Fashion Content-based Image Retrieval Project

Cimmino Fabio 807070
Lotterio Roberto 807500
Puleri Gianluca 807064

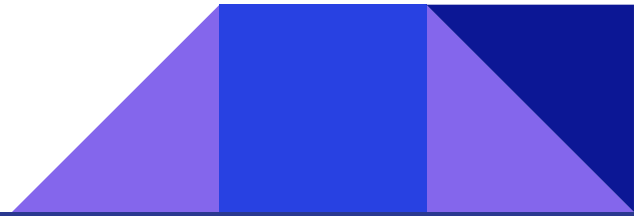
Introduzione

L’obiettivo del progetto è quello di realizzare un sistema che consenta di recuperare le immagini più simili da un database data una immagine query.

L’immagine query verrà inviata tramite smartphone utilizzando un bot di Telegram.

Le immagini più simili verranno restituite sul bot di Telegram.

Per effettuare il matching tra le immagini sono stati implementati 4 metodi di features extraction.



Preprocessing dataset

Il dataset utilizzato è composto da immagini appartenenti all'ambito del fashion e reperito dalla piattaforma Kaggle.

Dal dataset originale sono state eliminate le classi con un numero non sufficientemente alto di immagini. Inoltre sono state rimosse le classi di immagini poco inerenti al fashion (sport equipment, skin care, water bottle, etc.)

Infine sono state unite quelle classi la cui distinzione risultava ambigua anche per l'occhio umano.

Bottomwear

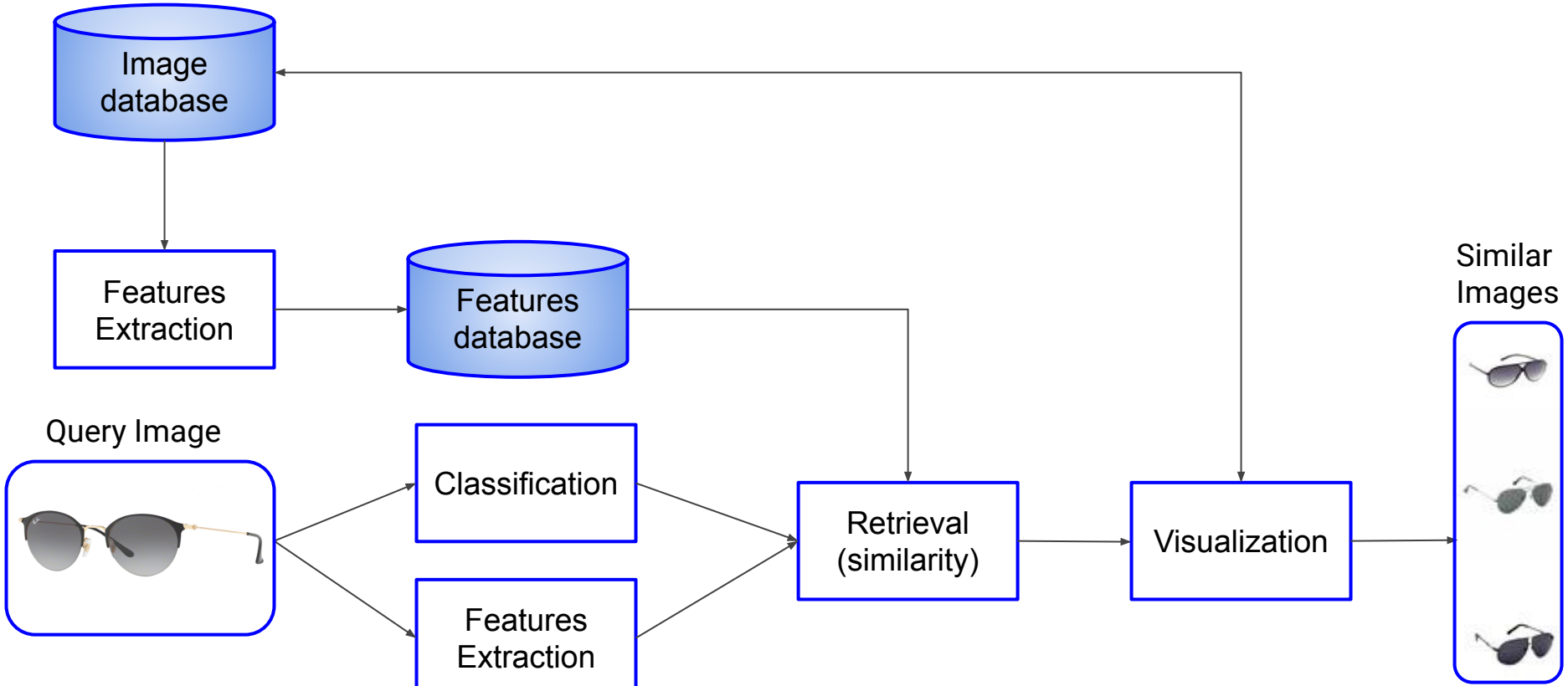
Loungewear

Classi del dataset

Il dataset contiene 10 mila immagini suddivise in 20 classi:

- Apparel Set
- Bags and Wallets
- Belts
- Bottomwear
- Cufflinks
- Dress
- Eyewear
- Flip Flops
- Fragrance
- Headwear
- Innerwear
- Jewellery
- Sandal
- Saree
- Scarves
- Shoes
- Socks
- Ties
- Topwear
- Watches

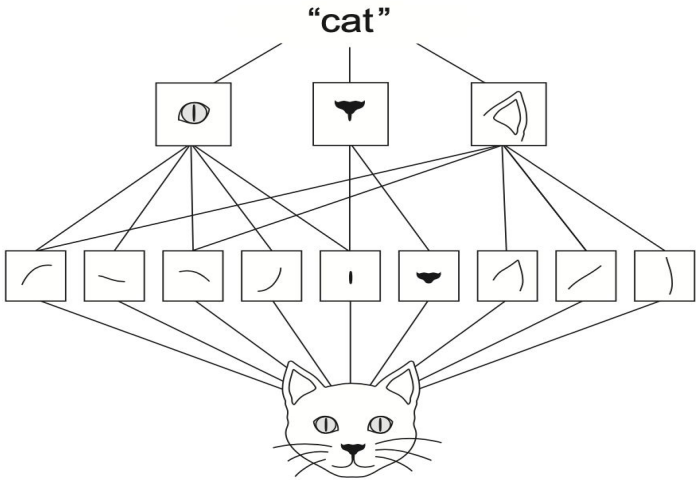
Solution Design



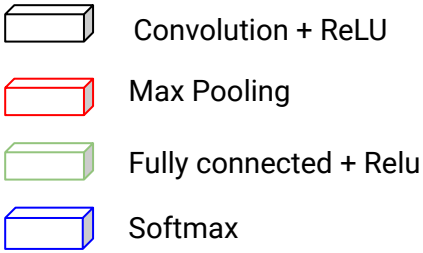
Classification

Per la classificazione dell’immagine query abbiamo scelto e implementato una rete neurale convoluzionale (CNN) per le seguenti proprietà:

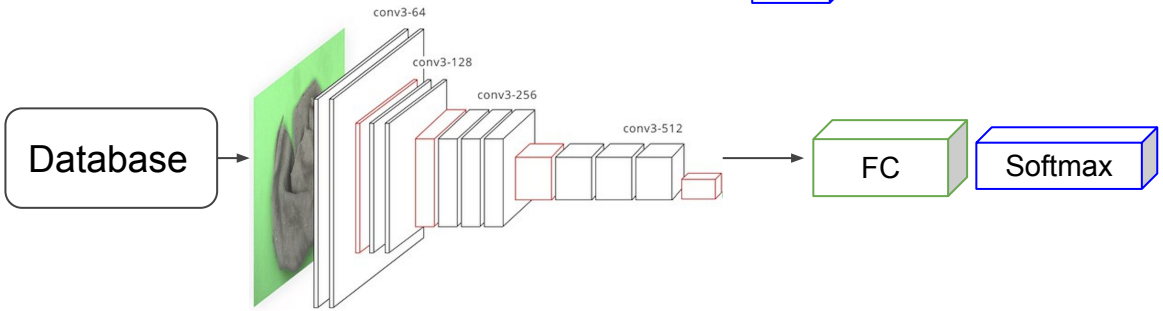
- I pattern che la rete apprende sono invarianti alle traslazioni
- Apprendono gerarchie spaziali di pattern



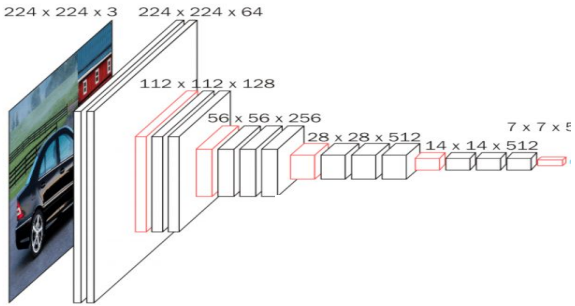
Architettura CNN



VGG-16



VGG-19



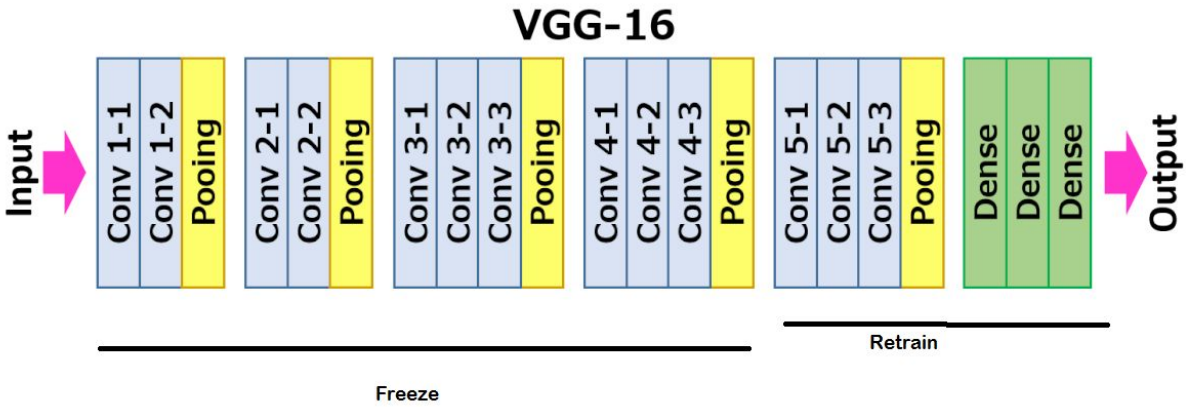
Abbiamo messo a confronto due architetture che differiscono nel tipo di base convoluzionale usata, entrambe pre-trainate su ImageNet dataset: VGG-16 e VGG-19

Il training è stato effettuato congelando i pesi della base convoluzionale e trainando sulla Fully Connected, infine è stato eseguito un fine-tuning.

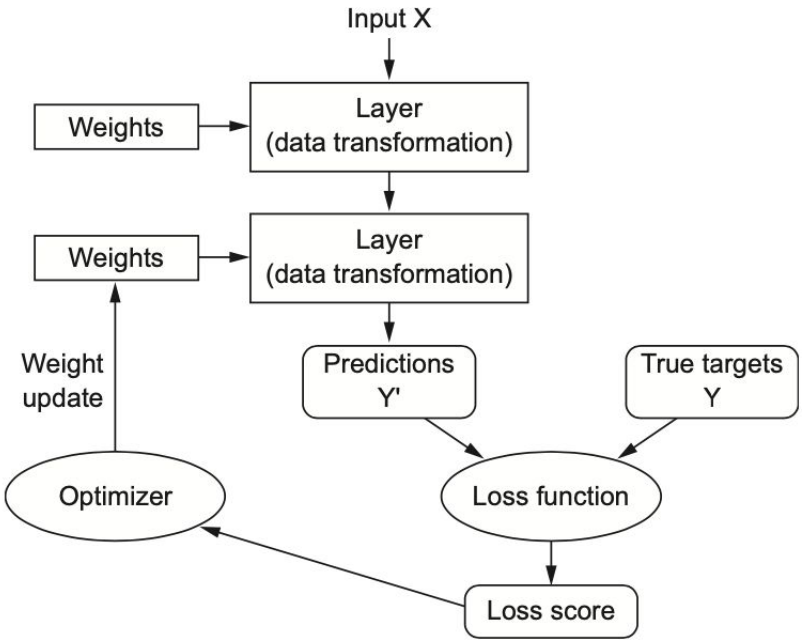
Fine-Tuning

Il fine-tuning consiste nello scongelare i pesi del blocco più vicino al FC, precedentemente congelati, per poi effettuare un re-train.

Questa procedura viene svolta per “ritoccare” i pesi del nostro modello.



Ottimizzatori



Per aggiornare i pesi di ogni layer è fondamentale la scelta dell'optimizer.

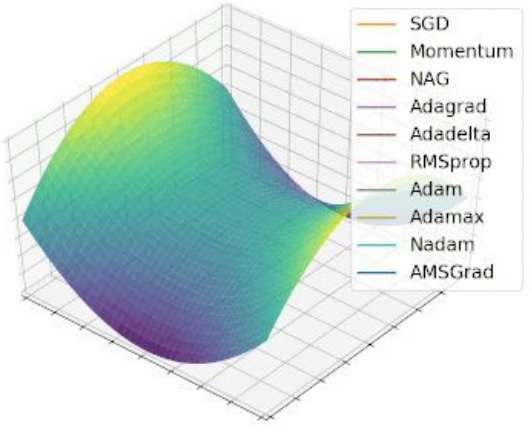
L' ottimizzatore implementa l'algoritmo chiamato backpropagation, quindi in base al valore del loss score provvede a regolare i pesi minimizzando il valore della loss.

Confronto Ottimizzatori

Sono stati confrontati 3 ottimizzatori su entrambi i modelli di CNN:

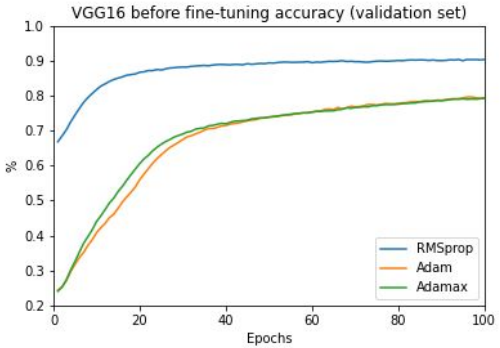
- RMSprop
- Adam
- Adamax

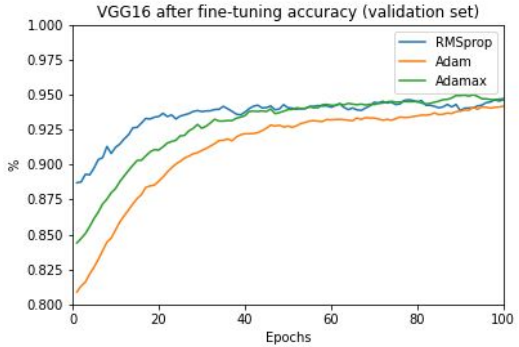
Per il confronto è stato utilizzato un valore pari a 2e-4 per il learning rate di training e un valore pari a 1e-4 per quello di fine-tuning.



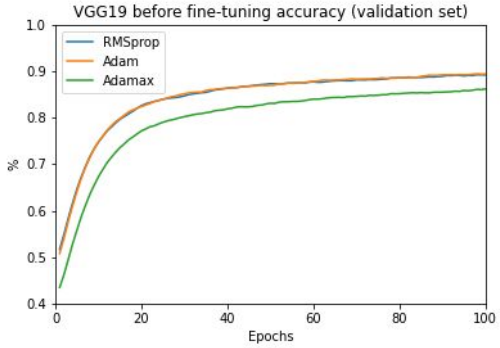
Confronto Ottimizzatori

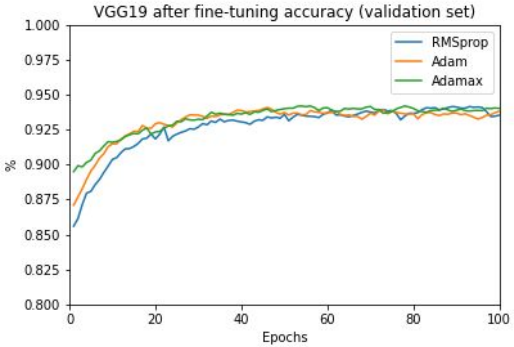
VGG-16





VGG-19





Il modello migliore risulta essere quello che usa la base convoluzionale VGG-16 con l'ottimizzatore RMSprop.

È stata ottenuta un'accuracy sul testset di 95,2%

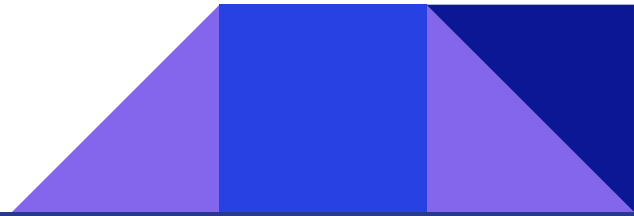
Image Features

Bag of visual words con SIFT feature

Bag of Visual Words

Il primo metodo implementato consiste nei seguenti passaggi:

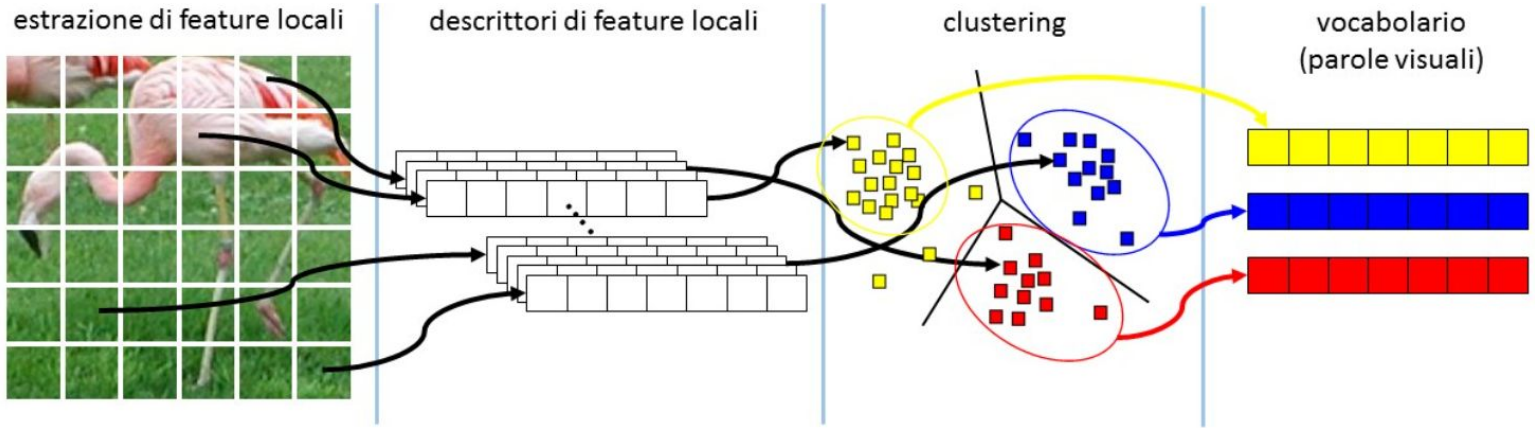
1. costruzione di un modello di tipo Bag Of Visual Words (BOVW)
2. usare il modello BOVW creato per la rappresentazione delle immagini
3. utilizzare le rappresentazioni delle immagini per effettuare il retrieval delle immagini basato sul contenuto visuale



Step 1: Costruzione del modello BOVW con SIFT features

E’ stato costruito un modello BOVW, che sarà un dizionario di “parole visuali”, apprendendo il vocabolario in maniera non supervisionata:

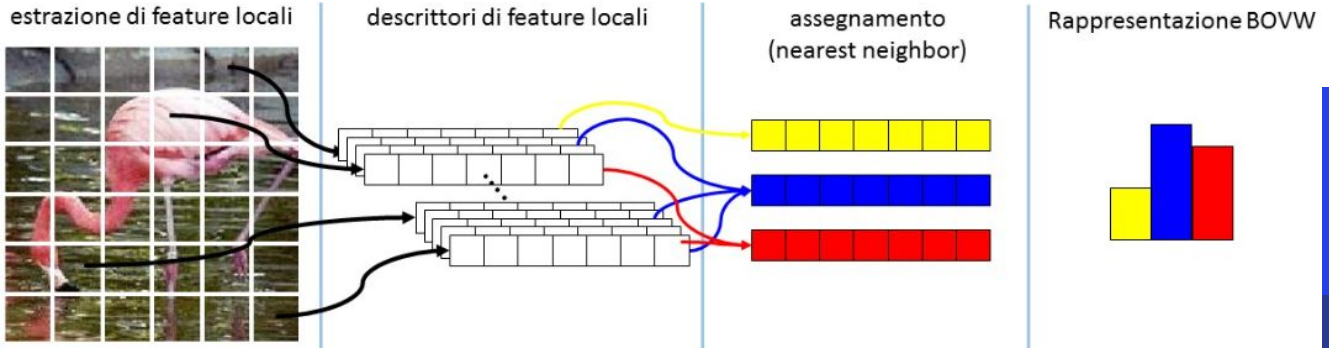
1. Estrazione delle SIFT features dalle immagini (descrittori 128-dimensionali)
2. Utilizzo di tecniche di clustering (K-means) sull’insieme di features estratte



Step 2: Rappresentazione delle immagini tramite BOVW

E’ stato utilizzato il vocabolario appreso mediante K-means per descrivere ogni singola immagine:

1. Ogni descrittore estratto dall’immagine viene associato al centroide più vicino
2. Si crea un istogramma che conta il numero di feature locali associate a ogni parola del dizionario
3. Normalizzazione dell’istogramma



Normalizzazione con TF-IDF

La componente generica del vettore di rappresentazione (non normalizzato) rappresenta la frequenza della parola nell'immagine: $x_i = tf(d, t_i)$

Misurazione dell'importanza di una parola con la IDF: $idf(t_i) = \log \frac{n}{1 + df(t_i)}$

Costruzione del vettore trasformato secondo la codifica TF-IDF: $\overline{x}_i = x_i \cdot idf(t_i)$

Infine il vettore viene ulteriormente normalizzato con norma L-2: $\overline{X} = \frac{X'}{\sqrt{\sum_i x_i^2}}$

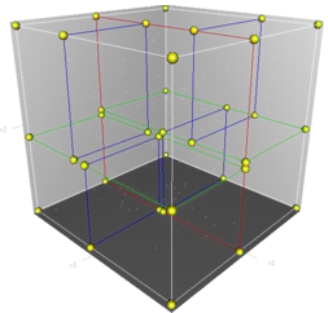
Step 3: Retrieval delle immagini

Il vettore di rappresentazione creato, a partire da un'immagine, in base al suo contenuto visuale può essere sfruttato per costruire il meccanismo di image retrieval.

Immagini dal contenuto simile hanno features simili (vettori "vicini" tra loro).

Quindi, data la rappresentazione di un'immagine query, viene calcolata la distanza euclidea rispetto a tutti i vettori di rappresentazione del feature DB.

Per il calcolo della distanza è stato utilizzato un k-d tree.

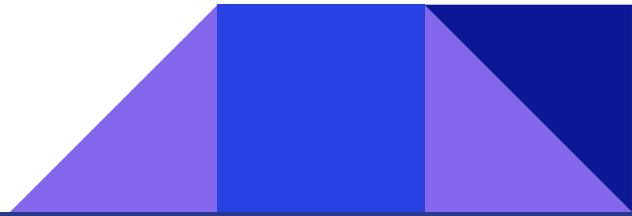


SIFT Features

L’algoritmo SIFT

SIFT individua dei keypoints fornendo delle informazioni quantitative (i cosiddetti descrittori) che possono essere utilizzati per il riconoscimento di oggetti.
L’algoritmo consiste nei seguenti passaggi:

1. Costruzione dello spazio scala
2. Approssimazione LoG
3. Ricerca dei keypoints
4. Eliminare i bad keypoints
5. Assegnare l’orientamento ai keypoints
6. Generare le SIFT features



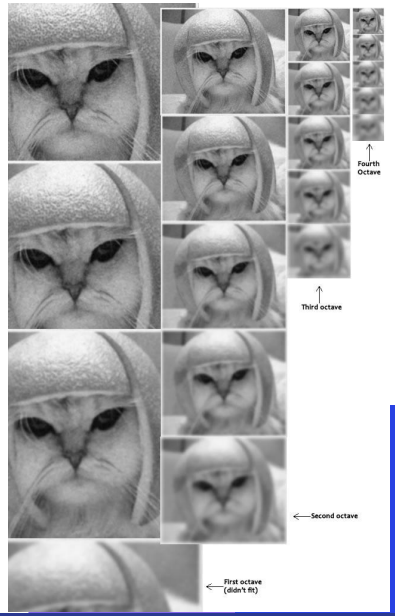
1 - Costruzione dello spazio scala

Il primo passo consiste nella costruzione di uno Scale Space tramite un filtro Gaussiano applicato ricorsivamente ad ogni livello.

$$L(x, y; \sigma) = G(x, y; \sigma) * I(x, y)$$

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} * e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

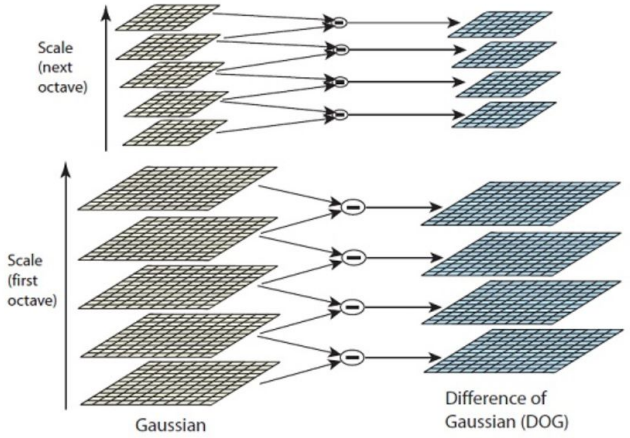
La sfocatura attenua il rumore e stabilizza la derivata del secondo ordine del prossimo step



2 - Approssimazione LoG

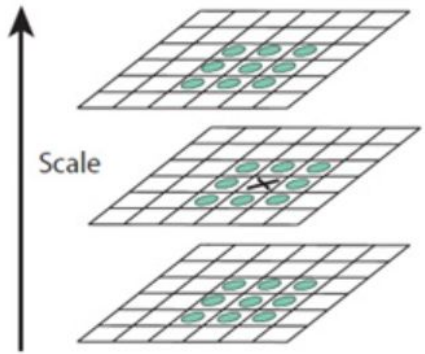
La derivata del secondo ordine (o “laplaciana”) consente di individuare i bordi e gli angoli dell’immagine utili per trovare i keypoints.

Il calcolo di tutte le derivate di secondo ordine è un’operazione molto intensiva a livello computazionale. Perciò utilizziamo lo Scale Space per generare le LoG velocemente nel seguente modo:



3 - Ricerca dei keypoints

A. Vengono individuati approssimativamente i massimi e minimi delle DoG.



X è un candidato keypoint se è il più grande o il meno di tutti i suoi 26 vicini.

B. Perfezioniamo le coordinate discrete degli estremi trovati:

$$D(s) = D + \frac{\partial D^T}{\partial s} s + \frac{1}{2} s^T \frac{\partial^2 D}{\partial s^2} s$$

4 - Eliminazione dei bad keypoints

In questa fase vengono eliminati i keypoint:

- che hanno un basso contrasto
  - valore assoluto dell'intensità inferiore ad una soglia
- che giacciono su un bordo
  - vengono calcolati due gradienti nel keypoint tra di loro perpendicolari. Intorno al keypoint l'immagine può essere:
    - Una regione piatta (entrambi i gradienti saranno piccoli) ⟹ Scartiamo il keypoint
    - Un bordo (un gradiente grande e l'altro piccolo) ⟹ Scartiamo il keypoint
    - Un angolo (entrambi i gradienti saranno grandi) ⟹ Teniamo il keypoint

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

$$Tr(H) = D_{xx} + D_{yy} = a + b$$

$$Det(H) = D_{xx} * D_{yy} - (D_{xy})^2 = a * b$$

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

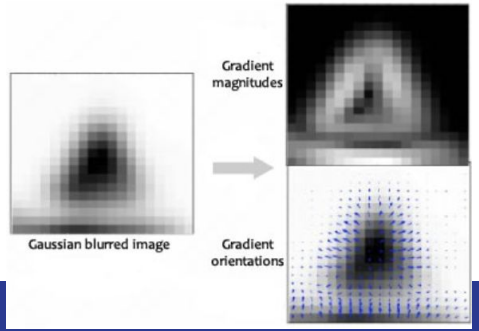
5 - Assegnazione dell’orientamento

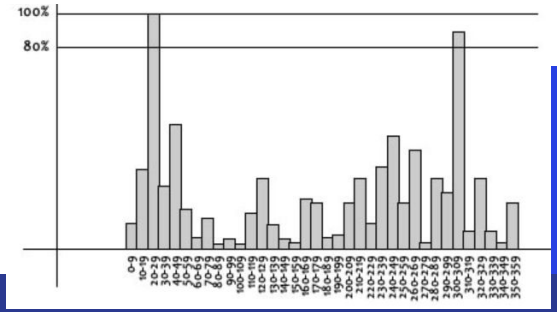
L’assegnazione dell’orientamento fornisce l’invarianza di rotazione.
L’idea è quella di calcolare le grandezze e direzioni dei gradienti attorno ad ogni keypoint.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

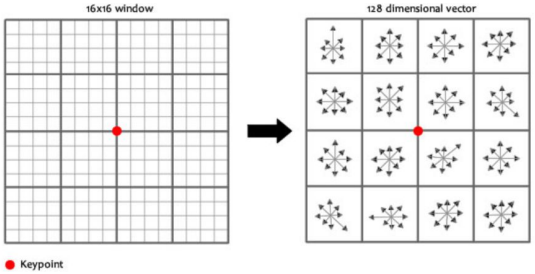
Viene creato un istogramma con 36 bins (10 gradi ogni bin). La quantità che viene aggiunta ad un bin è proporzionale alla grandezza del gradiente in quel punto.



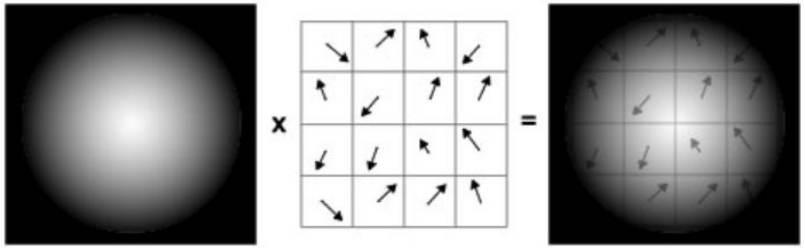


6 - Generazione delle features

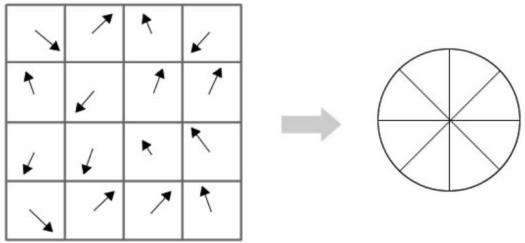
A.



B.



C.



D.

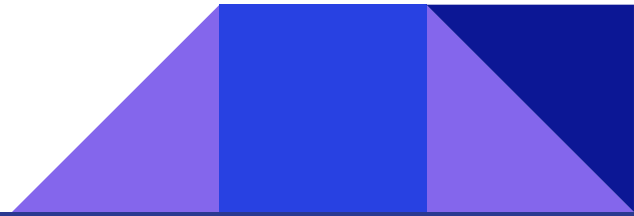
Calcolo di 8 bins per le 16 regioni 4x4.
Otterremo quindi 4x4x8= 128 numeri

Daisy Features

Daisy

Daisy è differente dagli altri algoritmi, non solo per quel che riguarda la sua implementazione, ma soprattutto per il campo di utilizzo; è infatti un approccio utilizzato nel dense wide-baseline matching.

Questo metodo utilizza dei descrittori locali simili a SIFT, tuttavia non essendo stato creato per il matching sparso, Daisy risulta computazionalmente più veloce e leggero per quello denso.



Daisy

Nel descrittore Daisy, le convoluzioni dell'immagine originale con derivate orientate di filtri Gaussiani, prendono il posto delle somme pesate di SIFT.

Una convolved orientation map si calcola con la seguente formula:

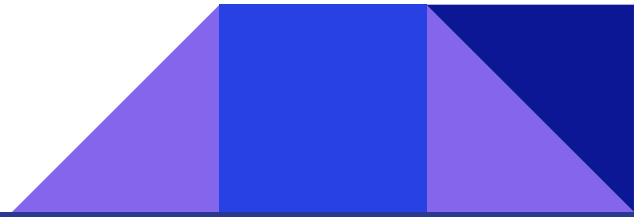
$$G_O^{\Sigma} = G_{\Sigma} \cdot \left(\frac{\partial I}{\partial o}\right)^{+}$$

Si costruisce il descrittore leggendo i valori nelle convolved orientation map.

L’algoritmo Daisy

Sull’immagine di input, per ogni pixel, vengono calcolate 8 orientation map, una per ogni direzione da rappresentare.

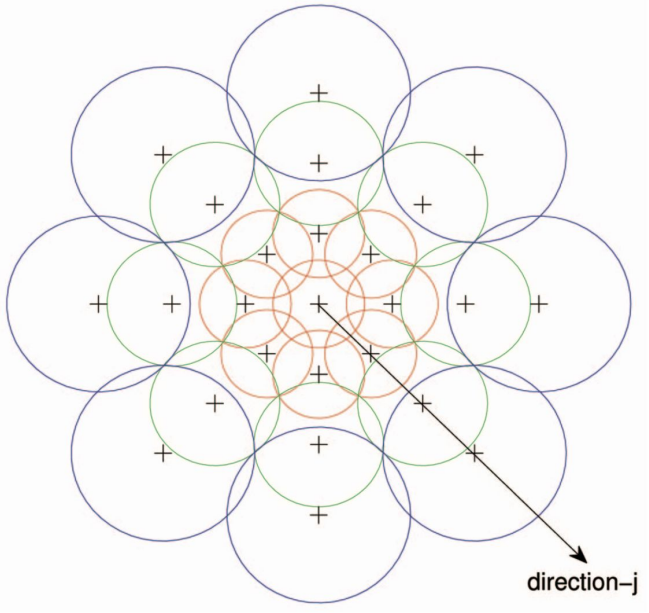
Di ogni orientation map viene effettuata una convoluzione ripetuta con kernel gaussiani di dimensioni differenti per ottenere convolved oriented maps per regioni di diverse grandezze.



Il descrittore Daisy

Si può vedere come per ogni pixel vengono scelte 8 direzioni lungo le quali calcolare le orientation maps per diverse regioni.

Ogni cerchio dell’immagine descrive una regione con raggio proporzionale alla deviazione standard del kernel Gaussiano.



Il descrittore Daisy completo

Dopo la convoluzione da parte di un kernel Gaussiano si ottiene il vettore:

$$h_{\sum}(u,v) = [G_1^{\sum}(u,v), ..., G_8^{\sum}(u,v)]$$

Il suo corrispettivo normalizzato è: $\tilde{h}_{\sum}(u,v)$

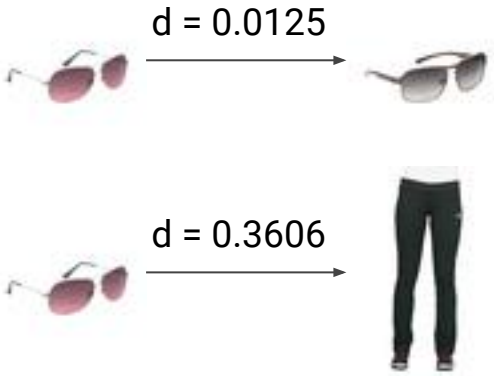
La concatenazione di vettori $\tilde{h}$ costituisce il descrittore Daisy completo:

$$\begin{aligned} D(u_0,v_0) = [&\tilde{h}_{\sum_1}(u_0,v_0),\ \tilde{h}_{\sum_1}(l_1(u_0,v_0,R_1)), ..., \tilde{h}_{\sum_1}(l_N(u_0,v_0,R_1)),\\ &\tilde{h}_{\sum_2}(l_1(u_0,v_0,R_2)), ..., \tilde{h}_{\sum_2}(l_N(u_0,v_0,R_2)),\\ &\tilde{h}_{\sum_3}(l_1(u_0,v_0,R_3)), ..., \tilde{h}_{\sum_3}(l_N(u_0,v_0,R_3))]^T \end{aligned}$$

Calcolo distanza vettori features

Per ottenere le 10 immagini con maggiore somiglianza all'immagine query è stata calcolata la cosine similarity tra: il vettore features dell'immagine query e i vettori features delle immagini del database.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}.$$



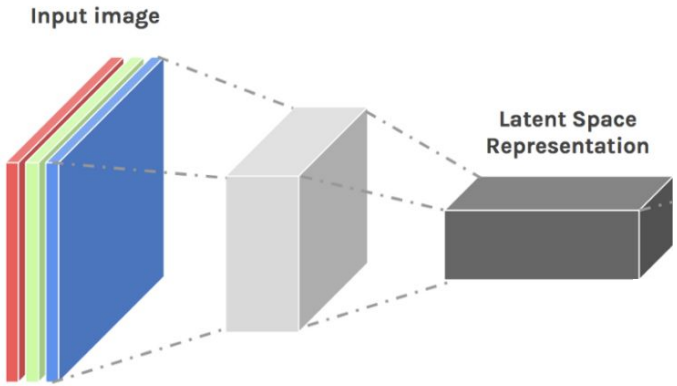
Learned Features: CNN-based

Deep Method

Deep Method

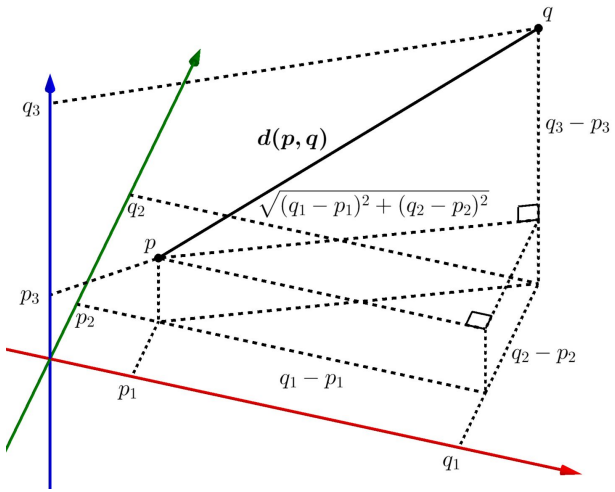
Il terzo metodo utilizzato per il retrieve delle immagini è il Deep Method, utilizzando la base convoluzionale della CNN sviluppata.

Data un'immagine query, questa viene inviata alla base convoluzionale per estrarne le features (latent space representation)



Deep Method

Come misura di similarità è stata scelta la distanza euclidea, effettuata per calcolare la differenza tra il vettore features della query con i vettori features del database.



$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

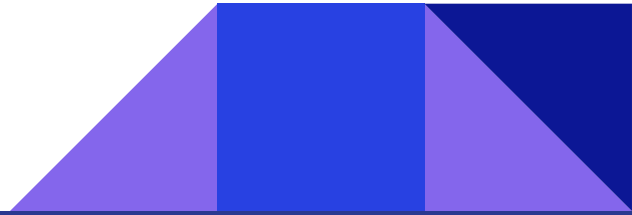
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Color Features

Steps Color Features Extraction

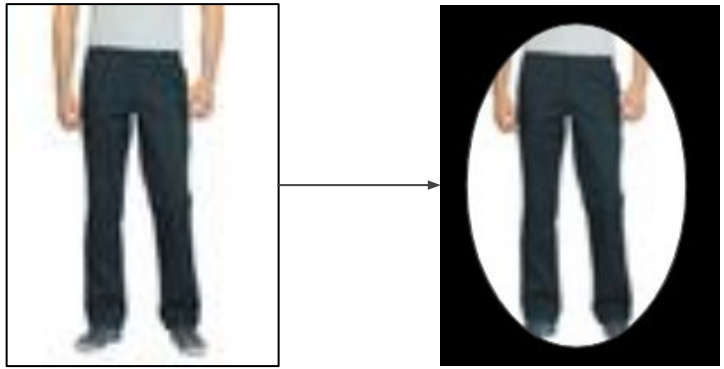
Il quarto metodo implementato consiste nei seguenti passaggi:

1. definizione del descrittore Color RGB
2. estrazione delle features del dataset e della query tramite utilizzo del descrittore
3. applicazione della metrica di similarità (Chi-Quadrato) tra dataset e query



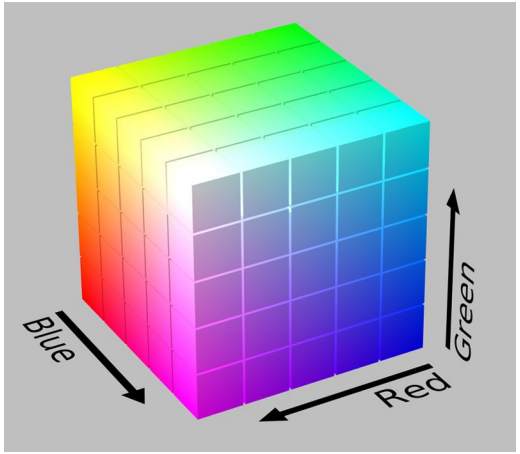
1 - Descrittore RGB

Su ogni immagine viene disegnata un’ellisse di altezza/larghezza pari al 90% dell’immagine in modo da attenuare il rumore creato dallo sfondo e/o da elementi di contorno.



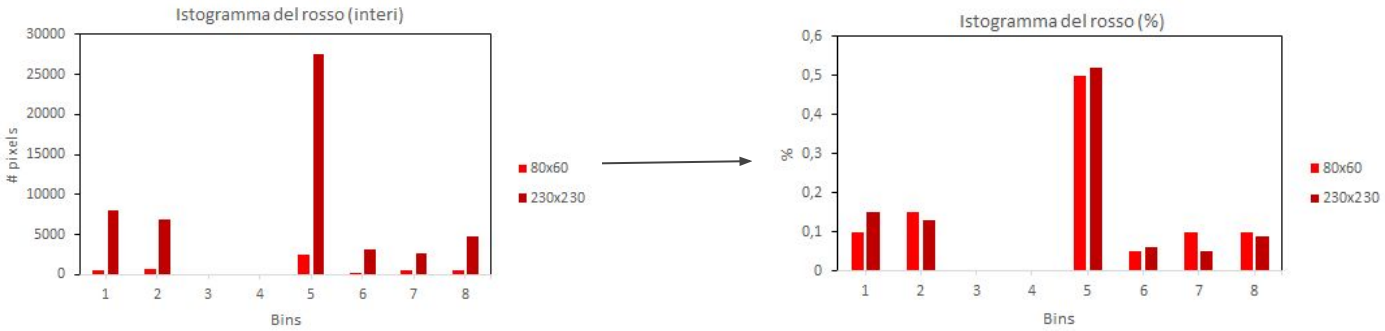
1 - Descrittore RGB

Il descrittore di immagini è un istogramma RGB 3D dei colori con 8 “bins” per ogni canale (rosso, verde e blu).



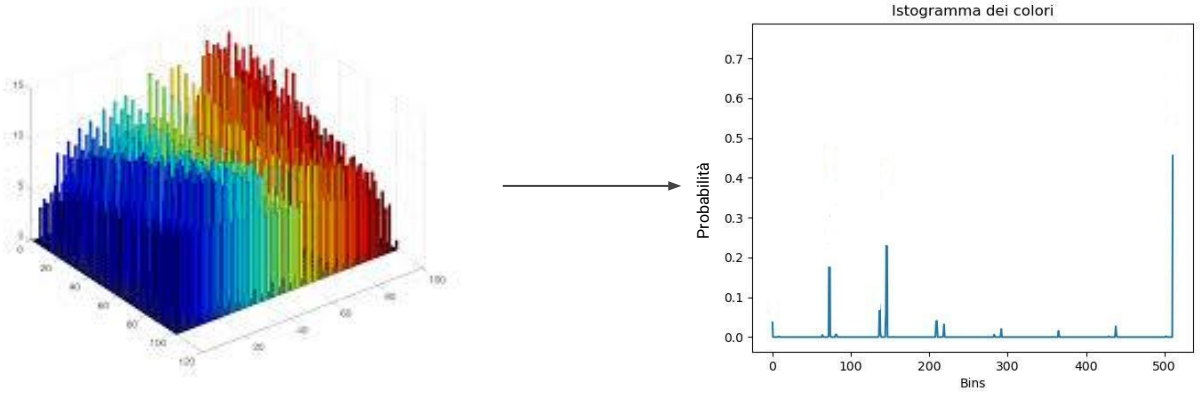
1 - Descrittore RGB

Ogni istogramma è normalizzato per ottenere l'invarianza della scala. Gli istogrammi sono rappresentati dai conteggi percentuali relativi ad uno specifico bin e non dai conteggi interi.



2 - Estrazione features

Utilizzando un descrittore con 8 “bins” per ogni canale, l’immagine è, dunque, rappresentata da un vettore 8x8x8=512-dimensionale.



3 - Applicazione distanza Chi-Quadrato

Per ottenere le top-10 immagini più simili viene utilizzata la distanza Chi-Quadrato, effettuata per calcolare la differenza tra il vettore features della query (a) con i vettori features del database (b).

$$chi2\_distance = \frac{1}{2}\sum_{i=1}^{n}\frac{(a_i - b_i)^2}{(a_i + b_i + \epsilon)}$$

Performance Evaluation

Misurazione performance

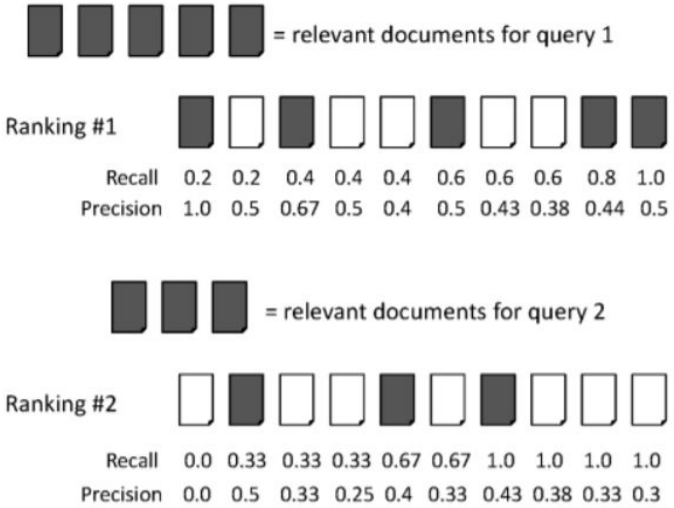
Come misura di valutazione delle performance è stata utilizzata la **Mean Average Precision:**

$$Precision@k = \frac{\#\text{ of recommended items @k that are relevant}}{k}$$

$$\text{AP@k} = \frac{1}{\text{GTP}} \sum_{i=1}^{k} \frac{\text{TP seen}}{i} \qquad \text{GTP} = \text{Ground Truth Positive}$$

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

Misurazione performance



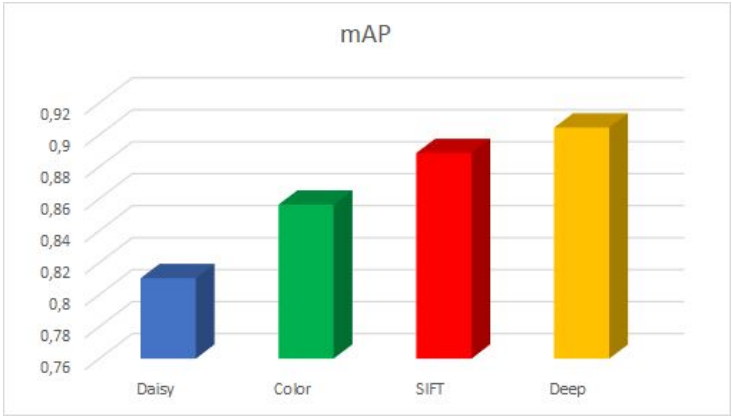
$$average\ precision\ query\ 1 = (1.0 + 0.67 + 0.5 + 0.44 + 0.5)/5 = 0.62$$
$$average\ precision\ query\ 2 = (0.5 + 0.4 + 0.43)/3 = 0.44$$

$$mean\ average\ precision = (0.62 + 0.44)/2 = 0.53$$

Confronto misurazione performance

Sono state scelte 20 immagini di query, una per ogni classe; 10 immagini prese dal database e 10 prese da internet.

È stata quindi calcolata la mAP per ogni metodo di feature extraction sulla base di queste 20 query.



Risultati

DAISY Results

1 2 3 4 5

6 7 8 9 10

Color Results

1 2 3 4 5

6 7 8 9 10

Query:

SIFT Results

1 2 3 4 5

6 7 8 9 10

Deep Method Results

1 2 3 4 5

6 7 8 9 10

DAISY Results

Color Results

Query:

SIFT Results

Deep Method Results

DAISY Results

Color Results

Query:

SIFT Results

Deep Method Results

Conclusioni

La parte iniziale di classificazione si è rivelata fondamentale per il retrieval delle immagini, di conseguenza tutti e 4 i metodi di feature extraction hanno avuto risultati soddisfacenti.

In particolare abbiamo potuto osservare che i descrittori CNN-based sono risultati, in media, più efficaci dei descrittori locali.

Il retrieval delle immagini è risultato ottimo quando la classe di appartenenza non presentava ulteriori sotto-classi.

Grazie per l’attenzione

Repository: https://gitlab.com/FabioCimmino/visual-information-processing