Recunoasterea faciala in desenul animat
"Laboratorul lui Dexter"

Task-ul 1- Detectarea tuturor fetelor din imagine

Generarea exemplelor pozitive si negative

Pentru a putea antrena clasificatorii este nevoie de extragerea unor poze cu fetele din desenul animat si cu decor sau non-fete. Am folosit imaginile cu adnotarile din folderul `antrenare/`.

In clasa `DatasetGenerator.py` am implementat functia `generate_positive_examples()` in care parcurgem folderul personajelor din `antrenare/`. Pentru fiecare imagine citim adnotarile specificate si adaugam un buffer de 10px inainte de crop. Dupa ce am taiat imaginea apelam functia `get_crop_type()` pentru a afla in ce forma geometrica poate fi incadrata imaginea. Apoi dam resize imaginii pentru a uniformiza dimensiunile, tinand cont de parametrii definiti in:

```
self.crop_sizes = {
    'rectangle_vertical': (54, 66),
    'square': (54, 54),
    'rectangle_horizontal': (66, 54)
}
```

Apoi imaginea redimensionata este salvata in folderul corespunzator.

Pentru generarea exemplelor negative am parcurs din nou toate imaginile si am selectat din fiecare 3 patch-uri cu limite de dimensiuni predefinite. De asemenea, am testat cu functia `iou` ca bucata selectata sa nu se intersecteze mai mult de 10% cu detectiile fetelor din fisierul text cu adnotari.

La final, in folderul `data` am avut un folder `ExemplePozitive` care includea 3 foldere separate cu imagini redimensionate in functie de forma fetei detectate si un folder `ExempleNegative` cu aproximativ 12.000 de patch-uri de dimensiuni diferite cu non-fete.

Extragerea trasaturilor folosind descriptori HOG

In `RunProject.py` apelam functia `get_positive_descriptors(self, shape)` pentru fiecare forma definita cu scopul de a antrena trei clasificatori. Functia este aceeasi de la laborator, dar a fost adaugata libraria `albumentations` pentru a mari numarul de trasaturi extrase de descriptorii HOG.

Functiile si parametrii alesi din libraria albumentations:

```
self.transform = A.Compose([
    A.GaussianBlur(blur_limit=(3, 5), sigma_limit=(0.1, 0.5)),
    A.RandomBrightnessContrast(),
    A.MedianBlur(blur_limit=3),
    A.Downscale(scale_range=[0.6,0.9])
], p=1.0)
```

Utilizarea lor in functie:

```
def get_positive_descriptors(self, shape):
        ....
        # adaugam 2 imagini transformate cu libraria albumentations
        for _ in range(2):
            transformed_image = self.transform(image=img)['image']

            features = hog(transformed_image,
                           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:
                features = hog(np.fliplr(transformed_image),
                               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)

    return np.array(positive_descriptors)
```

Se genereaza descriptorii negativi cu functia preluata din laborator `get_negative_descriptors(self, shape)`, modificata pentru a redimensiona imaginile in functie de forma geometrica a ferestrei clasificatorului. In final, toti descriptorii sunt salvati in `data/SalveazaFisiere/`.

Antrenarea clasificatorilor

O data ce descriptorii pozitivi si negativi au fost salvati pentru forma curenta, apelam functia `train_classifier(shape)`. Aceasta a fost modificata pentru a antrena un model pentru fiecare forma geometrica diferita. La final, fiecare model este salvat in `data/SalveazaFisiere/`.

Rularea pe folderul cu imagini de test

Pentru a incepe procesul de detectie apelam functia `run()`. Recuperam path-urile imaginilor din folder-ul de test si definim o lista de scale-uri la care vom redimensiona fiecare imagine. Fetele personajelor din desen sunt in general mari asa ca am selectat mai multe scalari la dimensiuni mai mici pentru a putea detecta fetele mari.

Totodata folosim un pas variabil in cadrul detectiei:

- pentru imaginile mici fereastra va glisa mai putin pentru a nu pierde detalii
- pentru imaginile mai mari va glisa mai mult pentru a nu detecta fete acolo unde este decor

Dupa ce fiecare model a facut propriile detectii pe toate scalarile apelam functia `non_maximal_suppression()` pentru a suprima din detectiile care se suprapun cu un procent mai mare de 30%. In final adaugam detectiile si scorurile la liste.

Salvam detectiile, scorurile si fisierele ca `numpy array` in folderul `341_Popeanga_Antonia/task1`. Acuratetea a fost de 62.4%, tinand cont de precizie si recall pentru detectia tuturor fetelor din folderul de 200 de imagini din 'validare/validare'.

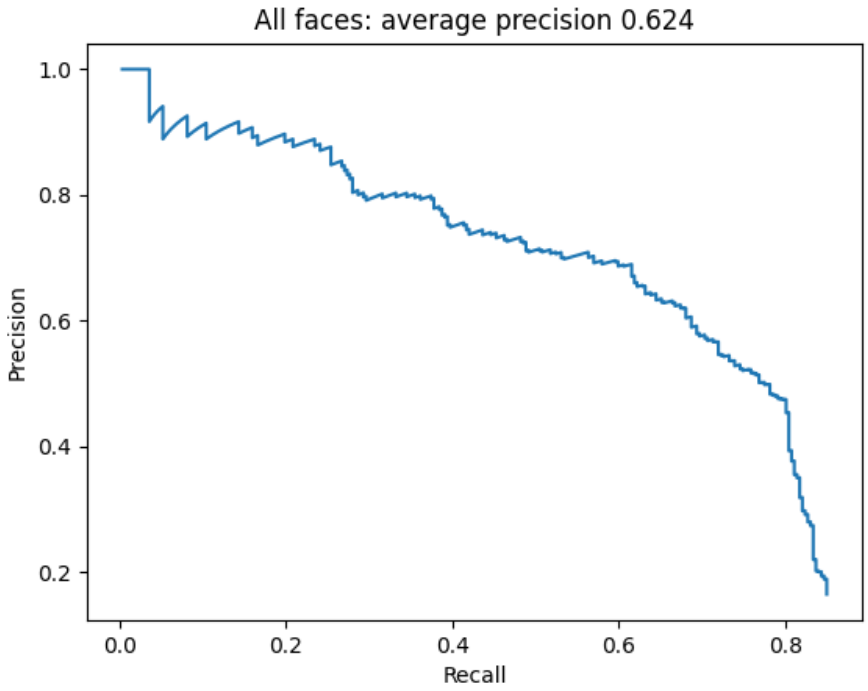

Figure 1: Precizia medie pentru detectia tuturor fetelor din desenul animat "Laboratorul lui Dexter"

Task-ul 2- Detectarea si recunoasterea fetelor personajelor principale

Generarea exemplelor pozitive si negative

Pentru generarea exemplelor pozitive, am preluat imaginile din folderele de exemple pozitive specifice pentru fiecare forma.In functia `generate_positive_examples_by_character()` le grupam pe personaje si le redimensionam in functie de dimensiunea definita in parametrii. Pentru exemplele negative vom prelua pentru fiecare personaj 7000 de imagini in mod aleatoriu din fisierul de exemple negative.

Generarea descriptorilor pozitivi/negativi si antrenarea

Functiile pentru generarea descriptorilor pozitivi si negativi sunt identice cu cele de la task-ul 1, folosind aceleasi functii din libraria albumentations pentru a augumenta numarul exemplelor pozitive. Singurele modificari au intervenit la fereastra fiecarui personaj, prin testare am ajuns la concluzia ca o fereastra mai mare cu dimensiunea celulei hog de 8px ofera cele mai bune rezultate.

```
self.character_sizes={
            'mom': (120,96),
            'dad':(96,80),
            'deedee':(80,112),
            'dexter':(96,96)
        }
```

Totodata, functia de antrenare este identica cu cea de la task-ul 1, doar ca in loc sa faca antrenarea modelelor pentru patrat/dreptunghi face pentru dad/deedee/dexter/mom.

Rularea pe folderul cu imagini de test

Pentru a testa detectorul facial specializat apelam functia run() din clasa CharacterFacialDetector.py. Procesul este asemanator, scalam imaginea originala de mai multe ori, si pentru fiecare scalare trecem cu fereastra glisanta a fiecarui personaj peste imagine. Fata de detectia pentru toate fetele, am observat ca pastrand pasul=1 pentru fereastra glisanta obtinem cele mai bune rezultate. Dupa ce fiecare model a facut detectiile pentru fiecare scalare, aplicam `non_maximal_suppression()` pentru a elimina detectiile care se suprapun prea mult. Adaugam detectiile impreuna cu scorurile si fisierele in dictionarul result[] la cheia potrivita fiecarui nume de personaj. La final, salvam aceste rezultate pentru fiecare personaj ca numpy array in folderul `341_Popeanga_Antonia/task2`.

Cu aceasta abordare am obtinut o acuratete de 27.3% dad, 46% deedee, 30.3%dexter si 36.7% mom.

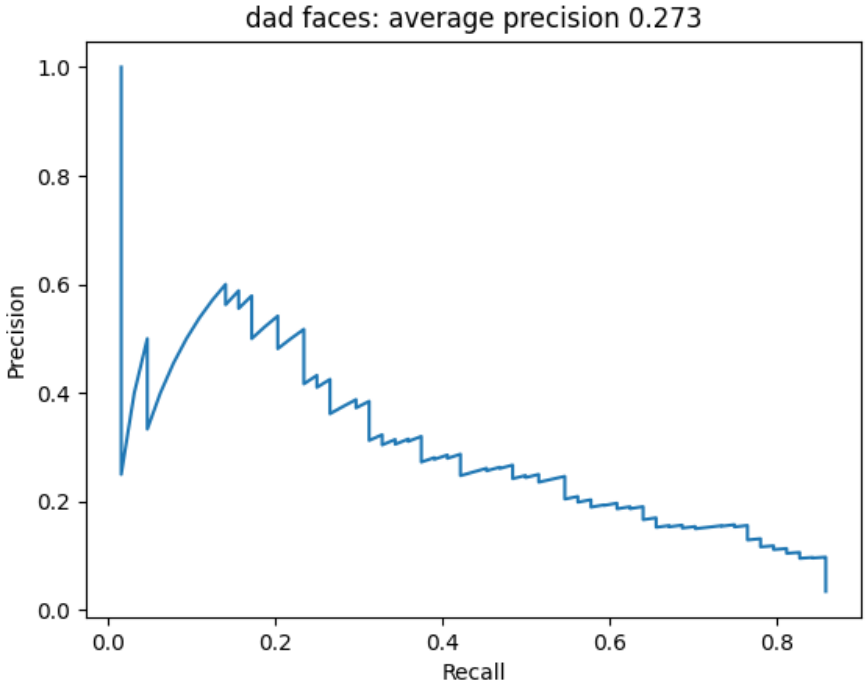

Figure 2: Precizia medie pentru detectia lui Dad

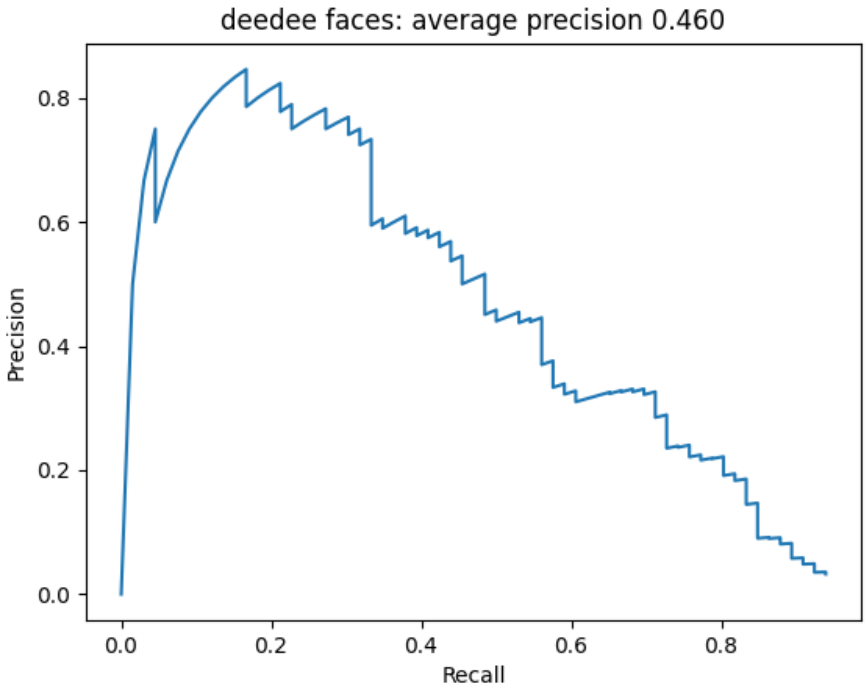

Figure 3: Precizia medie pentru detectia lui Deedee

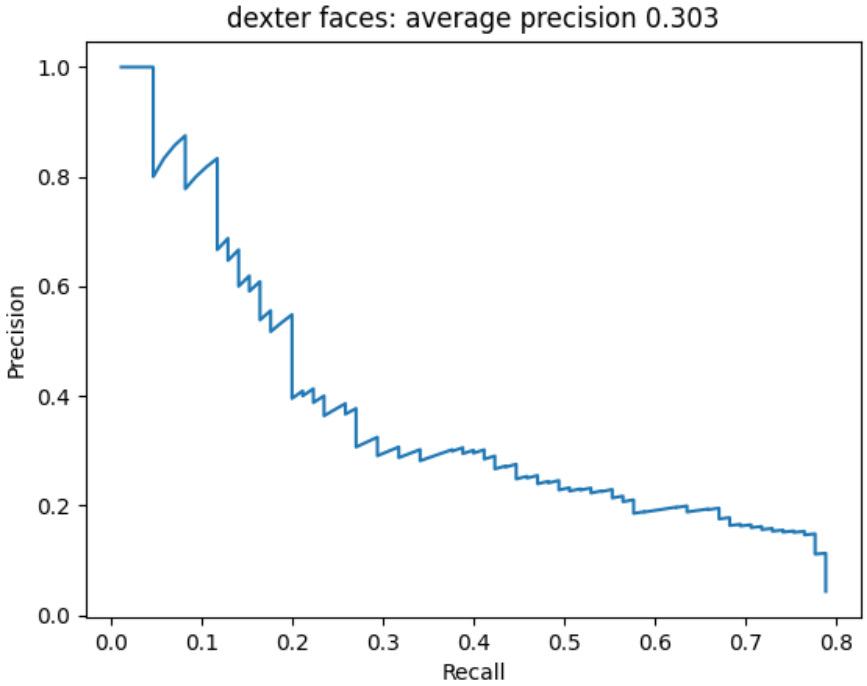

Figure 4: Precizia medie pentru detectia lui Dexter

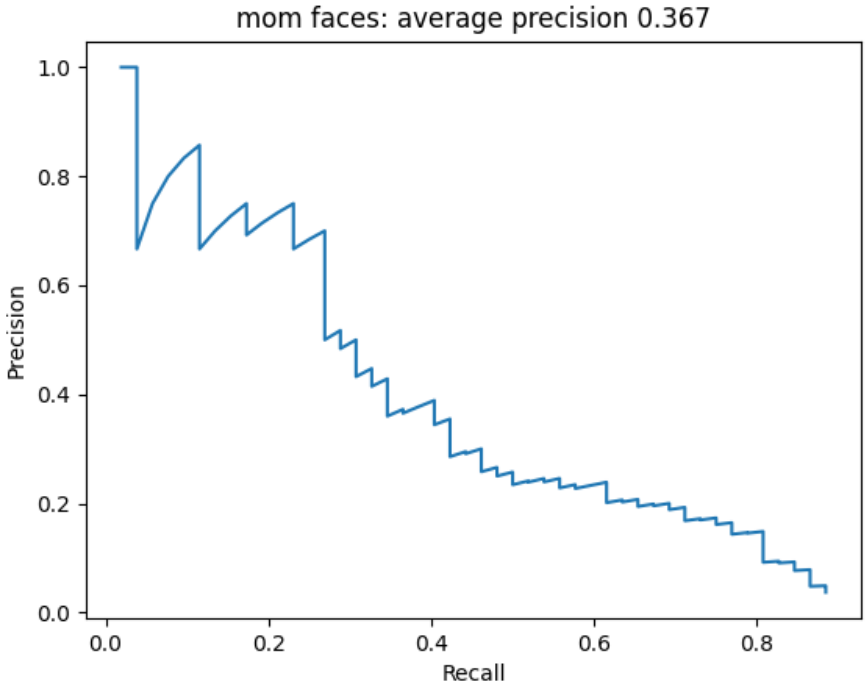

Figure 5: Precizia medie pentru detectia lui Mom