PP-Index Based Search Engine Using FaceNet with Facial Expressions Dataset

Federico Balestri, Stefano Petrocchi, Mirco Quintavalla, Giulio Silvestri

Datasets and Preprocessing

- **Facial Expression Dataset:** 35887 images belonging to 7 different classes
- **MirFlickr25k:** 25000 images without a specific class
- **Preprocessing:**
  - Image resized to $(160,160,3)$ in order to fit *FaceNet* input size
  - Creation of a *validation set* from the *training set* using 20% as *validation_split* (used in fine-tuning step)
  - Images normalization in the interval $[-1,1]$ using *mobilenet_v2.preprocess_input* function to fit *FaceNet* expected inputs

CLASS	TRAINING SET	PUBLIC TEST SET	PRIVATE TEST SET
Angry	3995 (13.91%)	467 (13.01%)	491 (13.68%)
Disgust	436 (1.52%)	56 (1.56%)	55 (1.53%)
Fear	4097 (14.27%)	496 (13.82%)	528 (14.71%)
Happy	7215 (25.13%)	895 (24.94%)	879 (24.49%)
Neutral	4965 (17.29%)	607 (16.91%)	626 (17.44%)
Sad	4830 (16.82%)	653 (18.19%)	594 (16.55%)
Surprise	3171 (11.04%)	415 (11.56%)	416 (11.59%)
TOTAL	28709	3589	3589

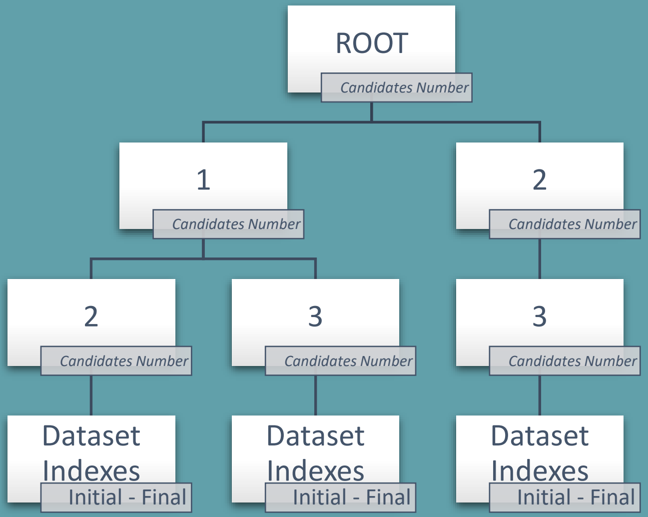
Feature Extraction Using Pretrained or Fine-Tuned FaceNet

FaceNet *extracts* 128 *features* from each image

Features are *scaled in* $[0,1]$ using *minmax_scale* function

Features *saved in CSV file* with image paths for easy image display in next steps

PP-Index Implementation



Pivot choice using *K-Medoids* or *Random* selection

Permutations database creation representing images as *permutations of pivots*

Prefix Tree construction (kept *in memory*)

Features dataset reordering (kept *on disk*)

Query Search

1. Query ***features extraction*** and *prefix computation*
2. ***Search*** for the *smallest subtree* with enough candidates
3. ***Sequential candidates features read*** in *reordered datastore* using initial and final indexes
4. ***K-NN selection*** using *cosine similarity* in *original features space*

Fine-Tuning FaceNet on Facial Expressions Dataset

- *Addition* of a *dense layer* of 7 neurons with *softmax activation* on top of the model
- *Training* the *classifier* (for comparison) for 10 epochs
- *Unfreezing FaceNet* completely and *training* for 100 epochs with *lower LR* using *Early Stopping* for *Fine-Tuning*

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
Angry	0.32	0.43	0.37	467
Disgust	0.10	0.45	0.16	56
Fear	0.27	0.11	0.16	496
Happy	0.54	0.48	0.51	895
Neutral	0.36	0.33	0.35	607
Sad	0.34	0.33	0.33	653
Surprise	0.41	0.53	0.46	415
Macro Avg	0.34	0.38	0.34	3589
Weighted Avg	0.39	0.38	0.37	3589
Accuracy			0.38	3589

CLASS	PRECISION	RECALL	F1-SCORE	SUPPORT
Angry	0.43	0.64	0.52	467
Disgust	0.60	0.38	0.46	56
Fear	0.47	0.32	0.38	496
Happy	0.83	0.80	0.82	895
Neutral	0.63	0.47	0.54	607
Sad	0.50	0.54	0.52	653
Surprise	0.68	0.83	0.74	415
Macro Avg	0.59	0.57	0.57	3589
Weighted Avg	0.61	0.61	0.60	3589
Accuracy			0.61	3589

Comparing the Performances of the two Search Engines

- **K** = *100* (*300* Candidates)
- **Pivot Selection** = *k-medoids*
- **Pivots** = *7*
- **Prefix Length** = *4*

Query images (used in all tests):

Angry *Disgust* *Fear* *Happy* *Neutral* *Sad* *Surprise*

Pretrained FaceNet:

MAP = *17.8%*

Fine-Tuned FaceNet:

MAP = *43.8%*

Optimization 1: finding the best *pivots-prefixes* configuration

- **Grid Search** approach
  - *Number of pivots* from 5 to 10
  - *Prefix length* from 2 to number of pivots - 1
- **Pivots selection** methods:
  - *Random* pivot selection
  - *K-Medoids* pivot selection
- **Summary Criterion**, useful for a quick evaluation:

$$Score = \frac{2}{3}MAP + \frac{1}{3}(1 - normalizedMNC)$$

  - **MAP:** Mean Average Precision
  - **MNC:** Mean Number of Candidates
  - **Normalized MNC :** MNC normalized considering 300 (minimum number of candidates to be retrieved) as minimum value and 57298 (cardinality of the whole dataset, private text set excluded) as maximum value
- **Noticeable results:**

Pivot Selection	N. Pivots	Prefix Length	M.A.P.	M.N.C.	Score
$K - Medoids$	7	4	45.1%	1 568	62.6%
$Random$	9	4	46.5%	1 475	64.4%
	10	4	50.7%	1 663	66.4%

  - In general, results seem to improve as *prefix length* approaches *number of pivots*
  - *K-Medoids* obtains best results for a lower number of pivots with respect to *Random selection*

Optimization 2: *K-Medoids* vs *Random Pivot* Selection

- ***K-Medoids*** cannot be performed in memory on the whole dataset, so it has been performed on a random sample (25000 objects ≈ 50% of the dataset)

Pivot Selection	N. Pivots	Prefix Length	M.A.P.
K-Medoids	7	4	**44.6%**
Random	10	4	**35.7%**

K-Medoids is more robust and generally better than *random* choice for pivots selection

Optimization 3: *Perturbation* vs *Non-Perturbation*

- ***Perturbated queries*** are generated by swapping pairs of the first 3 elements in the query prefix representation (6 queries = 5 perturbations + 1 original query)

Perturbated Queries (M.A.P.)	Single Query (M.A.P.)
43.1%	**52.2%**

The performances using query perturbation with k-medoids selection are inferior probably because perturbated queries retrieves more *non-relevant* candidates but with *higher similarity* than relevant ones from the original query

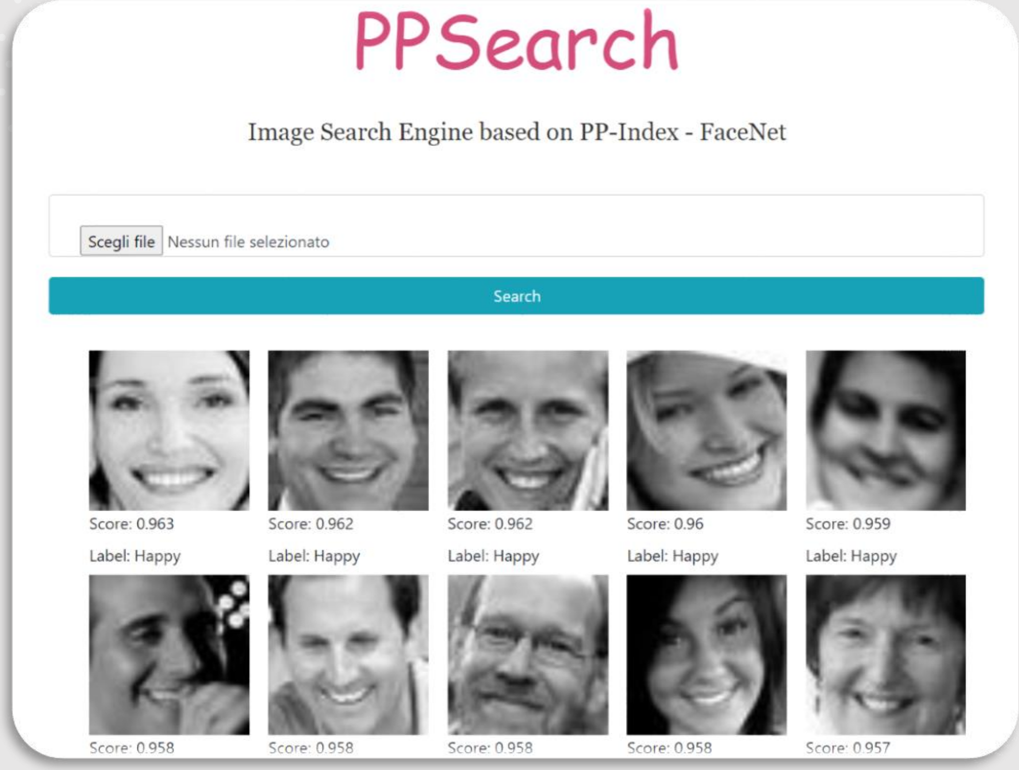
Optimization 4: *Multiple Indexes vs One Index*

- The use of ***random pivots*** with **multiple indexes** showed better perfomances than with a **single index**

- Using a **single index** or **multiple indexes** with ***K-Medoids*** doesn’t make significant differences in the search

Pivot Selection	Single Index (M.A.P.)	Multiple Indexes (M.A.P.)
Random	35.7%	43%
K-Medoids	44.6%	43.8%

Web App Developement



- ***Colab*** provides a **VM** that runs the app, which is exposed to a public URL using ***flask-NGrok***
- User can **query** any image
- **Web App** shows the 100 most similar images