Concepte și aplicații în Vederea Artificială

Tema 2

Detectarea și recunoașterea facială a personajelor din serialul de desene animate Familia Flintstone

Popovici Antonia-Adelina

Grupa 331

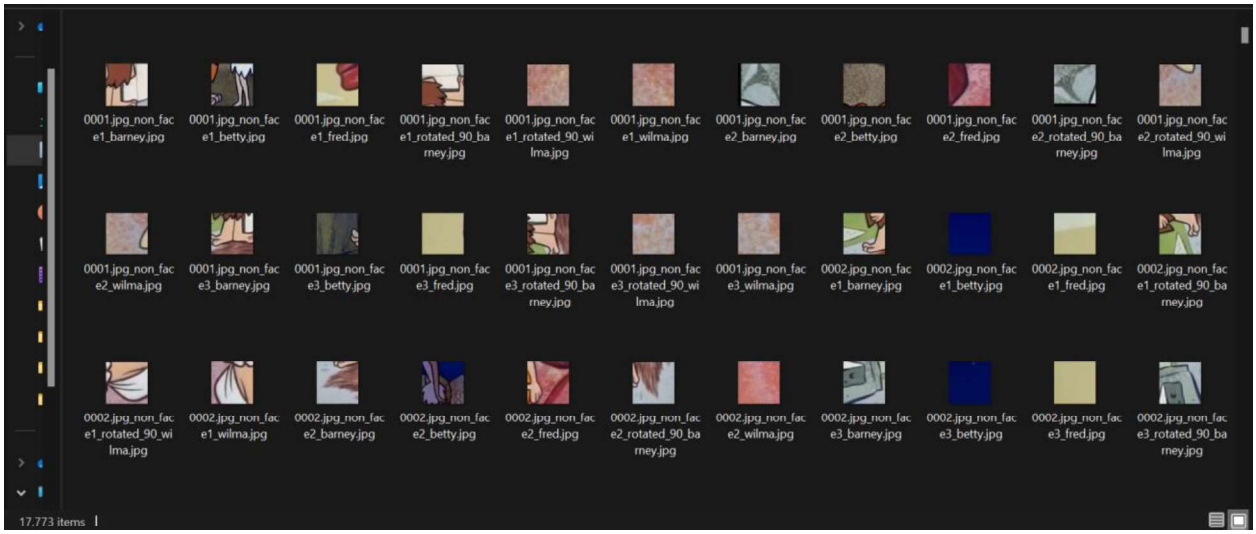
Task 1 – Detectarea facială

Pentru primul task, am implementat, precum în tema 3 de laborator, un algoritm de detectare facială folosind metoda glisării unei ferestre și histograme de gradient orientat.

Pentru realizarea descriptorilor pozitivi și negativi trebuie procesate imagini pozitive și negative. Imaginile pozitive au fost ușor de generat, întrucât m-am folosit de adnotările din fișierul de antrenare și de imaginile date. Am generat, astfel, 6977 de imagini pozitive, cu dimensiuni 64x64.

În cazul imaginilor negative, am încercat, inițial, să generez un set random de imagini care nu se intersectează cu fețele personajelor. Am observat un Average Precision slab, în jur de 30-40%, așa că am mai făcut un set de imagini negative care au un input în plus: zona să conțină culoarea pielii cu o toleranță de 20. Am urmărit generarea de zone cu mâini, picioare, gât etc. Acuratețea s-a mărit la 50%.

De asemenea, am rotit imaginile, am augmentat etc. pentru a genera cât mai multe imagini negative:



În clasa *FacialDetector.py* am generat descriptorii prelucrându-i în diverse moduri, în încercarea de a avea cât mai mulți descriptori pozitivi și negativi. Printre tipurile de descriptori HOG și metodele de prelucrare utilizate, se numără:

- imagine gri
- HOG standard
- HOG pentru imaginea oglindită (*flip*)
- HOG pentru imaginea rotită (45 de grade)
    - am rotit imaginea cu un unghi de 45 de grade
    - am aplicat algoritmul HOG pe imaginea rotită.
- HOG cu *gamma adjust* aleator între 0.5 și 1.5
- HOG cu transformare *shear* între -20 și 20 de grade

- HOG cu zgomot aleator între 0.01 și 0.1.

```
for i, image_file in enumerate(files):
    print(f"Imagine procesata: {i + 1}/{len(files)}")
    image = cv2.imread(image_file)
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    hog_image = hog(gray_image, pixels_per_cell=(6, 6), cells_per_block=(
        2, 2), feature_vector=True)  # pixels_per_cell = 6, 6, cells_per_block = 2, 2
    positive_descriptors.append(hog_image)

    hog_image = hog(np.fliplr(gray_image), pixels_per_cell=(
        6, 6), cells_per_block=(2, 2), feature_vector=True)
    positive_descriptors.append(hog_image)

    angle = 45
    rotated_image = rotate(
        gray_image, angle, reshape=False, mode='nearest')
    hog_image_rotated = hog(rotated_image, pixels_per_cell=(
        6, 6), cells_per_block=(2, 2), feature_vector=True)
    positive_descriptors.append(hog_image_rotated)

    jittered_image = exposure.adjust_gamma(
        image, gamma=np.random.uniform(0.5, 1.5))
    hog_image_jittered = hog(cv2.cvtColor(jittered_image, cv2.COLOR_BGR2GRAY), pixels_per_cell=(
        6, 6), cells_per_block=(2, 2), feature_vector=True)
    positive_descriptors.append(hog_image_jittered)

    sheared_transformer = transform.AffineTransform(
        shear=np.deg2rad(np.random.uniform(-20, 20)))
    sheared_image = transform.warp(gray_image, sheared_transformer)
    hog_image_sheared = hog(sheared_image, pixels_per_cell=(
        6, 6), cells_per_block=(2, 2), feature_vector=True)
    positive_descriptors.append(hog_image_sheared)

    blurred_image = util.random_noise(
        gray_image, var=np.random.uniform(0.01, 0.1))
    hog_image_blurred = hog(blurred_image, pixels_per_cell=(
        6, 6), cells_per_block=(2, 2), feature_vector=True)
```

În final, am ajuns la 41862 descriptori pozitivi și 99426 descriptori negativi, fiecare având 2916 caracteristici.

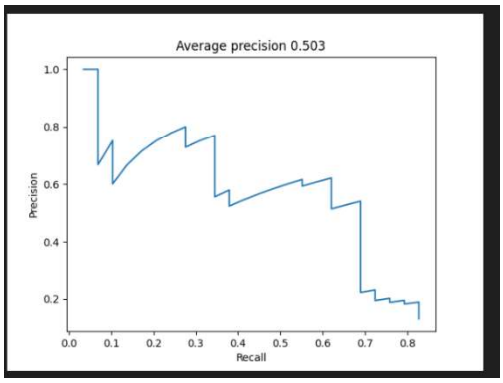
Antrenarea modelului a fost realizată similar cu modelul antrenat la laborator, folosind *LinearSVC*.

Pentru imaginile de testare, am procesat în același mod ca în generarea descriptorilor. Glisarea am realizat-o prin diverse scalări ale ferestrei:

```
scale_factors = [
    (0.8, 1.2), (1.0, 1.5), (1.2, 0.8),
    (0.7, 1.3), (1.5, 1.0),
    (0.9, 1.1), (1.1, 0.9),
    (0.5, 1.5), (1.5, 0.5),
    (1.8, 2.5), (2.0, 3.0),
    (0.4, 2.0), (2.0, 0.4),
    (0.3, 3.0), (3.0, 0.3),
    (0.2, 4.0), (4.0, 0.2),
    (0.1, 5.0), (5.0, 0.1)
]
for scale_factor_w, scale_factor_h in scale_factors:
    scaled_window_size = (
        int(original_window_size[0] * scale_factor_w),
        int(original_window_size[1] * scale_factor_h)
    )
```

Am adăugat și o verificare a culorii feței, în încercarea de a mări procentajul (cu toate că există și personaje cu fețe colorate verde, albastru etc., am considerat că e mai util să detectez fețe ce păstrează *skin tone*-ul, pentru că sunt mai multe).

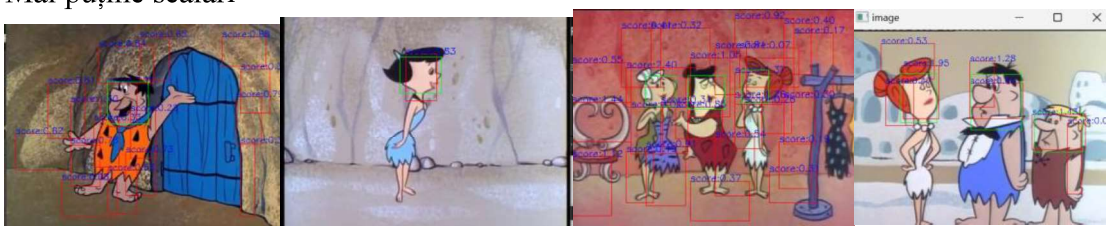
```python
def check_skin_color_percentage(img):
    hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    lower_skin = np.array([0, 20, 70], dtype=np.uint8)
    upper_skin = np.array([20, 255, 255], dtype=np.uint8)

    skin_mask = cv2.inRange(hsv_img, lower_skin, upper_skin)

    skin_percentage = np.sum(skin_mask > 0) / (img.shape[0] * img.shape[1])

    return skin_percentage


def is_uniform_color(patch, threshold=30):
    std_dev = np.std(patch)
    return std_dev > threshold
```



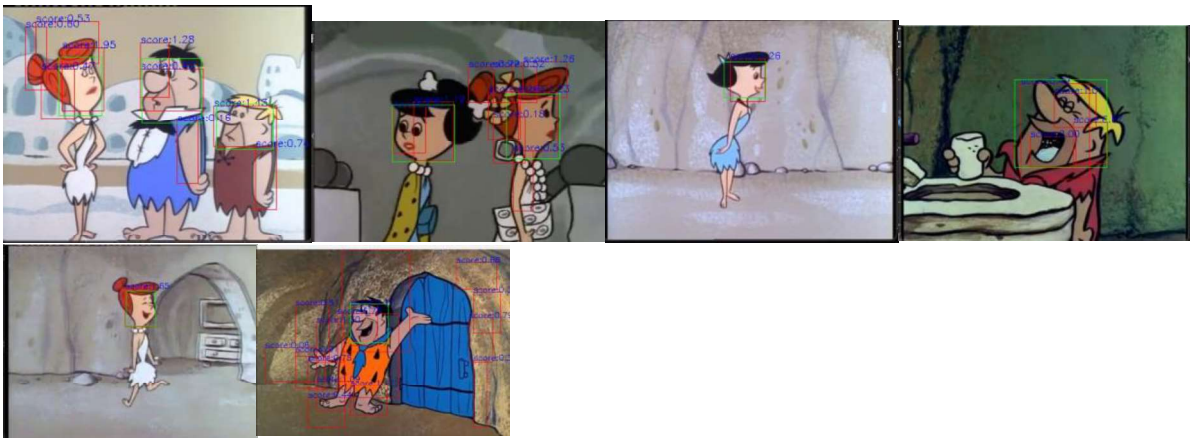
În plus, am aplicat *non-maximal-surpression*, precum la laborator. În final, Average Precision este în jur de 50%.

AP-ul pentru mai puține dimensiuni ale ferestrei de glisare era mai scăzut. Am adăugat câteva poze cu mai puține scalări. Am adăugat noi dimensiuni și AP-ul a crescut. Aș fi putut adăuga și mai multe dimensiuni, însă timpul de procesare al unei imagini ar fi crescut semnificativ.

1. Mai puține scalări



2. Mai multe scalări



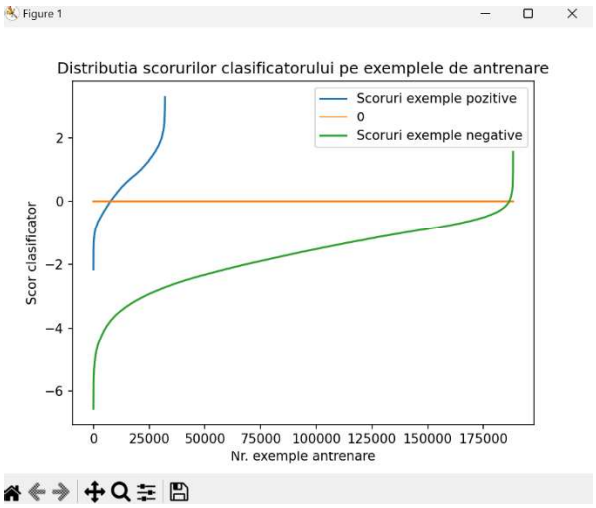
Task 2 - Recunoaștere Facială

Am implementat codul doar pentru recunoașterea facială a lui Barney. Partea complexă a fost să generez imagini pozitive și negative.

Am considerat poze pozitive tot ce conține fața lui Barney și ca imagini negative restul fețelor și imagini generate pentru task-ul anterior.

```
negative_images_folder_barney = os.path.join(
    'exemple_caractere/barney', 'imaginiNegative')
negative_images_folder_betty = os.path.join(
    'exemple_caractere/betty', 'imaginiNegative')
negative_images_folder_fred = os.path.join(
    'exemple_caractere/fred', 'imaginiNegative')
negative_images_folder_wilma = os.path.join(
    'exemple_caractere/fred', 'imaginiNegative')


copy_images('fete_caractere/barney_poz',
            positive_images_folder_barney, 'barney', 'barney')
copy_images('fete_caractere/barney_neg',
            negative_images_folder_barney, 'barney', 'barney')
copy_images('fete_caractere/betty_pos',
            negative_images_folder_barney, 'barney', 'betty')
copy_images('fete_caractere/fred_pos',
            negative_images_folder_barney, 'barney', 'fred')
copy_images('fete_caractere/wilma_pos',
            negative_images_folder_barney, 'barney', 'wilma')
copy_images('exemple_caractere/imaginiNeg',
            negative_images_folder_barney, 'barney', 'img_neg')
```

Am antrenat tot un *LinearSVC* cu un număr mai mare de descriptori negativi, care au fost generați în același mod ca la task-ul 1.



Din lipsă de timp, nu am realizat AP-ul și pentru celelalte personaje, însă acestea ar fi fost realizate în exact același mod. Din același motiv, am testat numai pe 15 poze în speranța că rezultatul pe setul mare de poze o să fie într-o oarecare măsură acceptabil.

Rezultatul final pentru recunoașterea fețelor lui Barney a fost egal cu 50.8%. Nu îl consider un rezultat relevant, având în vedere cele menționate anterior.