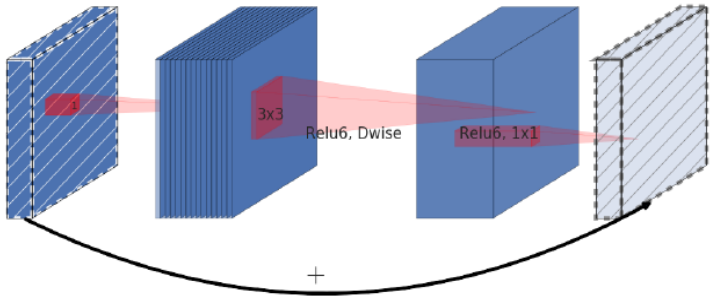


(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

3. Linear bottlenecks & 4. Inverted Residuals



Classic Residuals



Inverted Residuals

Metric

Mean Average Precision

- Calculate Precision for each relevant image at each query

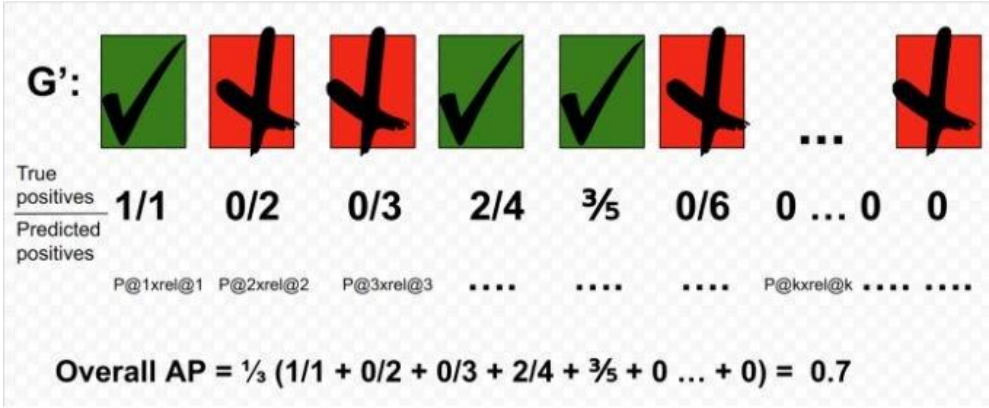
$$Precision = \frac{TP}{TP+FP}$$

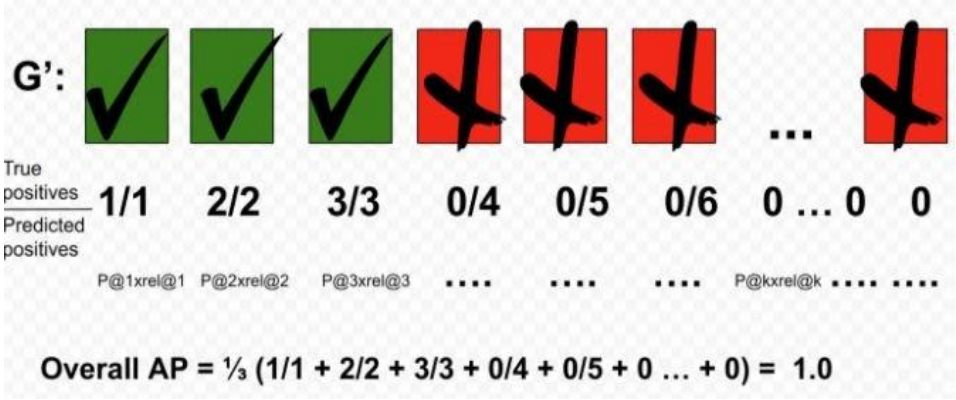
- Calculate the Average of Precision for each query:

$$AP@k = \frac{1}{GTP}\sum_{k}^{n} P@k\ Rel@k$$

- Calculate the Mean of the Average Precisions :

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$


G':
True positives / Predicted positives
1/1
0/2
0/3
2/4
3/5
0/6
0 … 0
0
P@1xrel@1
P@2xrel@2
P@3xrel@3
P@kxrel@k
Overall AP = ⅓ (1/1 + 0/2 + 0/3 + 2/4 + 3/5 + 0 … + 0) = 0.7


G':
True positives / Predicted positives
1/1
2/2
3/3
0/4
0/5
0/6
0 … 0
0
P@1xrel@1
P@2xrel@2
P@3xrel@3
P@kxrel@k
Overall AP = ⅓ (1/1 + 2/2 + 3/3 + 0/4 + 0/5 + 0 … + 0) = 1.0

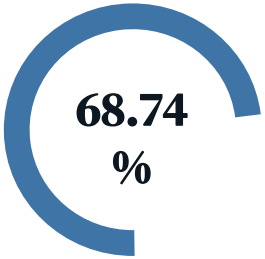
Benchmarking Dataset

Paris6k

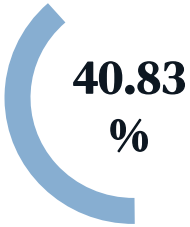
Image Retrieval Based on Convolutional Features

Model	mAP@10 (first 10 ranked images)	mAP@100 (first 100 ranked images)	mAP (at all dataset)	Length of feature vector	No. of parameters
VGG-16 Max-Pooling	92.41%	66.37%	60.3%	512	138,357,544
VGG-16 Average-Pooling	84.26%	45.89%	40.83%	512	138,357,544
ResNet-50 Max-Pooling	81.67%	45.94%	42.53%	2048	25,636,712
ResNet-50 Average-Pooling	52.06%	56.3%	52.06%	2048	25,636,712
EfficientNet-B1 Max-Pooling	89.44%	59.94%	54.18%	1280	7,856,239
EfficientNet-B1 Average-Pooling	93.15%	71.3%	68.74%	1280	7,856,239

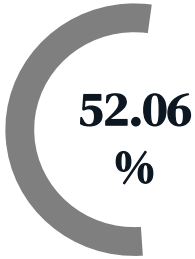
Snap Shot of Baseline Results



EfficientNet-B1 using Average Pooling



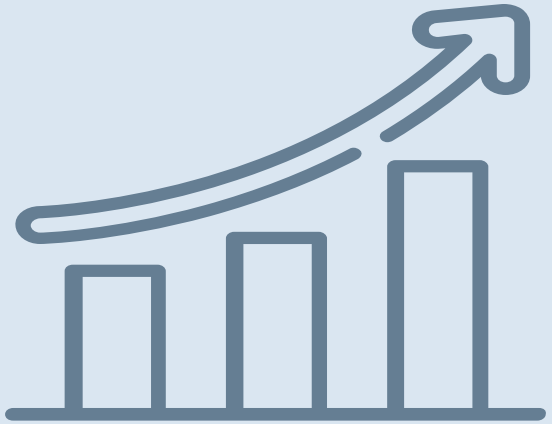
VGG-16 using Average Pooling



ResNet-50 using Average Pooling

*All mAP results are calculated for the entire dataset

Improvements: R-MAC Vectors



Why R-MAC?

- CNN models provides end-to-end feature descriptors that outperforms pre-CNN feature descriptors, but the problem with using traditional max/avg pooling is that these model aren't geometry-aware of objects location in the images.

- R-MAC pooling provide better object localization matching for given images, this done by Encoding several image regions and performing simple aggregation methods.

Obtaining R-MAC Vectors

- Set of R square regions of different sizes are defined on our feature maps, The number of square regions (R) is dependent on the pyramid level L specified :
  For Example: **L=3** the feature map is divided 3 times at each level **n=3,n=2,n=1**

$$R_n = 4^{(n-1)}$$

$$windowSize = \lceil a/n \rceil$$

$$strideSize = \left\lfloor \frac{a}{n} \right\rfloor$$

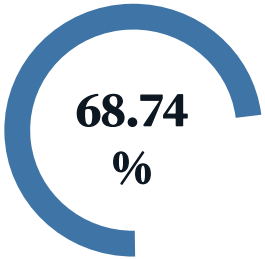
Cont.

- For each square region in the feature map the corresponding the feature vector is obtained.
- An $\boldsymbol{L}_2$-normalization is carried before implementing the PCA.
- PCA whitening is carried on the obtained vectors.
- Another $\boldsymbol{L}_2$-normalization is carried as a final post processing measure for the R-MAC vector.
- Final step, is to aggregate all regional feature vector into one vector.

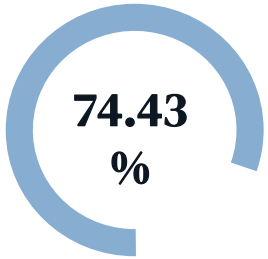
Obtained R-MAC Results

Model	mAP@10 (first 10 ranked images)	mAP@100 (first 100 ranked images)	mAP (at all dataset)	Length of feature vector	No. of parameters
VGG-16/R-MAC	80.19%	71.06%	63.69%	512	138,357,544
ResNet-50/R-MAC	95%	71.52%	67.74%	2048	25,636,712
EfficientNet-B1/R-MAC	94.81%	75.17%	74.43%	1280	7,856,239

Snap shot comparison



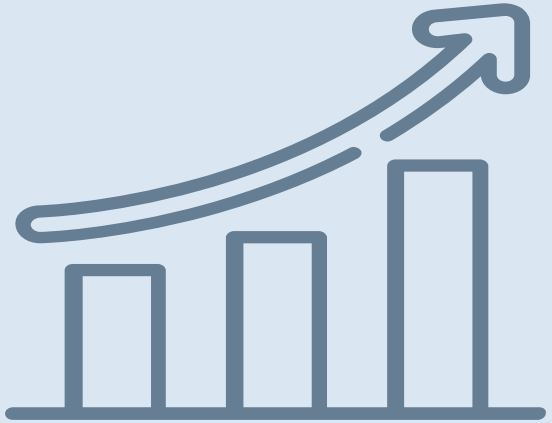
EfficientNet-B1 using Average Pooling



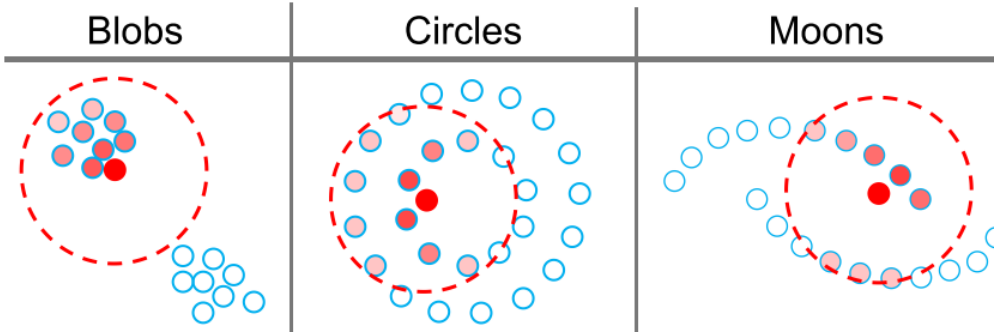
EfficientNet-B1 With R-MAC

*All mAP results are calculated for the entire dataset

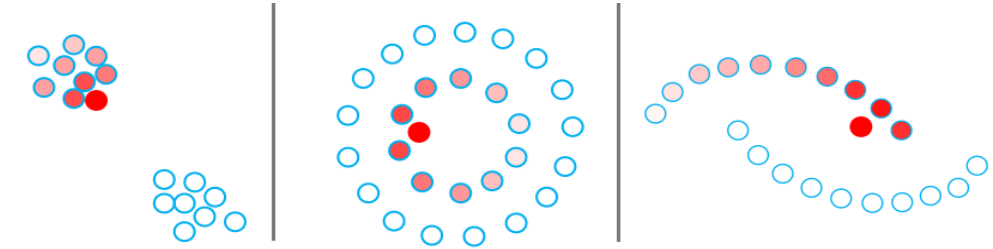
Improvements: Diffusion Process



- ***Problem with traditional ranking techniques***: Regular image matching techniques (e.g. K-nearest neighbor) fails when the manifolds of the data don't resemble a circular shape



- ***Solution proposed by Diffusion ranking technique*** **:** Diffusion is a mechanism used to captures the data manifold in the feature space



A **random walk** on a graph is a process that begins at some vertex, and at each time step moves to another vertex.”

- When the graph is **unweighted**, the walk moves to a chosen uniformly at random among the neighbors of the present vertex.
- When the graph is **weighted**, it moves to the neighbor with the highest probability proportional to the weight of the corresponding edge

  A special case of random walks are **lazy walks**; in which the walk stays put with probability 1/2 at each time step, and walk to a random neighbor the other half of the time.

  (always converges to a stable distribution in which every vertex is visited with probability proportional to its weighted degree)

Step 1: After feature extraction we construct an affinity matrix A such that

$$For\ a\ dataset\ \mathcal{X} := \{x_1, x_2, \dots, x_n\} \subset \mathbb{R}^d$$
$$a_{ij} := s\left(x_i, x_j\right) \quad , \qquad \forall\ (i,j) \in [n]^2$$
$$where\ s(x_i, x_j)\ is\ a\ similartiy\ measure$$

Step 2: calculate the diagonal matrix D

$$D = diag(A1_n)\ \ diagonal\ matrix\ with\ the\ row - wise\ sum\ of\ A$$

Step 3: Symmetrically normalize A to obtain the transition or stochastic matrix S :

$$S := D^{-1/2} A D^{-1/2}$$
$$D = diag(A1_n)\ \ diagonal\ matrix\ with\ the\ row - wise\ sum\ of\ A$$

Step 4: Define a query vector y, such that:

$$y = (y_i) \in \mathbb{R}^n \ \ specifies\ a\ set\ of\ query\ points\ in\ \mathcal{X}\ with:$$
$$y_i = 1\ , \qquad if\ \ x_i\ \ is\ a\ query$$
$$y_i = 0\ , \qquad otherwise$$

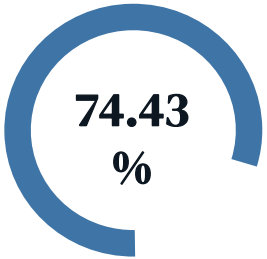
Step 5: Using matrix *S* and **y** we define the following 'random walk' on the graph: with probability *α* one jumps to an adjacent vertex according to distribution stated in *S*, and with 1 - *α* to a query point uniformly at random. the diffusion mechanism carries out such that, given an initial vector f^0, it iterates according to:

$$f^t = \alpha S f^{t-1} + (1-\alpha)y$$

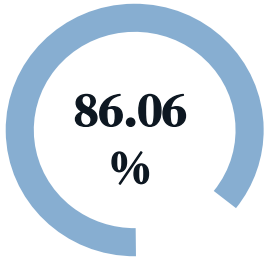
Obtained Diffusion Results

Model	mAP@10 (first 10 ranked images)	mAP@100 (first 100 ranked images)	mAP (at all dataset)	Length of feature vector
VGG-16 R-MAC+Diff	80%	80.07%	78.11%	512
ResNet-50 R-MAC+Diff	95%	82.43%	82.54%	2048
EfficientNet-B1 R-MAC+Diff	94.81%	83.65%	86.03%	1280

Snap shot comparison



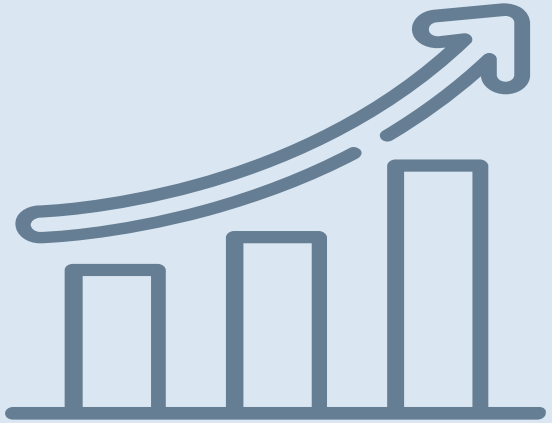
EfficientNet-B1
with R-MAC



EfficientNet-B1
With R-MAC and
Diffusion

*All mAP results are calculated for the entire dataset

Fine Tuning



- ***How Fine – Tuning is carried***:

Freeze all layers except the final classification layer, replace it with new layers and train them on top of the existing pretrained network, this can enhance detection for specific data (i.e. to improve landmarks detection; one would use a dataset of landmarks for fine tuning).

- ***Limitations***

  - Resource constraints (Local download, RAM limitations)
  - Dataset is too large → using small portions leads to overfitting

Conclusion

- Efficient-Nets have a very promising claim to dominate the field of retrieval tasks, as they are fast, very computationally efficient and provide better accuracy.
- Efficient content image retrieval depends on various factors, not only the features used as previously advertised by older less intuitive systems, thus considerable improvements can be realized after understanding the retrieval pipeline

Questions?