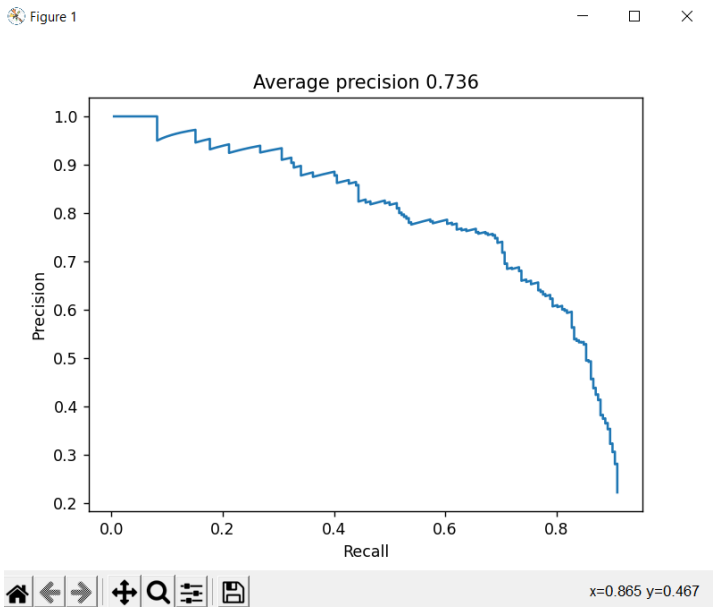
Dima Oana-Teodora 341

Concepte și Aplicații în Vederea Artificială - Tema 2
Detectarea și recunoașterea facială a personajelor din serialul de desene animate Familia Simpson



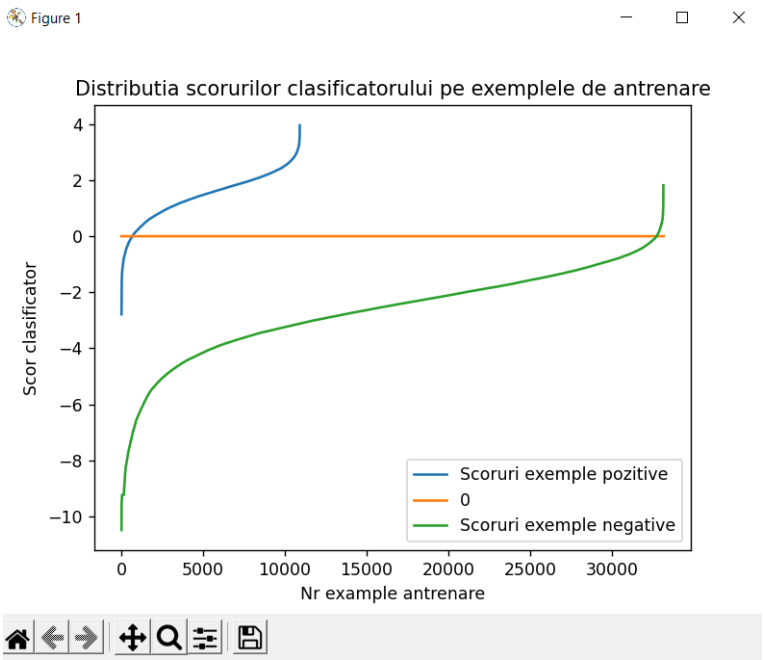
Task 1 - detectarea facială (Av P: 0.736)
Task 2 - recunoaștere facială

opencv-python==4.5.4.60
numpy==1.21.4
scipy==1.7.2
imutils==0.5.4
numpy==1.22.0
sklearn==1.0.2
skimage==0.19.1
matplotlib==3.5.1

*Rularea proiectului:
- nu necesită argumente
- se vor rula în această ordine
    1. *GeneretePositiveExamples.py* (o singură dată)
    2. *GenerateNegativeExamples.py* (o singură dată)
    3. *task1.py* sau *task2.py*

*Structura directoarelor de lucru:

```
- ./antrenare/
        - bart/
        - homer/
        - lisa/
        - marge/
        - bart.txt
        - homer.txt
        - lisa.txt
        - marge.txt
- ./diverse/
        - descriptoriFeteBune.npy
        - descriptoriNonFete.npy
        - model
- ./feteBune/
- ./nonFete/
- ./validare
        - ./simpsons_validate./
        - simpsons_validare.txt
        - task1_gt.txt
        - task2_bart.txt
        - task2_homer.txt
        - task2_lisa.txt
        - task2_marge.txt
```



*Structura proiectului

Parameters.py
Căile directoarelor de lucru cu fișierele proiectelor, dar și anumite valori constante necesare, se află definite în constructorul clasei *Parameters*.

Visualize.py
Conține două funcții (*show_detections_without_ground_truth*(detections, scores, file_names, params: Parameters) și *show_detections_with_ground_truth*(detections, scores, file_names, params: Parameters)) definite pentru afișarea grafică și salvarea imaginilor adnotate, alături de desenarea bounding boxurilor corespunzătoare.

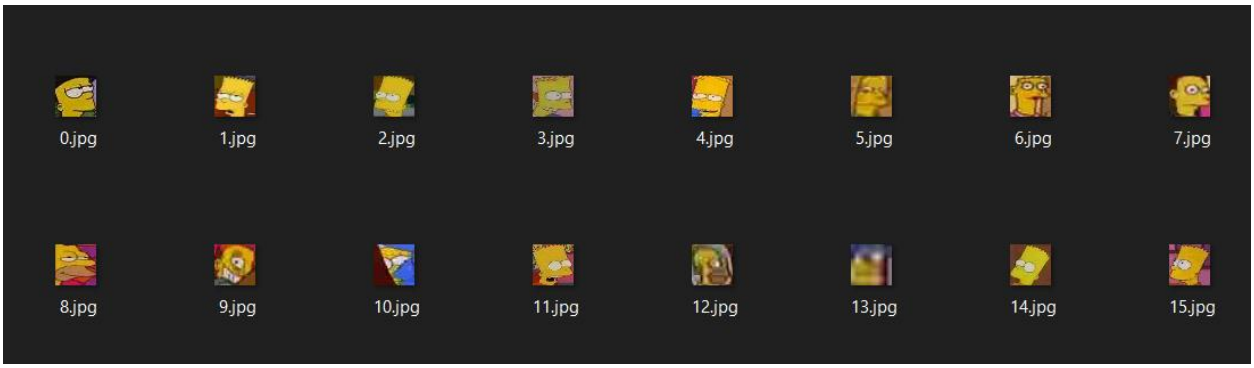
** Generarea fețelor pentru training:
GeneratePositiveExamples.py

Cele 4 fișiere *.txt* din folderul *./antrenare* conțin coordonatele fețelor, așa cum ar trebui să fie corect detectate. Fiecare față va fi citită din fișierul respectiv, salavată împreună cu coordonatele sale și cu numele caracterului corespunzător, apoi va fi redimensionată la 36 x 36 px, pentru o omogenitate a datelor de training. Mai jos, este reprezentat procedeul de procesare și de redimensionare al fiecărei fețe.

```
#compute positive examples using 36 X 36 template
for idx, img_name in enumerate(image_names):
    # print(idx,img_name)
    img = cv.imread(img_name)
    bbox = bboxes[idx]
    xmin = bbox[0]
    ymin = bbox[1]
    xmax = bbox[2]
    ymax = bbox[3]
    # print(xmin,ymin,xmax,ymax)
    face = img[ymin:ymax, xmin:xmax]
    # print("original face shape:",face.shape)
    face_warped = cv.resize(face, (height_hog, width_hog))
```

```
    # print("warped face shape:", face_warped.shape)
```

Rezultat: imagini 36 x 36 px
Numărul de fețe generate: 5454 (număr care se va dubla prin rotirea fiecărie imagini - *np.fliplr(img)*, pentru un set de date de antrenare care devine mai mare => procesul de augumentare a datelor)

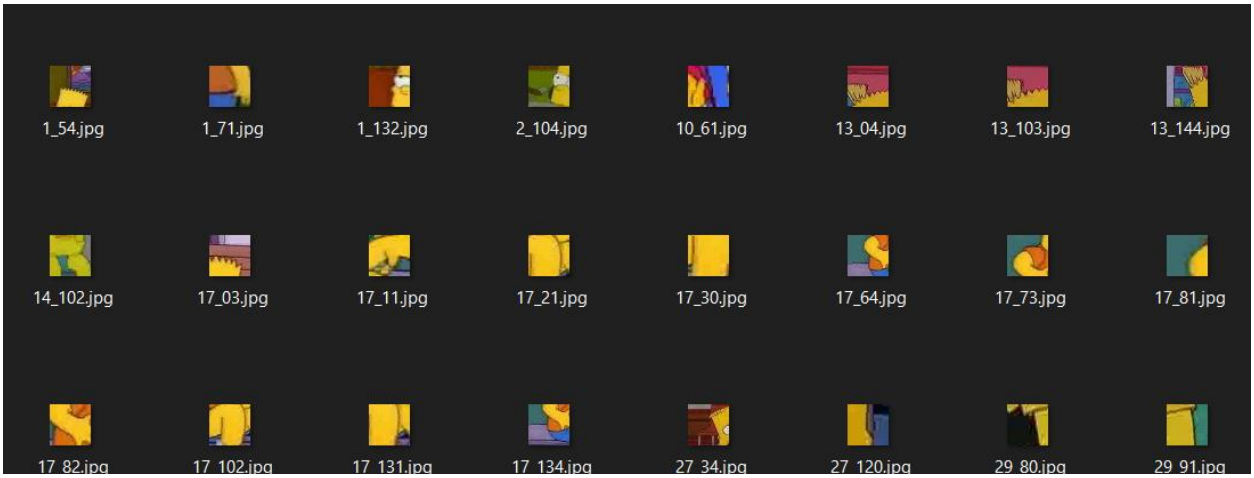


**Generarea non-fețelor pentru training:
GenerateNegativeExamples.py

Generează pentru fiecare imagine de antrenare maxim câte 15 non-fețe, care conțin și culoarea galbenă (în cazul de față, galbenul este culoarea pielii de pe fețele personajelor și nu ar ajuta generarea unor non-fețe care nu conțin culoarea pielii). Funcția *cv.inRange()* colorează cu negru orice pixel care are valoarea în afara intervalului *params.non_faces_yellow_min* și *params.non_faces_yellow_max* (două nuanțe de galben).

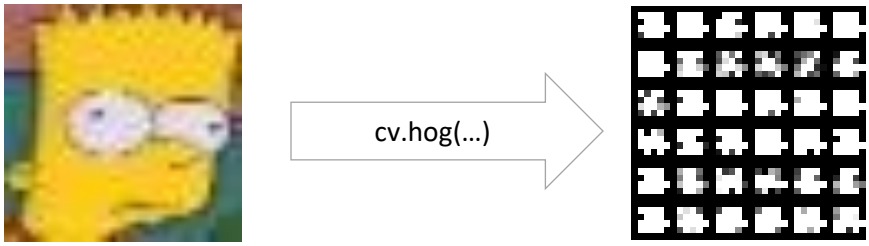
```
xmax = x + new_width_hog
ymax = y + new_height_hog
negative_example = img[y : ymax, x : xmax]

# ce e galben ramane galben, restul tranforma in negru
frame_treshold = cv.inRange(cv.cvtColor(negative_example, cv.COLOR_BGR2HSV),
                            params.non_faces_yellow_min,
                             params.non_faces_yellow_max)

if 50 <= frame_treshold.mean():
       # media valorilor pixelilor este mai mare ca 50
       negative_example = cv.resize(negative_example,
                                    (height_hog, width_hog))
```

Rezultat: imagini 36 x 36 pixeli
Numărul de non-fețe generate (aproximativ): 16.303



**Procesarea fetelor:

Se face prin intermediul The Histogram of Oriented Gradients (HOG) care facilitează detectarea obiectelor. În metodele din clasa *FacialDetector* se apelează *cv.hog(...)* pentru fiecare față și non-față pentru extragerea însușirilor descriptorilor pozitivi sau negativi.




```
for i in range(num_images):
            print('Procesam exemplul pozitiv numarul %d...' % i)
            img = cv.imread(files[i], cv.IMREAD_GRAYSCALE)
            features = hog(img, pixels_per_cell=(self.params.dim_hog_cell,
self.params.dim_hog_cell), cells_per_block=(2, 2), feature_vector=True)
            positive_descriptors.append(features)
            if self.params.use_flip_images:
                # augumentare de date prin utilizarea imaginilor inversate
```

```
                features = hog(np.fliplr(img),
pixels_per_cell=(self.params.dim_hog_cell, self.params.dim_hog_cell),
                               cells_per_block=(2, 2), feature_vector=True)
                positive_descriptors.append(features)


        positive_descriptors = np.array(positive_descriptors)
```

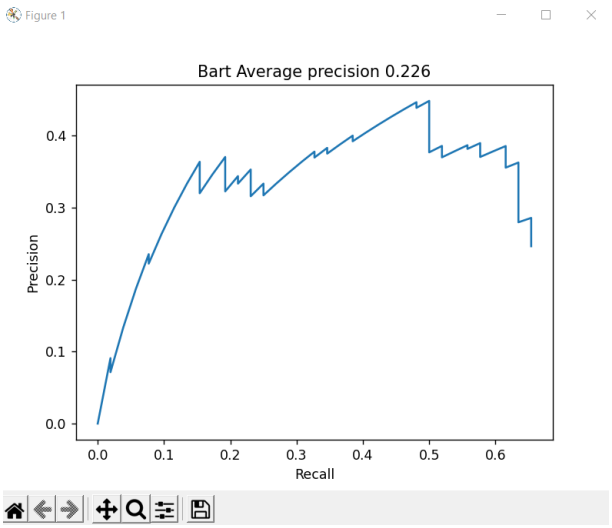
**Modelele utilizate:

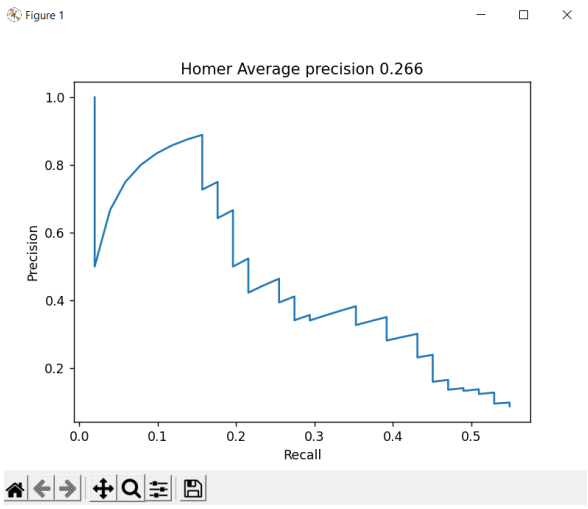
În învățarea supervizată, datele de instruire constau în: un set de exemple de antrenament, în care fiecare exemplu este o pereche formată dintr-o intrare și un valoarea de ieșire anticipată. Un algoritm de învățare supravegheat analizează datele de antrenament și apoi prezice clasificarea corectă pentru setul de date de intrare.[1]

Algoritmul de clasificare este de tipul Support Vector Machine (SVM).

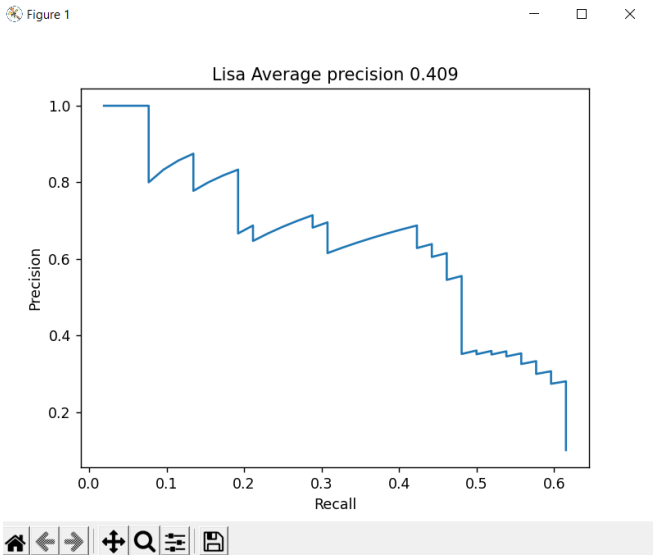
Pentru recunoașterea individuală a fețelor (task 2) am utilizat un SVC(C) (*CharacterRecognition.py*)

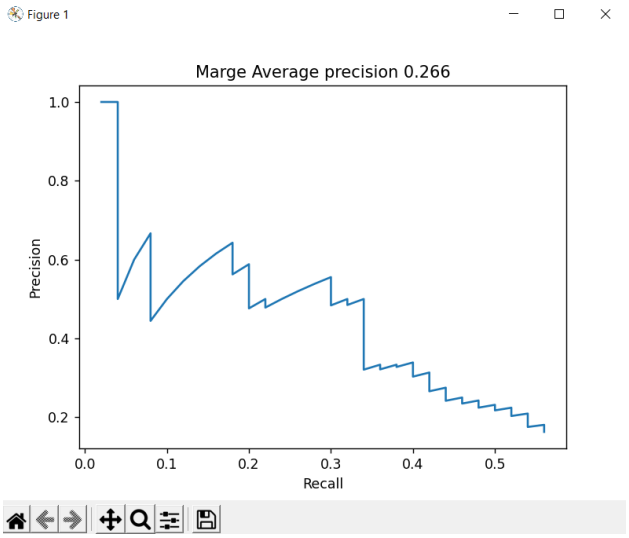
- Pentru recunoașterea tuturor fețelor (task 1) am utilizat LinearSVC(c) (*FacialDetector.py*).





[1] pagina 23, capitolul 2.5 Support Vector Machine - *Face Detection from Images Using Support Vector Machine*, Parin M. Shah





Bibliografie și resurse:

1. Codul de bază: Laborator 10 CAVA
2. https://docs.opencv.org/3.4/da/d97/tutorial_threshold_inRange.html
3. https://www.geeksforgeeks.org/python-os-path-exists-method/
4. https://www.geeksforgeeks.org/python-os-mkdir-method/
5. https://www.geeksforgeeks.org/python-opencv-cv2-imwrite-method/
6. https://numpy.org/doc/stable/reference/routines.array-creation.html
7. https://docs.python.org/3/library/os.path.html
8. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
9. https://www.tutorialkart.com/opencv/python/opencv-python-resize-image/
10. https://medium.com/featurepreneur/colour-filtering-and-colour-pop-effects-using-opencv-python-3ce7d4576140
11. https://book.pythontips.com/en/latest/enumerate.html
12. https://stackoverflow.com/questions/37535080/os-mkdir-under-if-not-working-python
13. https://www.w3schools.com/python/ref_math_pow.asp
14. https://theailearner.com/tag/cv2-inrange-opencv-python/
15. https://www.thepythoncode.com/article/hog-feature-extraction-in-python
16. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
17. https://numpy.org/doc/stable/reference/generated/numpy.fliplr.html
18. https://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/
19. https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/
20. https://matplotlib.org/stable/tutorials/introductory/pyplot.html
21. https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1322&context=etd_projects